



TIBCO Business Studio™ - BPM Edition

Application Designer's Guide

Version 5.4.0

April 2023

Document Updated: October 2023



Contents

Contents	2
Accessing Help	10
Modeling Processes	12
TIBCO Business Studio - Cloud BPM Edition Usage	12
Process Modeling Methodology	12
Model a Process	13
Projects, Assets, and Project Organization	14
Projects	14
Project Lifecycle	15
Creating a Project	17
Deploying a Project	19
Deploying Projects from the Command Line	21
Application Deployment Build Pipeline Example	22
Project References	22
Copy and Reuse an old Project	26
Process Packages	27
Creating a Package	27
Copying a Package	28
Processes	29
Pageflow Processes and Business Services	30
Pageflow Processes	30
Creating a Business Process	37
Creating a Case Action Process	39
Service Process Modeling	42
Activities	47
Using Forms with User Tasks	55

Using Data Fields and Parameters with Process User Tasks	68
Process Components	72
Sending an Email Message from a Process	73
Using Email Templates	80
Deleting a Process	84
Process and Service Process Interfaces	85
Process Interfaces	86
Process Migration	91
Design Considerations for Process Migration	91
Migration Point Restrictions	91
Migration Point Viewing	93
Example of Process Instance Migration	94
Process Fragments	96
Using the Predefined Fragments	96
Creating Custom Fragments	97
Importing Custom Fragments	97
Resource Patterns and Work Distribution	98
Separation of Duties	98
Retain Familiar	101
Chained Execution	104
Piling	105
Distribution Strategy	106
Sub-Processes	107
Refactoring Sub-Processes	109
Synchronous and Asynchronous Sub-Process Invocations	110
Creating Call Sub-Processes	111
Embedded Sub-Processes	115
Inline Sub-Process in Detail	128
Event Sub-Processes	131
Dynamic Sub-Processes	133
Making a Sub-Process Call to a Process Interface	138

Controlling Flow	140
Controlling Flow in a Process	140
Flows	140
Loops	141
Gateways	149
Configuring a Task as Automatic Ad-Hoc	155
Configuring a Task as Manual Ad-Hoc	156
Refactoring Ad-Hoc Activities	157
Exclusive Event Based Gateway Example	158
Controlling Flow from an External Application	159
Using Data	161
Using Process Data	161
Process Data Overview	161
Data Fields and Parameters	161
Using the Properties View to Create a New Data Field or Parameter	166
Using the Wizards to Create a New Data Field or Parameter	166
Participants	169
Dynamic Organization Participants	181
Auditing Process Data	183
Correlation Data	185
Associating Process Data with Events and Tasks	188
Using Process Data to Create Tasks	192
Using Business Data	193
Business Object Models	193
Business Object Modeler Tasks	199
Business Data Project Deployment	207
Business Data	208
Business Data Definition	217
Business Data Scripting	230
Business Data Modeling Best Practice	252
Business Data Services Glossary	255

Advanced Business Data Concepts	255
Using Case Data	260
Case Data Overview	260
Case Data Terminology	262
Creating and Deploying a Case Data Model	264
Upgrading a Case Data Model	278
Deleting a Case Data Model	287
Using a Case Data Model in a Business Process	287
Operators	308
Values	309
Generating a Business Service to Create/Update/Delete Case Data	320
Configuring Events	323
Events	323
Start Events	323
Intermediate Events	324
End Events	325
Setting Event and Task Visibility (Private and Public)	326
Throw and Catch Events	327
Creating References Between Throw and Catch Events	327
Throw and Catch Signal Events	328
Throw and Catch Error Events	330
Event Handlers	333
Signal Event Handlers	334
Using the Cancellation Event Handler	335
Event Handlers in Pageflows	338
Event Persistence	338
Compensation Events Configuration	339
Timer Event Scripts	340
Configuring Timer Event Scripts	341
Task Scripts on Events	342

Timer Event Rescheduling	342
Signals	344
Creating a Global Signal Definition Project	346
Creating Payload Data for Global Signals	347
Sending signals between processes	349
Creating a Global Signal Definition Project	349
Creating Payload Data for Global Signals	349
Correlation Data	351
Using Scripts	352
Using Scripts	352
Implementing Scripts	353
Using Process Data in Scripts	355
Factory Methods	356
WorkManager and Process Classes	357
Enumerations	357
JavaScript Exclusions	358
Data Mapping	359
Sample Scripts	363
Supplemental Information	365
Process Manager and Work Manager Scripting	369
Troubleshooting Scripting	382
Calling REST Services	386
Defining the Interface to an External REST Service	386
Creating JSON Schemas	388
Creating JSON Schemas from a JSON Sample	389
Configuring the Process Project from Which you Want to Call the REST Service ..	390
Using a System Participant to Identify the REST Service Shared Resource	391
Defining Input and Output Mappings	391
Exclusion of Empty Content from REST Service Input	392

Workflow Process and Data Pattern Support	395
Workflow Data Patterns Support	395
Workflow Process Patterns Support	400
Workflow Resource Patterns Support	408
Using Organization Models	413
Organization Model Concepts	413
Organization Model Deployment	413
Data Types	413
The Organization Model as an Analysis Tool	414
The Organization Model at Runtime	415
Elements of Organization Models	416
Benefits of Organization Models	417
Organization Model Creation	418
Creating a Project Containing an Organization Model	418
Adding an Organization Model in an Existing Project	419
Creating an Organization Model	419
Creating a Schema	421
Organization Modeler Diagram Editors	429
Adding Objects in Organization Modeler Diagrams	431
Adding Notes and Labels	436
Tailoring the Appearance of Organization Modeler Diagrams	437
Organizations	439
Creating an Organization	439
Hierarchies and Associations	441
Organization Units	442
Creating an Organization Unit	443
Large Organizations with Branch Networks	444
Dynamic Organizations	445
Creating a Dynamic Organization	446
Dynamic Organization Units	447

Position	449
Creating a Position	450
Groups	451
Creating a Group	452
Participants	452
Participants from the Organization Model	453
Dynamic Organization Identifier Mapping	454
Dynamic Organization Participants	455
Capabilities and Privileges	455
Creating Capabilities	456
Creating Capability Categories	457
Creating Privileges	458
Creating Privilege Categories	459
Locations	459
Creating a Location	460
Resources	460
Resources and TIBCO Business Studio - BPM Edition	461
Creating a Resource	462
Queries	462
Creating Queries	463
System Actions	464
System Actions Reference	465
Schemas	472
Organization Modeler Default Schema	474
Attributes	477
Managing Work Using Organization Models	480
Using Organization Models in a Process	481
Creating or Importing an Organization Model	481
Organization Model Entities as Process Participants	482
Mapping Resources to the Organization Model	484
About Participants	485

Defining How Work is Assigned to Users	492
Resource Query Language	499
Best Practices When Using Resource Query Language (RQL)	499
RQL Structure	501
RQL Cleanup Configuration	511
Work List Facade	512
Work Item Attributes	512
Creating a Work List Facade	513
Setting the Display Label for Work Item Attributes	514
Mapping Process Data to Work Item Attributes	515
Importing Pre-Version 5.x Projects into TIBCO Business Studio - BPM	
Edition Version 5.x	517
What's Changed?	517
Importing Pre-Version 5.x Projects	520
Post migration tasks	521
Business Object Model Data Types	522
WSDL Activities	522
Other Service Tasks	523
Data Mapping	524
Participant Configurations	524
Multi-Instance Sub-Processes	524
Scripting	525
Forms	528
TIBCO Documentation and Support Services	531
Legal and Third-Party Notices	533

Accessing Help

You can access Help either online when using TIBCO Business Studio™ - BPM Edition or by downloading it for use at any time.

i Note: TIBCO provides Out-Of-Cycle updates of product documentation. If you have downloaded the help already, and want to be notified of these updates for TIBCO Business Studio - BPM Edition, navigate to the current version of TIBCO Business Studio - BPM Edition documentation on the TIBCO Product Documentation page, and click **Subscribe** to subscribe to the RSS feed. When notified of an update you can re-download the html from the Product Documentation site to use the latest version. Copy it to the location of the existing help and access it from TIBCO Business Studio - BPM Edition, or download it as described below to replace previously downloaded help. Alternatively, you can go back to Preferences and re-download the help from there.

i Note: If your environment requires internet access via a proxy, then go to **Window > Preferences > General > Network Connections** and configure the proxy settings.

Procedure

1. Go to the **Help** menu.
2. Select **Download Offline TIBCO Help...**. This guides the user to the appropriate Preferences page (**Preferences > TIBCO Help > Content**).
 - **Content access strategy: Prefer offline help (if available).** (default) is already selected, but online help is used until help is downloaded. The alternative selection is **Always use online help.**
 - To define where you want the offline help to be located, browse to the location in which you want to save the **Base offline documentation folder** and click **Apply**. You can click **Location** to go to the location that is defined.
 - Select the help you want to download and click **Download** and **OK**.

- The help is downloaded to the location specified in the **Base offline documentation** field. If the **Prefer offline Help (if available)**. content access strategy is selected then the downloaded help content is used. Once the help is downloaded, then the **Help > Menu** items open the downloaded offline help.

Modeling Processes

TIBCO Business Studio - Cloud BPM Edition Usage

TIBCO Business Studio™ - Cloud BPM Edition is intended to be used by business analysts and solution designers, including those responsible for the analysis, design, implementation, and deployment of business processes.

TIBCO Business Studio - Cloud BPM Edition is an offline application development environment in which applications can be designed and deployed to the run-time deployment manager service.

Process Modeling Methodology

Modeling a process can be achieved in several different ways, however the general approach described in this topic reflects best practices.

Define the "As is" Process

- Interview business end users about their current practices.
- Capture the process flow (either on paper first or directly in modeling tool).
- Capture the process relevant data (either on paper first or directly in modeling tool).
- Attach the process-relevant data at key points (for example, at decision points or certain activities).

Define the "To be" Process (Optional) This is an iterative exercise in which you:

- Propose optimizations (process changes and new automation of existing processes).
- Define the Business Object Model and Organization Model (Optional).
- Define a *business object model* that defines key business terms specific to your

corporate environment (for example, in an insurance environment, a claim, claimant, and so on). This can be used as an analysis tool.

Hand Over for Implementation (Optional)

You should do the following:

- Check the Problems view for any warnings or errors in the process.
- Hand the process off to the solution engineer for implementation. The solution engineer underpins the process with the necessary details (such as calls to REST APIs and so on) that enables the process to execute.

Result

Deployment is part of the software development cycle (design, deploy, execute). After preparing the software, some transformation, packaging, physical delivery, configuration and initialization takes place. All of these, some of which might be optional, are aspects of deployment.

Model a Process

TIBCO Business Studio™ - BPM Edition is used to model your business processes, providing a project-based view that gives you access to all the files related to the project. You can add details to and manage your processes at a project level.

To launch TIBCO Business Studio - BPM Edition, go to **Start > Programs > TIBCO > Studio for Designers**.

i Note: Typically, TIBCO Business Studio - BPM Edition is simply run from Windows. It can, however, be run from a Docker image, if desired. But because Docker containers are completely isolated from their Windows host, you must configure the "File Sharing" option in Docker before trying to run TIBCO Business Studio - BPM Edition from within the Docker image.

To configure File Sharing, from Docker, select **Settings > Resources > File Sharing**, then specify the local drives you want to be available to Docker containers.

Projects, Assets, and Project Organization

Projects help to facilitate the sharing and organization of resources. Before modeling your business process or defining your organization model, you must create a project to contain your assets.

A project can contain the following assets:

- Processes and forms - A process models the business process, and forms that are used to collect user input in a user task within a business process.
- Business Object Model - A business object model is a set of business terms and relationships specific to your corporate environment (for example, in a financial environment, broker, counterparty, and so on).
- Organization models - An organization model defines the organizational structure of your enterprise and the relationships between the different components (for example, organization units and positions) within your organization.
- Work List Facade - This project type is the single place where you can define display values for work list attributes that are used throughout an organization.
- Global Signal - These are signals that are thrown and caught across process instances, allowing processes to collaborate with each other.
- REST Service - Processes can be published as a REST (Representational State Transfer) service, which allows client applications to use the BPM REST API to invoke the published REST service.

Projects

This product supports the full project life cycle, bringing together all artifacts in a single place. The *project* is the container for these artifacts. As such, projects help to facilitate sharing and organization of resources.

For example, team members might have different responsibilities but need to use the same resources that are made available through the project.

You must create a project. Each project has a corresponding directory in the file system (specified when you create the project). Projects can also refer to other projects (see [Project References](#)).

Project Lifecycle

The user manages a project through its creation, deployment to a development server, and to the testing phase. The project is then locked for production and deployed on the production server for the 'end user' to use.

The project lifecycle consists of the following stages:

- Create or import projects in TIBCO Business Studio - BPM Edition
- Generate test deployment artifacts
- Deploy the generated artifacts
- Test on the development server
- Go back to TIBCO Business Studio and make any changes required to the project
- Repeat the deployment and test until successful completion
- Lock the projects for production
- Generate production deployment artifacts
- Deploy on the production server

Further Project Development

- Once a project has been locked for production, you cannot work on it until you unlock it by creating a new draft version. Right-click the project, then go to **Deployment > Create New Draft**. At this stage, the project's minor version is incremented. See [Project Versioning](#).

i Note: Any other dependent project that references the project requires that it is deployed before the other dependent project (because it could contain new content that the dependent project relies on for correct function). This is verified against the deployed project's major and minor version number during deployment.

Project Versioning

When you create a project, it is assigned a version number that is used to control the compatibility requirements between projects, and to indicate revisions to the project. The

version number is shown after the project name in Project Explorer (for example, Insurance v1.0).

The version number consists of a major number and a minor number. The format for a version number is:

major.minor

You lock a project for deployment to a production server. When more work is required in the project, you create a new draft and the minor version number is incremented.

i Note: Dependent projects depend on a minimum of the *major.minor* version of the project in the current workspace. This is the minimum version that must be deployed before or with the dependent project in order to successfully deploy the dependent project.

If you wish to make a backwards incompatible change (for example, remove business data from a class, or change the type of some business data), you must update the major version number of a project before you can successfully deploy it. This is in order that existing deployed applications that depend upon the items in the project are not broken by this backwards incompatible change. Right-click the project, then select **Properties > Lifecycle** and then click **Increment Major Version**.

After incrementing the major version and deploying this project, existing projects that depend on it might not use the new version until they are regenerated and deployed. Increment the major version only if you need to make changes to this project that are not backward-compatible.

Example of Versioning

You have a simple application with a data project and a process project.

- Create the projects. They are versioned **1.0**.
- Develop and test these projects until you are ready to put your app into production for real end users to use. You then lock them for production and deploy them to a production server.
- You decide to make some changes later to add some extra data for your users to enter. So, you create a new draft of your data project (which becomes **1.1**) and add the necessary attributes for the new data.
- Create a new draft of your process project and configure the new data on a user

form.

- Your process project now depends on the new version of the data project **1.1** and does not deploy to a development or production server unless the data project version **1.1** is deployed first or at the same time.

Creating a Project

You can create a new project using a template.

This illustrates creating a process project.

i Note: However, other project types do not have separate, selectable templates.

Procedure

1. Go to **File > New > BPM Process Project**.
2. Complete the New BPM Process Project dialog box as follows:

Field	Description
Project name	Enter a descriptive name for the project.
Location	Either accept the default location for the project (your workspace) or de-select the Use default location check box and click Browse to select a different location.
Status	Package life cycle status for informational purposes. How or whether you use life cycle statuses is up to you, but they are typically used as follows: <ul style="list-style-type: none"> • Under revision - for packages in development • Under test - for packages in User Acceptance Testing (UAT) • Released - for packages in production
Id	The identifier for the project. This defaults to: <code>com.example.<projectName></code>

3. Click **Finish** on this dialog box (or on any of the subsequent dialog boxes) to create a project with the settings you have made to that point. Click **Next** to modify the default project settings and create a process using a template.
4. On the Business Processes dialog box, enter the name of the folder that you want to designate for Business Processes.

Under Package Details, you can accept the pre-selected check box and either enter a file name or accept the default file name.

By default, when you create a project, a package and process is created as well. The default packages folder is called **ProcessPackages** and the default packages file is `<projectName>.xpdL`. Either accept the default names or rename the packages file and folder. Click **Finish** if you are done, or **Next** to specify more options.

5. The Package Information dialog box is displayed. Either accept the default properties of the package, or modify them as necessary and click **Finish** if you are done, or **Next** to specify more options.

- **Package Label**

Descriptive label for the package. Defaults to the same name as the file name of the package.

- **Package Name**

This defaults to the same name as the project.

- **Author**

Username of the user who created the package.



Note: The name of the author can be defined in the User Name: field on **Window > Preferences > User Profile**. If no user is defined there, it uses the default user system property.

- **Created**

Displays the date/time that the package was created.

- **Description**

Text description of the package.

- **Documentation Location**

URL or file name of any supporting documentation.

- **Language**

Provides context for user-visible language in processes. For example, annotations in a process might be in a language or character set unfamiliar to the user of the process. Use this field to specify the language used. Note that this field is informational; it does not change any system or language settings.

6. The Select Template dialog box allows you to create a process using a template. Select a template from those available and click **Finish** if you are done, or **Next** to specify more options.
7. The Set Special Folders dialog box displays the default special folders for the other asset types related to the project type you have created. Either accept the default names or enter your own names and click **Finish**.

The newly-created package, process, and project are displayed in the Project Explorer.

Deploying a Project

After you have developed an application, and resolved all issues highlighted in the Problems view, its constituent elements must be deployed to the BPM runtime so that the application can be run.

The projects for your business application must be deployed to the BPM runtime in the correct order of dependency, that is, if "Process Project A" uses data defined in "Data Project X", then "Data Project X" must be deployed first to ensure that "Process Project A" is successfully deployed.

In general, a project is deployed in the following order:

- Organization projects
- The Work list facade project
- Business data projects (in their own interdependency order)
- Global Signal projects

- Business Process projects (in their own interdependency order)

i Note: REST Service projects do not require deployment to the BPM runtime.

You can access a visual representation of your projects using the "View Dependencies" option from the project explorer context menu (see [Using the Dependency Viewer](#)).

Deploying Projects for Development Testing

- For process projects that use email or REST service tasks, you must create the shared resources named in each of the email and REST participants before deployment. This is done in the Shared Resources Manager in Administrator. See the "Shared Resources" topic in the *TIBCO® BPM Enterprise Administration*.
- Click **Launch Admin** in the deployment artifact generation wizard to open the New Deployment administration tools in a web browser. Alternatively, log in to the Work Manager and select **Administrator > Deployment Manager > New Deployment**.
- You can select, or drag-and-drop the .rasc file you generated, and deploy the artifacts (.rasc files) to a development server.

i Note: The projects need to be deployed in order of dependency. For example, you must deploy a business data project before the process projects that reference it and you must deploy projects with sub-processes before projects with the main process that invoke these and so on. If you do not deploy in the correct order, the deployment fails and you must redeploy the referenced projects in the correct order.

Deploying Projects for Production

- When your development and test cycle is complete, you can lock your projects and deploy them into production BPM Enterprise environments.

i Note: Draft deployment artifacts cannot be deployed to a production server.

- To lock your projects for production in TIBCO Business Studio - BPM Edition, right-

click the project, then select **Deployment > Lock for Production**. This ensures that production deployment artifacts required for deployment to a production server can be created.

- Generate the deployment artifacts for a production server. Right-click the project, then go to **Deployment > Generate Production Artifacts**.
- You can also generate and deploy your projects to the production environment using the automated CI/CD commands. For more information, see [Deploying a Project from the Command Line](#).

Deploying Projects from the Command Line

You can automatically deploy projects from the command line or as part of your CI/CD build pipeline using the Studio Automated CI/CD feature.

For detailed information about generating and deploying the runtime artifacts to BPM Enterprise Edition, see the `TIBCO_HOME/docker_cicd/readme.txt` file.

Generating deployment artifacts

After you have installed the Studio Automated Build support components and created the Docker image, you can use the Docker image to generate deployment artifacts (RASC files) from the BPM Studio projects.

The `generate-rascs` command also creates a `deploy.manifest` file in the target folder. This file details the generated `.rasc` deployment artifacts and the correct order to deploy them for the `deploy-rascs` command.

The `generate-rascs` command also creates a `deploy.info` file in the target folder. The `deploy.info` file is in JSON format and lists the set of shared-resource definitions that the generated artifacts require.

The **sharedResources** property in the `deploy.info` file lists the name and type of the shared resources that are required for the successful deployment.

You can navigate to the shared resources in the BPM Administrator client to verify if the shared resources are created with the name, type, and other properties as defined in the `deploy.info` file.

Deploying the generated artifacts

The `deploy-rascs` command deploys the `.rasc` runtime artifacts created in a folder by a previous `generate-rascs` execution.

Application Deployment Build Pipeline Example

When you install the Studio automated build support components, the `TIBCO_HOME/docker_cicd/sample_deploy_pipeline` folder contains template files to help you create a Jenkins pipeline job that can build and deploy BPM applications.

The sample shows how a build pipeline can check out BPM projects from a source repository such as a Subversion, generate deployment artifacts, and then deploy them to a TIBCO BPM Enterprise runtime environment. You can then create a simple, repeatable build pipeline for your BPM application.

For more information, see the `TIBCO_HOME/docker_cicd/sample_deploy_pipeline/readme.txt` file.

Project References

Projects can refer to other projects. You can invoke a process or use an organization model or BOM in another project.

The process of allowing projects to refer to other projects involves the following actions:

- Invoking a sub-process from another project (see [Sub-Processes](#)).
- Using a process interface from another project (see [Process Interface](#)).
- Creating a data field or formal parameter of the type external reference (see [Data Fields and Parameters](#)).

You can add project references explicitly or automatically.

- Explicitly: see [Creating References in the Project Properties](#).
- Automatically: see [Creating Project References in a Selection Dialog Box](#).

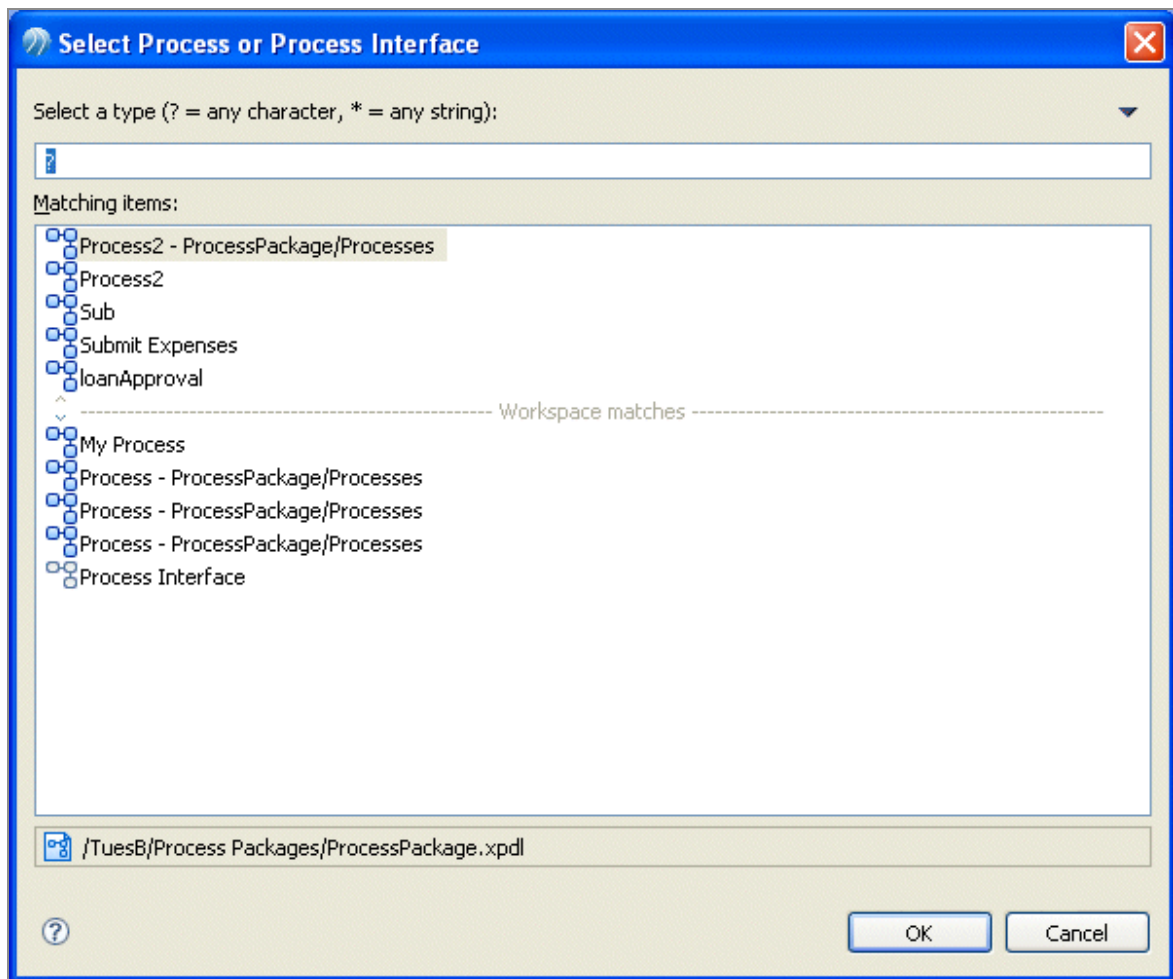
Note: When you reference one project from another, it is important that their major version numbers match.

Creating Project References in a Selection Dialog Box

Project references are automatically created when an object from outside the project is selected.

Procedure

1. For example, when defining a sub-process call, you select from a list of all the sub-processes in the workspace:



2. If the selected sub-process is not part of the project in which the sub-process call is

located, a message is displayed asking if you want to add the project (which contains the selected sub-process) as a reference project. Click **Yes** to create a reference to the project that contains the sub-process.

Creating References in the Project Properties

You can create a reference from one Project to another in the project properties. You can also remove project references in this way if you no longer reference assets in that project.

Procedure

1. Right-click the project, select **Properties**, and highlight **Project References**.
2. On the Properties dialog box, select any projects that you want to refer to from the selected project.

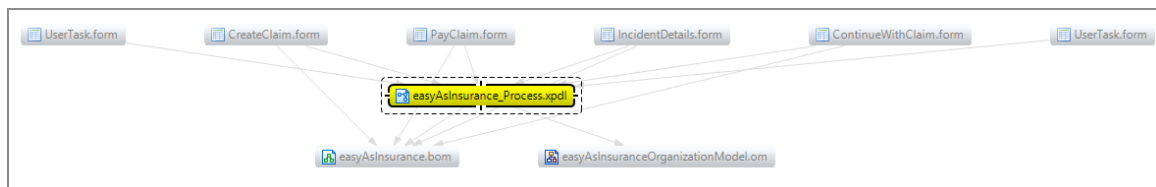
Using the Dependency Viewer

Use the Dependency Viewer to show an interactive, graphical representation of your project's file dependencies.

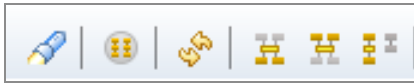
Procedure

1. Right-click a resource in Project Explorer (for example, a project, a BOM file, or a form file) and select **View Dependencies**.
2. The Dependency Viewer displays the dependencies.


For example, right-clicking the `easyAsInsurance_Process.xpdl` file shows references from the form files to the XPDL file and references from the XPDL file to the `easyAsInsurance.bom` (business object model) and `easyAsInsuranceOrganizationModel.om` (organization model).





Note: You can select different views in the Dependency Viewer to make these relationships clearer:





To focus on a resource, do one of the following:

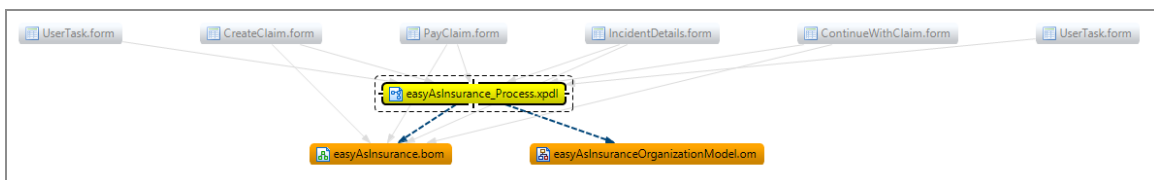
- Select the resource in the Project Explorer.
- Right-click the resource in Dependency Viewer and choose **Focus on** or **Focus on Resource name**.
- Click the  button.


To reset the layout and refresh the contents of the viewer, click .

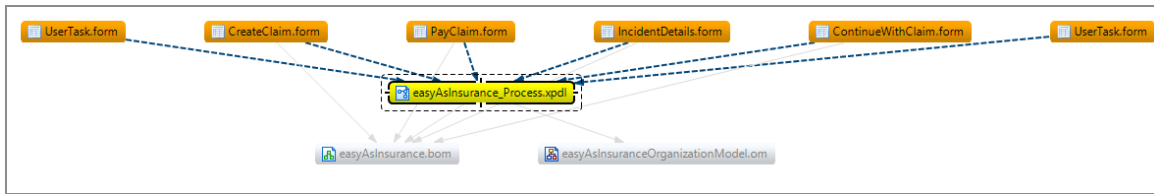
To show everything in the workspace, including resources which are not related to the resource you have focused on, click .


Note: To break the link between the Project Explorer and the Dependency Viewer, click . Doing this means that resources that you select in the Project Explorer are not focused in the Dependency Viewer.

To highlight resources that are referenced *from* the resource you are focusing on, click . In the following graph, the `easyAsInsurance_Process.xpd1` file references `easyAsInsurance.bom` (Business Object Model) and `easyAsInsuranceOrganizationModel.om` (Organization Model).



To highlight resources that have a reference to the resource on which you are focused, click . In the following graph, a number of form files reference the `easyAsInsurance_Process.xpd1` file.



To show all the resources which are part of the relationship graph of the focused resource (that is, all the resources which are directly or indirectly related to the focused node), click .

Finding Cyclic Dependencies

The Dependency Viewer highlights any connections that are part of a cycle.

A simple example of a cyclic dependency is where Project A refers to Project B, which refers back to Project A.

Procedure

1. Display the list of projects.
2. Selecting every resource as the focus using the Dependency Viewer, check for cyclic connections. You can identify these by this connection:



For a given resource, get the list of resources it depends on and make each of these the focus in Dependency Viewer, and check for cyclic dependencies.

Copy and Reuse an old Project

TIBCO Business Studio - BPM Edition allows you to copy and reuse an old project.

Procedure

1. Clean the entire project
2. Copy, rename and refactor the project
3. Rename everything with the new project name
4. Ensure that the old references are removed in the rasc generation process

5. Deploy the project as usual

i Note: The deployment fails with a 500 generic server exception, if the project is not cleaned before it is copied. This is because the generated artefacts of the first project get copied.

Process Packages

A *process package* is a mandatory container for processes and their infrastructure (such as *participants* and *data fields*).

The package and any processes stored in it are saved in XPDL format.

For example, in the insurance environment, separate packages could contain the processes used by the Claims Department, the Policy Origination/Maintenance Team and the IT Department. Processes can be shared between packages and projects so libraries of process components can be created and reused.

i Note: If the *Data Fields* folder is empty, it is hidden by default. This is because the preferred usage is to define Data Fields at the Process level.

Creating a Package

You can create a package as part of creating a project or independent of project creation.

See [Creating a Project Package and Process](#).

Procedure

1. Right-click the Process Package folder under which you want to create the package and select **New > Process Package**.
2. If you want to change the Packages Folder under which the package is created, click **Browse** or enter an existing folder name.
3. Enter the name of the **File** or accept the default file name. The file name must have **.xpdL** as the extension. Click **Next**.

4. The Package Information dialog box shows the name of the package and its default properties (for a full explanation, see [Creating a Project](#)).

Either accept the default properties of the package, or modify them as necessary. Then, click **Finish** to create a process using the default template, or click **Next** to create a process using a different template.

5. By default, a process is created with start and end events connected by sequence flow. The Select Template dialog box allows you to create a process using a different template. Select a template from those available and click **Finish**.

Result

The newly-created packages and any processes that you elected to create are displayed in the Project Explorer.

i Note: The number of open editors is restricted to 8 by default to avoid adverse performance issues. To edit this setting, select **Window > Preferences > General > Editors** and under **Close editors automatically**, edit the value for **Number of opened editors before closing**.

Copying a Package

You can copy a package into the same project, or into a different project in your workspace.

Procedure

1. In the Project Explorer, highlight the package file you want to copy.
2. Press Ctrl+C to copy the package.
3. Open the project into which you want to copy the package.
4. Select the folder into which you want to copy the package (for example, the **Process Packages** folder).
5. Press Ctrl+V to paste the package. If copying the package within the same project, you are prompted to enter a new name. If copying the package to a different project when a package of that name already exists, you are prompted to overwrite the file.






Note: After copying a package, errors similar to this are displayed in the Problems view:

✖ XPDL 2.0 : The following files have duplicate process IDs: /mond/Process Packages/ProcessPackage.xpdl

To correct this, right-click the problem, and select **Quick Fix**. Accept the suggested fix for this problem to resolve the duplicate IDs.

Processes

There are the following types of process: the business process, pageflow process, business service, case action process and service process.

- A *business process*  models actual and future processes in your organization that usually involve more than one person. Business processes are short or long-lived.
- A *pageflow process*  is a short-lived process (always executed in a single sitting) designed to implement a user interface dialog box. It is always executed by one person (the person that initiates the process instance).
- A *business service*  is a short-lived process (always executed in a single sitting) designed to implement a user interface dialog box. Unlike a pageflow it is executed outside of the context of a user task's work item. If there are further, longer-term tasks to perform after the initial business service tasks, then it is usual to asynchronously invoke a business process (using a sub-process task) to perform the remaining tasks.
- A *case action process*  is a special type of business service (always executed in a single sitting), associated with a case class, that defines an action a user can perform that is related to a case. You can either create a new case action or generate one directly from a case class.
- A *service process*  is a stateless, non-persistent process that is not audited. Service processes should be of short duration.

The Process Editor provides tools on a palette that use Business Process Modeling Notation (BPMN). By creating your process this way, you can fully prepare it for implementation by a specialist in your organization.

Pageflow Processes and Business Services

Pageflow processes and business services are short-lived processes designed to display user interface pages to desktop and mobile users.

A business service can start a business process. A pageflow process requires a trigger from a task within a business process. TIBCO Business Studio includes a **Publish** option for business services to set the **Target Device** to either **Desktop** or **Mobile**.

Pageflow Processes

A pageflow process is a special type of business process that can be used to provide an animated user interface for a single work item to the same user - that is, a sequence of forms rather than just a single form.

A pageflow process can also include other activities - such as service, scripts and conditional logic - which can be used to drive the interaction with the user.

i Note: Unlike a normal business process, a pageflow process is *stateless* and *non-transactional*. If the process is not completed in full, any data set earlier in the process is lost. If a pageflow process performs a *stateful* action - something that cannot be reversed, such as writing to a database or starting a process instance - it should be the final action performed by the pageflow process.

Pageflow Process Modeling

A *pageflow process* is a short-lived process designed to present user interface pages to the user in sequence. They are always executed by one person (the person that initiates the process instance).

All tasks that are available in a business process are available within a pageflow process with the exception of a *business process user task*. Pageflow processes have a special variant of a business process user task that does not have participants, and does not generate work items. These are referred to as *pageflow user tasks*.

A pageflow process is stored under the **Processes** branch of the Project Explorer alongside business processes.

A user task in a pageflow process differs from a user task in a business process in several key respects:

- Pageflow user tasks do not have participants assigned to them (this is because the user who initiates the process instance completes all the tasks in the pageflow process).
- Pageflow processes cannot contain lanes or pools.
- Pageflow user tasks do not create work items. The user interface pages are presented to the user without them needing to access their work queue.
- Pageflow user tasks do not restrict the type of technology used to create the user interface page that is displayed. For example, you could use TIBCO Business Studio Forms or a different technology. This allows the same process to be deployed to several runtime environments that utilize different user interface display technologies.

There are also special considerations to observe when using pageflows, specifically:

- Pageflow processes are not persistent (if the user cancels out of a pageflow process, data entered to that point is lost).
- Pageflow processes are not audited.
- Pageflow processes are not transactional (for example, there is no provision to roll back changes if a service task fails). If transactional control is required, chaining might be a better choice than a pageflow process (see [Creating a New Embedded Sub-Process](#)).
- Deployed pageflows are held in memory, and in some cases, having pageflows in different XPDL package files or in different Applications can result in errors. This is because one pageflow is available sooner than another dependent pageflow which might not have loaded yet.

Creating a Pageflow Process

All tasks that are available in a business process are available within a pageflow process with the exception of a *business process user task*. Pageflow processes have a special variant of a business process user task that does not have participants (because the participant is implied by the person who opens the work item), and does not generate work items. These are referred to as *pageflow user tasks*.

A pageflow process is stored under the **Processes** branch of the Project Explorer alongside business processes.

i Note: To convert a business process to a pageflow process, right-click the business process in Project Explorer, and then click **Convert To Pageflow Process**. Alternatively, a pageflow process can be converted to a business process in a similar way (**Convert To Business Process**).

Procedure

1. In the Project Explorer, under Process Packages, select the package you created, or Processes within an existing package, right-click and select **New > Pageflow Process**.
2. The **New Pageflow Process** wizard is displayed.

i Note: If you start this wizard from the **File > New** menu, the first dialog box is Project and Package, where you must specify a valid project and package. This dialog box is not displayed if you right-click at the package level to start the dialog box; however, you can click **Back** to display it if necessary.

3. Enter the **Label** of the process. If you want to use a template to create the process, select the template and click **Next**.

In addition to the process templates, you can select a process interface as the basis for your new pageflow process. This creates a process with the necessary events and parameters that are specified in the process interface.

4. In the Description dialog box, add optional text that describes the process, an optional URL that links to documentation about the process, and click **Next**.

i Note: The **Documentation URL** field is intended for design-time collaboration; it is not displayed in the runtime environment.

5. The **Extended** dialog box is displayed. This allows you to add optional supplemental information to the XPDL for the process.
6. Click **Finish**.
7. The process that you created is displayed in the Process Editor:



When you first start the Process Editor, the palette (on the right side of the diagram) might collapse; if so, expand it. You can expand this window to fill your screen by double-clicking the title bar. A pageflow process has a different default color scheme from a business process.

Note: Pageflows do not contain pools or lanes because they are short lived and do not span different parts of the organization.

Business Services

A pageflow process can be published as a business service.

A business service provides a user with direct access to a set of actions that accomplishes some sort of business function. A business service can (but does not have to) be used as a "process starter" mechanism to trigger an instance of a stateful business process.

The following sections provide two simple examples of how business services can be used:

[Example 1 - Starting a Business Process](#)

[Example 2 - Updating External Data](#)

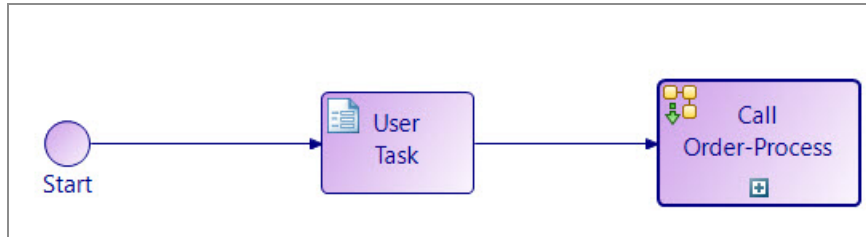
Generate a Business Service

A business service can call a business process. A business service provides input data to the business process with parameters. A sub-process in the business service sends the parameters to a Start Event in a business process.

Before you begin

- A business process with a Start Event, and any required tasks.
- The business process should have at least one input parameter.
- The Start Event must have a Trigger Type set to Start Request Event.

The generated business service finishes with a Task that calls the business process.



Procedure

1. Right-click the Start Event, and select **Generate > Business Service**.

A new business service opens, with a default **Label** similar to ProcessPackage-Process.

The business service consists of a Start Event, a User Task, and a final Task.

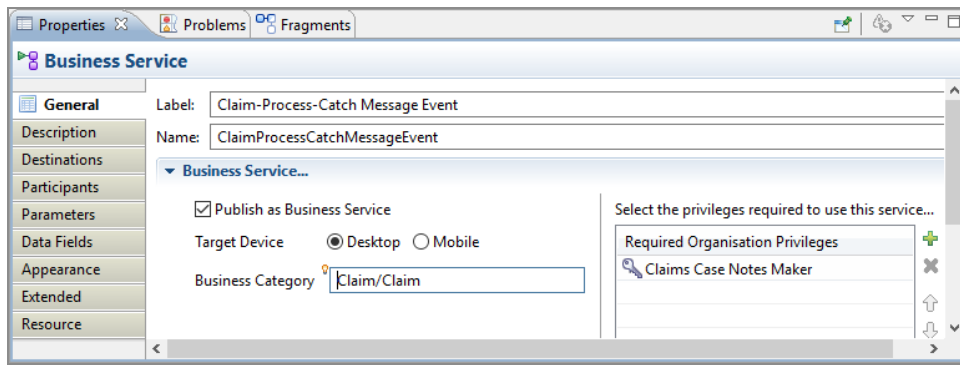
2. Rename the business service with a more descriptive name, so that users can easily identify it from the list of available business services.
3. Add any additional User Tasks, as required.
4. From the **General** tab, use **Select the privileges required to use this service** to specify all of the organization model privileges that a resource must have to be able to use this business service.

i Note: If there are no required privileges for the business service, then all resources can use the business service.

5. Choose the **Target Device**:

- Desktop
- Mobile

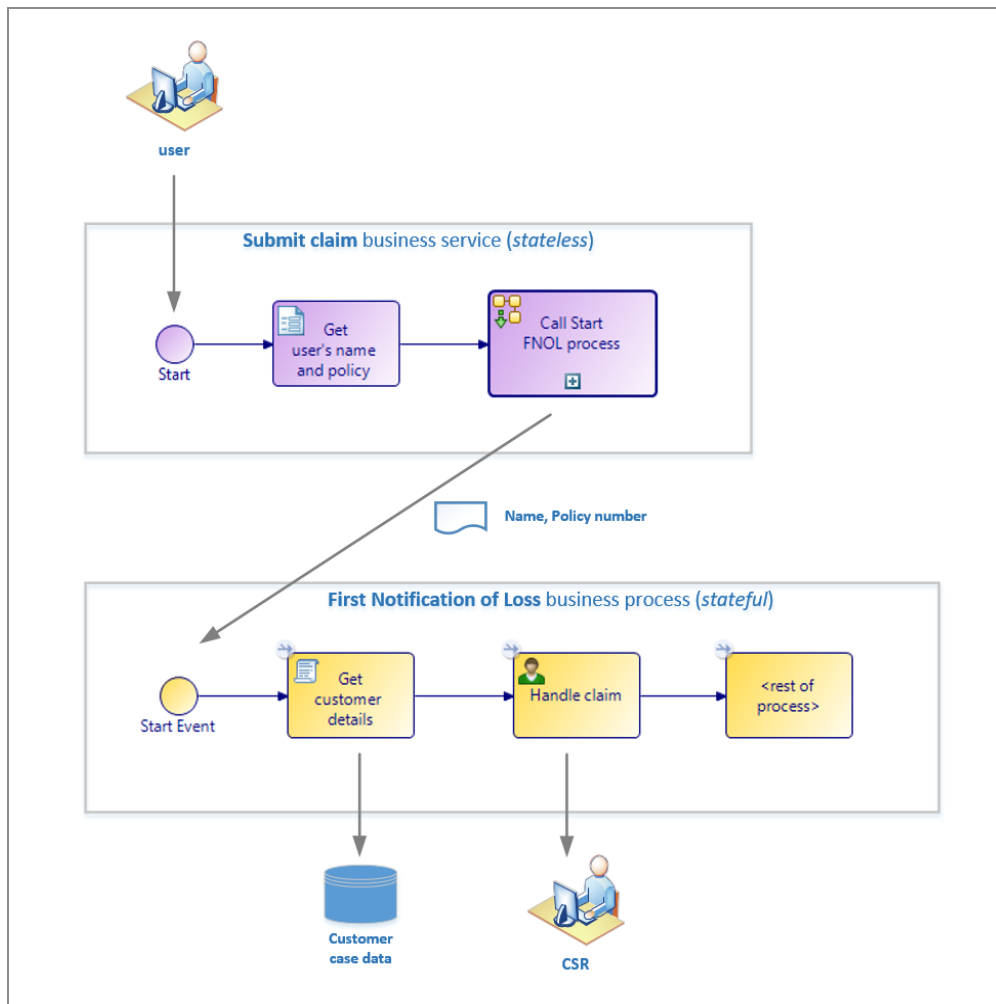
Publish as Business Service is already checked, and the default is **Desktop**.



6. Press Ctrl+S to save the changes.

Example 1 - Starting a Business Process

This example shows how a business service can be used to gather data for and start an instance of a business process.



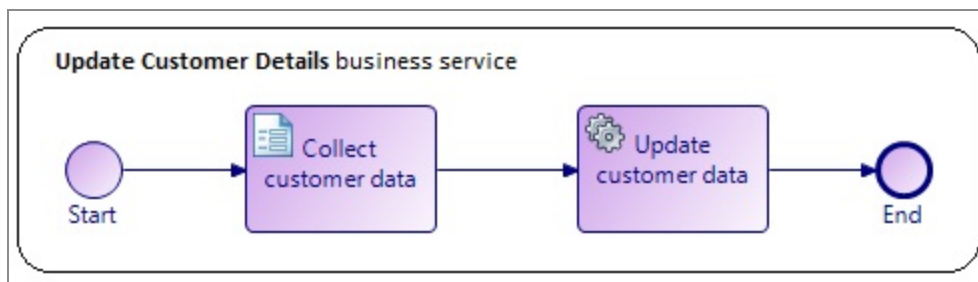
Procedure

1. Using Work Manager, a user selects and clicks the **Submit claim** business service. This starts the **Submit claim** business service.
2. The user immediately sees the **Get user's name and policy number** form, enters their name and policy number, then clicks **Submit**.
3. The **Start FNOL process** send task starts an instance of the **First Notification of Loss** business process, passing the name and policy number as inputs to the process. The **Submit Claim** business service terminates.
4. The **First Notification of Loss** business process uses the received name and policy number to obtain the customer's full details from the customer database, then generates a **Handle claim** work item for a Customer Service Representative, as the first step in processing the claim.

i Note: During the design phase of the **First Notification of Loss** business process, the business analyst or solution designer can automatically generate the **Submit Claim** business process - there is no requirement to design and implement it from scratch.

Example 2 - Updating External Data

This example shows how a business service can be used to provide a simple business function, independent of any business process.



Procedure

1. Using Work Manager, a user selects and clicks the **Update Customer Details** business service. The **Collect customer data** form is displayed.
2. The user completes the form and clicks **Submit**.
3. The **Update customer data** REST service task updates the customer database with the new data.
4. The business service terminates.

Creating a Business Process

You can create a Process and its containing package and Project in one operation or you can create a business process.

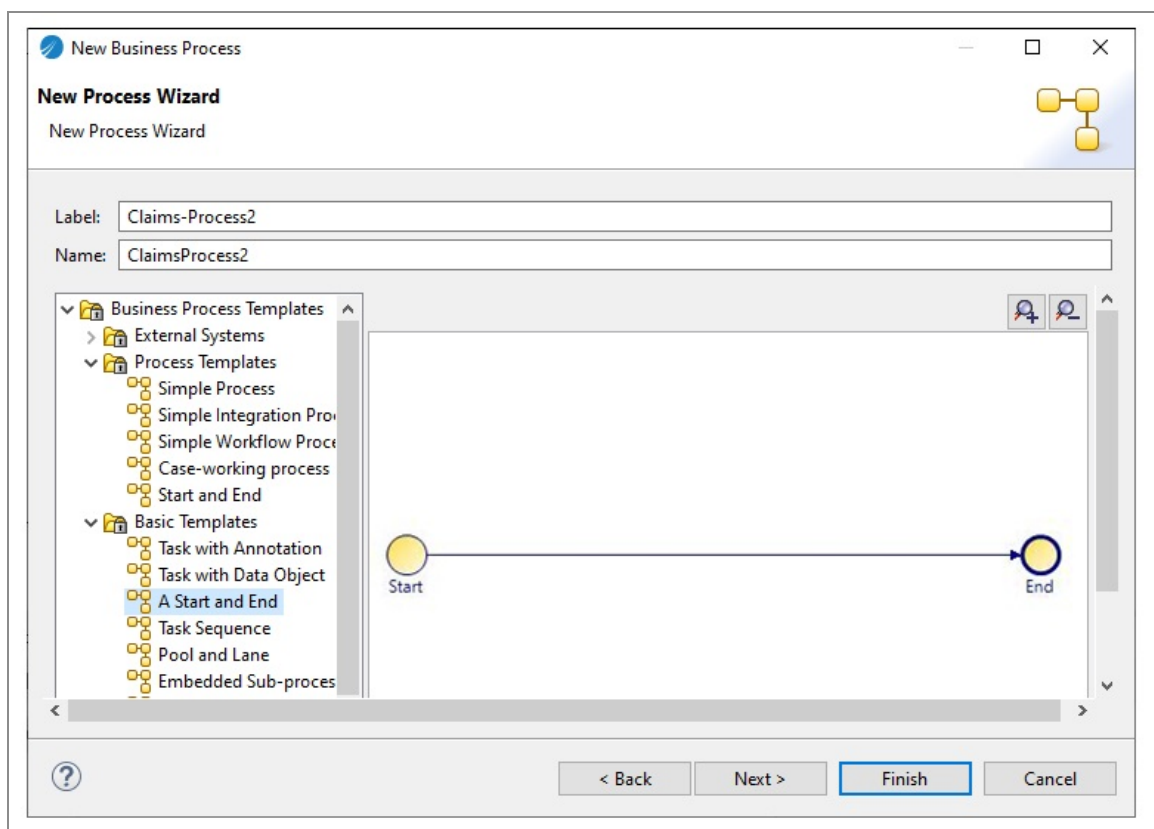
See [Creating a Project](#).

Procedure

1. Before creating a process, you must create a project and a package to contain your process.
2. In the Project Explorer, select the package you created, right-click and select **New > Business Process**.
3. The New Business Process wizard is displayed.

i Note: If you start this wizard from the **File > New** menu, the first dialog box is Project and Package, where you must specify a valid project and package. This dialog box is not displayed if you right-click at the package level to start the dialog box; however, you can click **Back** to display it if necessary.

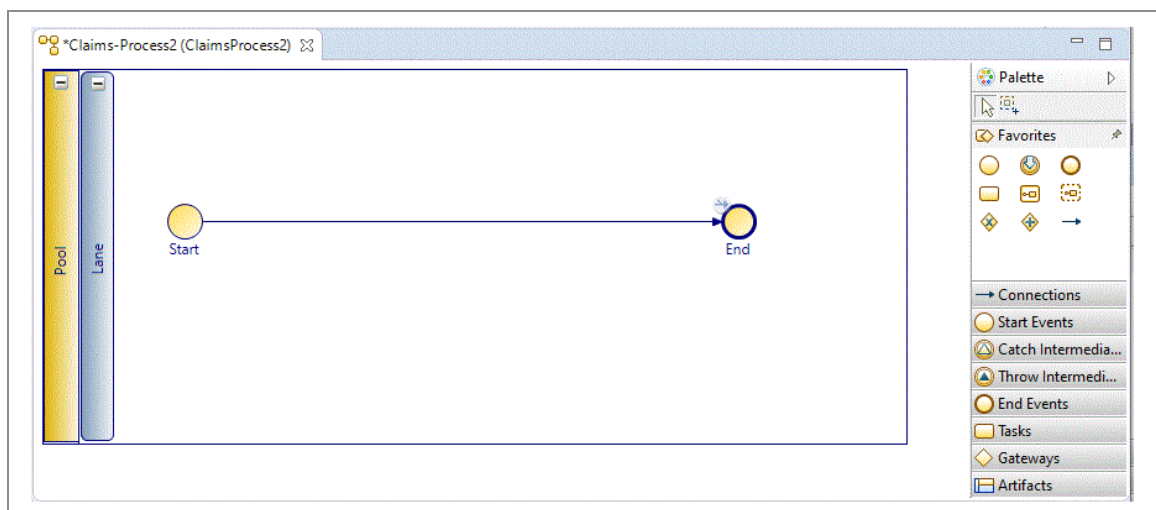
4. Enter the **Label** of the process. To use a template to create the process, select the template and click **Next**. This example shows one of the basic templates (a task sequence).



In addition to the process templates, you can select a process interface as the basis

for your new process. This creates a process with the necessary events, parameters that are specified in the process interface.

5. In the Description dialog box, add optional text that describes the process, an optional URL that links to documentation about the process, and click **Next**.
6. The Extended dialog box is displayed. This allows you to add optional supplemental information to the XPDL for the process.
7. Click **Finish**.
8. The process that you created is displayed in the Process Editor. When you first start the Process Editor, the palette (on the right side of the diagram) might collapse; if so, expand it:



You can expand this window to fill your screen by double-clicking the title bar).

The Process Editor provides a **Pool** and **Lane** that you can rename if you plan on using pools and lanes (see [Controlling Flow in a Process](#)). You add elements of your business process using the Palette.

Creating a Case Action Process

You can either create a new case action or generate one directly from a case class. Template case action processes are provided that allow you to view or update the contents of a case object, but you can modify these templates to provide whatever functionality you need for a particular case class.

i Note: You can generate a Case Action from a Case Class definition in the BOM editor. See [Generating Case Actions \(to View or Update Case Data\) From Case Classes](#) for more information.

Before you begin

Before creating a case action, you must create/import a business data project with a case class and add a case state attribute to that case class. You must also create/import a TIBCO Cloud BPM project and have a package to contain your case action.

Procedure

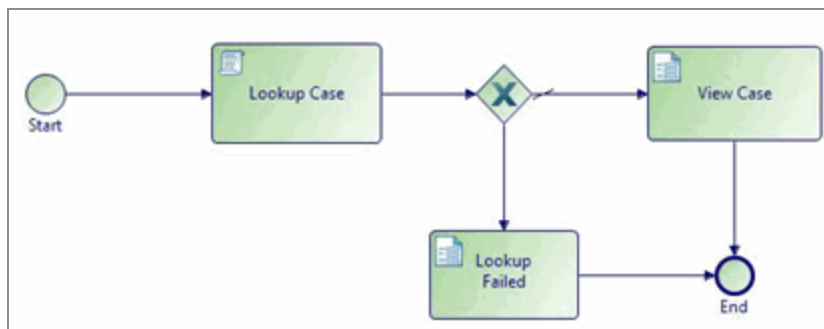
1. In the Project Explorer, select the package you created, right-click and select **New > Case Action**.

You can also create a case action from context menus on the case class in the BOM. See [Generating Case Actions \(to View or Update Case Data\) From Case Classes](#).

2. The Case Action wizard is displayed. Enter a **Label** and **Name** for the Case Action.
3. (optional) On the **Case Class** selector, **Select a Case Class type from a Business Object Model** to select the Case Class that this action applies to.

If you do not set this here, you can set it from the Properties for the Case Action when you have created the case action using the wizard.

4. Select a template for the case action from **View Case Action Process** or **Update Case Action Process**.
5. Click **Finish**.
6. The case action that you created is displayed in the Process Editor. For example, **View Case Action process**:



Generating Case Actions from Case Classes

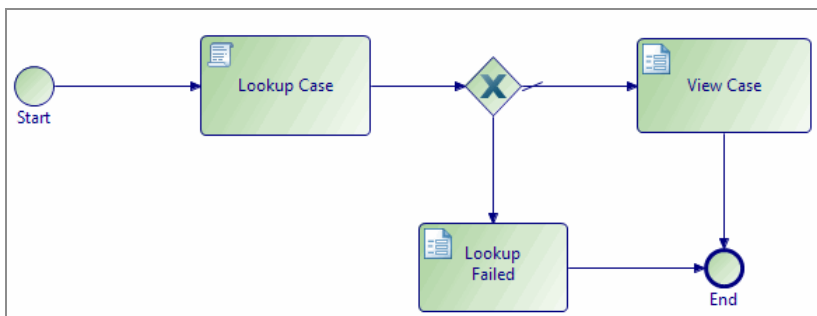
You can generate a Case Action from a Case Class definition in the BOM editor.

Procedure

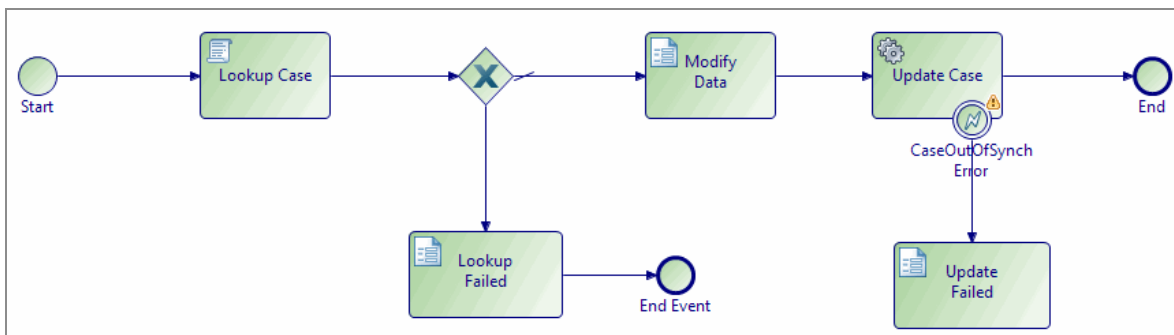
1. Right-click on a Case Class in the BOM and select **Generate Case Action**.
2. Select one of the following actions:
 - **To View Case Data:** Fetches the data for the case reference provided by Work Manager and shows that data to the user in a form.
 - **To Update Case Data:** Fetches the data for the case reference provided by Work Manager and shows that data to the user in a form. The user can change that data and the case object is subsequently updated.
3. From the **Generate Case Action** dialog box, open the project and Process Packages where you want to place the xpd. You can also select an existing xpd. Click **Finish**.

The case action is generated with the appropriate template (**View Case Data** or **Update Case Data**) and you can edit it to meet your needs.

View Case Data:



Update Case Data:



Service Process Modeling

A *service process* is a stateless, high performance process that is not audited or persistent. Service processes run in-memory and so they should be of short duration.

Service processes should not change anything outside the scope of the process itself, like updating mission critical data. This is because service processes are not transactional and therefore, there is no provision to roll back changes if a task fails.

When modeling a service process, you must specify a deployment target. You can deploy service processes to either:

- the process engine
- the pageflow engine

The deployment target you choose depends on whether you are invoking your service processes from business or pageflow processes.

You can invoke a service process from the following process types:

- As a sub-process (either static or dynamic) from all process types. You can invoke a service process from a sub-process in the following process types:
 - business process; when the service process has the process deployment target selected.
 - pageflow process; when the service process has pageflow deployment target selected.
 - service process; when, at the minimum, the invoked service has the same deployment targets selected.
 - business service; when the service process has the process deployment target selected.
 - case action process; when the service process has either the process or pageflow deployment target selected.
- As a REST service. Using the TIBCO Business Studio - BPM Edition REST API.

When you are invoking a service process from a sub-process, your service process must have a start event of type Start Request Event. When you are invoking a service process as a REST process, your service process must have a start event of type Start Request Event. If you are invoking a service process from both a sub-process and as a REST process, your service process must have both a start event of type Start Request Event and a start event of type Start Request Event. See [Example - Calling a Service Process from a Sub-Process](#).

If your service process is only going to be invoked as a REST service, it is more efficient to create your service process with a start type of Start Request Event and select the pageflow engine as the deployment target. This is because a service process deployed into pageflow run-time is already invocable using a REST service.

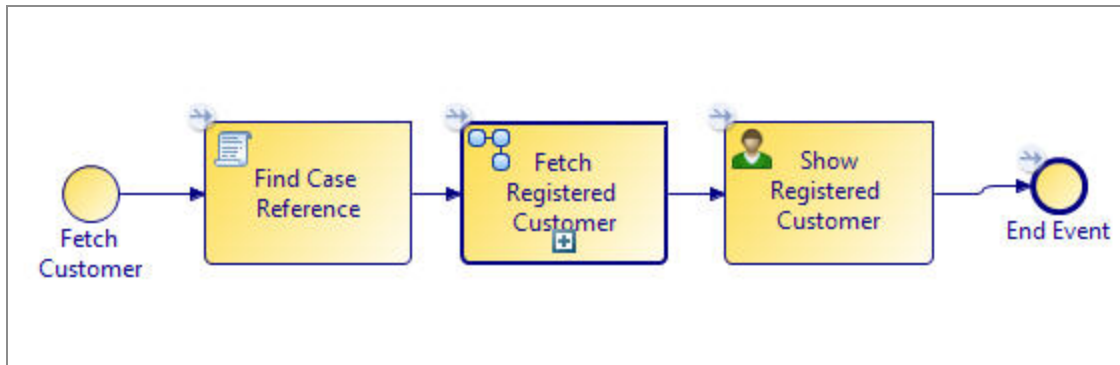
As service processes should be of short duration, the following activity types are not supported in service processes.

Activity Type	Description
Incoming request event	An incoming request event might be waiting to be triggered for an arbitrary period of time.
Signal event handlers	A signal event handler event might be waiting to be triggered for an arbitrary period of time.
User tasks	No direct user interaction with a service process is supported.
Attached Sub-process calls to non-service process types.	Service processes can only synchronously invoke other service processes. (They must have the same deployment targets set). However, they can invoke conventional stateful business processes in asynchronous-detached mode.
Non-boundary timers	Timers that pause rather than monitor the execution of a thread of a service process are not supported.

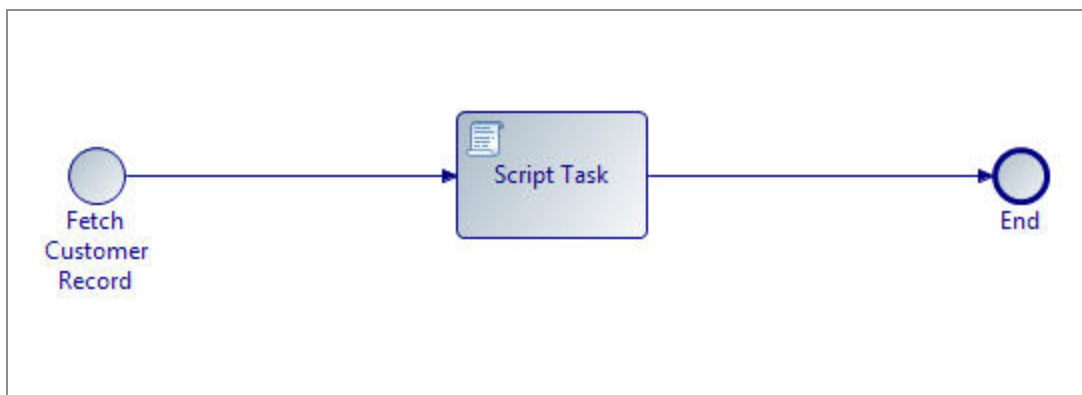
A service process is stored under the **Processes** branch of the Project Explorer alongside business and pageflow processes.

Example - Calling a Service Process from a Sub-Process

This example shows how to model a business process that invokes a service process from a sub-process:



In this example, a Find Case Reference script task finds the customer's case reference. The Fetch Registered Customer sub-process calls the following service process:



The Fetch Registered Customer sub-process invokes a service process. To enable your service process to be called from a sub-process, the service process must have a start type of Start Request Event. The script task in the service process retrieves the customer details for the supplied customer case reference. The Show Registered Customer user task displays the details of the registered customer.

Creating a Service Process

You can create a service process and its containing package and project in one operation or create a service process only and add it to existing package/project.

See [Creating a Project](#).

As service processes should be of short duration and have only one function, some activity types are not supported in service processes.

Activity Type	Description
Incoming request event	An incoming request event might be waiting to be triggered for an arbitrary period of time.
Signal event handlers	A signal event handler event might be waiting to be triggered for an arbitrary period of time.
User tasks	No direct user interaction with a service process is supported.
Attached Sub-process calls to non-service process types	Service processes can only synchronously invoke other service processes. However, they can asynchronously invoke conventional stateful business processes.
Non-boundary timers	Timers that pause rather than monitor the execution of a thread of a service process are not supported.

When modeling a service process, you must specify a deployment target. You can deploy service processes to either:

- the process engine
- the pageflow engine

The deployment target you choose depends on whether you are invoking your service processes from business processes or pageflow processes. When you create a service process, the service process is set to deploy to the process engine by default.

Procedure

1. Before creating a service process, you must create a project and a package to contain your process.
2. In the Project Explorer, under Process Packages, select the package you created, or Processes within an existing package, right-click and select **New > Service Process**. The **New Service Process** wizard is displayed.
3. Enter the **Label** of the process. You can change the label on the **General** tab in the Properties view if necessary. If you want to use a predefined template to create the service process, select the template and select **Next**.

In addition to the service process templates, you can select a service process

interface as the basis for your new service process. This creates a service process with the deployment target, events and parameters that are specified in the service process interface. Specifying a service process interface allows the dynamic selection of service processes at runtime. See [Service Process Interfaces](#).

4. In the **Description** dialog box, add optional text that describes the process, an optional URL that links to documentation about the process, and click **Next**.

Note: The **Documentation URL** field is intended for design-time collaboration; it is not displayed in the runtime environment.

5. The **Extended** dialog box is displayed. This allows you to add optional supplemental information to the XPDL for the process.
6. Select **Finish**.

The service process that you created is displayed in the Process Editor.

When you first start the Process Editor, the palette (on the right side of the diagram) might collapse; if so, expand it. You can expand this window to fill your screen by double-clicking the title bar. A service process has a different default color scheme from other processes.

Note: Service processes do not contain pools or lanes.

7. Once you have created your service process, you must configure the deployment target that you want the service process to be deployed to.

When you create a service process, the deployment target is automatically set to **Deploy to Process Run-time** so, if you want your service process to deploy to the pageflow engine, you must select **Deploy to Pageflow Run-time** in the **General** tab of the Properties view.

Refactoring Service Processes and Service Process Interfaces

You can convert an existing business, pageflow or sub-process into a service process or, alternatively, a service process into a business, pageflow or sub-process. You can also refactor a service process interface into a process interface, and vice versa.

There are some activities that are not supported in a service process or service process interface. If the business process or pageflow process you want to convert to a service process contains any of the unsupported activities, validation errors occur. To avoid

validation errors, you must remove any of the unsupported activities before conversion.

The following activity types cannot be used:

Activity Type	Description
Incoming request event	An incoming request event might be waiting to be triggered for an arbitrary period of time.
Signal event handlers	A signal event handler event might be waiting to be triggered for an arbitrary period of time.
User Tasks	No direct user interaction with a service process is supported.
Attached sub-process calls to non-service process types	Service processes can only synchronously invoke other service processes. However, they can asynchronously invoke conventional stateful business processes.
Non-boundary timers	Timers that pause rather than monitor the execution of a thread of the service process are not supported.

Right-click the business, pageflow or sub-process you want to convert and select **Convert to service process**. In the same way, you can convert a service process to a business process or pageflow process.

You can also refactor a service process interface into a process interface by right-clicking a service process interface and selecting **Convert to Process Interface**. Similarly, you can convert a process interface into a service process interface by selecting a process interface and selecting **Convert to Service Process Interface**. When you convert a process interface into a service process interface, the deployment target is automatically set to **Deploy to Process Run-time** so, if you want your converted service process interface to implement service processes deployed to the pageflow engine, you must go back and configure this in the service process interface after conversion.

Activities

An **activity** represents work that a company or organization performs using business processes.

An activity can be atomic (it is not broken down into a finer level of detail) or non-atomic. Atomic activities are represented in the Process Editor by tasks. For more information about how BPMN defines activities and tasks, see the [BPMN website](#).

When creating the process, each time a different person, group, role, or system needs to call an API outside the BPM system, an activity should be added to the process.

Activities might be directly or indirectly triggered by events such as the receipt of an email, phone call or workflow item, and might involve making a judgment on the presented facts and performing an action (such as entering data to a computer system, phoning someone in the same or a different organization, and so on).

A task in a process diagram represents an atomic activity (one that cannot be further broken down). A task of an unspecified type looks like this in the Process Editor:



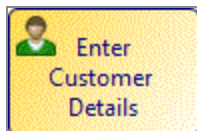
If the activities need to be broken down into finer steps, they must be represented as [Sub-Processes](#). The Activity Type is set in the **Properties** view.

Types of Tasks

Tasks can be of a number of types, each of which is represented by a different icon. Tasks can be user tasks, service tasks, script tasks, send tasks, or receive tasks.

User Tasks

User tasks are those that require human interaction with a software application.



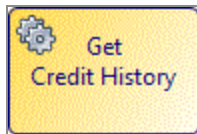
User tasks can be further configured for inbound and outbound parameters. Forms can be generated from the task's input and output parameters, representing the information you want to present to and capture from the user.

You can also generate a pageflow process from a task.

By using TIBCO Business Studio Forms, you can design, view, and test the forms you need to collect user input in a business process. You can create sophisticated forms without programming, and associate them with user tasks in order to provide richer user experiences for business process participants.

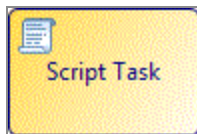
Service Tasks

Service tasks can ideally be completed without human interaction (for example, sending an automatic email notification).



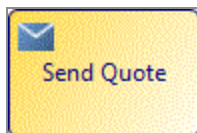
Script Tasks

Script tasks contain a set of instructions written in a scripting language (usually added to the step by the solution engineer) that are executed in the runtime environment when the process is deployed and executed.



Send Tasks

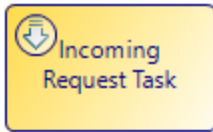
Send tasks are used to call a REST service without waiting for a response before following the send-task's outgoing flow.



They can be paired with a receive task or event to form a request response operation.

Incoming Request Tasks

Incoming Request tasks are used to wait for an invocation -- via the process REST API -- from a system or person outside of the process.







This type of task can be used to start a process as long as it has no incoming sequence flow and there are no start events in the process.

On the **Interface** tab, you can add parameters to an incoming request task. You can also add correlation data. However, you cannot add data fields because data fields are used internally in a process, and parameters are in this case, inputs from an external process (for more information, see [Data Fields and Parameters](#)).

Activity Markers

You can select BPMN Activity Markers on the Properties of the activity. The currently selected Activity Marker is indicated by a symbol on the activity.

- **Multiple Instance Loop with Parallel Ordering**  Indicates a task or sub-process activity that is replicated a fixed number of times based on the evaluation of an expression. The ordering is parallel.
- **Multiple Instance Loop with Sequential Ordering**  Indicates a task or sub-process activity that is replicated a fixed number of times based on the evaluation of an expression. The ordering is sequential.
- **Standard Loop**  Indicates a task or sub-process activity that might have more than one instance, depending on the conditions of the loop. A standard loop consists of a Boolean expression that is evaluated before or after each cycle of the loop. If the expression evaluates to True, the loop continues.

The conditions of the loop are set on the **Loops** tab (for more information, see [Loops](#)).
- **Ad-hoc**  Indicates an embedded sub-process that contains activities that have no predefined sequence. This also means that the number of times the activities are repeated is completely determined by the performers of the activities and cannot be defined beforehand.

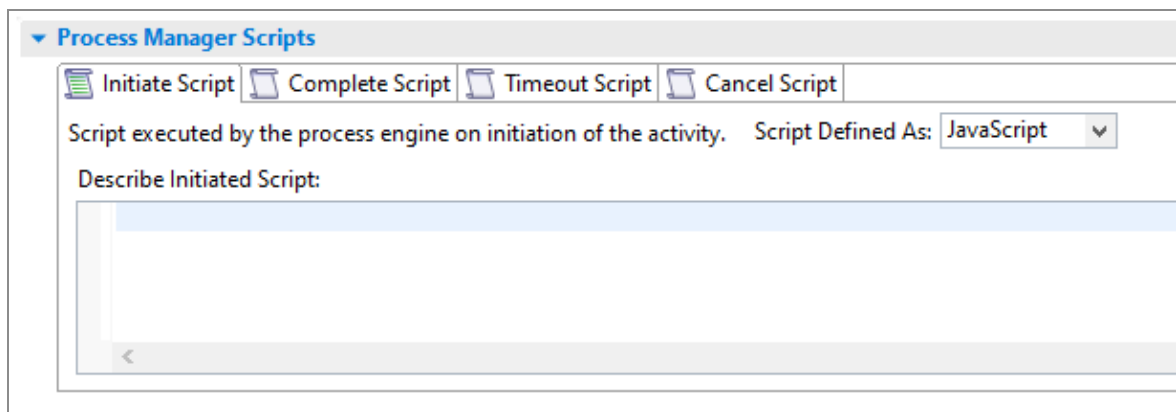
Creating Scripts

All types of task can have scripts that are performed at certain stages of the lifecycle of the task. User tasks can have scripts that are run for example, when the work item is opened or closed.

The **Scripts** tab allows you to add text that describes these scripts.

Procedure

1. Click the task to which you want to add a script.
2. In the Properties view for the task, click the **Scripts** tab.
3. The Scripts available are divided into two sets, **Process Manager Scripts** and **Work Manager Scripts**. Open the set you require (if one set is not available to you for this task, it is disabled)

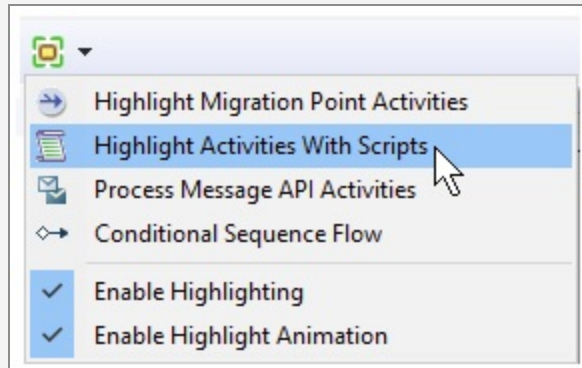


4. Click the tab for the desired type of script. Any tabs that have scripts defined have a script icon before the script name. Tabs with no script defined have an empty script icon. So, in the example above, a script is defined for the **Initiate Script** and **Cancel Script** tab.
5. Select **Free Text** from the **Script Defined As** list. Describe your script in the area provided.

Note: You can highlight tasks in a process that contain scripts. Click in the Process Editor for the process. On the toolbar, you can see a button:



Click this to see the dropdown, and select **Highlight Activities with Scripts**.



This highlights all activities in a process that contain scripts, and all other activities are disabled.

User Tasks and Pageflow Processes

In addition to creating a standalone pageflow process, you can do other actions from a user task to select or create a pageflow process.

- Select an existing pageflow process.
- Create a pageflow process that is referenced from the selected user task.

In the runtime environment, the pageflow process is run when the work item associated with the user task is opened.

When you right-click a user task, the **Pageflow** menu provides these options:

- **Open:** Select this option to open the associated pageflow process in the Process Editor (if a pageflow process has already been selected for the task).
- **Synchronize Parameters:** Select this option if you have made changes to the parameters in the user task, and want the changes reflected in the corresponding parameters in the associated pageflow process (see [Synchronizing Parameters With a Pageflow Process](#)).

- **Generate:** Select this option to generate a new pageflow process that is associated with this user task. The parameters associated with the user task are replicated for the pageflow process.
- **Use Existing:** Opens the Select Form dialog box from which you can select an existing pageflow process.
- **Delete:** Deletes the reference from the user task to the pageflow process. The pageflow process itself is not affected.

Selecting a Pageflow Process

If you have already created a pageflow process, you can select it from a user task.

Procedure

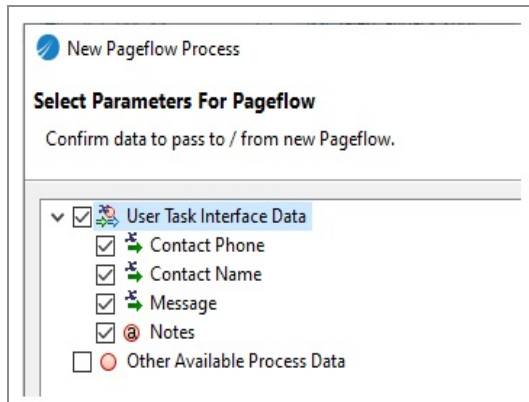
1. Select the user task.
2. Do one of the following:
 - On the **General** tab for the user task, select the Pageflow option and click the picker to display the Select Pageflow Process dialog box.
 - Right-click the user task and select **Pageflow > Use Existing** to display the Select Pageflow Process dialog box. Expand the projects to select a pageflow process and click **OK**.
3. The **Pageflow** field is automatically completed with a URL that points to the pageflow process (relative to the **Forms** special folder). You can open the pageflow process by clicking **Open Pageflow Process**.

Creating a Pageflow Process from a User Task

You can create a pageflow process from a user task.

Procedure

1. Select the user task to which you want to associate a pageflow.
2. Right-click and select **Pageflow > Generate**. The New Pageflow Process dialog box is displayed.



3. Confirm the data that you want to associate with the pageflow process and click **Next**:

- **User Task Interface Data:** Select from the process data specified on the **Interface** tab of the user task. If no explicit process data is selected on the **Interface** tab, all process data is available.
- **Other Available Process Data:** Select from the process data that is not associated with the user task.



Tip: The parameters that you associate with the pageflow process are available to user task forms after the pageflow process is created.

4. Enter the **Label** of the process. If you want to use a template to create the process, select the template and click **Finish** to create the pageflow process or **Next** to specify additional options.

Synchronizing Parameters with a Pageflow Process

When a pageflow process is first generated from a user task, a dialog box is displayed that allows you to create the pageflow process with its own set of parameters (either replicating the existing data fields and parameters of the user task, or a subset thereof).

User tasks in the pageflow process can display forms that utilize these parameters.

If you add or remove new parameters to the user task in the parent process (after the pageflow process is generated), you can update the list of user task parameters that are known to the pageflow process as follows:

Procedure

1. Right click the user task, and select **Pageflow > Synchronize Parameters**. The Synchronize Pageflow Parameters With User Task dialog box is displayed:
2. Confirm the changes and click **Finish**. For example, if parameters have been added to the pageflow process but not the user task, synchronizing deletes them from the pageflow process. If parameters have been added to or removed from the user task, synchronizing adds or removes the corresponding parameters in the pageflow.

Result

After synchronizing, the parameters in the Project Explorer listed for the pageflow process should be identical to the parameters listed for the parent process.

Mandatory Parameters and Pageflow Processes

When a pageflow process has a mandatory parameter, that parameter must have a value before the pageflow can be considered complete.

This is different to sub-process parameters, where the mandatory flag controls whether mapping to the parameter is required or optional.

This means that a pageflow process and associated user task are expected to have the same data available. As a result, mandatory pageflow parameters are mandatory in all user task forms within the pageflow. However, if this is not the desired behavior, explicitly associate the parameter with the user task in the pageflow and de-select the mandatory flag.

Using Forms with User Tasks

This section explains how to work with forms associated with user tasks and page flows within the TIBCO Cloud BPM environment. It provides information that is specific to using forms within a TIBCO Cloud BPM environment.

This information is for users responsible for designing and implementing business processes that include user interfaces for presenting and capturing information from users. The forms and business processes you design can be deployed to a TIBCO Cloud BPM node and can be accessed via the TIBCO Cloud BPM Client or a custom client application.

Creating a New Form

When you create a BPM project, a default form is created for each user task defined in the business process. You can use the default form or if required edit the default form.

You can also use any of the methods listed below to create a new form for a user task:

- [Creating a New Form for an Existing User Task](#)
- [Creating a New Form Manually from the Project Explorer](#)
- [Creating a Free-standing Form](#)
- If required, [Switching Back to the Default Form](#)

Creating a New Form for an Existing User Task

For forms that are used by a business process, editing the default form for an existing user task is the most efficient way to create a form.

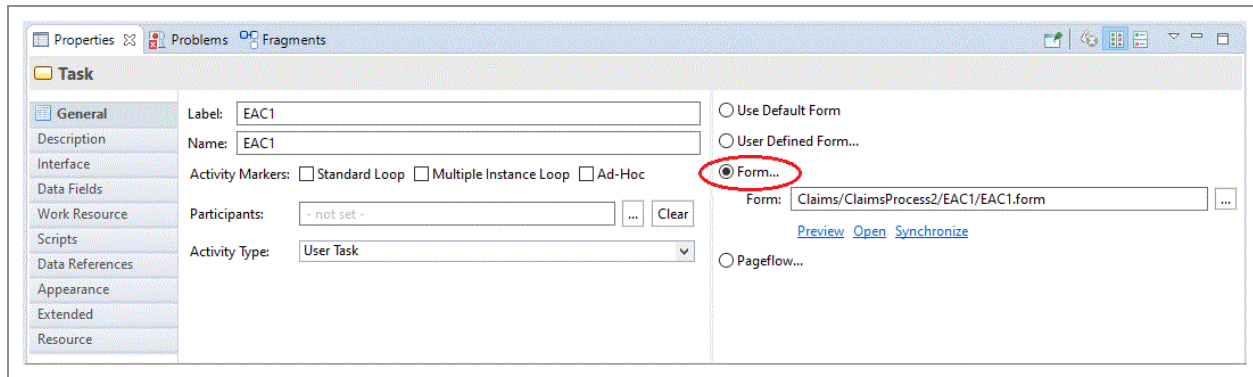
This is efficient for the following reasons:

- From the beginning, the default form exists in the context of the user task with which it is associated in the business process. This means that the **Form** field for the user task (on the **General** tab of the user task's Properties view) is automatically filled in with the URL of the new form.
- If there are existing parameters associated with the user task, the new form includes user interface components, bound to the corresponding form parameters.

You can create a new form for any user task in one of the following ways:

- Go to the context menu of a user task in a business process and click **Form > Open**.
- On the **General** tab of a user task's Properties view, select the **Form...** radio button.

In both of these cases, the existing default form is set on the user task as the custom form. You are prompted with a warning message informing you that "the customized form is no longer automatically kept in sync with the activity interface."



At this point, you are using the automatically created default form as a starting point for further modifications. If there are any changes to the user task interface, those are to be synchronized with the form. The form displays error-level problem markers when the parameters defined for the form are out of sync with the activity interface from which it was created.

Form Generator Preferences

There are two **Form Generator Preferences** settings that can be accessed from **Window > Preference > Form Designer > Generator**.

- **Generate text area for a string exceeding this length:** Used to specify the text length threshold value above which the form generator generates a text area rather than a text input. The default value is 100.
- **Generate text area for an XSD string with no length restriction:** Used to generate a text area for a BOM primitive type or property that was originally defined in an XML schema as base type `xsd:string` with no length restriction.
 - Checked: A text area is generated.
 - Unchecked: A text input is generated.

i Note: You can set the **Form Generator Preference** both globally at the workspace level and override it locally at the project level. Local preference overrides must be applied to the project in which the form is generated, not the project containing the BOM.

Creating a New Form Manually from the Project Explorer

A form might be created manually from the Project Explorer within any **Forms** special folder.

Procedure

1. Go to the context menu of the **Forms** special folder, or any child folder under the Forms special folder in the Project Explorer and click **New > Form**. This triggers the opening of the New Form dialog box.
2. Specify the **Form type** on the New Form dialog box. The type of form that is selected here determines the components that are initially part of the form model. The form types are as follows:
 - **Process task**: This creates a form that is the same as one created from a User Task in a process definition. It contains a root pane, a toolbar with **Cancel**, **Close**, and **Submit** buttons, and a messages pane for displaying error messages.
 - **Pageflow task**: This creates a form that is the same as one created from a User Task in a Pageflow Process. The only difference to a Process task form is that the toolbar contains only **Cancel** and **Submit** buttons. The **Close** operation is not supported in pageflows since there is no way to re-open a step in a pageflow once it has been closed.
 - **Embeddable**: This creates a form that is suitable for embedding within another form. This only contains a single root pane. This is because the parent form would typically contain the toolbar and messages pane, so these components are not needed in an embeddable form. For more information about Embeddable Forms, see the *TIBCO Business Studio™ - BPM Edition Forms User's Guide*.

Creating a Free-standing Form

Although it is convenient to generate new forms from existing user tasks, there are situations where a forms designer prefers to create a free-standing form before the corresponding user task is available.

There is no impediment to doing this. A form can be created by either of the methods listed below, and linked to a user task later.

- Go to the context menu of the **Forms** special folder, or any folder under the Forms

special folder in the Project Explorer and click **New > Form**.

- On the **File** menu, click **New > Other > TIBCO Forms > Form**.

Procedure

1. Go to the context menu of the user task and click **Form > Use Workspace**.

The Select Form dialog box opens.

2. Select an existing form from the Project Explorer folder structure and click **OK**.

Note that you must choose a form within the same project as the user task.

3. Synchronize the user task parameters with the form by going to the context menu of the user task and clicking **Form > Synchronize...** This establishes a permanent link from the form to the user task. This also enables the form synchronization validator to check that the form remains in sync with the user task's data interface and for the form synchronization wizard to modify the form as necessary to preserve consistency.



Note: The bidirectional linkage between the user task and form means that any given form might only be used by a single user task. The pairing of a user task to a form can be subject to automatic synchronization validation. If you share a form between multiple user tasks, you need to manually synchronize the form with the different user tasks. By doing this, the synchronization validation link to the previously associated user task is broken. Hence the second approach is not recommended. A better way to share forms is by embedding the reusable parts.



Warning: Forms must be created only under a Forms special folder, or within folders under a Forms special folder. If you choose a location other than the default location, be sure that the folder you select meets this requirement.

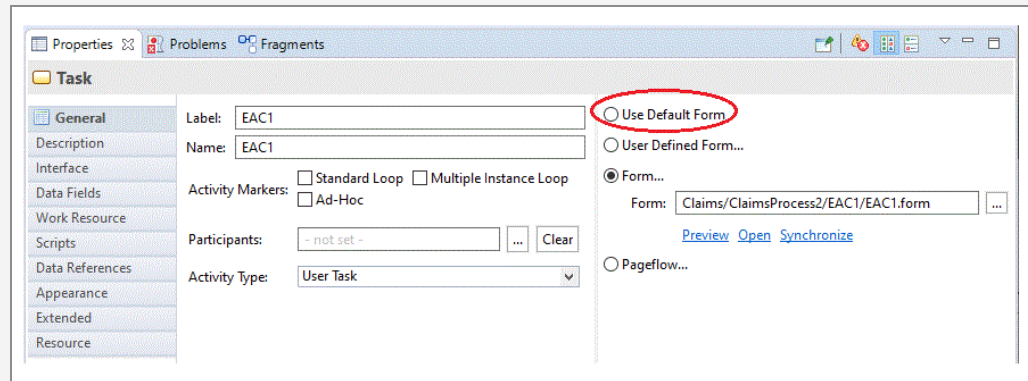
Switching Back to the Default Form

After creating an auto-generated form using the **Form > Open** context menu action or the user task General Properties tab **Form...** radio button and manually customizing it, there can be a situation where you would like to start over again with a default form.

Procedure

1. Go to the context menu of the user task and click **Form > Use Default**. This option is enabled only if the **Form...** radio button is selected on the user task General Properties tab.

Note: You can also select the **Use Default Form** radio button in the user task General properties tab as shown below.



2. The Confirm Delete dialog box appears.
 - Choose the **Delete** option if you need to delete the existing customized form. The customized form in the Forms folder in the Project Explorer is deleted.
 - Choose the **Keep** option if you need to retain the customized form. You would select this option if the Form were still to be used by other user tasks.
3. After providing the delete options, you need to generate the new form. Go to the context menu of the user task and click **Form > Open**. You can also click the **Form...** radio button in the user task General Properties tab.
4. If you previously specified the **Delete** option on the Confirm Delete dialog box, a default auto-generated form is created.
5. If you previously specified the **Keep** option on the Confirm Delete dialog box, the Overwrite Form dialog box appears. Choose the **Overwrite** option if you need to overwrite the existing customized form with a default form, or choose the **Reuse** option to continue using the existing customized form.
6. The system remembers your responses; the next occasion on which it presents either of these dialog boxes your previous choice becomes the 'default action' (the one executed if you press the Enter key). If you select the **Do not ask this question again** check-box, the dialog box is not displayed on subsequent occasions and the

system automatically takes the action that you choose on this occasion. These settings might be changed using **Window > Preferences > Form Designer > Synchronization**.


Updating Forms with the Synchronization Wizard

If the set of task parameters changes after a form has already been created, you can use the **Synchronize Form** wizard to update the set of form parameters, controls, and bindings in the linked form. Likewise, if a change is made in the business object model that impacts the form (for instance, if an attribute is added to a class that defines a type on the form), the synchronization wizard updates the form appropriately.

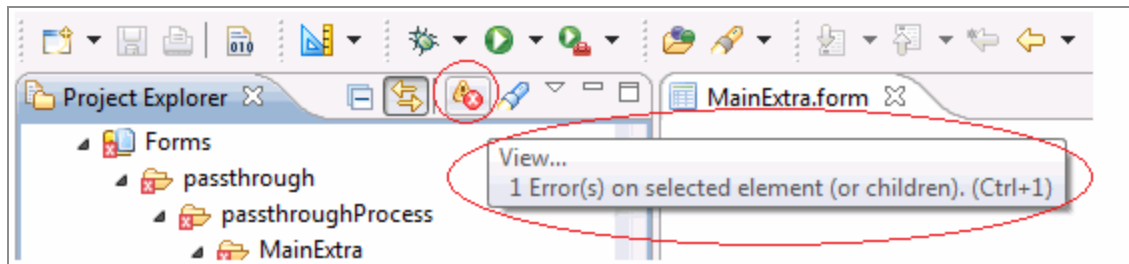
When a form needs to be synchronized, a problem marker appears on the form in the Form Designer, usually near the element that is out of sync, as well as in the Properties and Content Outline views. Mouse over the problem marker on the form to see a tool tip containing a description of the problem and a link to launch the synchronizer, as shown below.

Problem Marker: Form out of sync

The screenshot shows a form titled 'User Task' in the Form Designer. A yellow tooltip is displayed over a red problem marker icon in the top-left corner. The tooltip contains the text: 'Forms Process 2.x : Form 'UserTask' is out of sync', '1 quick fix available for this problem...', and a link 'Synchronize Form' with a green icon. The form itself has a hierarchical structure with expandable sections. The 'Student' section is expanded, showing three text input fields: 'Name', 'Student Number', and 'Current Courses'. The 'Course' section is also expanded, showing a 'Master' section with two text input fields: 'Course Title' and 'Course Number'. To the right of these fields are two buttons: 'New' and 'Delete'.

You can also use the  button available in the Project Explorer view, Outline view, and Properties view. When the problem markers are in scope for the active view's input, this action is visible and can be clicked to inspect the markers and execute quick fixes.

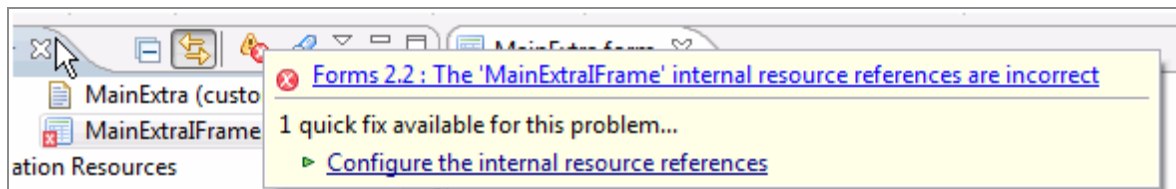
Problem Marker in the Project Explorer View



On clicking the  button, two hyperlinks are available:

- The marker link (opens the offending resource in its appropriate editor)
- The quick fix link (opens the Quick Fix wizard)

Problem Marker in the Outline View



If there are multiple markers, the tooltip window displays left and right arrow buttons to navigate the markers. If a marker issue has multiple quick fixes, they are all listed in the lower section.

Options for Synchronization

If multiple problem markers are associated with the selected element on the form, the tool tip includes left and right arrows for navigating through them. Also, in the case of multiple synchronization problems, the tool tip can present different quick fixes for handling the problems.

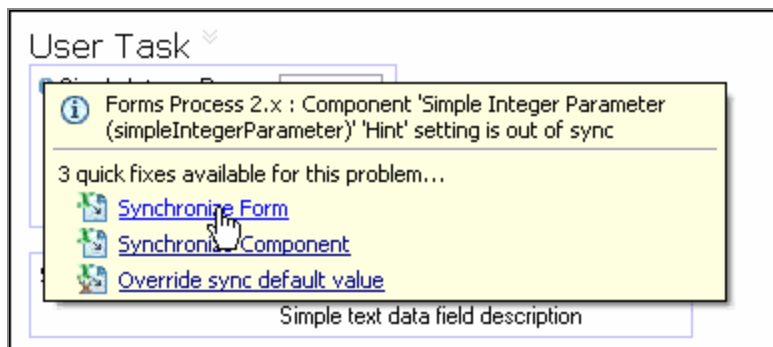
The first two quick fixes are: choose to synchronize the entire form, or synchronize just the component whose problem marker was used to launch the wizard. In the latter case, although fixes for all out-of-sync components are displayed, *only* the component whose problem marker you "mouse over" has the check boxes checked for its available synchronization actions, that is, selected for updating by the wizard.

There is a third option when multiple components are out of sync. You can choose to *override* the sync default value, that is, to leave the form as it is (perhaps because the problem is minor or because you intend to fix the user task parameter, instead, to bring the user task and the form into sync). This option causes the wizard to ignore the suggested fix or fixes and remove the problem marker for the component whose marker was used to launch the wizard.

i Note: If the wizard is subsequently launched with the **Synchronize component** option using the problem marker of *another* component, fixes for the component for which the override was done is still displayed, but disabled and deselected to indicate the override. Nonetheless, the grayed-out fixes can be selected, undoing the earlier override. In this case, the wizard displays a warning message to the effect that user overrides are overwritten with generator defaults.

The image below shows an example of three possible synchronization quick fixes presented by the tool tip: synchronize form, synchronize component, and override sync default value.

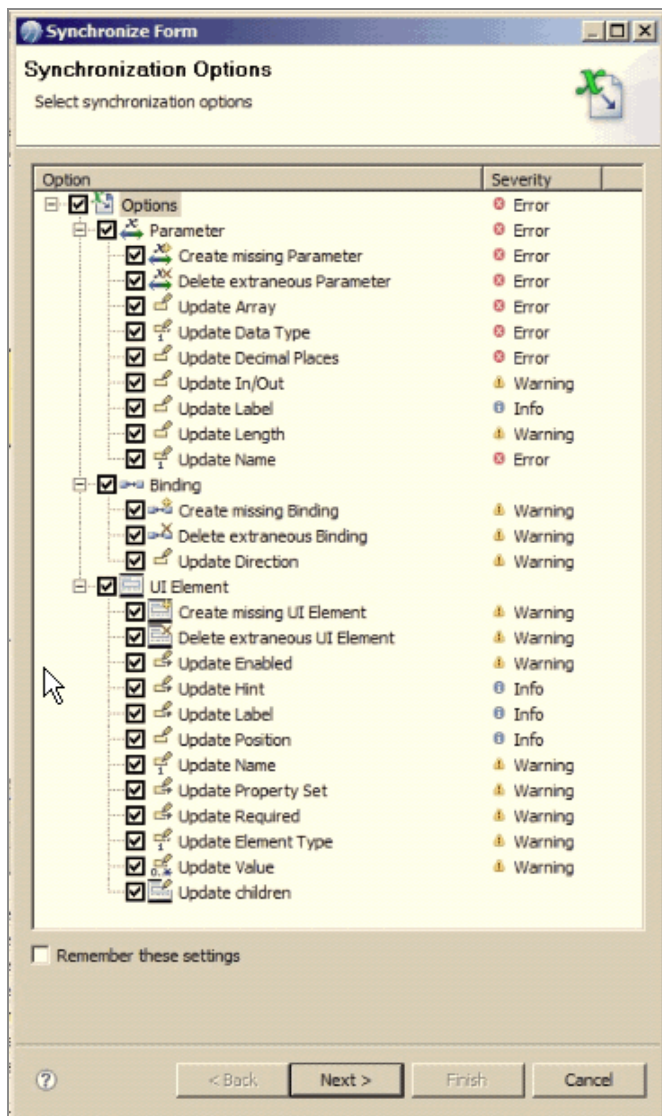
Synchronization Quick Fixes



Click the desired link in the tool tip to launch the **Synchronize Form** wizard.

To launch the wizard directly, without using a tool tip, right-click the user task, and click **Form > Synchronize**. The menu for launching the wizard is shown in the illustration below.

The first screen of the wizard, shown below, presents a list of all the possible conditions on which the wizard can check and report, and, in each case, the severity associated with the item being out of sync. Normally, all of the check boxes can be left checked.

First Page of Synchronization Wizard

If you check the **Remember these setting** check box, the synchronization options wizard does not start on page one in the future, **Synchronization Options**, but takes you straight to page two, Synchronization Actions. From page two, you can still reach page one by clicking the **Back** button.

Click **Next** to advance to the **Synchronization Summary** page, which displays the changes that can be made, such as parameters to be added, modified, or removed from the form. Clicking the Expand All (+) or Collapse All (-) button on this page expands or collapses the list of actions displayed. The check boxes enable you to select the actions the wizard performs when you click the **Finish** button.

Actions performed by the wizard respond to the **Undo** and **Redo** commands invoked on the **Edit** menu. Note that if a synchronization fix is undone, recreating the out-of-sync condition, the problem marker might not be displayed again until the synchronization wizard is invoked again.

Synchronization Wizard Preferences

The list of conditions that are checked by the wizard (shown on the wizard's first page), and the severity level indicated by the problem marker icon (and displayed in the tool tip) for each item, can be modified by editing the **Synchronization Options**. In the menu bar, click **Window > Preferences > Form Designer > Synchronization** to view and edit the synchronization settings. You can also click **Window > Preferences > Form Designer > Synchronization** to open the dialog box and check or uncheck the check boxes to remove or add an item to the list of items that are checked when the wizard is run. Click the severity for an item to display an option list for selecting a different severity. In most cases, the default severity choices are suitable.

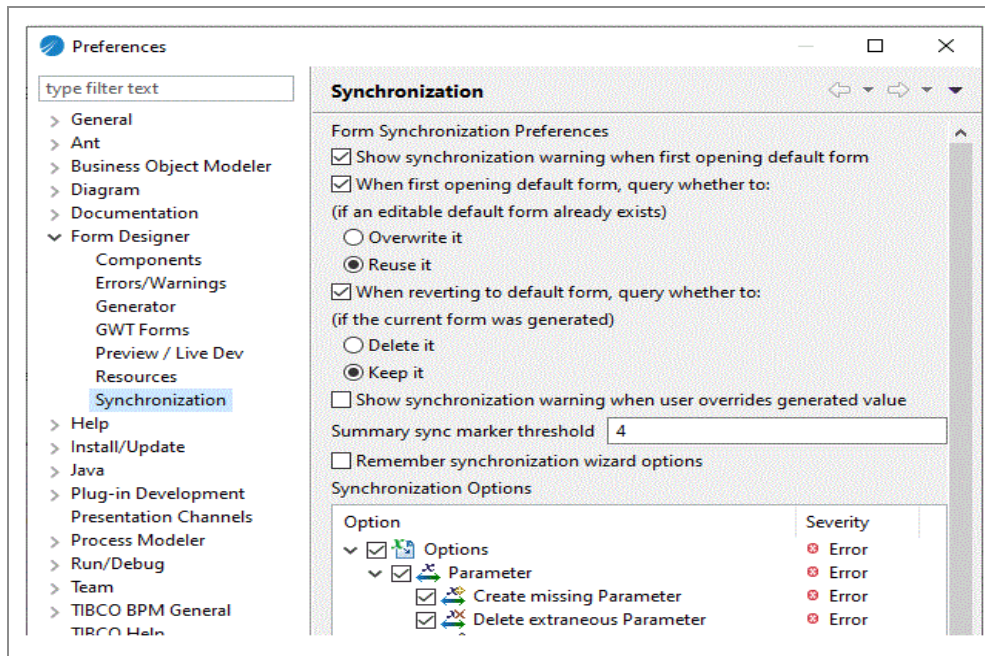
You can also click **Window > Preferences > Form Designer > Errors/Warnings > Forms Synchronization** to view descriptions of the items that can be synchronized by the wizard, and view or edit the setting for the severity associated with an out-of-sync condition for each item.



Warning: The presence of an **Error** marker prevents deployment of the form, while **Warning** and **Info** markers do not. If the marker's severity for an item is changed from **Error** to **Warning** or **Info**, the form is deployable, but doing so is strongly discouraged, since the severity level of **Error** indicates a likelihood of severe problems with the form at runtime if the condition is not addressed. For this reason, the severity levels should normally be left unchanged.

There are additional preferences affecting synchronization that can be edited in the Synchronization dialog box (**Window > Preferences > Form Designer > Synchronization**), shown below.

Preferences for TIBCO Forms Synchronization Support



Most of the settings here are self-explanatory. The **Summary sync marker threshold** setting deserves mention, however. This is the number of sync markers on the form above which individual problem markers are not shown. By default, the threshold is 4. Change this setting by typing the desired number in the text field. If the number of sync markers on the form are above the **Summary sync marker threshold** value, only one summary marker reading "Form <form name> is out of sync," is displayed.

If you check the **Remember synchronization wizard options** check box, the wizard does not start on page one in the future, **Synchronization Options**, but takes you straight to page two, **Synchronization Actions**. From page two, you can still reach page one by clicking the **Back** button.

Using Live Development


Use Live Development when you want to make quick changes to your project, and test the results immediately without having to redeploy an entire project. This is especially relevant to Forms development, where small changes require a rapid turnaround for retesting.

Procedure

1. Deploy the entire TIBCO BPM Enterprise application using Deployment Manager (if

not already deployed).

See "Deployment Manager" in *TIBCO BPM Enterprise Administration*.

2. Switch to the **BPM Live Dev** perspective (select **Window** > **Perspective** > **Open Perspective** > **BPM Live Dev** in the top right of the pane).
3. Edit the information in **Work Manager View Connection** for **Work Manager/Client Base URL**. Replace the `<domain>` element with your own Work Manager URL domain.
4. Click  Launch Live Dev Work Manager to open the TIBCO BPM Enterprise Work Manager in Live Dev mode in a browser window.
5. Provide the login credentials to the Work Manager, if prompted.
6. Click **Business Service**, **Work Views**, or **Case Manager** depending on the form that you are developing.
7. Start a process and proceed until the required work item arrives (or start a business service and progress to the appropriate form). In case it is a case action form, click on the case action.
8. Access a form on opening a work item/business service/case action. The form from your design-time workspace is used instead of the deployed form.
9. Iteratively:
 - Test the form.
 - Edit the form in the local workspace and click **Save**.
 - Either use the **Refresh** button provided on the form which reloads the form with the latest changes without the need to re-open it or restart the pageflow for instance, or close the work item or business service in Work Manager or browser. A **Cancel** button is also provided on the form in cases where Form loading fails. The **Cancel** button cancels the form if the form fails to load.
 - Reopen the work item / business service in Work Manager on the browser.

The process flow works as normal and you can complete/edit data in the same way as in normal Work Manager.



Note: The Browser tab continues to use the Live Dev mode, even after the window refresh, until the tab is closed.

Using Data Fields and Parameters with Process User Tasks

Data fields and process parameters represent data that can be used as input to, or output from, steps in a business process. When mapped to a user task, process parameters represent the inputs and outputs of the user task. The parameters of a user task, in turn, can be mapped to controls on the form associated with that user task.

i Note: The term *parameter* is used as follows:

Formal Parameters

Parameters created at the process level of a process package represent interfaces for *other* processes that want to provide input to, or receive output from, the process that contains these parameters. They can be used, in turn, as input to, or output from, any user task in the business process by being added to the user task's interface.

User Task Parameters

User task parameters are properties of a specific user task within a business process, and they are mapped to data fields or process parameters using the **Interface** tab of the user task's Properties View.

The Mode Property of User Task Parameters

The Mode property of a user task parameter refers to whether it is an **In**, **Out**, or **In/Out** parameter. The Mode is set differently depending on whether the parameter is derived from a process parameter or a data field:

- **Process Parameters**

Mode value set on the parameter's Properties view.

- **Process Relevant Data**

Mode value set on the user task's Interface Properties view.

Using Data Fields and Parameters

Incoming data can populate form controls with initial values or dynamically set runtime values for certain control properties, such as the text of a control's label, or whether it is initially visible on the form. Outgoing data are submitted when the user clicks the **Submit** button. This data can be used in various ways in subsequent phases of the business process, sent to an external process, or written to a persistent store.

Relationship Between User Task Parameters and Form Parameters

A user task and a form have their own separate models for the input and output parameters for the task or form. When a form is created from a user task, the form parameters are automatically set to match those in the user task.

The Synchronize Form operation on the user task again updates the form parameters to match those in the user task if there have been any changes since the form was originally generated.

Data Types for Data Fields and Process Parameters

Data fields and process parameters must either have one of the predefined basic data types or an existing complex type.

Basic Data Types

The following simple data types can be used for a data field or process parameter:

Boolean

True or false.

Date

Date only.

Date Time and Timezone

A combination of date, time, and time zone.

Fixed Point Number

Fixed point number is represented as a standard JavaScript BigNumber, where the number of digits and of decimal places are defined, and should be used for financial calculations that use decimals.

Floating Point Number

Number is represented as a standard JavaScript BigNumber, where the number of digits and of decimal places are not defined.

Performer

A special type of data field/parameter that you can select as a participant for a user task. By assigning a value to the performer data field/parameter earlier in the process, you can dynamically define a participant for a user task.

Text

A set of alphanumeric characters, with a specified maximum length.

Time

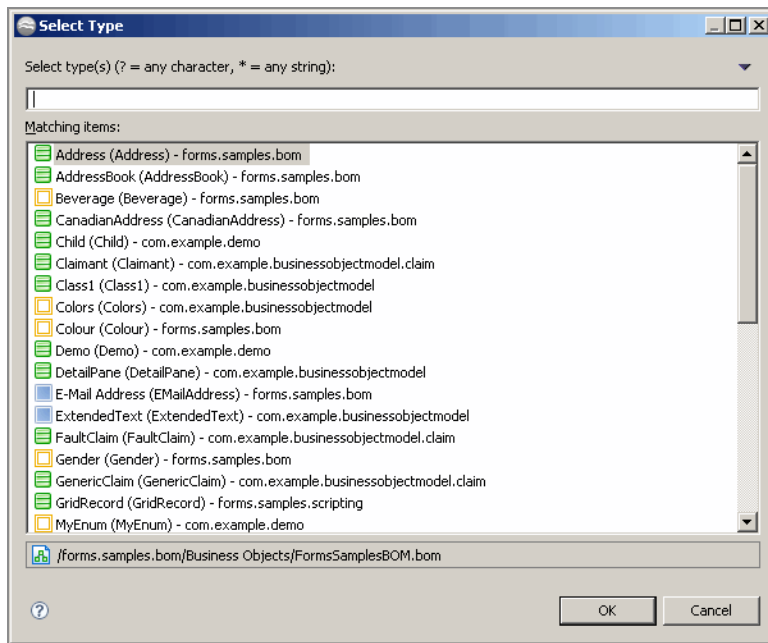
Time only.

URI

A URI (Uniform Resource Identifier) is a string of characters that unambiguously identifies a particular resource.

Complex Data Types

Complex data types must first be created in a business object model to be available as types for data fields and parameters. All existing complex data types available to the business process appear in the Select Type dialog box that opens when **External Reference** is chosen as the type for the parameter or field, and the browse button is clicked:

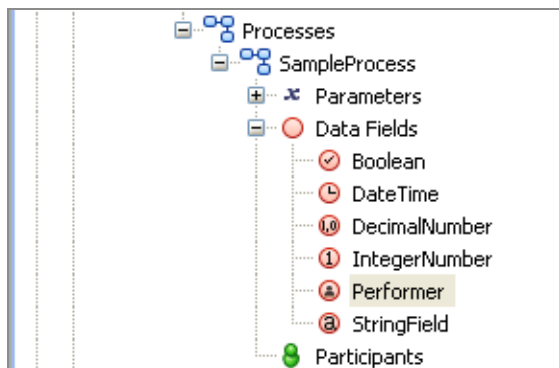


Process Data Fields

These fields are available to any form associated with a user task within that process, but not to forms in other business processes in the package.

Data fields created at this level appear beneath the business process in the Package Explorer.

Data Fields at the Process Level

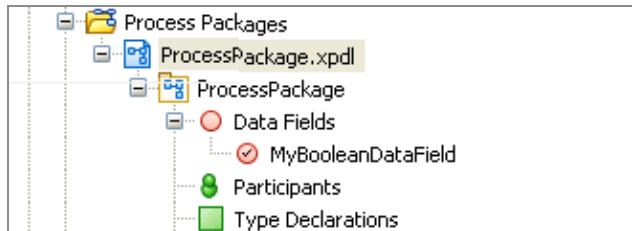


Data Fields at the Process Package Level

Data fields created at the package level appear under a process package in the Project Explorer, and are available to any form in any business process within the process package.

In other respects, data fields at the package level are identical to those at the process level.

Data fields created at this level appear beneath the process package in the **Package Explorer**.

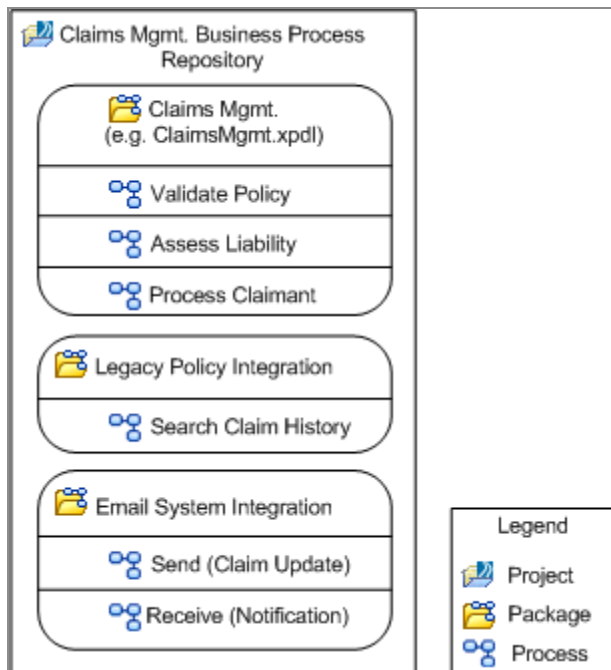


Process Components

Process components represent reusable building blocks that encapsulate the management of a particular item in a business process. The process components form a reusable library that you can call upon in different contexts.

For example, you might have an item to "verify the caller's address/contact details" in the business process for taking out an insurance policy. This could be implemented as a sub-process and this particular process component could be used in the context of renewing an insurance policy.

The following example shows a project and the associated packages and processes used in an insurance environment.



In this example, the Validate Policy process might call a sub-process in another package (for example, the Search Claim History process). This sub-process is in the same project in this example, but it could be located in a different project.

Sending an Email Message from a Process

You can send email messages from a process by configuring the service task. These messages could be internal notifications giving information about the progress of the process, or external messages such as automatic order confirmations sent to customers.

To set up your server to deliver email you need to do the following:

- An SMTP resource instance must exist in the TIBCO Cloud BPM runtime before you can deploy an application that uses a service task to send an email. This resource instance defines the connection information used by TIBCO Cloud BPM to contact the SMTP mail server.

To define a resource instance, you must use TIBCO Cloud BPM Administrator to:

- create a resource template. A resource template specifies configuration properties for resource instances.
- create a resource instance based on the resource template. A resource instance represents a resource shared between applications.

- install the resource instance on a host. The resource instance is then available to applications running on that node.
- There are several different types of resource instance. An SMTP resource instance must be used to provide a connection to an SMTP mail server.
- You use TIBCO Cloud BPM Administrator to define an SMTP resource instance.
- You can bind to the resource instance when you deploy the process.

Configuring Service Tasks to Send Email Messages from a Process

A service task that is configured as an email task:

- Defines the email addresses which the message is to be sent from, and to which replies are to be sent, as well as the address of the recipient of the message. Addresses can be specified explicitly or taken from available data fields; you can include fields in the message and the contents of the fields are used at runtime.
- Defines the body of the text. This can also be typed in explicitly or taken from available data fields.
- Defines any files or field contents that are to be sent as attachments to the email message.

As well as configuring the service task, you need to:

- Create a system participant (of type email),
- Assign that participant to the email service task.
- At deployment, bind the participant to the appropriate SMTP resource instance (which must already exist).

Defining an E-Mail Service Type from a Service Task

Procedure

1. Select the service task. On the **General** tab of the Properties view for the service task, select the **E-Mail** option from the **Service Type** drop-down list.


2. Enter an email address for the recipient in the **To:** field, a subject, and the body text for the message. You can enter actual email addresses and plain text, or you can specify data fields, delimited by percent characters. For example, if you specify:

%ManagerID%

in the **To** field, this is replaced by the contents of the **ManagerID** data field. (See [Setting up Dynamic Data Inputs to an Email Message](#) for more information.)

This is the minimum configuration necessary to send an email message. For further options, click **More Details** (which navigates you to the **E-Mail** tab) or the **E-Mail** tab and continue to specify further parameters.

3. On the **E-Mail** tab, you can specify further parameters for the **Definition** of the email message.


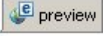


Note: All of the parameters on the **E-Mail** tab can be specified using a data field or a parameter. Click the  button to display the Select Data Field or Formal Parameter dialog box and choose the required data field or parameter. You can also mix text with data field values, as shown in the **Subject** field of the example illustration.

- **From:** Use **Custom Configuration** to choose a data field or parameter which specifies a different **From:** address for this email.
- **To:** Specify the recipient of the email, either as an explicit email address or by selecting a data field or parameter. This is a mandatory field and you receive an error if it is not present.
- **Cc:** Specify any recipients to whom you want to send a copy of the email either as explicit email addresses or by selecting a data field or parameter. Their email address is visible to other recipients of the email.
- **Bcc:** Specify any recipients to whom you want to send a blind copy of the email either as explicit email addresses or by selecting a data field or parameter. Their email address is not visible to other recipients of the email.
- **Reply To:** Use this parameter to specify a different email address to which recipients of a message can reply. Alternatively, select a data field or parameter.
- **Headers:** Message headers provide a list of technical details, such as who sent the mail message, the software used to compose it and the email servers it

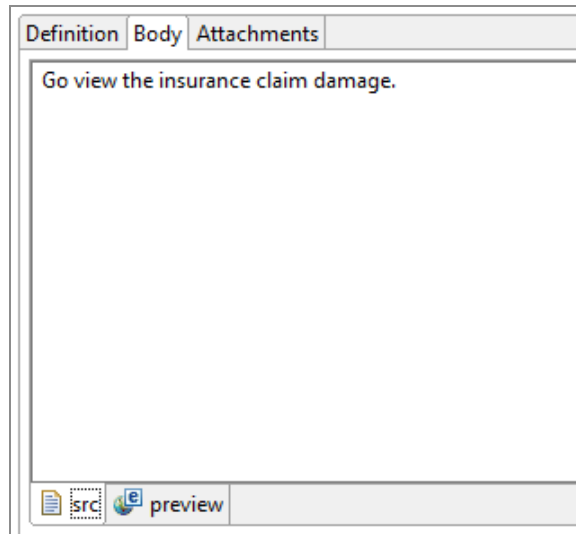
passed through to get to the recipient. Use this parameter to specify additional information in the header of the email.

- **Priority:** Select a priority from the drop-down list (**Low**, **Normal**, or **High**) or select a data field or parameter.
 - **Subject:** Type the Subject line for the email message or select a data field or parameter.
4. Click **Body** to specify the main text of the message. Click in the main part of the field to enter text. You can enter either plain text or HTML. If you enter HTML, then when you use the preview option below, you can see what the output of the HTML is. You can also specify the contents of data fields or files to be inserted.

At the bottom of the text area are the following buttons:

- Click the  icon to view the source of the body text of the email.
- Click the  icon to display a preview of how the body text looks to the recipient.
- Click the  icon to open the Select Data Field or Formal Parameter dialog box, which you can use to select a field and specify that the contents of the field should be inserted in the email. This button is only available when you are viewing the source of the body text, not when you have selected a preview.
- Click the  icon to open the Open dialog box, which you can use to select a file and specify that the contents of the file should be inserted in the email.

You can alternate between viewing the source of the body text and previewing how it looks to the recipient:



5. Click **Attachments** to specify a document to be attached to the message:

- **Field Contents:**

At run time, the resulting e-mail message produced by this task has an attachment named field.txt, where field is the name of the attached data field.

- **Files:**

Use this option to browse the file system and attach one or more files to the email message.

Note: To be successful, file attachments must use a consistent directory structure and file name at design time and runtime. So, for example, if you are attaching a file `C:\temp\readme.txt` at design time, then that same file, in the same directory, needs to be available to the runtime.

This also means that in a distributed environment, this needs to be available on any host running a BPM Server node. It would be good practice to have a mapped drive available to all hosts where the attachment is available.

Setting up Dynamic Data Inputs to an Email Message

Data fields available to the process can be used to dynamically set parts of the email at runtime, using either `%FieldName%` notation or the Select Data Field or Formal Parameter

dialog box.

See [Defining an E-Mail Service Type from a Service Task](#).

However, a process formal parameter, or a package-level or process-level data field that is defined as an external reference to an attribute of a class defined in a business object model (that is, as structured data), cannot be used in this way.

Procedure

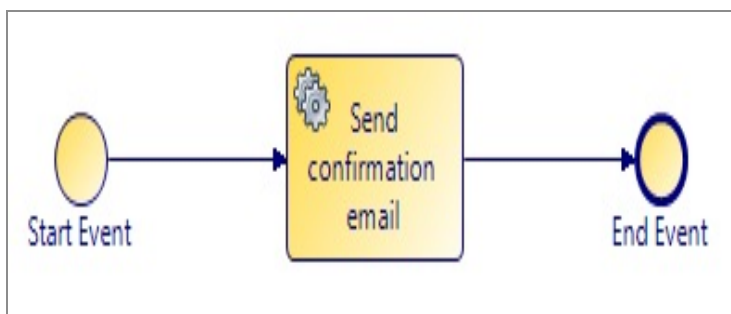
1. Create an activity-level data field on the **Data Fields** tab of the email service task.
The data field:
 - must be defined as a simple type that matches the type of the attribute you want to reference.
 - must not have the same name as an existing formal parameter, or package-level or process-level data field.
2. On the **Scripts** tab, define an **Initiate Script** to assign the value defined in the desired attribute to the data field.

Note: The attribute must have been populated earlier in the process - for example -- by collecting the data on a form.

3. On the **General** or **E-Mail** tab, enter the local data field name at the appropriate place, using either `%FieldName%` notation or the Select Data Field or Formal Parameter dialog box (as described in [Defining an E-Mail Service Type from a Service Task](#)).

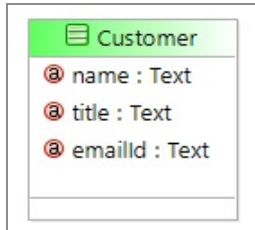
Example of Setting up Dynamic Data Inputs to an Email Message

A project contains a process that uses an email service task to send a delivery confirmation email to a customer.



The process is triggered by an external application that sends in the customer's email address, name and title.

The **Start Event** collects this data via a formal parameter, **customer**, which is defined as an external reference to a business object model that defines a **Customer** class.



The **Send confirmation email** service task uses the email address, name, and title to dynamically build the email message. However, the service task cannot use the **customer** formal parameter to obtain this data.

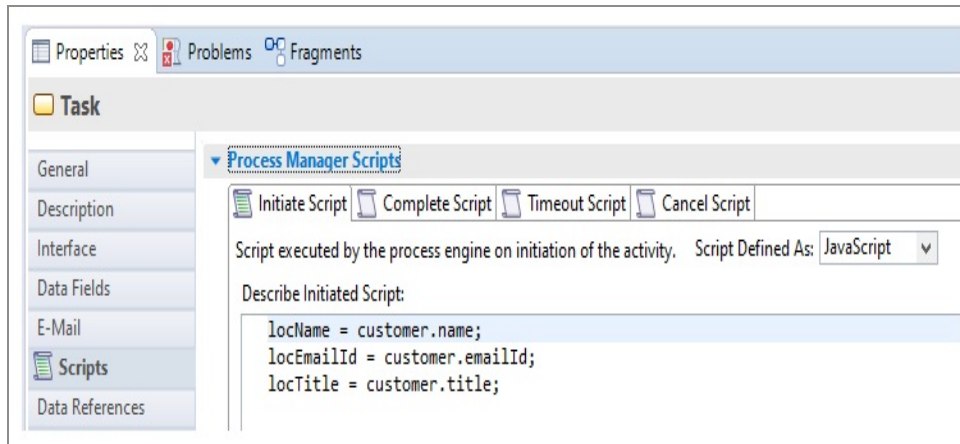
Instead, to access the **Customer** attributes **name**, **title** and **emailId** on the service task, you must perform the following steps:

Procedure

1. Define the following activity-level data fields on the **Data Fields** tab of the **Send confirmation email** service task.

Label	Name	Read ...	Type	Length	Decimal ...	Array	BOM Type	Case Class R...	Type Declara...
@ locName	locName	<input type="checkbox"/>	Text	50	NA	<input type="checkbox"/>	NA	NA	NA
@ locEmailId	locEmailId	<input type="checkbox"/>	Text	50	NA	<input type="checkbox"/>	NA	NA	NA
@ locTitle	locTitle	<input type="checkbox"/>	Text	50	NA	<input type="checkbox"/>	NA	NA	NA

2. On the **Scripts** tab, define the following **Initiate Script** to assign the values defined in the **Customer** object to these data fields.



- On the **General** tab, enter the field names as shown below.

Service Type:	E-Mail
To:	%locEmailId%
Subject:	Order delivery
	Dear %locTitle% %locName%,
	Your order has now shipped.
	Thank you for shopping with us!

Result

At runtime, the following actions occur:

- the email message is sent to the email address defined in the **emailId** attribute of **Customer**.
- the email message contains the **title** and **name** defined by those **Customer** attributes.

Using Email Templates

Email templates contain organization-specific information, such as company logos and corporate style.

Creating your Email template files

Procedure

1. In Project Explorer, navigate to the project you want to add the Email template to.
2. Right-click the project and select **New > Folder** and name your folder.
3. Right-click the folder you created, and select **Special Folders > Other > Use as Presentation Resources Folder**.
4. On the Project Assets dialog box, click **Finish**.
5. Right-click the Presentation Resources folder you created, and select **New > File** and create the files you require.

Alternatively:

- Right-click **Presentation Resources > New > File**, and create the files you require.

i Note: If you create the files you require elsewhere, copy and paste them by copying them and then right-clicking on **Presentation Resources** (or your new Presentation Resources folder) and selecting **Paste**.

i Note: This email template is available to this project only. If you attempt to access it from another project, you might receive a validation error, as the files contained are not available outside this project.

Applying the Email template to a Project

Procedure

1. From the project, right-click **Properties > Presentation Channels**. Select the check box **Enable project specific settings**.
 - or select **Window > Preferences > Presentation Channels**, and click **Configure Project Specific Settings...** Select the project you want to configure and click **OK**.
2. Expand **Default Channel** and select **Work Manager Email**.

3. Click in the **Value** field for **mailTemplateLocation**, and you can see the picker:
4. Use the picker, to navigate to the HTML file you created in the Presentation Resources folder, and click **OK**.
5. Click **Apply**.

Result

Next time you push an email, it uses this template.

i Note: Email templates are scoped to one project only. If the template is applied on the workspace level, then all relevant projects should have the specified custom template files available inside them. It is then recommended to overwrite the workspace Presentation Channel definition on the project level when the custom email template needs to be provided.

Creating an Alternative Email Template

You can create your own email template that can contain organization-specific information, with your own logos and corporate style.

After deployment, it can be used as the default email template for any Push Destination using the **Work Manager Email** channel type.

The files you can use to create your email template are as follows:

- **HTML file** (Mandatory) This contains the information about what you want the email to contain. It typically includes:
 - a reference to the .css file as its stylesheet. This can be included using an HTML line similar to the following:

```
<link rel="stylesheet" type="text/css" href="cid:Easycss" />
```

- references to any of the graphics files contained in the cid.properties file (referred to by their cid: identifier, prefaced by 'cid:'), for example:

```

```

refers to NewProductLogo.png in the example in cid.properties above.

Referring to them in this way means they can be shown externally as part of the email when the HTML file is used to create an email after deployment.

- the content you want to include in the email.

This HTML can contain any of the following tokens, each of which is replaced by an actual value at runtime;

%%token.workItemUrl%%	URL of the work item.
%%token.workItemId%%	The id of the work item.
%%token.entityName%%	The entity name who requires push notifications as defined in the Organizational Model.
%%token.mailDate%%	The date and time the pushed mail message was sent.
%%token.mailFrom%%	The name of the sender of the pushed mail message.
%%token.mailSubject%%	The subject line for the pushed mail message.
%%token.mailTo%%	The user/s who the pushed mail message is sent to.
%%token.mailCc%%	The user/s who the pushed mail message is copied to.
%%token.mailBcc%%	The user/s who the pushed mail message is blind-copied to.
%%token.hostIPAddress%%	The IP address of the host.
%%token.hostMachineName%%	The host machine name.
%%token.baseurl%%	The base URL.

- **CSS (Cascading Style Sheet) file** This is a stylesheet file, typically for your organization, which contains standard information such as banners or background colors, that you want to be in every email you send.
- **graphics** you are referencing from the `cid.properties` file, which would then be used by the HTML file. These must be available locally, and can either be in the same directory (**Presentation Resources**), or in a sub-directory.

Note:

- **cid.properties** This allows you to reference any graphics files you have available locally (in the same directory, or a subdirectory) which you want to be used when you push the work item (which then sends an email to notify the user of the work item). It is specifically used for embedded images / css in emails, rather than providing the option of external calls.
- Each file name has a `cid:<name` identifier which allows it to be accessed remotely as well as locally. For example, the `cid.properties` file might contain lines such as the following, where `cidOne` is the `cid:` identifier for `NewProductLogo.png` when you reference it later in the HTML file:

```
cidOne:NewProductLogo.png
```

Deleting a Process

You can delete a process and decide whether to delete referenced processes.

Procedure

1. Do one of the following:
 - Right-click the process in the Project Explorer, and select **Delete**.
 - Highlight the process and select **Edit > Delete**.
 - Highlight the process and press **Delete**:
2. If the process you are deleting is referenced in the parent package (for example, you are deleting a sub-process), a dialog box is displayed:

- To invalidate the process that is referring to the process you are deleting, click **Yes**.
 - To retain the process, click **No**.
3. If the process you are deleting is not referenced in the parent package, it is still possible that it is referenced by a process in a different package. TIBCO Business Studio cannot validate against this however and a dialog box is displayed:
- If you are sure that processes in different packages do not reference the process you are deleting, click **Yes**.
 - If you do not want to delete the process, click **No**.

i Note: It is also possible to delete a process by deleting the XPD file of the package that contains the process in Windows Explorer. This deletes all processes in that package. If you delete a package in Windows Explorer while TIBCO Business Studio is open, you must refresh the Project Explorer to see an updated view of the workspace.

Process and Service Process Interfaces

Process interfaces and *service process interfaces* specify the events and their parameters that must be present in processes created using that interface. At runtime, any of the processes that implement the interface might be chosen based on data available at that time.

You can:

- use a process interface with a business process or a pageflow process.
- use a service process interface with a service process.

The use of a process and/or service process interface allows the dynamic selection of sub-processes at runtime, allowing separation of the design-time and runtime environments.

A process interface consists of start events, intermediate events, and their associated formal parameters and errors. A service process interface is the same as a process interface, except that it also specifies a deployment target.

Process Interfaces

Using the Process Interface Editor, you can modify the interface to add events or parameters.

Process interfaces are used in applications where the specific sub-process to execute in a sub-process task needs to be decided at runtime. The process interface defines the parameters required to start a sub-process. Each process that might possibly be executed from the same sub-process task must have the same process interface selected (so that it has a common parameter set). Then the sub-process task is set to reference a process-interface and a data field whose content at runtime specifies the actual sub-process to execute.

The screenshot shows the 'Process Interface (ProcessInterface)' editor window. It has a title bar with two tabs: '*Claims-Process2 (ClaimsProcess2)' and '*Process Interface (ProcessInterface)'. The main area is divided into four sections:

- General Information:** This section describes the general information about this process interface. It contains a 'Name' field with the value 'Process Interface (ProcessInterface)' and a 'Description' field which is empty.
- Start Events:** Specify events that can be used to start processes that implement this interface. It contains a list with one item: 'Start Event (StartEvent)'. To the right of the list are 'New...' and 'Remove' buttons.
- Formal Parameters:** Specify the formal parameters used by events in this interface. It contains an empty list box. To the right of the list box are 'New...' and 'Remove' buttons.
- Intermediate Events:** Specify in-flow events that can be triggered in processes that implement this interface. It contains an empty list box. To the right of the list box are 'New...' and 'Remove' buttons.

At the bottom of the window, there is a tab labeled 'Process Interface (ProcessInterface)'.

Once created, a process interface can be used by several different processes. If a process is created using a process interface, all the events and parameters specified in the interface are present in a process that implements that interface. You can add additional events or parameters.

✓ **Tip:** If you have additional parameters that are local to a process that implements a process interface, you can move the parameters into the process interface by right-clicking the parameter and selecting **Move Parameter to Interface**.

This option is only available for processes that implement a process interface in the same package, and only for parameters that do not have problems in the Problems view.

If you create a process using a process interface, the process that you create inherits the events and parameters created in the interface.

Start Events

A process interface must have one start event.

- You must specify one start event. The start event is the start activity for processes that are started by sub-process tasks or manually. See [Dynamic Sub-Processes](#).
- A start event for a process interface can be associated with mandatory formal parameters using the **Interface** tab. For example:

Process Data Name	Mode	Mandatory	Description
FP1	In / Out	<input checked="" type="checkbox"/>	

If a process instance is invoked using this start event, the parameter **FP1** must be specified.

Both input and output (including combined in/out) parameters can be associated with the event. Specifying output parameters means that these are the parameters that are returned when the process is invoked using this start event.

Creating a Process Interface

You can create a process interface from Project Explorer.

Procedure

1. Right-click the Processes Interfaces folder in Project Explorer and select **New > Process Interface**.
2. In the **Label** field, enter a name for the interface. The Name field is auto-filled with the same name, with no spaces.
3. (optional) Click **Next**, then specify a description for the interface.

Note: The **Documentation URL** field is intended for design-time collaboration; it is not displayed in the runtime environment.

4. (optional) Click **Next** to specify extended attributes to be used in the interface.
5. Click **Finish**.

Result

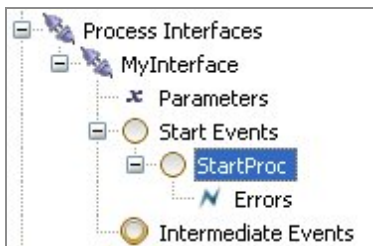
The Process Interface editor is displayed. Use this to add or remove start events, intermediate events, and parameters.

Creating Error Events

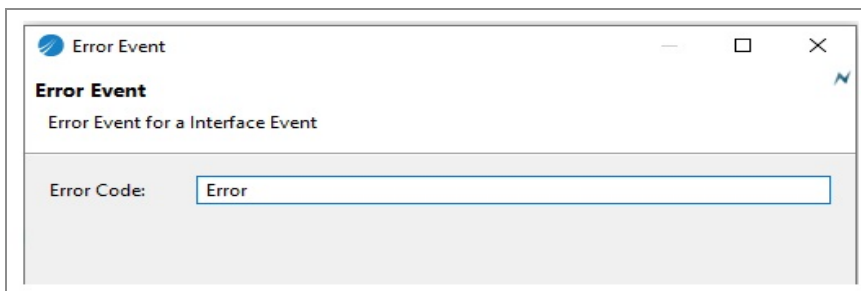
Error events are created in the Project Explorer.

Procedure

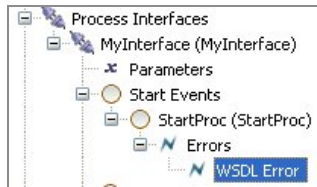
1. Create a process interface.
2. In the Project Explorer, under the process interface, expand an event. For example:



3. Right-click **Errors** and select **New > Error**. The following dialog box is displayed:



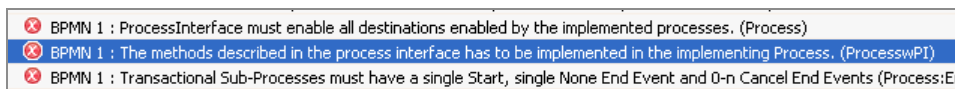
4. Enter a name for the error code and click **Finish**. The error code you created is displayed in the Project Explorer, and in the Process Interface Editor:



Modification of a Process Interface

You can modify a process interface using the Process Interface editor, however any changes made cause validation errors in processes that have already implemented the interface.

For example:



To correct this problem, ensure that the changes made to the process interface are reflected in any processes that have already implemented the interface.

Creating a Service Process Interface

A service process interface allows the dynamic selection of service processes at runtime. A service process interface specifies the deployment target, events and their parameters that must be present in the service processes created using that service process interface. At runtime, any of the service processes that implement the interface might be chosen based on the data available at that time.

Note: You can only use service process interfaces with service processes.

Using the Service Process Interface Editor, you can add the events and parameters you require.

Once created, a service process interface can be used by several different service processes. If a service process is created using a service process interface, the events and parameters specified in the interface must be present in a service process that implements the interface. You can add additional events or parameters, but removing any of those required by the interface invalidates the service process.

If you do have additional parameters that are local to a service process that implements a service process interface, you can move the parameters into the service process interface by right-clicking the parameter and selecting **Move Parameter to Interface**. However, this option is only available for service processes that implement a service process interface in the same package, and for parameters that do not have any problems in the Problems view.

If you create a service process using a service process interface, the service process that you create inherits the events and parameters created in the interface.

Procedure

1. In the Project Explorer, select the package where you want to create your service process interface, right-click and select **New > Service Process Interface**.
2. Enter the **Label** of the service process interface and select **Next**.
3. In **Description**, add optional text that describes the process, an optional URL that links to documentation about the process, and select **Next**.



Note: The **Documentation URL** field is intended for design-time collaboration; it is not displayed in the runtime environment.

4. (Optional) In **Extended**, add any optional supplemental information to the XPDL for the process.
5. Select **Finish**.

The Service Process Interface editor is displayed. Use this to configure the deployment target for the service process interface and add or remove start events, intermediate events, and parameters.

Process Errors

You can specify errors that might be thrown by a process that implements the process interface.

This is useful where the process interface is used for dynamic sub-process invocation. In this case, the error events are translated into end error events in process implementations. Call sub-process activities that reference process interfaces can then identify the thrown error events directly from the process interface.

Process Migration

Process migration provides the ability to migrate a long running process instance from one version to another version of the same process. In other words, migrated process instances continue executing using the new process definition. Process migration is controlled by the use of *migration points* and *migration rules*.

- A *migration point* is a task in the process template at which a process instance can be migrated to a different version. Not all tasks are valid migration points - for example, tasks that have a parallel path, or tasks that might have parallel executions due to multiple tokens flowing on a single path. Valid migration points are automatically identified by TIBCO Business Studio at design time and are denoted by an icon next to the task in the Process Modeler.
- A *migration rule* defines when, how, and to what version a process instance migrates to. A migration rule identifies the migration point in the process template from which a process instance migrates, and the process template version to which it migrates to.

Design Considerations for Process Migration

A process instance can only migrate to a new version of a process when certain criteria become satisfied.

These are automatically identified by TIBCO Business Studio at design-time, but you should be aware of the points described in [Migration Point Restrictions](#).

Migration Point Restrictions

You should be aware of a number of restrictions when designing your process to be sure that it can migrate correctly.

- The task name is used to identify a migration point. This means that all tasks must have names.

i Note: By default, gateways do not have names. This means that you must specify a name for all gateways in your process if you want them to be valid migration points.

Note that:

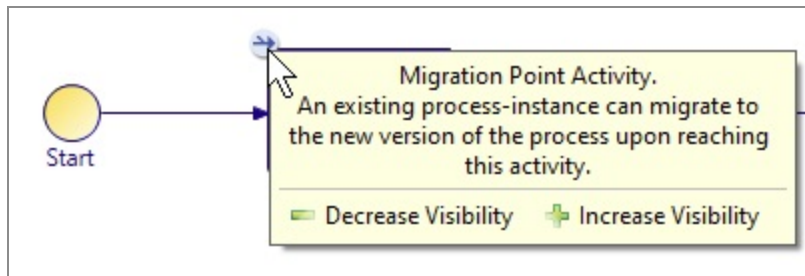
- task names must not be duplicated.
- task names in source and destination process templates must be the same.
- Migration cannot be performed on pageflow processes. Pageflow processes are short-lived processes whose data is not persisted, therefore migration should not be required.
- For a user task, migration should be set before the user task is scheduled. Migration does not happen if the changes are made after the user task is OFFERED to a user.
- Valid migration points are points of the process template where a single process thread executes. Therefore, a task that starts a parallel path is a valid migration point but any tasks that follow are not until those paths are rejoined.
- Part of a process that does not appear to be a parallel flow might be treated as a parallel flow. A primary example of this is incoming request tasks without inflows and any sort of task boundary event. If the flows out of these tasks are not merged back into the process then everything after that point is treated as a parallel flow. This is because without an explicit merge back into the flow then there is an implicit merge at the end of the process (hence all other activities on those flows are counted as "in-parallel" with the incoming request task).
- Any tasks inside embedded sub processes cannot be migration points, but the embedded sub process itself might be a migration point. See [Embedded Sub-Processes](#). Migration takes place before starting the sub process.
- These events cannot be migration points:
 - start events
 - events placed on the boundaries of tasks.
 See [Events](#).
- Tasks that follow these gateways are not valid migration points:
 - Inclusive
 - Complex

- Parallel
- Tasks following tasks that have Timer events placed on their boundary with **Continue Task on Timeout** selected are not valid migration points:
- Tasks following tasks with multiple instance loops with ordering set to Parallel and flow conditions that are set to One are not valid migration points. This is equivalent to an exclusive gateway, and means that only the completion of the first activity instance causes flow to continue.
- **BPM Validation Configuration** on the Resource properties tab of a process is used to suppress problems markers for the 'No migration point activities in the process' Problem. It gives you the choice of validation, suppressing the error until the next process flow change, or suppressing the error until a manual reactivation via Resource properties.
- Migration points cannot be set on an event handler flow but it is possible to modify an event handler flow which can then be triggered after a process migration from a migration point set somewhere on the main flow.
- Event handlers are automatically re-initialized during process migration. This means that they are re-activated using the new values for correlation data.
- When adding an ad-hoc event to a new process version, the initializer for the ad-hoc event must be defined as the migration point or come after the migration point in order for the ad-hoc event to be available after process migration. If the initializer is in a part of the process that has already been executed before migration, the ad-hoc event does not take effect in process instances that have been migrated.
- Migration is delayed with an audit message if the event handler thread is not complete. The migration completes once all outstanding event handler threads have been completed.
- When you add a new ad-hoc activity, and then upgrade and migrate, you must define an initializer for the activity. This could actually be the migration point itself.

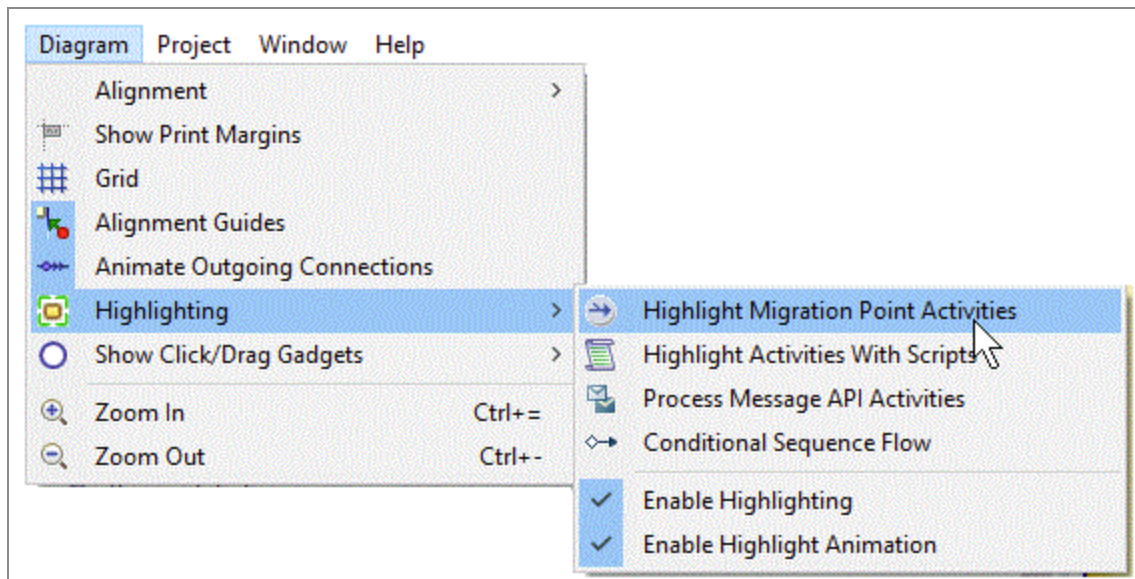
Migration Point Viewing

Valid migration points in a process are denoted by an icon next to the task in the Process Modeler, and by process diagram annotations which allow identification of individual activities as migration points as and when they are viewed in the diagram.

You can increase or decrease the visibility of these icons by hovering over the icon and using the selections (shown in the screen capture below).



Each selection you make is additional to those you have made before. For example, when you select a data field, all of the activities that reference that data field are highlighted in the process diagram. If you then also select the **Highlight Activities With Scripts** option, only the activities that reference the data field **and** have scripts are highlighted.

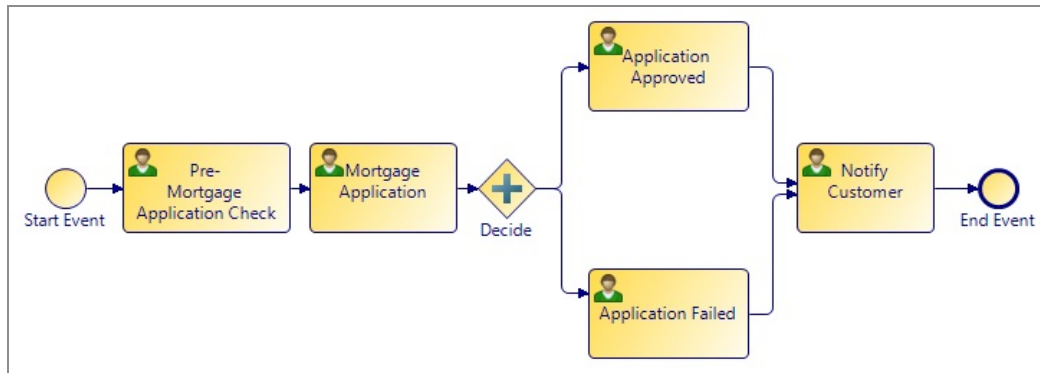


To clear selections, click in the Process Editor window again, and it navigates back to its default setting, which is to show migration points.


You can also choose to highlight migration points by selecting **Highlight Migration Point Activities** from the drop-down toolbar.

Example of Process Instance Migration

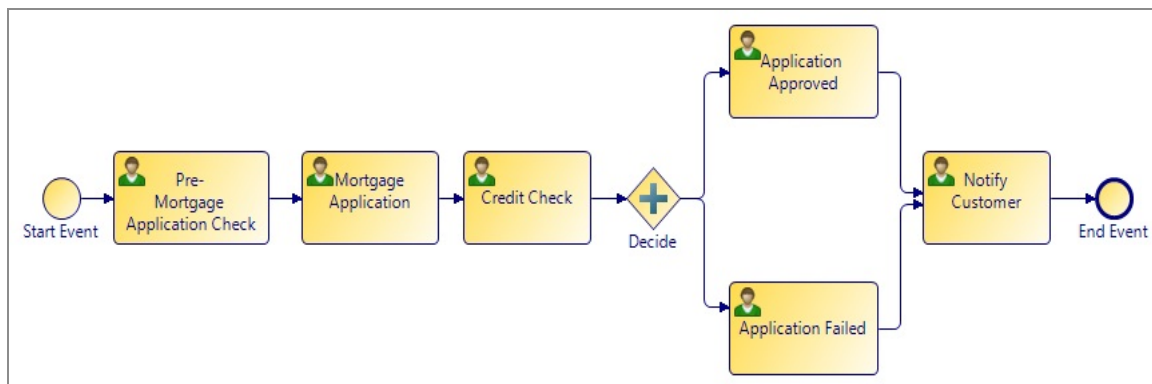
The following figure shows version 1 of a process.



In this process:

- an icon  next to a task shows that it is a valid migration point.
- the **Pre-Mortgage Application Check**, **Mortgage Application** and **Decide** tasks are valid migration points.
- The tasks **Application Approved** and **Application Failed** following the **Decide** gateway are not valid migration points (as more than one task at a time could be active).

The following figure shows version 2 of the same process. A new task called **Credit Check** has been added to the process.



To migrate the process instances from Version 1 to Version 2, a migration rule is created that specifies the **Mortgage Application** task as the migration point.

i Note: A migration point must exist in both the source and destination versions of the process template. **Credit Check** is therefore not a valid migration point for migration between these two versions.

Note that:

- Process instances that have started executing against version 1 migrate to Version 2 when they have finished executing the **Pre-Mortgage Application Check** task but not yet started to execute the **Mortgage Application** Task.
- Process instances that have already started executing the **Mortgage Application** task continue with Version 1 of the process.

This means that all process instances that have started executing against Version 1 of the process template migrate to Version 2 unless they have reached the **Mortgage Application** task.

Process Fragments

Predefined fragments are provided in the folder **Business Process Fragments**.

For example, **Basic Fragments** contains simple fragments such as an embedded sub-process, split conditional, and task sequence.

Rather than reusing an entire process, you can use the **Fragments** view to quickly create new processes.

You can also create your own process fragments. For example, there might be process patterns that you frequently use. By storing these patterns or "fragments" you can easily use them to construct new processes.

Custom Fragments

The Fragments that you create are stored in the category you specify and can be used to create new processes in the same way as predefined fragments.



Tip: You can drag fragments and drop them into other categories.

Using the Predefined Fragments

You can use one of the predefined fragments.

Procedure


1. Open the process.
2. Make sure that the Fragments view is visible. If it is not, select **Window > Show View > Fragments**.
3. In the Fragments view, there is a navigation tree that looks similar to the Project Explorer. This is the **Fragment Explorer**. Expand **Business Process Fragments** folder. This contains categories of BPMN process fragments.
4. Expand the category that contains the fragment you want to use.
5. Select the fragment.
6. Drag the fragment onto the process.



Tip: You can also use the **Copy** and **Paste** menu options either by right-clicking or by selecting from the **Edit** menu.

Creating Custom Fragments

If a portion of your process is one that you are likely to reuse in other processes, you can capture the useful part of the process and save it as a custom fragment.

Create a new category by either right-clicking **Business Process Fragments** in the Fragment Explorer and selecting **New > New Category** or by clicking the **Create New Category** button (.

Procedure

1. Double-click in the name field of the newly added category and enter a name.
2. Do the following:
 - a. Holding down the **Ctrl** key, select the activities and sequence flows in the process that comprise the fragment.
 - b. Copy and paste the fragment into the Fragments view.

Importing Custom Fragments

You can import custom fragments from an archive file or a `.bsProjects` folder.

Procedure

1. Create an archive file that contains the **.bsProjects** folder from your workspace. This folder contains your custom fragments.
2. The recipient of the archive file can then import it. To do this, select **File > Import > General > Existing Fragments Projects into Workspace**, and click **Next**.
3. In the Import dialog box, select the **Select archive file** check box and click **Browse** to select the archive file that contains the fragments you want to import.
4. Click **Finish**. The fragments are imported into the current workspace.

Result

i Note: Alternatively, to import from the **.bsProject** folder, select **File > Import > General > Existing Fragments Projects into Workspace**, and click **Next**. In the Import dialog box, select the **Select root directory** check box and click **Browse** to select the location of the **.bsProject** folder from the file system. Click **Finish**. The fragments are imported into the current workspace.

Resource Patterns and Work Distribution

A number of patterns are available to model how you want work to be distributed to resources.

Resources are the people who carry out the work, and are represented by participants. How these patterns are interpreted depends on your runtime environment, which might not support all the patterns.

Separation of Duties

This pattern stipulates that you want specific tasks executed by different resources.

For example, the resource that prepares a contract is different from the one who witnesses it. There are several ways to specify this pattern:

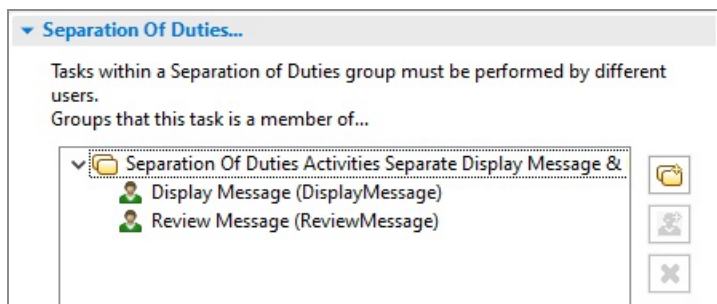
- On the right-click menu in the Process Editor
- On the **Work Resource** tab in the Properties view for the task

- On the **Work Resource** tab in the Properties view for the process

Specifying Separation of Duties

Procedure

1. With the process open in the Process Editor, select the user tasks that you want performed by separate resources (use the Ctrl or Shift keys to select multiple tasks).
2. With the cursor on one of the selected user tasks, right-click and select **Resource Patterns > Create a Separation of Duties Group**.
3. Use the **Work Resource** tab in the Properties view to see separation of duties groups:



If you are viewing the **Work Resource** tab from the process level, it shows all separation of duties groups that are defined in the process.

If you are viewing the **Work Resource** tab from the user task level, it shows all separation of duties groups in which the selected user task is a member.

Note: If you are viewing the **Work Resource** tab for an individual user task, rather than for the process, you can click the **See all task groups(s) in the process** link to display the **Work Resource** tab for the process, which shows all separation of duties groups for the process.

Specifying Separation of Duties from the Work Resource Tab

The [Specifying Separation of Duties](#) topic describes how to create a separation of duties group by selecting multiple user tasks in the process. Separation of duties groups can also be created, and modified, using the **Work Resource** tab from either the process level, or the user task level.


Procedure

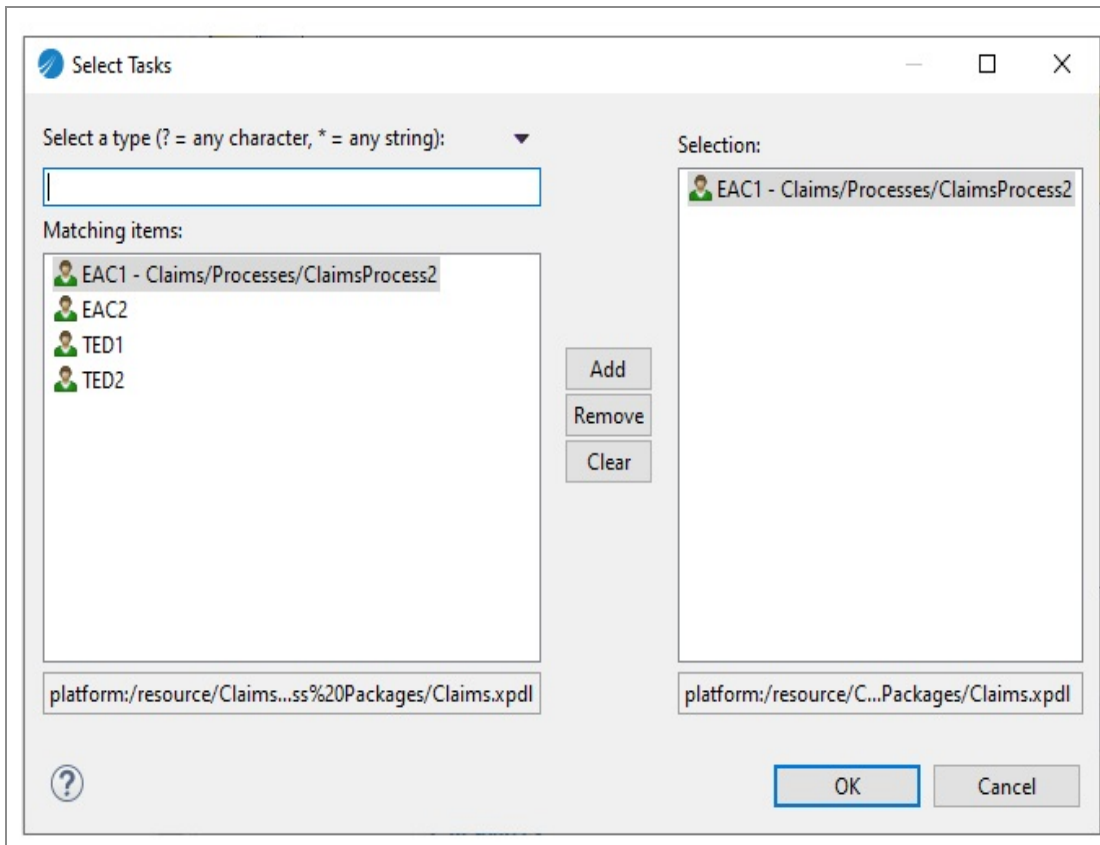
1. Select the process, or a user task in the process.


2. On the **Work Resource** tab, expand the **Separation of Duties** section.


If you are viewing the **Work Resource** tab from the process level, it shows all separation of duties groups that are defined in the process.

If you are viewing the **Work Resource** tab from the user task level, it shows all separation of duties groups in which the selected user task is a member.

3. To create a new separation of duties group, click , then use the Select Table dialog box to specify the user tasks in the group by moving the desired tasks from the left pane to the right pane, then click **OK**.



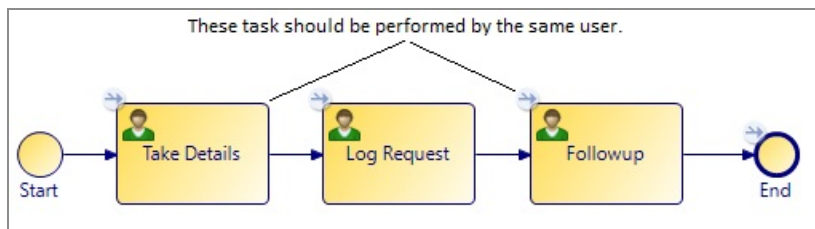
4. To modify the user tasks in an existing separation of duties group, select the group in the **Separation Of Duties** pane, click , then add or remove user tasks from the group using the Select Tasks dialog box.

5. To delete an existing separation of duties group, select the group in the **Separation Of Duties** pane, then click .

Retain Familiar

Retain Familiar stipulates that you want a specific task to be executed by the same resource that executed a previous task in the same process instance. For example, the resource that handles the initial customer contact is the same one that handles the follow-up call.

Consider the following process:



In this example, all three tasks are assigned to the same organization unit. The **Take Details** task is allocated to a resource from the organization unit. Because the **Followup** task is in the same Retain Familiar group as the **Take Details** task, they are allocated to the same resource as well.

i Note: This pattern specifies that you would ideally like tasks to be executed by the same resource. However, this is not always possible in the runtime environment. For example, it might be that by the time the later tasks in a retain familiar group are reached, the original participant is unavailable or not in the participant set. In such cases, the work item is delivered as normal according to the distribution strategy.

If you *delete* a resource, then Retain Familiar does not work as the resource is no longer available, but the work item remains offered to the participant(s) defined, and you might receive an error message. If you *remove* a resource from a position or group, Retain Familiar still works as the task is delivered to the resource even though they are no longer a member of the original position or group.

When there are parallel threads, it is possible for the second task to become active before the other has been completed. In such case, Business Studio do not know who the "preferred individual" is and therefore, cannot validate the pattern.

If the pattern is broken for any reason the 'familiar' resource becomes the most recent resource used. For example, resource A performs the first step in a process, but is not available when the second task is allocated, so the task is allocated to resource B. When the third task is allocated, it is allocated to resource B, who has now become the 'familiar' resource.

There are several ways to specify this pattern:

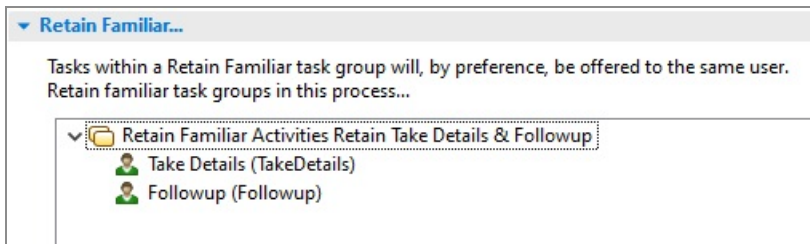
- On the right-click menu in the Process Editor
- On the **Work Resource** tab in the Properties view for the user task
- On the **Work Resource** tab in the Properties view for the process

Specifying Retain Familiar

Procedure

1. With the process open in the Process Editor, select the user tasks that you want performed by the same resource (use the Ctrl or Shift keys to select multiple tasks).
2. With the cursor on one of the selected user tasks, right-click and select **Resource Patterns > Create Retain Familiar Group**.

3. Use the **Work Resource** tab in the Properties view to see retain familiar groups:



If you are viewing the **Work Resource** tab from the process level, it shows all retain familiar groups that are defined in the process.

If you are viewing the **Work Resource** tab from the user task level, it shows all retain familiar groups in which the selected user task is a member.

Note: If you are viewing the **Work Resource** tab for an individual user task, rather than for the process, you can click the **See all task groups(s) in the process** link to display the **Work Resource** tab for the process, which shows all retain familiar groups for the process.

Specifying Retain Familiar from the Work Resource Tab


The [Specifying Retain Familiar](#) topic describes how to create a retain familiar group by selecting multiple user tasks in the process. Retain familiar groups can also be created, and modified, using the **Work Resource** tab from either the process level, or the user task level.

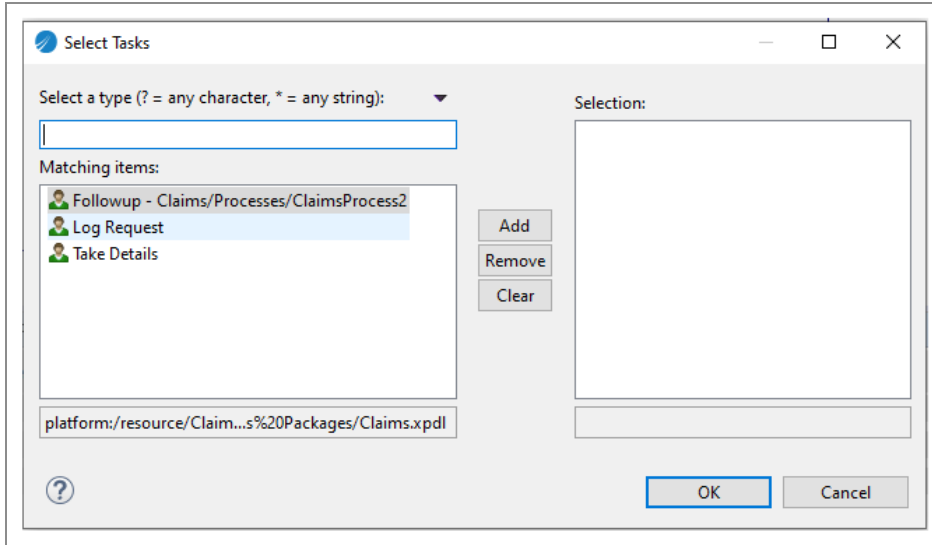
Procedure



1. Select the process, or a user task in the process.
2. On the **Work Resource** tab, expand the **Retain Familiar** section.

If you are viewing the **Work Resource** tab from the process level, it shows all retain familiar groups that are defined in the process.

If you are viewing the **Work Resource** tab from the user task level, it shows all retain familiar groups in which the selected user task is a member.

3. To create a new retain familiar group, click , then use the Select Table dialog box to specify the user tasks in the group by moving the desired tasks from the left pane to the right pane, then click **OK**.



4. To modify the user tasks in an existing retain familiar group, select the group in the **Retain Familiar** pane, click , then add or remove user tasks from the group using the Select Tasks dialog box.
5. To delete an existing retain familiar group, select the group in the **Retain Familiar** pane, then click .

Chained Execution

Chained Execution allows you to specify that a user automatically starts the next work item in the same process instance as soon as the previous one has been completed.

To set up a chained execution pattern, you refactor the required user tasks as an embedded sub-process, and mark the sub-process to use chained execution.

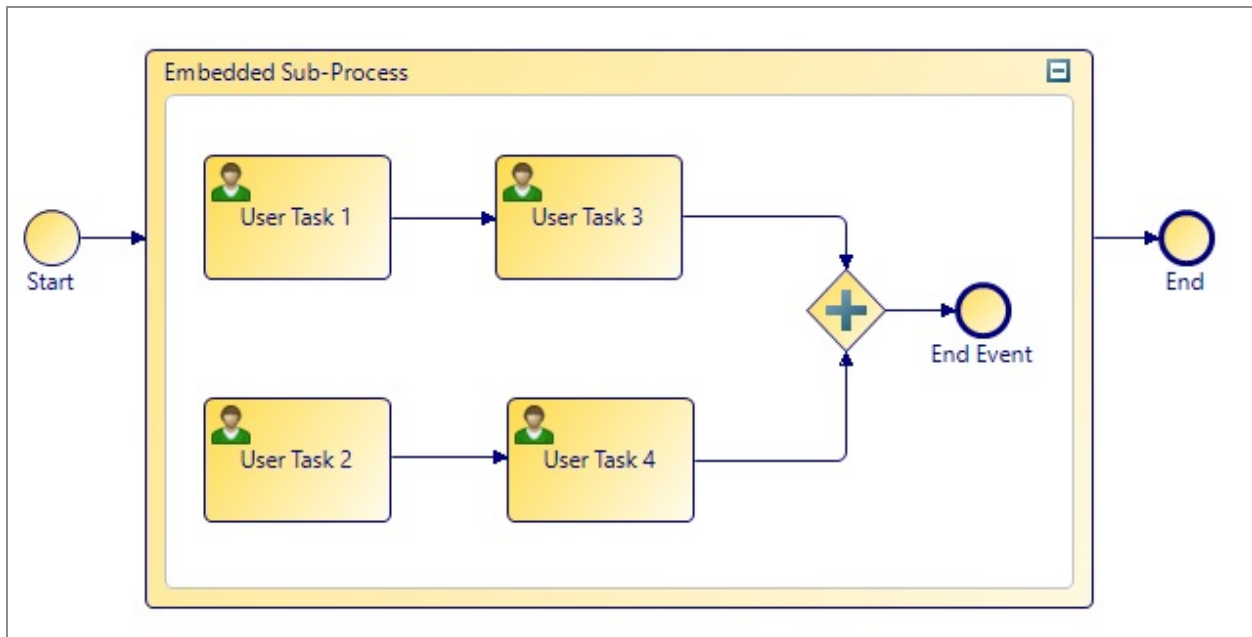
User tasks that are to be chained together must meet the following requirements:

- They can be assigned to different Participants, but they still are offered to the same user *if* that user qualifies under the participant definitions for each task. For example, if one task is assigned to a position and the second to a group, a user who both holds that position and belongs to that group can be given both tasks. If the user is not a member of the group, the second task is offered to each of the group members.
- They must have **Offer to All** as their **Distribution Strategy** (see [Offering and Allocating Work](#)).

Chained Execution- Multiple Parallel Paths in a Chaining Group

When multiple parallel paths exist in a chaining group, items are chained in the order they are scheduled, and not the order of process flow.

For example:



In the example above, User Task 1 or User Task 2 would be performed first. Both would appear in the relevant user's work list. If User Task 1 was opened first, then on completion, User Task 2 would be performed (it would be the only user task in the chained group available at that point). It is likely that while User Task 2 is completed that User Task 3 would be scheduled. This would be performed and then User Task 4 would be performed last. The user tasks would therefore be performed in scheduled order and not according to the connections between the user tasks.

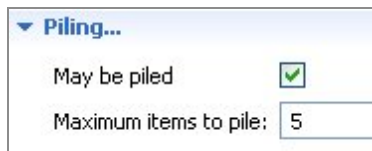
Piling

Specifying that a user task can be piled means that multiple instances of that user task in a user's work queue is presented to the user in sequence (in preference to other work items).

Piling allows you to specify that a particular participant completes work items that relate to a specific task, sequentially. The work items might be in different processes. Once the

work item is completed, if another work item that corresponds to the same task is present in the user's work list, it is immediately started. The benefit of this is that the same work is given to a specific resource who then gains experience in processing this particular task.

To specify piling on a work item, select the **might be piled** check box on the **Work Resource** tab and specify the maximum number of items to pile. For example:



▼ Piling...

May be piled ☒

Maximum items to pile: 5

i Note: The order in which piled items are presented in the user's work queue is based on the sort order of the work list. Setting a default sort order dictates the order of the work list and, therefore, the order of piled items.

The order of piled items cannot be configured with Work Views.

Distribution Strategy

Set the distribution strategy for the task to offer or allocate the work to a user:

i Note: The exact method with which work items are offered and allocated might differ, depending on the runtime environment.

- **Offer To All:** Select this option to specify that you want *all* users that match the participant definition to have the opportunity to accept or decline the work item. For example, if there is a claims handler organizational entity (such as a group), the work item is offered to all users in that group. Once a user opens the work item, it is allocated to them and removed from the work lists of other users in that group.
- **Offer To One:** Select this option to specify that you want only *one* user that matches the participant definition to have the opportunity to accept or decline the work item. If the user declines the work item, it is offered to another user that matches the participant definition.
- **Allocate To One:** Select this option to automatically allocate the work item to a user that matches the participant definition.
- **Allocate To Offer-set Member:** Select this option to allocate the work item to a

specific user who is a member of the offer set defined by the user task's participant definition. A Performer Field must also be specified with this option, which the process must populate with the GUID of the specific member of the offer set to whom the work item should be allocated.

Sub-Processes

Some activities can contain further steps, or sub-processes. There are three types of sub-process: embedded, reusable and event.

Note: A sub-process that is embedded can be re-factored into a reusable sub-process and vice versa. For more information see [Refactoring Sub-Processes](#).

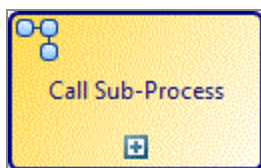
Call Sub-processes

To facilitate the reuse of process components, an activity (or several activities) can call another process as a sub-process, and those sub-processes can be reused from call sub-process activities in many processes.

The sub-process could be a process that you have already created, or you can refactor activities in your current process into a call sub-process activity that calls a new sub-process.

See [Refactoring Sub-Processes](#).

Activities that call a sub-process look like this in the Process Editor:



Click the plus sign (+) in the activity to view the sub-process.

An activity of this type defines a call-out to another process:

- The called process exists as a separate process from the parent process, and because of this it can be started from other processes.
- The called process does not have access to data fields and parameters of the calling

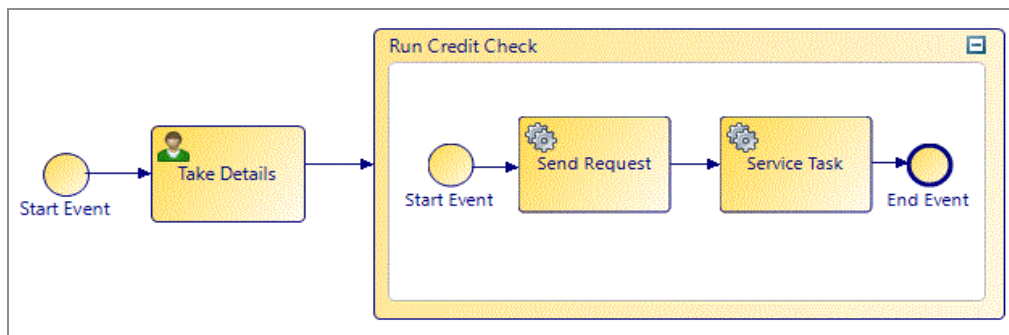
process and package. For this reason, data mapping must be performed to and from the called process.

When a process is canceled, any sub-processes that the parent process created are canceled first. A sub-process canceled by a parent process can execute a cancellation event handler flow to perform any necessary compensatory actions.

Compensatory actions are used to 'undo' or perform appropriate remedial action for partially completed sub-processes.

Embedded Sub-processes

An embedded sub-process is one that is fully contained within the parent process; it does not exist as a separate process.



In this example, Run Credit Check is an embedded sub-process. This implies that running a credit check is an activity that is not needed by other processes. If you subsequently decide that you want to be able to run a credit check from within other processes, you can expose the embedded sub-process as a reusable sub-process by refactoring it. See [Refactoring Sub-Processes](#).

An embedded sub-process has the following characteristics:

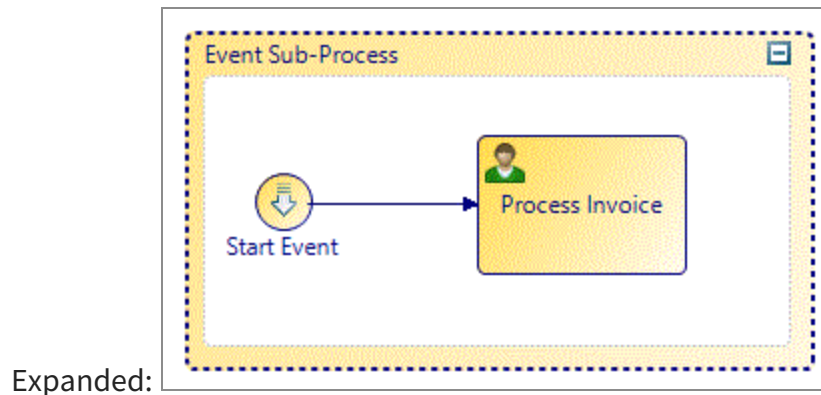
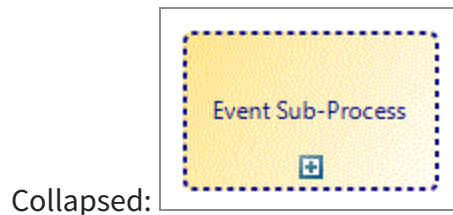
- It is fully contained within the parent process, and is executed within the parent process.
- Activities within the embedded sub-process have access to the same data fields and parameters as the parent process and package.
- No data mapping is required.
- It cannot contain lanes and pools.

To create an embedded sub-process, refactor one or more objects in your process as described in [Refactoring Sub-Processes](#).

i Note: If you want to use the chained execution resource pattern, you can do so by selecting the **Chained Execution** check box in the Properties view for the embedded sub-process. For more information, see [Chained Execution](#).

Event Sub-processes

An event sub-process executes an internal sub-process when an event is triggered. You can use event sub-processes within business processes and pageflows. Click the plus sign (+) in the activity to view the sub-process.



Refactoring Sub-Processes

You can refactor activities into a reusable sub-process or an embedded sub-process.

Refactoring a sub-process allows you to do the following:

- Create a new embedded sub-process from selected objects.
- Create a new call sub-process activity from selected objects and replace the selected objects with a call to the newly created sub-process.
- Convert an existing embedded sub-process (and its contents) into a call sub-process activity and replace it with a call to the newly created sub-process.

- Select an existing single event handler activity or a whole event handler flow and refactor it into an event sub-process.
- Replace a call sub-process activity with a copy of the content of the sub-process.

Synchronous and Asynchronous Sub-Process Invocations

Sub-process invocations can be performed synchronously or asynchronously in relation to the invoking process flow.

Use **synchronous sub-process invocation** when you want the process to wait for the sub-process to complete before the main process can proceed.

Use **asynchronous sub-process invocation**:

- when you want sub-processes to run separately from the main process, and the sub-process does not need to complete for the main process to complete. The invoking process can complete faster without waiting for the sub-process.
- when you want to run sub-processes in parallel with the main process without sacrificing migratability, and when you do not want to hold up the main process.

i Note: Use asynchronous attached sub processes when you want to run sub-processes in parallel with the main process without sacrificing migratability, and when you do not want to hold up the main process.

Choose the invocation mode of a Call Sub-Process activity from the Properties tab, by selecting **General > Lifecycle > Invocation Mode**):

- **Synchronous:** the invoking process flow waits for completion of sub-process.
- **Asynchronous Attached:** The invoking call activity completes immediately and flow continues. The sub-process is canceled with parent process, the parent process is not considered complete until sub-process completes.
- **Asynchronous Detached:** The invoking call activity completes immediately and flow continues. The sub-process lifecycle is independent of the invoking process.

Pageflow Processes (including Business Service, Case Action) can invoke Business Processes in asynchronous detached mode).

Creating Call Sub-Processes

There are several different ways of creating a call to a sub-process:

You can create a call to a sub-process:

- By refactoring objects in your process (see [Creating a New Embedded Sub-Process](#)).
- By dragging a process from the Project Explorer and dropping it onto your process ([Creating a Call Sub-Process Activity Call Using drag-and-drop](#)).
- Using the call sub-process activity tool from the palette (see [Refactoring Activities into a Sub-Process](#)).

i Note: If you modify a sub-process (for example, by adding a parameter) and that sub-process is referenced by a main process in a different package, validation in the main process does not occur until you save the sub-process package.

Creating a Call Sub-Process Activity Call Using drag-and-drop

Procedure

1. Expand the Project Explorer to locate the process that you want to be the sub-process.
2. Click the intended sub-process, holding down the mouse button, drag the pointer to the calling process (open in the Process Editor), and release the mouse button.

i Note: You can select multiple processes for drag-and-drop operations using the Ctrl key.

3. If you are dropping more than one process, a menu is displayed with two options:
 - **Create Sub-Process Task Sequence:** Selecting this option allows you to create sub-process tasks connected by sequence flow. A dialog box is displayed to allow you to control the order of the tasks: Use the **Move Up** and **Move Down** buttons or drag-and-drop to control the order of the tasks. When you are

finished, click **OK** to place the tasks.

- **Create Unsequenced Sub-Process Tasks:** Selecting this option places the tasks in the process without a connecting sequence flow.

Creating a call sub-process call using the palette

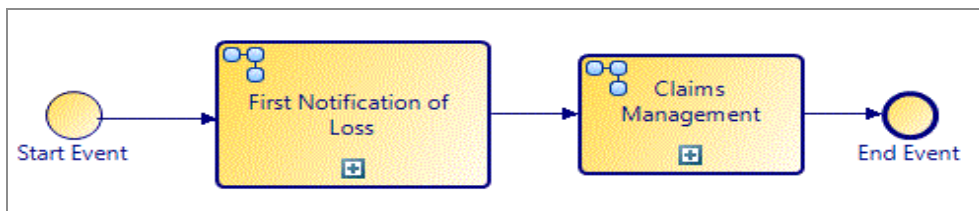
Procedure

1. In the Process Editor, select the **call sub-process** tool.
2. Click in the process where you want to place the activity that calls the sub-process.
3. On the Properties view for the activity, browse for the process you want to call as a sub-process.

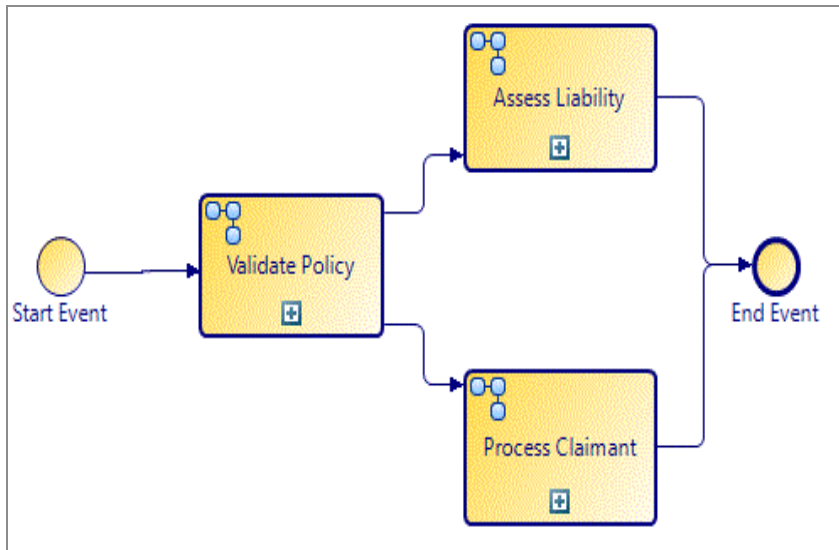
Note: If the process you select is not in the current project, you are prompted to create a project reference.

Call Sub-Process Example

The following example demonstrates how a top-level process could use sub-processes in an insurance environment.



Each of the activities in the process calls sub-processes that are executed in the runtime environment. For example, the sub-process called from the Claims Management activity could look like this:



Note the following about this sub-process:

- The Validate Policy sub-process can be re-used. For example, when a customer calls to renew their policy, the first step of the Renewal Process could be to confirm that they have an existing policy. The Validate Policy Activity could be re-used for this purpose in the Renewal process.
- The Process Claimant activity has a **Loop** Activity Marker, that indicates that the activity is repeated for each Claimant.
- The *actual* sub-process called by the Assess Liability activity is a manual Process. However, the *to be* sub-process (planned for the future), could be implemented as an automatic process that consists of a series of questions used to determine liability.

Expanding a Sub-Process

You can expand a sub-process associated with an activity.

An activity that calls a sub-process looks like this:



To expand the sub-process associated with this activity, either click the plus symbol, or do the following:

Procedure

1. In the Process Editor, select the activity that calls the sub-process.
2. In the Properties view, select the **General** tab.
3. Click **Open Sub-Process**.
4. The sub-process opens in the Process Editor.

Configuring a Sub-Process


You can configure the sub-process invocation mode, and whether it should execute immediately or be queued.

Use the Lifecycle option, on the **General** tab in the **Properties** view to select the invocation mode from the following:

- Synchronous
- Asynchronous Attached
- Asynchronous Detached

For definitions of the different types of invocations, see [Synchronous and Asynchronous Sub-Process Invocations](#). There are a number of restrictions that are indicated via validation problem markers.

You can access the process-id of the new process instance as a sub-process output mapping. This is mappable to a string-typed field or parameter.

 **Note:** You cannot map from sub-process output parameters when invoking asynchronously because the invoking process does not wait for the sub-process to complete and return.

Use the Lifecycle option, on the **General** tab in the **Properties** view to select whether the sub-process should execute immediately or whether its start request should be queued.

- Select **Schedule Start Request** to set the initial priority for the queued sub-process start request and the tasks within that using the options in the **Priority** radio buttons.
- Select **Start Immediately** to set the priority of the task within the process in the process **Resource** tab.

- Specify whether sub-process instances should suspend and resume when the parent process is suspended or resumed by selecting or deselecting the **Suspend/Resume Sub-Process With Parent Process** check box.

▼ Lifecycle...

Invocation Mode: ☒ Synchronous ☐ Asynchronous Detached ☐ Asynchronous Attached

Start Scheduling: ☒ Start Immediately ☐ Schedule Start Request

☐ Inherit From Parent process

☐ Low (100)

☐ Normal (200) (default for main processes)

☐ High (300) (prioritize over main processes)

☐ Higher (400)

☐ Custom (100-400):


Suspension: ☒ Suspend/Resume Sub-Process With Parent Process

Invoking a Sub-Process Asynchronously

You can create asynchronous sub-processes from a process to allow you to run sub-processes independent of the main process, when you are not concerned if they complete before the main process.

See [Synchronous and Asynchronous Sub-Process Invocations](#).

Procedure

- Create the main process containing a call sub-process activity.
- On the Properties for the call-sub-process activity, use the Lifecycle option, on the **General** tab in the **Properties** view to select **Asynchronous** invocation mode.
- Click the  sign on the call sub-process activity to open the asynchronous sub-process in a new window.
- Create the content of your asynchronous sub-process.

Embedded Sub-Processes

To create an embedded sub-process, refactor one or more objects in your process.

See [Creating a New Embedded Sub-Process](#).

i Note: If you want to use the chained execution resource pattern, you can do so by selecting the **Chained Execution** check box in the Properties view for the embedded sub-process. For more information, see [Chained Execution- Multiple Parallel Paths in a Chaining Group](#).

Adding Local Data Fields

You can add data fields that are local to a sub-process (they are not used in the process that contains the embedded sub-process).

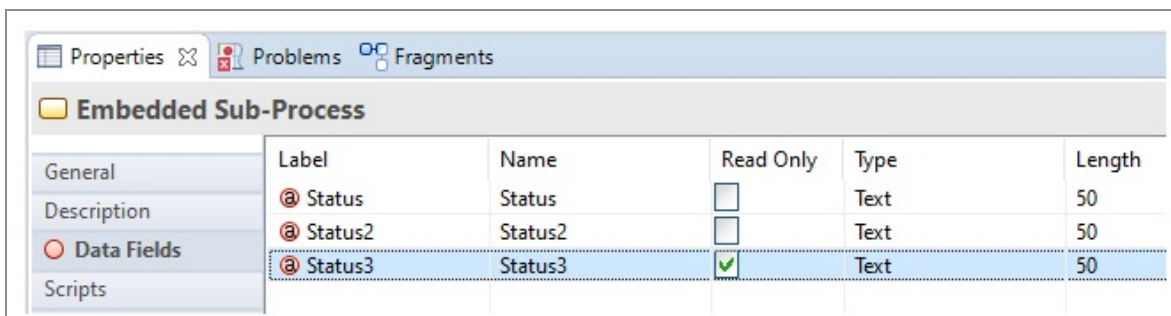
By default, activities in the embedded sub-process have access to all local data and process data. However, using the **Interface** tab, a subset of data can be selected.

Procedure

1. Click the **Data Fields** tab in the Properties view for the embedded sub-process.
2. At the right of the **Data Fields** tab, click the plus sign to add local data fields. The properties of the data fields that you create are the same as for process data (see [Adding Data Fields or Parameters to a Package or Process](#)).

i Note: Local data fields are not displayed in the Project Explorer.

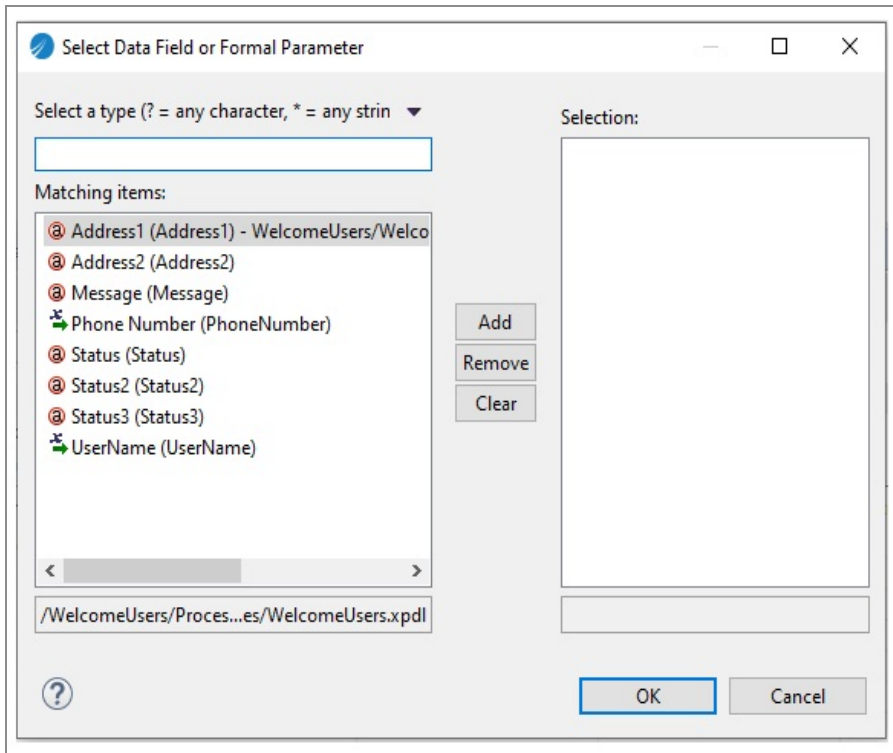
For example:



	Label	Name	Read Only	Type	Length
General	@ Status	Status	<input type="checkbox"/>	Text	50
Description	@ Status2	Status2	<input type="checkbox"/>	Text	50
Data Fields	@ Status3	Status3	<input checked="" type="checkbox"/>	Text	50
Scripts					

In this example, the data fields **Status**, **Status2**, and **Status3**, are local to the embedded sub-process.

On the **Interface** tab for activities within the embedded sub-process, you can access the local data as well as the process data:



In this case the address data fields (process data) and the local data defined on the **Data Fields** tab are both available.



Tip: You can quickly create an embedded sub-process that has local data by using the fragment **BPMN Process Fragments > Basic Fragments > Embedded Sub-Process with Data Fields**. For more information about fragments, see [Creating Call Sub-Processes](#).

Local Data Fields in Loops

When using local data fields in loops, the value taken by the local data field during each iteration depends on the type of loop.

Multi-instance loop

Each instance of a multi-instance loop has a separate instance of the local data field, allowing parallel loops to execute without interfering with each other. However, the local data fields cannot be used in the loop expression.

Standard loop

The local data field is shared between all instances of a standard loop. It can be used to pass data between instances, and can be used in the loop conditional script.

Note: You cannot initialize a local data field prior to the first "Before" condition test.

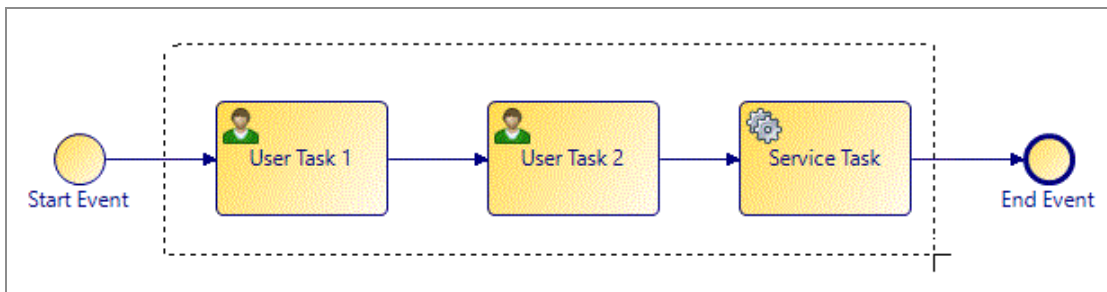
Note: For both multi-instance and standard loops, the task initiate script runs just before executing the task for each iteration of the loop.

Creating a New Embedded Sub-Process

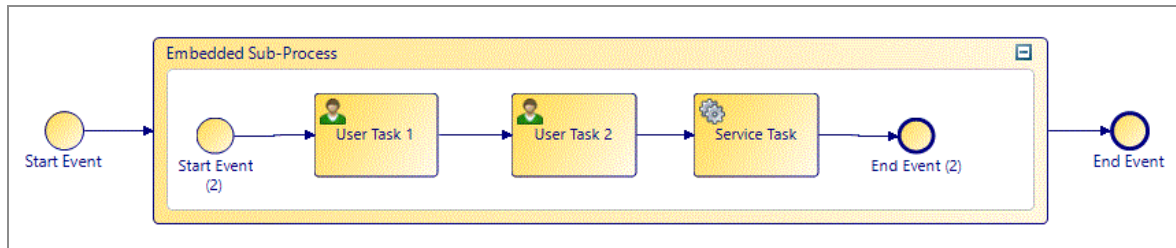
You can create a new embedded sub-process by refactoring existing objects into an embedded sub-process.

Procedure

1. Select the objects that you want to put in the embedded sub-process. For example:



2. Right-click and select **Refactor > Move into New Embedded Sub-Process**.
3. Complete the dialog box as follows:
 - a. Enter a name for the embedded sub-process that you want to create.
 - b. Select the **Insert start event in new sub-process** and **Insert end event in new sub-process** check boxes to control whether start and end events are added to the refactored sub-process.
4. Click **Finish**. The objects that you selected are placed within a new embedded sub-process (with start and end events if those options were selected):

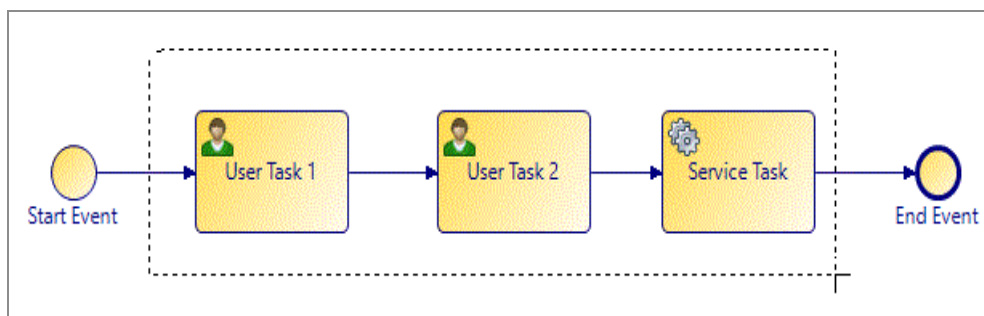


Refactoring Activities into a Sub-Process

You can create a new call sub-process activity and sub-process by extracting existing objects.

Procedure

1. Select the objects that you want to put in the sub-process. For example:



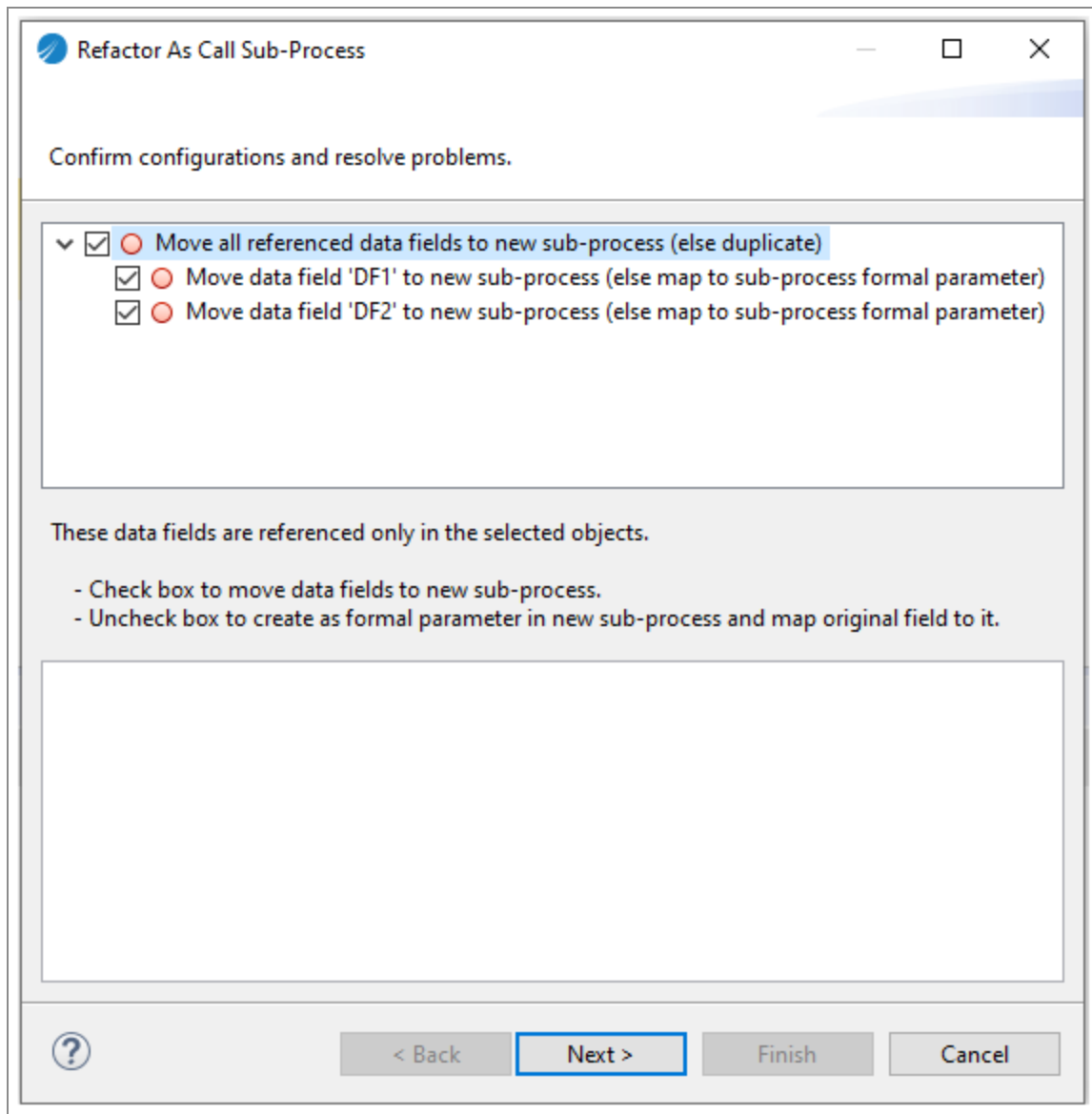
In this example, the two user tasks have the following parameters (two data fields and one formal parameter):

Task			
General	Visibility: <input checked="" type="radio"/> Private <input type="radio"/> Public	Reschedule work item: <input type="checkbox"/> Overwrite data already modified in work item.	
Description	<div>Parameters</div> Select a subset of data that is accessible for this activity. <input type="checkbox"/> No interface data association required.		
Interface	Process Data Name	Mode	Mandatory
Data Fields	Param1	In	<input checked="" type="checkbox"/>
Work Resource	DF2	In / Out	<input type="checkbox"/>
Scripts	DF1	In / Out	<input type="checkbox"/>
Data References			
Appearance			

The user tasks also have a participant associated with them.

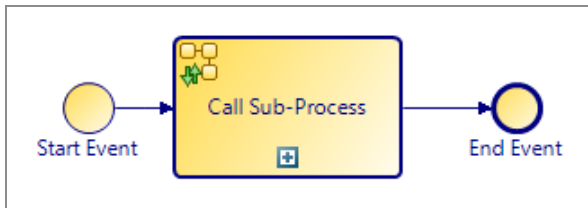
2. Select all three tasks.

3. Right-click and select **Refactor > Extract into New Sub-Process**.
4. Because the data fields and participants are only referenced in this activity (and not used by any other activities), the following message is displayed:

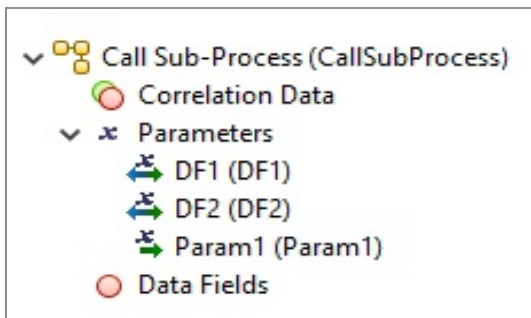


- If you select the data fields, they are moved into the sub-process.
 - If you do not select the data fields, they are created as formal parameters in the sub-process (and mapped to those formal parameters).
5. Complete the dialog box as follows:

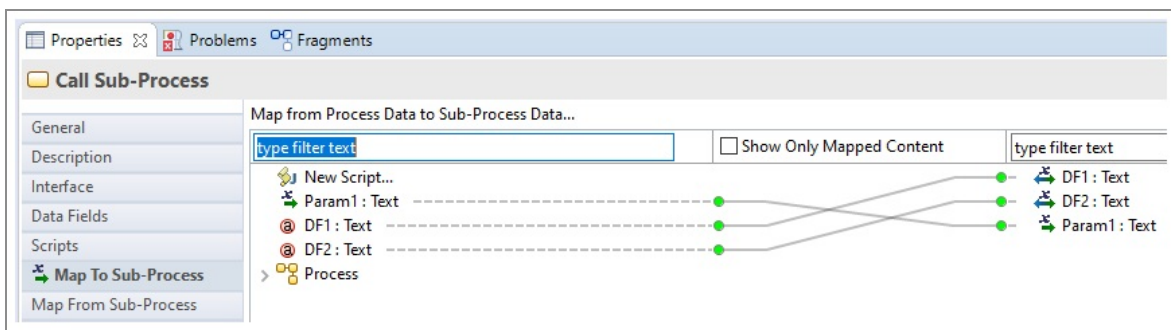
- a. Enter a name for the sub-process that you want to create.
 - b. Select the **Insert start event in new sub-process** and **Insert end event in new sub-process** check boxes to control whether start and end events are added to the refactored sub-process.
6. Click **Finish**. The objects that you selected are copied to the new sub-process and the selected objects are replaced with a task that calls the call sub-process.

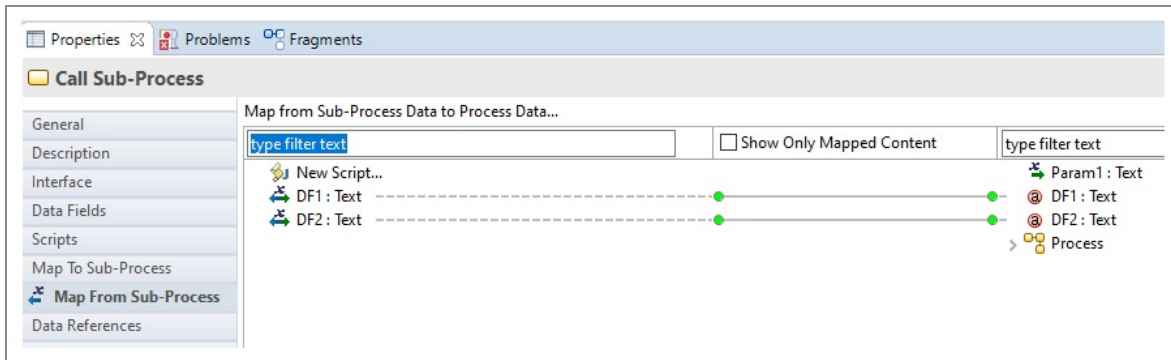


If you chose to create formal parameters for the data fields, you can see them in the Project Explorer, under the sub-process:



In addition, if you click the task in the parent process that calls the sub-process and go to the Properties view, you can see the mappings that have been created between data fields and formal parameters of the parent process and the formal parameters created in the sub-process. For example:





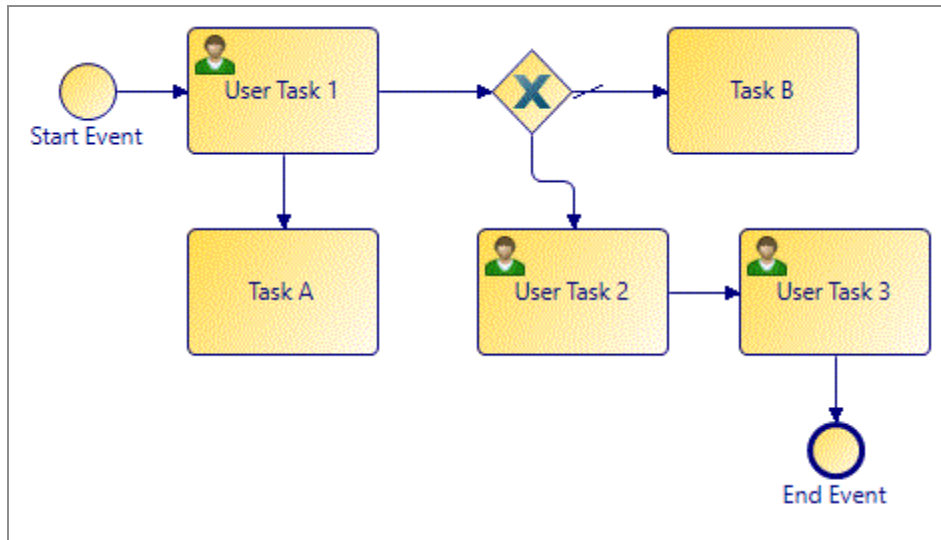
Notes on Refactoring Objects into Sub-Processes:

- Formal parameters in the parent process are created as formal parameters in the sub-process. They are mapped according to their mode as defined in the process API.
- If a data field is referenced only in the selection that you are refactoring, you have the option of moving or copying it. Data fields that are *copied* into the sub-process are created as formal parameters and mapped. Data fields that are *moved* into the sub-process are created as data fields. By default, data fields are *moved*. In order to *copy* them as formal parameters, you must uncheck the selection boxes on the dialog box.
- If a data field is referenced in one or more of the tasks you select for refactoring but is also referenced elsewhere, it is *automatically copied* into the sub-process as a formal parameter.
- A mapping is created between data fields and formal parameters of the sub-process and any corresponding formal parameters that are created in the sub-process. You can view this mapping by selecting the task that calls the sub-process and clicking the **Map To Sub-Process** and **Map From Sub-Process** tabs in the Properties view.

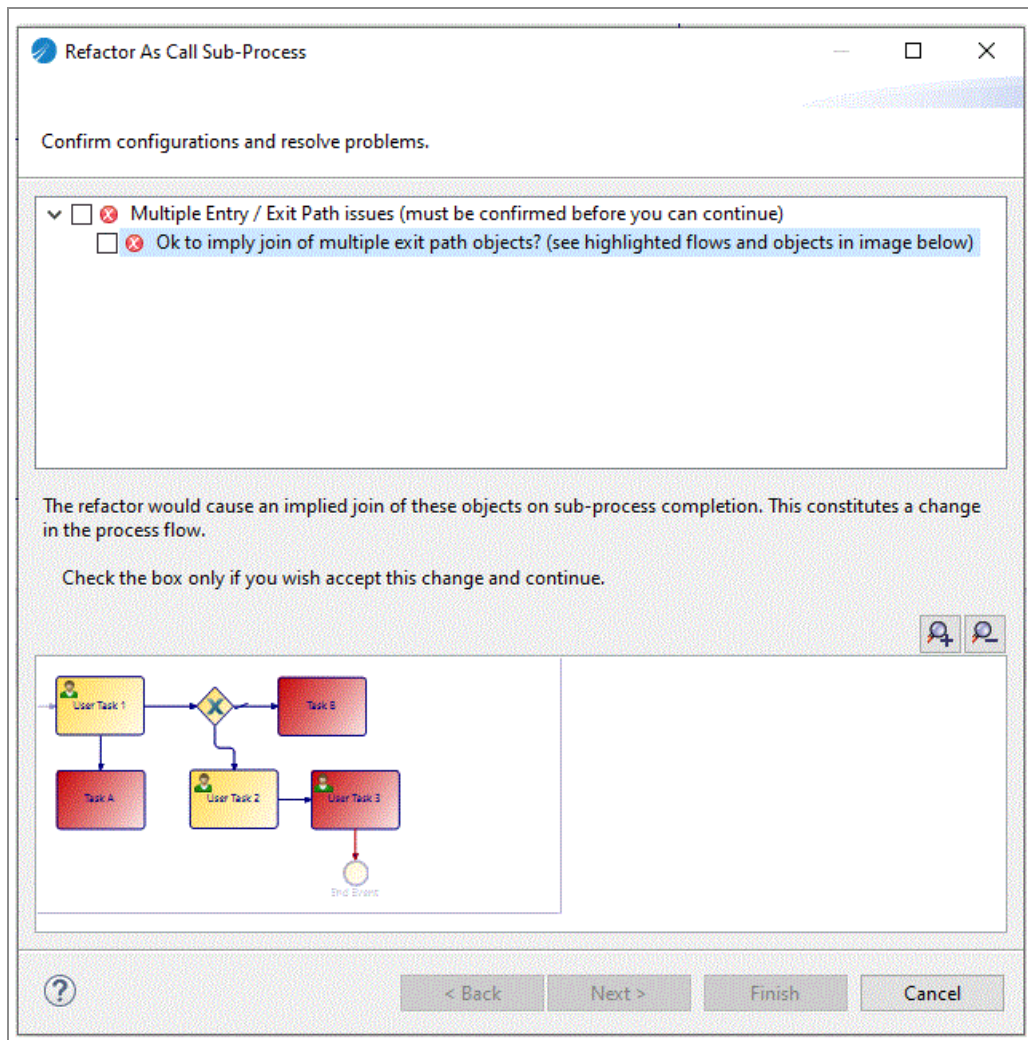
Changes in Process Logic:

When refactoring objects into a call sub-process, you can potentially change the logic of the process flow.

Consider the following process:



In this process, flow proceeds from **User Task 1** to the gateway without necessarily waiting for **Task A** to finish. If **User Task 1** and **Task A** are refactored into a sub-process, an end event is inserted into the sub-process, effectively synchronizing the flow. TIBCO Business Studio prompts you to confirm this change to the process:



Inline Sub-Process Content


This option creates an embedded sub-process from a call sub-process activity. The reason for making a call sub-process inline is that in some cases the tasks in the sub-process execute relatively quickly and the overhead of invoking a sub-process can be comparatively high.

Refactoring a Call Sub-Process into an Embedded Sub-Process

You can refactor a call sub-process activity into an embedded sub-process.

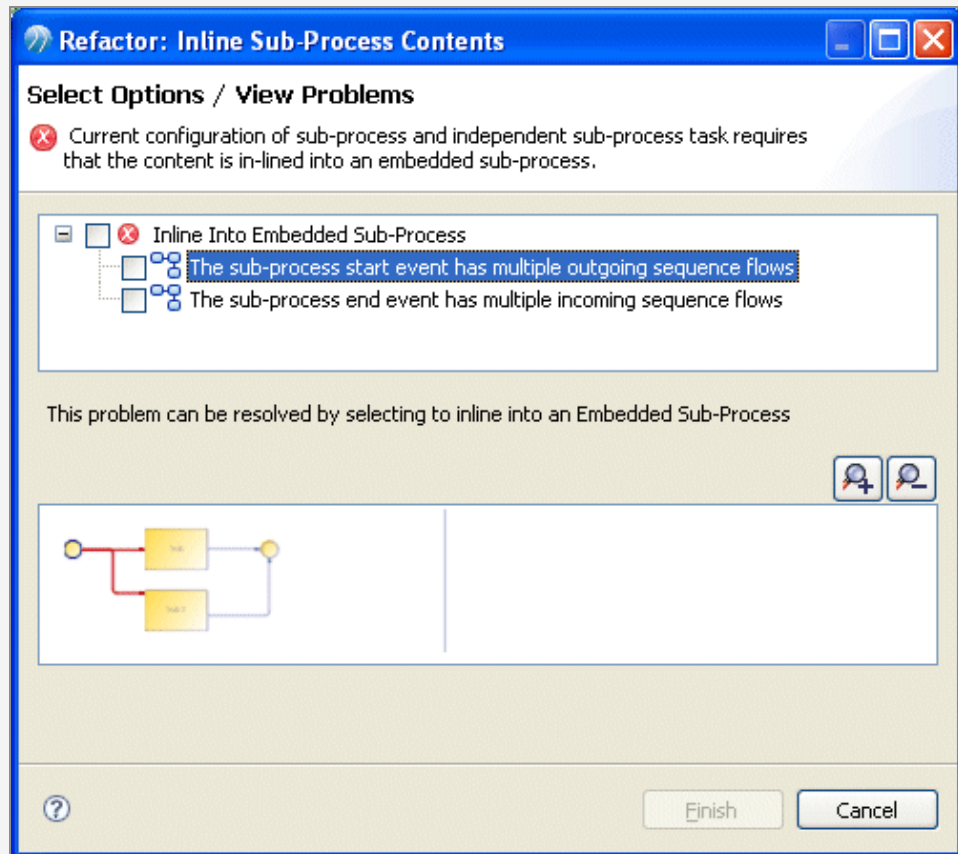
Procedure

1. Right-click the call sub-process activity, and select **Refactor > Inline > Sub-Process Content**.

 **Note:** Only a single level of a process hierarchy can be made in line at a time. If the sub-process you want to make inline contains a sub-process task, after refactoring, it is brought into the embedded sub-process as a sub-process task. (the refactor does not extend into additional levels of the process hierarchy).

2. The sub-process called by the selected task is analyzed, and a dialog box is displayed with the results:
 - If there are no problems with the refactoring operation, a dialog box is displayed. Select **Inline into Embedded Sub-Process** and click **Finish**.
 - If there are problems with the refactoring, but the problems can be resolved by making the sub-process inline, a dialog box is displayed showing the problems:

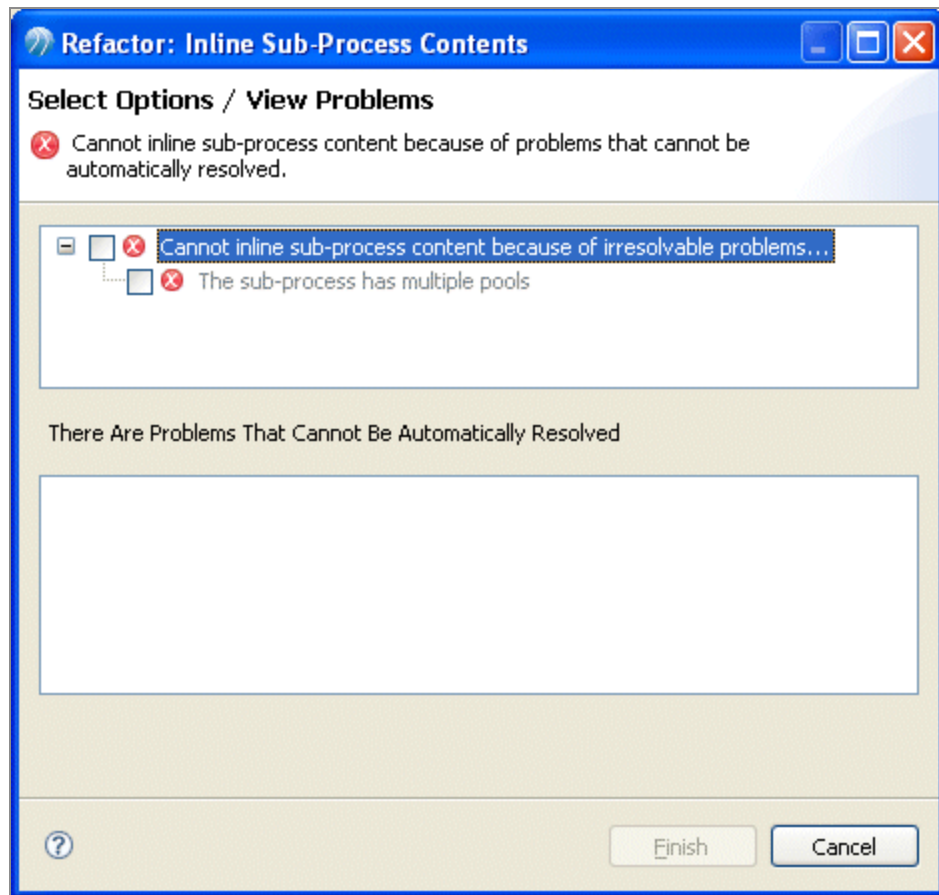
- Note:** The possible problems displayed in this dialog box are similar to those for process package optimization, except that some of these problems that are not resolvable for package optimization are resolvable by refactoring into an embedded sub-process (such as those reported in the previous dialog box).



Also, problems dealing with nesting of sub-processes are not applicable because manual refactoring affects only a single level of sub-process hierarchy at a time.

You can see the location of each problem by highlighting it. If you select all of the problems in this dialog box, the **Finish** button is enabled and you can refactor the sub-process.

- If there are problems that cannot be resolved by refactoring, a dialog box similar to the following is displayed:



Because the errors cannot be resolved, click **Cancel**, resolve the problems manually, and retry the refactoring.

3. If the sub-process was able to be refactored, its contents are placed in an embedded sub-process, however the sub-process that was made inline is not removed.

Result

- i Note:** The following are potential consequences of a refactoring. For more information, see [Inline Sub-Process in Detail](#):
- References to sub-process parameters in the sub-process content are swapped for the calling process data fields that are mapped to them.
 - Sub-process data fields and unmapped parameters are copied up to the calling process and renamed with a sequence number if the data field or parameter already exists in the calling process.
 - Sub-process participants are copied to the calling process (if they do not already exist).
 - Type declarations referenced by data copied up from the sub-process are copied to the package of the calling process *only* if the sub-process is in a different package.
 - If the sub-process implements a process interface (see [Process Interfaces](#)), the start or intermediate events that implement interface-defined events are changed so that they are no longer flagged as such.

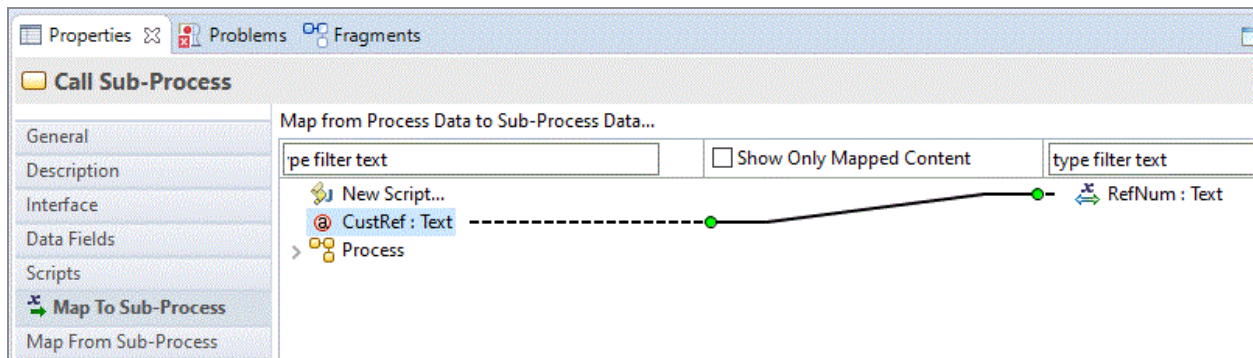
Inline Sub-Process in Detail

This section describes in detail what happens when you use an inline sub-process (including its parameters, participants, and so on).

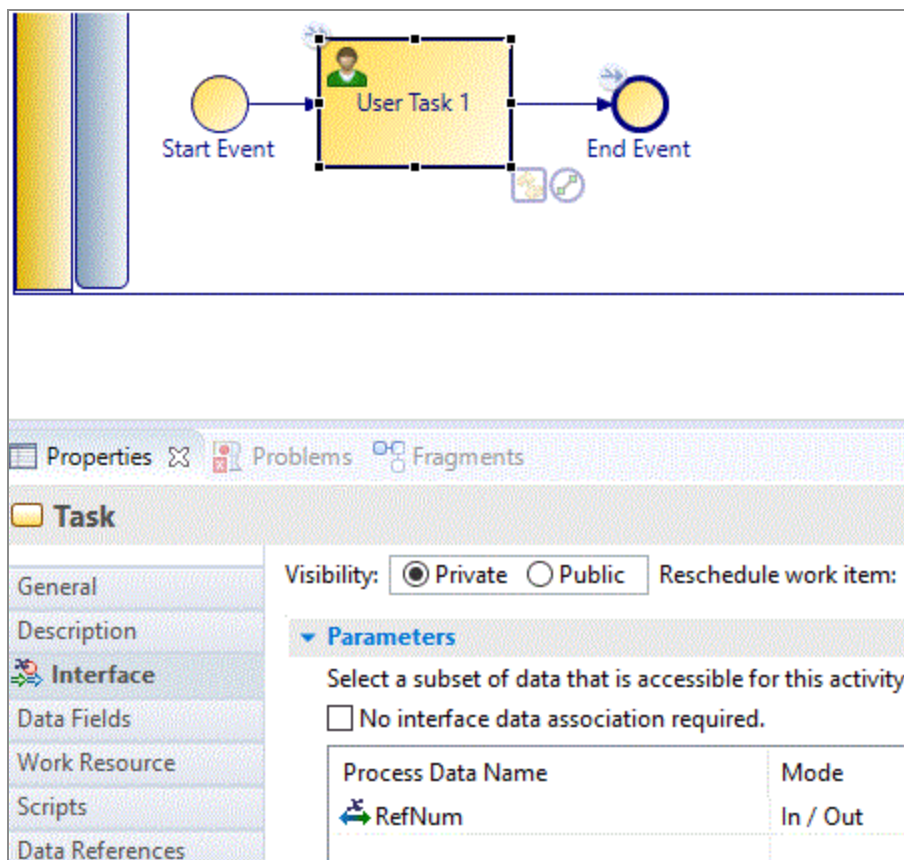
Data Fields and Mapped Parameters

References to parameters in sub-processes are replaced according to the parameter mapping.

For example, consider the parameter mapping to the following sub-process:



The data field **CustRef** in the parent process is mapped to **RefNum** in the sub-process. In the sub-process, the parameter **RefNum** is used by the single user task:



After refactoring as an inline sub-process, the user task is brought into the main process with **RefNum** as its parameter.

i Note: Every effort is made to ensure that when a sub-process is in-lined, it is semantically the same as if it were not in-lined. However, not every potential scenario can be catered for.

For example, if the same calling-process field is used in the parameter mappings for two parallel call sub-process activities, the calling-process field is now used in-parallel in the two sets of sub-process contents that were moved up into the calling-process. This might have undesired effects.

Sub-Process Data Fields and Unmapped Formal Parameters

Sub-process data fields and unmapped formal parameters are moved up to the calling process after refactoring. In order to preserve the original semantics of the sub-process, any data fields or unmapped formal parameters in the sub-process that also exist in the calling-process are renamed (each new instance of the same named data is suffixed with a sequence number). Any references to the renamed sub-process data is updated when the sub-process activities and flows are moved up to the calling process.

For example: A sub-process has a data field called CustomerField which its activities use internally. The calling process also has a field called CustomerField which it uses internally.

When the sub-process is in-lined, the sub-process CustomerField is copied into the calling-process as CustomerField2. All references to CustomerField in sub-process activities and flows are replaced with references to CustomerField2.

Multiple Calls to a Sub-Process

This rule also applies when several sub-processes are called from a single process. If the sub-processes have the same data fields names, then each invocation causes separate, sequence-numbered instances of data fields in the calling process when they are made inline.

For example: There are two calls to the same inline sub-process from a single calling process. The sub-process has a field called CustomerField.

The copy of sub-process activities and flows ‘moved up’ in place of one call sub-process activity operate on CustomerField and the other operate on CustomerField2.

Field Name Conflicts

This rule also applies when inline sub-processes are nested, and have conflicting field names.

For example: The calling-process (MainProcess) calls an inline sub-process (SubProcess) which in turn calls a nested inline sub-process (SubSubProcess).

Each process has a field called CustomerField.

- SubSubProcess is in-lined into SubProcess so that its instance of CustomerField becomes CustomerField2 in SubProcess.
- SubProcess now has 2 fields, CustomerField and CustomerField2. When this is in-lined into MainProcess CustomerField in SubProcess is dealt with first (alphabetically) and is therefore renamed as CustomerField2 in MainProcess.
- Now when the SubProcess field CustomerField2 is subsequently copied into MainProcess, a CustomerField2 already exists so it is renamed as CustomerField3.

Sub-Process Participants and Type Declarations

Participants are moved up to the calling process if the calling process does not already have a participant with the same name. If the Participant already exists in the calling process, references to the sub-process participant are exchanged for references to the calling process participant in the sub-process content that is moved up.

Similarly, type declarations (that are referenced by sub-process data fields and unmapped formal parameters) are moved up to the calling process if the calling process does not already have a type declaration with the same name. Otherwise, the calling-process type declaration is used.

Event Sub-Processes

An event sub-process executes an internal sub-process when an event is triggered. You can use event sub-processes within business processes and pageflows.

See also [Event Handlers](#).



Note: Event sub-processes and event handlers provide similar functionality.

Note the following when using event sub-processes:

- A process can contain zero or more event sub-processes of any supported type or configuration.
- Each event sub-process must start with a single start event.
- There is no in-process communication between event sub-processes (sequence flow, signals, and so on).

You can use event sub-processes with the following start events:

- Event Sub-Processes placed in a Business process: Start Request or Signal Event
- Event Sub-Processes placed in a Pageflow : Start Request or Signal Event

Business processes support interrupting/non-interrupting event sub-processes.

i Note: Pageflows support non-interrupting event sub-processes. They do **not** support interrupting event sub-processes.

This is defined on the start event of the event sub-process by checking **Interrupt Process Flow** or **Continue Process Flow**. This affects the process as follows:

Interrupt Process Flow:

i Note: An interrupting event sub-process start event has a solid border.

- The main process flow is suspended until completion of the event sub-process.
- This does not affect the processing and completion of individual activities that are already in progress related to the process instance. However, those activities' outgoing flow is not processed until the event sub-process completes.
- This does not affect the processing of other event handler flows, event sub-processes and ad-hoc activities.
- This does not affect the processing of activities in active reusable sub-processes.
- Incoming events, submits, sub-process completions and so on in the main process are preserved but ignored until the event sub-process has been completed.

Continue Process Flow:

i Note: A non-interrupting event sub-process start event has a dashed border.

- The main process flow is not suspended during processing of the event sub-process.
- All activity can continue as normal.

Dynamic Sub-Processes

Dynamic sub-processes are used when a process (which can be either a business process or a pageflow process) calls one of several sub-processes at runtime, but it is not known at design time which of these sub-processes are called.

The exact sub-process to be called on any given occasion is chosen at runtime, depending on the process data. For example, a corporate HR Department's recruitment process might need to call different sub-processes for determining a candidate's eligibility for employment depending on the country where the candidate is recruited.

In order for the main process to be able to accommodate any of the sub-processes that might be called, the sub-processes must all take and return a common set of parameters, which are known at design-time. This common data is specified by a process interface on which all the sub-processes are based. A call sub-process activity in the main process then specifies a call to the process interface instead of naming an individual sub-process.

Creating Dynamic Sub-Processes

A dynamic sub-process can be created by modifying a call sub-process activity.

Procedure

1. Create a process interface that specifies the start event and its input/output parameters. Each process that is to be invoked from the dynamic sub-process task **must** implement the same process interface. See [Process Interfaces](#).
2. Create one or more call sub-process activities in your process.
3. From the General tab for each call sub-process activity:
 - a. Using the picker, select the process interface created in [step 1](#).
 - b. In the Runtime Identifier Field select a formal parameter or data field using the picker. This must be a text field or an array. Arrays can be used for multi-instance sub-process tasks where potentially different sub-processes are required for each instance of the same task.

<input type="checkbox"/> Is A Transaction
Invocation of Sub-Process: (Open Sub-Process)
Sub-Process location: <input type="text" value="- the same package -"/>
Sub-Process name: <input type="text" value="BookAppointmentInterface (BookAppointmentInterface)"/>
Runtime Identifier Field: <input type="text" value="ProcessIdentifierAr"/>

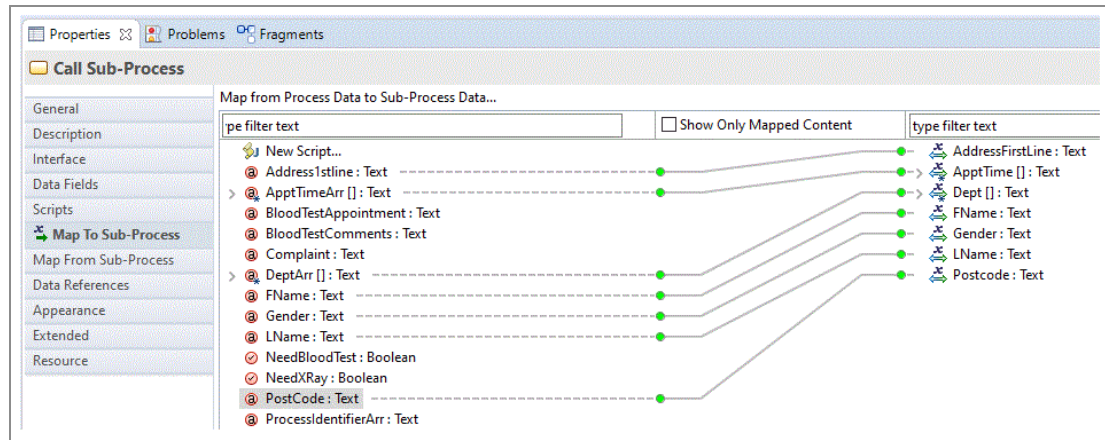
4. Initialize the Runtime Identifier Field. For example, for a multi-instance sub-process task you could create a Script task before it to populate the array data field with a sub-process name element for each task instance. For a single instance sub-process task simply set the runtime identifier field to the required sub-process name in a script prior to the task.

For a sub-process implementation which resides in an external sub-process library., you could use a script something like the script shown in the following illustration:

Script Defined As: <input type="text" value="JavaScript"/>
Describe Task Script:
<pre> if (type.toString() == "One") { runtimeID = "/APPE/Process Packages/APPE.xpdl.APPEProcess" ; } else if (type.toString() == "Two") { runtimeID = "/APPE/Process Packages/APPE.xpdl.APPEProcess2"; } </pre>

5. In the call sub-process activity, map data to and from the called process (to pass data between the process and sub-process):
 - Map the input to the interface in Map To Sub-Process.
 - Map the output from the interface in Map From Sub-Process.

For example:



Automatically Passing Different Data to and from Sub-Process Instances

If the call sub-process activity is multi-instance (or loop) it is possible to automatically pass different data to and from each separate sub-process instance.

Inputting Different Data to Each Sub-Process Instance

Procedure

1. Specify an array data field that matches the type of a non-array sub-process/interface input parameter.
2. In a script prior to the sub-process task, populate this array data field with different data for each sub-process instance. For example:
 - `appointmentTypeArray.set(0, "X-Ray");`
 - `appointmentTypeArray.set(1, "PlasterDept");`
 - `appointmentTypeArray.set(2, "Physio");`
3. In the "Map to Sub-Process" property tab, map the array data field to the non-array sub-process/interface parameter. The sub-process invoked from each instance of the task receives the list element corresponding to the instance index of that task (see 'getActivityLoopIndex()' in [Process Instance Attributes and Methods](#)). For example:
 - The first instance (activity loop index=0) receives the data "X-Ray" into its "appointmentType" parameter.

- The second instance (activity loop index=1) receives the data "PlasterDept" into its "appointmentType" parameter.
- The third instance (activity loop index=2) receives the data "Physio" into its "appointmentType" parameter.

Returning Different Data from Each Sub-Process Instance

Procedure

1. Specify an array data field that matches the type of a non-array sub-process/interface output parameter.
2. In the "Map From Sub-Process" property tab, map the output parameter to that array field.

Sub-process instances can complete in any order, so output must be appended to the target list. For single to multi-instance output mappings, the target list must be configured as **Append to Target List**.

In other words, the mapped output values are appended in the order of sub-process completion, and therefore the location of data in output arrays might not match the location of input data for the corresponding sub-process instance.

Result

The way that this data is passed behaves in the same way for multi-instance statically defined sub-process tasks (tasks that reference an actual sub-process at design time).

Correcting Validation Errors

Any problems that result from validation are shown in the Problems view.

To correct the problem, do one of the following:

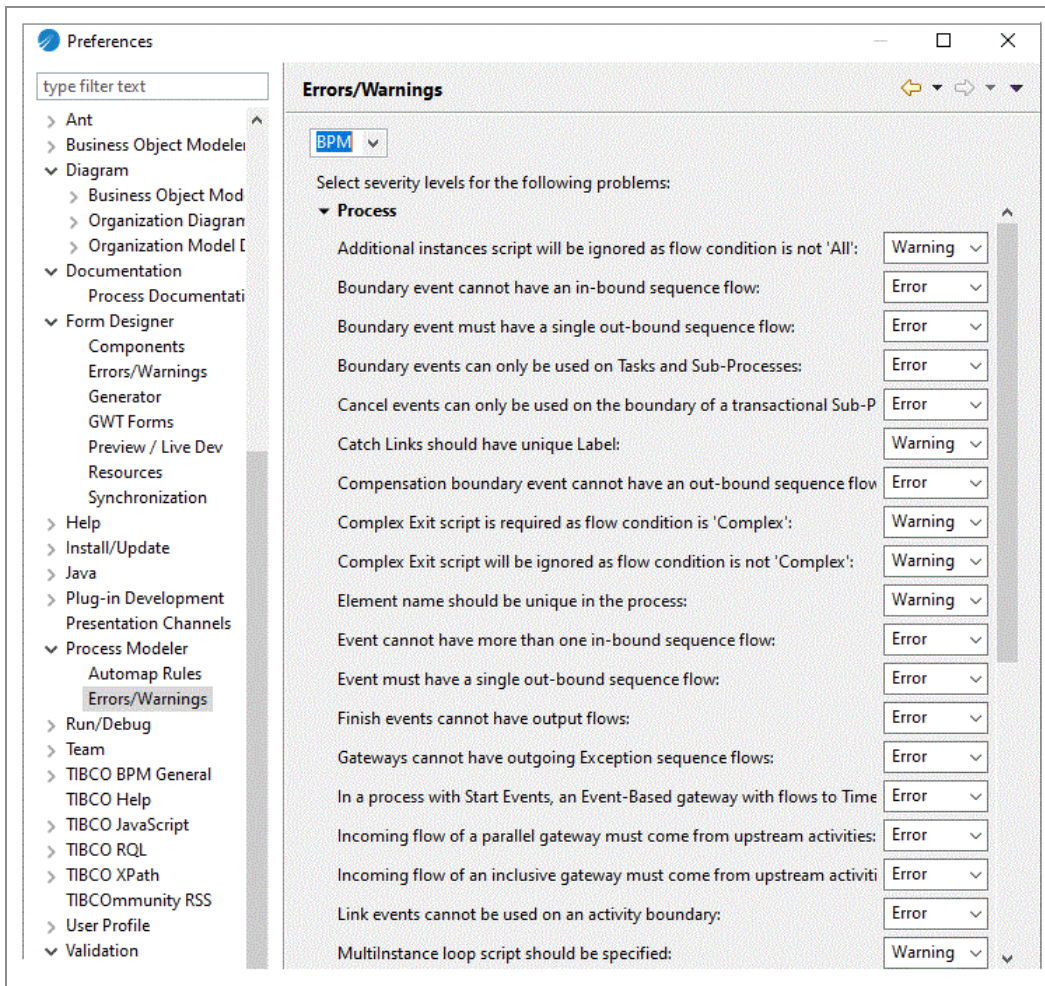
- Right-click the problem and select **Quick Fix** (if enabled for the current problem). This gives you the option of having TIBCO Business Studio correct the problem for you.
- Double-click the problem or right-click the problem and select **Go To**. This displays the Process in the Process Editor, highlighting the offending object and allowing you to manually correct the problem.

Setting the Validation Preferences

You can customize the validation that is performed in the Process Editor. Specifically, for each validation error you can specify its severity level as **Error**, **Warning**, **Info**, or **Ignore**.

Procedure

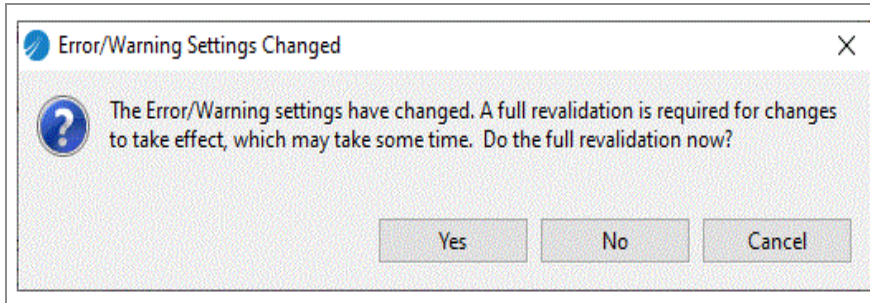
1. Select **Window > Preferences**.
2. Expand **Process Monitor** and select **Errors/Warnings**. The following dialog box is displayed:



3. If you want to change the severity level for a Process Editor problem, select from the drop-down list. When you have finished, click **Apply** to save any changes you have made.

Note: You can downgrade the severity of BPM errors, however some errors are not displayed because downgrading them would allow processes to be deployed that would be invalid in the runtime environment.

The following dialog box is displayed:



4. Make one of the following selections:

- Click **Yes** to revalidate your workspace. Depending on the size of the workspace and the number of errors, there is a delay while the revalidation occurs.
- Click **No** to revalidate your workspace later. The revalidation takes place when the concept file next changes or is saved, or when you explicitly request a rebuild of the project or workspace.
- Click **Cancel** if you do not wish to apply your changes.

Making a Sub-Process Call to a Process Interface

There are two different ways of creating a call to a process interface.

- By dragging a process interface from the Project Explorer and dropping it onto your process.
- Using the call sub-process tool from the palette.

Creating a Call Sub-Process Call using drag-and-drop

Procedure

1. Expand the Project Explorer to locate the process interface that you want to

implement.

2. Click the process interface, holding down the mouse button, drag the pointer to the calling process (open in the Process Editor), and release the mouse button.



Note: You can select multiple processes for drag-and-drop operations using the Ctrl key.

3. If you are dropping more than one process interface, a menu is displayed with two options:

- **Create Sub-Process Task Sequence**

Selecting this option allows you to create sub-process tasks connected by sequence flow. A dialog box is displayed to allow you to control the order of the tasks: Use the **Move Up** and **Move Down** buttons to control the order of the tasks. When you are finished, click **OK** to place the tasks.

- **Create Unsequenced Sub-Process Tasks**

Selecting this option places the tasks in the process without a connecting sequence flow.

Creating a Call Sub-Process Call Process Interface using the Palette

Procedure

1. In the Process Editor, select the **call sub-process** tool.
2. Click in the process where you want to place the activity that calls the sub-process.
3. On the Properties view for the activity, browse for the process interface you want to call as a sub-process.



Note: If the process interface you select is not in the current project, you are prompted to create a project reference:

Controlling Flow




Controlling Flow in a Process

You can use flows, loops, and gateways to control flow in a process.

Flows

Sequence flows indicate the order in which activities are performed. You can set up sequence flows between flow objects (activities, events, and gateways).

When you create a sequence flow, you can highlight it, right-click and select one of the following types of sequence flow:

-  **Uncontrolled Flow** - Indicates a flow that does not have a condition associated with it (the default). This path is taken in all cases.
-  **Default Flow** - Shows the default flow from a gateway or activity that is taken if no conditional flow has its condition met.
-  **Conditional Flow** - Shows a flow that is only followed if the associated condition is met.

Associating a Script with a Conditional Flow

Scripts are associated with a Conditional Sequence Flow by entering the script in the Properties view for that Sequence Flow object. You can use a script to define the conditions that determine whether a conditional sequence flow is followed.

Procedure

1. Create the Conditional Sequence Flow object. On the **General** tab of the Properties view, note that the **Conditional** button is selected by default.
2. In the **Script Defined As** list, select **JavaScript**.

3. Enter the script in the **Describe Sequence Flow Condition** area. The script can be a multi-line script and the result of the last statement of the script should be of Boolean type and that determines the result of the script. This is an example of a simple script:

The screenshot shows the 'Sequence Flow' configuration window. On the left, the 'General' tab is active. The 'Label' and 'Name' fields are empty. The 'Type' section has three radio buttons: 'Uncontrolled', 'Conditional' (which is selected), and 'Default'. Below this is a link 'Hide Implementation Details'. On the right, the 'Script Defined As' dropdown is set to 'JavaScript'. The 'Describe Sequence Flow Condition' text area contains the script `data.boolUrgent`.

This is an example of a more complex script:

The screenshot shows the 'Describe Sequence Flow Condition' text area with a more complex JavaScript script:

```
var points = 10;
if (data.orderAmount > 100000)
{
    points +=20;
}
if (data.orderType.toString() == "CPU")
{
    points >=10;
}
points > 25;
```

Note: On the **General** tab, there might already be a text description of the required script. This description is converted to comments if you select **JavaScript** from the **Script Defined As** list. If you select **Unspecified** from the **Script Defined As** list, the description is lost. However, you can recover the description by pressing Ctrl+Z.

Loops

Standard and multi-instance loops are supported, as defined in BPMN.

- **Standard loop:** A standard loop consists of a Boolean expression that is evaluated

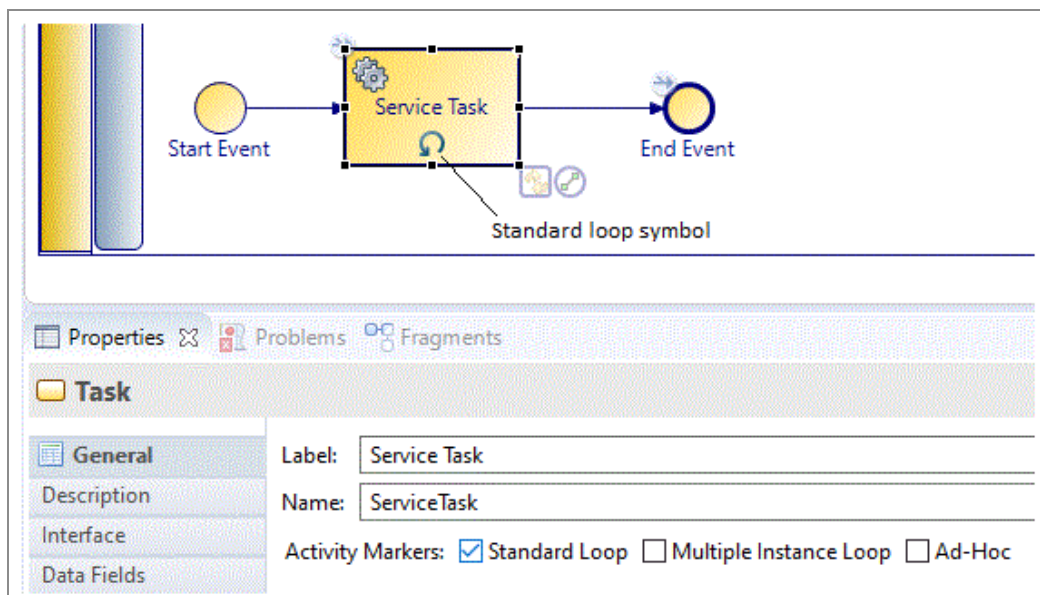
either **before** (loop while condition is true) or **after** (loop until condition is true) each cycle of the loop.

- **Multi-instance loop:** A multi-instance loop has an expression that evaluates to an integer, and is evaluated only once (before the activity is performed). The resulting integer specifies the number of times the activity should be performed.

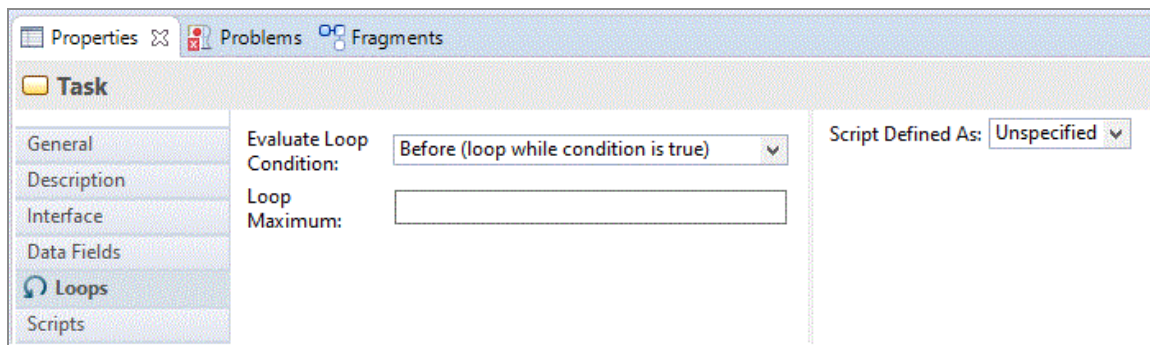
You can select the type of loop applied to a task either in the Properties view for that task (on the **General** tab), or by right-clicking the task and selecting from the **Activity Markers** menu. The **Loops** tab in the Properties view for the task is where you specify the details of the loop.

Creating a Standard Loop

A standard loop is indicated by a symbol.



Click the **Loops** tab to set the details of the loop:



Set the following parameters as appropriate for the loop you want to create:

- **Test Time:** Select either **Before** or **After**. The **Before** option is equivalent to the programming construct "while": the expression is evaluated *before* the activity is performed, and therefore if the expression evaluates to False, the activity is not performed. The **After** option corresponds to the programming construct "Do: while": the expression is evaluated after the activity has been performed, guaranteeing that the activity is performed at least once.
- **Loop Maximum:** Optionally specify an integer to control the maximum number of times the activity is performed. In the event of conflict between **Loop Maximum** and the result of the JavaScript defined under **Script Defined As**, the **Loop Maximum** overrides the script calculation. **Script Defined As:** Select one of the following options:
 - **Free Text:** Allows you to describe how you would like the loop to be tested if you prefer to leave the specific implementation of the loop expression to someone else.



Note: A BPMN warning is generated if you leave the script area blank. To remove the warning, enter a text description of the intended script implementation.

- **JavaScript:** Allows you to enter JavaScript that is evaluated for the loop expression.
- **Unspecified:** Allows you to specify that there is no expression or description for this loop.

Creating a Multi-Instance Loop

A multi-instance loop is indicated by one of two symbols:

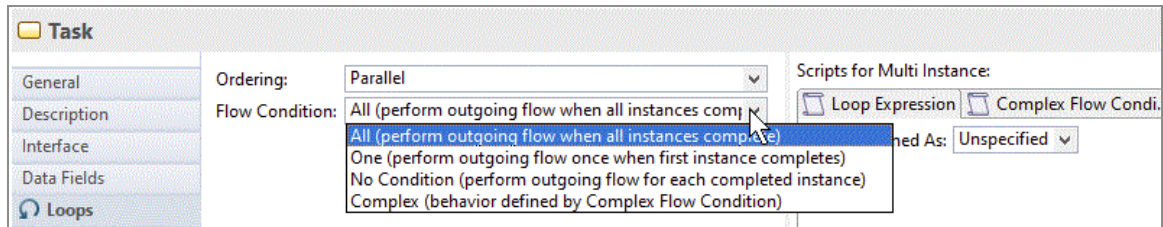
- Parallel multi-instance loop:



- Sequential multi-instance loop:



Click the Loops tab to set the details of the loop:



Set the following parameters as appropriate for the loop you want to create:

Ordering

Select either **Sequential** or **Parallel** ordering. The Sequential option causes the instances of the activity to occur in succession rather than at the same time. The Parallel option causes the instances of the activity to occur at the same time.

Flow Condition

This property can be used to mimic the functions of a gateway. Select one of the following options:

- **All (perform outgoing flow when all instances complete)** The outgoing flow is processed once after all activity instances have completed.
- **One (perform outgoing flow once when first instance completes)** The outgoing flow is processed when the first activity instance completes (existing instances can still be completed but the outgoing flow is not processed when they complete).
- **No Condition (perform outgoing flow for each completed instance)** The outgoing flow is processed for every activity instance as it completes.
- **Complex (behavior defined by Complex Flow Condition)** The outgoing flow is processed for each instance for which the Complex Flow Condition tab evaluates to true.

Scripts for Multi-Instance

Select **Free Text** for the **Script Defined As** field if you want to describe how you would like the loop to be tested, and prefer to leave the specific implementation of the loop expression to someone else. You can also select **Unspecified** if there is currently no condition or description for the loop.

i Note: Depending on the destination environment you have selected, **JavaScript** might be available as an option. Use this option if you want to enter JavaScript that is evaluated for the loop expression.

There are three tabs on which you can specify scripts:

Loop Expression

Specify either a script that evaluates to an integer or a description of the desired script.

Complex Flow Condition

This expression is evaluated if you selected Complex as the Flow Condition.

Additional Instances Expression

This expression is used for control-flow pattern WCP-15 (additional activity instances might be required at runtime) when you only need to add instances when the task is complete.

See the table "Supported Control Flow Patterns" in [Workflow Process Patterns Support](#). If you need to add instances while the task is in progress, see [Adding Additional Instances to a Multi-instance Loop While the Task is Still in Progress](#).

The additional instances script is evaluated after the last instance completes. If the script evaluation calls for additional instances, they are done and the script is evaluated again once the last additional instance completes.

i Note: If an Additional Instances Expression is specified then this expression must eventually evaluate to zero so no more instances occur.

Adding Additional Instances to a Multi-instance Loop While the Task is Still in Progress

This implements control-flow pattern WCP-15.

See the table "Supported Control Flow Patterns" in [Workflow Process Patterns Support](#).

i Note: If you have a multi-instance user task and a loop which allows you to generate multiple copies of that task, if you have concurrent copies of the multi-instance user task, each with multiple instances, and then "add additional activity instances", the additional instances are added to every copy of the multi-instance user task.

To do this, you need to execute a script somewhere on a parallel path whilst the loop task is active. The script can add instances to the loop task using an expression similar to the following:

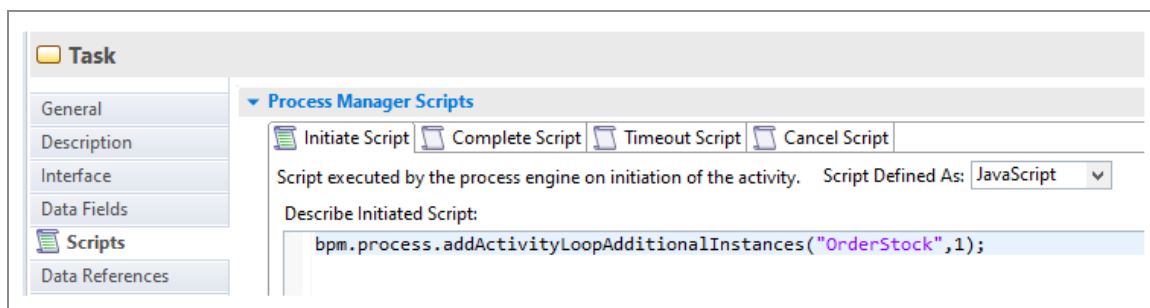
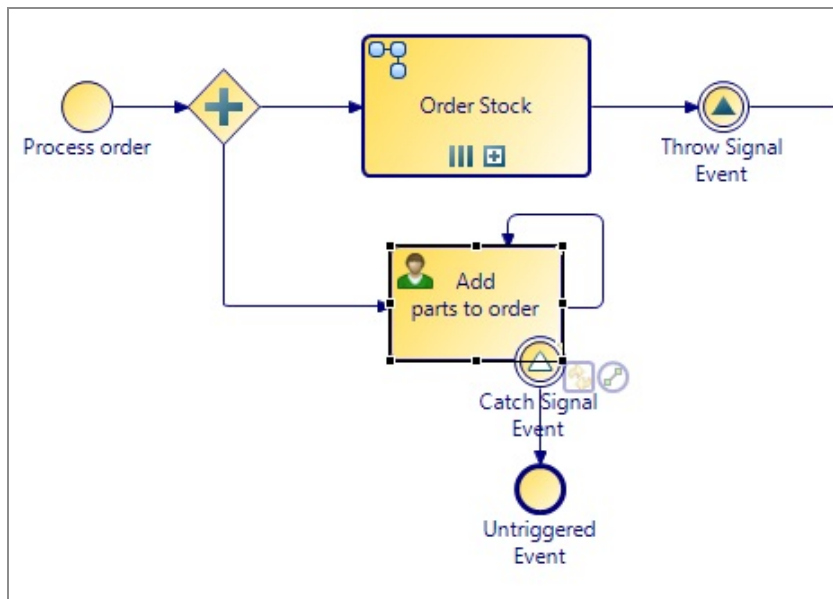
bpm.process.addActivityLoopAdditionalInstances ("OrderStock",1);

which contains a string for the name of the task (in this example, OrderStock), and an integer for the number of additional instances required (in this example, 1).

i Note: If you wish to add additional instances to a dynamic sub-process task, you need to pre-populate the array that is chosen as the runtime identifier field with the name or names of the sub-process implementation that you wish to add, as well as any associated arrays used to provide input into the dynamic sub-process.

You can also add additional instances to a loop in the loop task itself, **but only** in the task Complete script.

In this example, the task **Add parts to order** would order an additional item each time it was used.

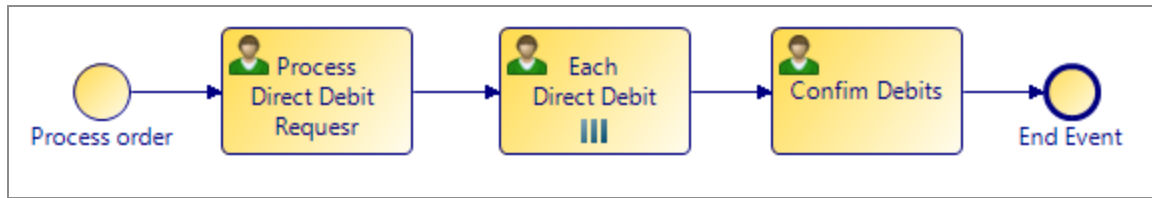


This workflow pattern specifies that multiple instances of an activity should be created, each instance being synchronized and able to run concurrently.

See the table "Supported Control Flow Patterns" in [Workflow Process Patterns Support](#). Note that:

- The number of instances required is known and specified at design time.
- Each task instance has access to all the process data (data fields, parameters, and so on).
- All the instances must be complete before the next task is started.

The following process has three activities:



There can be multiple instances of the Each Direct Debit task. This is indicated on the General tab of the Properties view:

Task	
General	Label: Each Direct Debit
Description	Name: EachDirectDebit
Interface	Activity Markers: <input type="checkbox"/> Standard Loop <input checked="" type="checkbox"/> Multiple Instance Loop <input type="checkbox"/> Ad-Hoc
Data Fields	Participants: - not set -
Loops	Activity Type: User Task
Work Resource	
Scripts	

The details are specified on the Loops tab:

Task	
General	Ordering: Parallel
Description	Flow Condition: All (perform outgoing flow when all instances com
Interface	Scripts for Multi Instance:
Data Fields	<input type="button" value="Loop Expression"/> <input type="button" value="Complex Flow Cond..."/> <input type="button" value="Additional Instan..."/>
Loops	Script Defined As: Free Text
Work Resource	Describe when outgoing flow should be processed for each instance:
	3

The **Parallel** ordering setting and **All** flow condition means that the activity instances are performed at the same time and can be synchronized. The number of instances is set to **3** on the right side of the Properties view.

Associating a Script with a Loop

Scripts can be associated with processing loops in the Properties view for the task to which a loop has been applied.

Procedure

1. On the **General** tab of the Properties view for the task, select either **Standard Loop** or **Multi-instance Loop**.

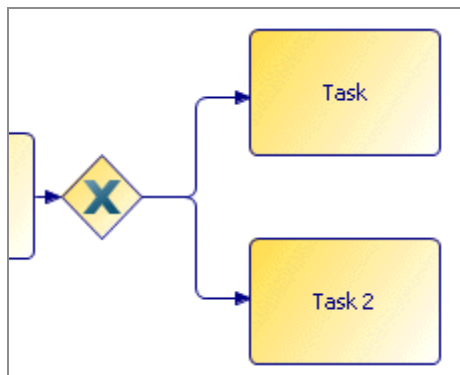
i Note: When using a multi-instance loop, you should set an additional instance expression which must evaluate to 0. When the loop has finished, it looks to see if this exists. If you do not set this to 0, then your multi-instance loop does not end.

2. When either of these is selected the **Loops** tab becomes available. Select **JavaScript** from the **Script Defined As:** field and enter the script itself in the area provided.

Gateways

Gateways are a control mechanism for the sequence flow in the process.

They are represented by a diamond:



Although the gateway resembles a decision box in a flow chart, gateways provide a variety of behaviors besides conditional decisions.

i Note: By default, gateways do not have names. This means that you must specify a name for all gateways in your process if you want them to be valid migration points.

As shown on the Properties view of a gateway, these are the different types of gateway that you can create.

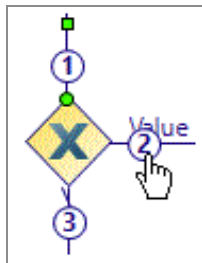
Gateway type:

- ☒ Exclusive Decision/Merge (XOR) Data Based
- ☐ Inclusive Decision/Merge (OR)
- ☐ Exclusive Decision/Merge (XOR) Event Based
- ☐ Complex Decision/Merge
- ☐ Parallel Fork/Join (AND)

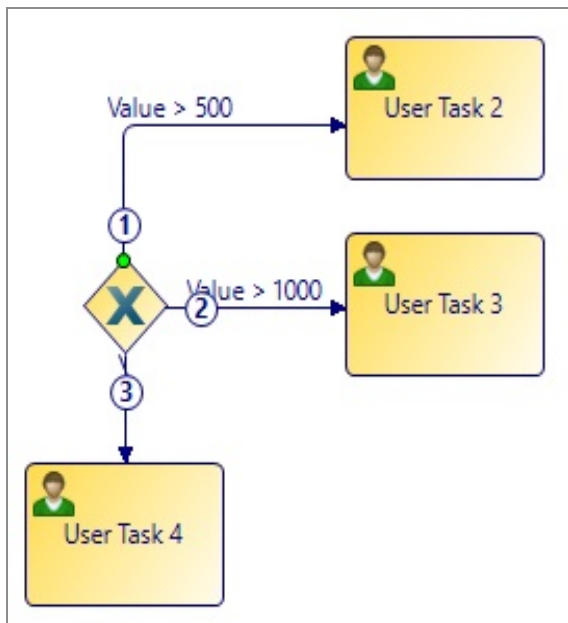
Order of Flow Evaluation

When a gateway has multiple sequence flow output, you can specify the order in which the outgoing sequence flow is processed.

You can view the current order by highlighting one of the sequence flows and placing the pointer over the outline numbers that appear. For example:



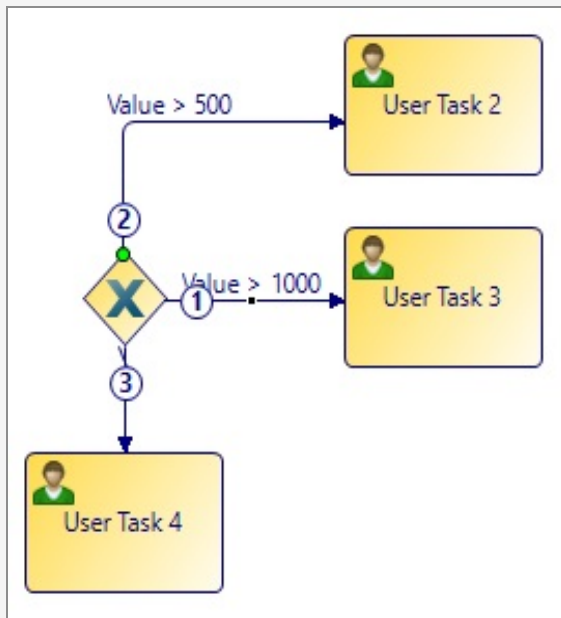
This is especially significant for evaluating the conditions on sequence flows attached to exclusive gateways. For example:



In this case, the **> 500** sequence flow is processed first. Assuming that the conditions are set up as their labels imply, if the value is greater than 500, **User Task 2** is executed. The **> 1000** sequence flow and **User Task 3** can never be reached.

To change the order of evaluation, drag the numbers that appear on the sequence flows. For example, dragging the 2 and dropping it onto the 1 changes the order of sequence flow evaluation as follows:

i Note: Regardless of the selected order of evaluation, conditional sequence flows are always evaluated *before* default sequence flows.



Exclusive (XOR)

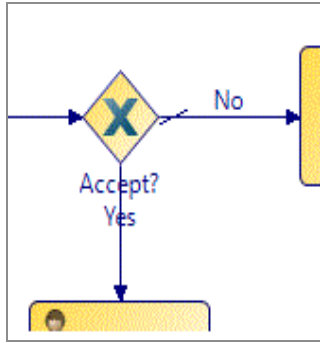
In an exclusive gateway, there are several paths through which the process can continue, but only one is actually chosen when the process is run. There are two types of exclusive gateway: exclusive (data) and exclusive (event).

Exclusive (Data)

The sequence flow is chosen based on an expression using data from the process. This type of gateway is indicated in the process as follows:



The following shows a typical exclusive (data) gateway:



There is one uncontrolled input sequence flow to the gateway, and conditional and default output sequence flows.

i Note: An XOR (data) gateway displays an **X** as a visual cue to the gateway type. However, BPMN does not require this.

To disable the display of the **X**, deselect **Show "X" Marker** in the Property view for the gateway.

Exclusive (Event-based)

The sequence flow is chosen based upon an external event, for example, an incoming request event. This type of gateway is indicated in the process as follows:



The gateway follows the outgoing path of whichever event happens first.

Inclusive (OR)

In an inclusive gateway used for branching, each output Sequence Flow is independently evaluated according to an expression.

This means that anywhere from zero to the maximum output sequence flows can be taken. In practice, you should either provide a default sequence flow or ensure that at least one sequence flow evaluates to True.

When used to merge flow, any upstream sequence flows are synchronized, but the gateway does not wait for all sequence flows.

An inclusive gateway looks like this:



Complex

A complex gateway is used to fork or merge depending on how an expression evaluates. When used as a decision, the expression determines which of the outgoing sequence flow are chosen for the process to continue.

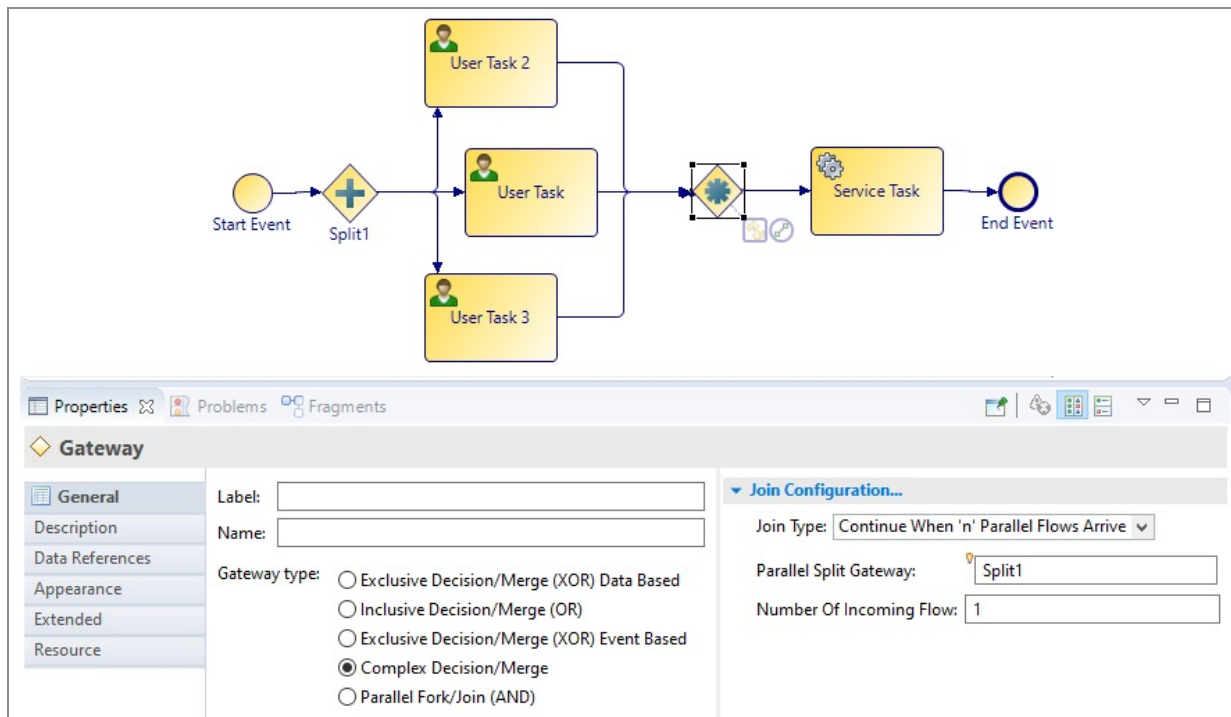
When used to merge flow, the expression determines which of the incoming sequence flows is required for the process to continue. This type of gateway is indicated in the process as follows:



Join Configuration

Although you cannot enter an expression on a complex gateway, there is a **Join Configuration** section in the Properties view.

This allows you to specify how many incoming sequence flows are received before flow continues. For example:



There are three sequence flows going into the complex gateway. On the properties of the gateway, the **Continue When 'n' Parallel Flows Arrive** join type is selected. The parallel gateway is specified indicating that this complex gateway is handling an earlier parallel split (named **Split1**), and that flow should continue when only one of the sequence flows reaches the complex gateway.

Parallel (AND)

A parallel gateway is used to fork or merge several parallel paths (synchronization). When several sequence flows enter a parallel gateway, the process flow waits until all arrive at the gateway before continuing.

This type of gateway is indicated in the process as follows:



Configuring a Task as Automatic Ad-Hoc

The automatic ad-hoc activity allows you to define one or more standalone tasks (with no incoming / outgoing flow) in a business process that are executed zero or more times during the lifetime of a process instance. An automatic ad-hoc activity is executed when it becomes enabled by any of the initializer activities being processed and the transition of the given precondition from false to true.

i Note: This feature is not available for pageflows.

You can have a number of ad-hoc activities in a process. Their preconditions determine if and when they are executed.

Procedure

1. Check the **Ad-hoc** Activity Marker on the **General** tab of the properties on a User task or Call Sub-Process task.
2. Click **Ad-Hoc Configuration Properties**. The **Ad-Hoc Configuration** tab opens.
3. **General:** optionally select **Interrupt main process until completion**.
This pauses the main process flow, resuming on completion of the ad-hoc activity. It does not stop you processing currently active work items or sub-processes, event sub-processes or other ad-hoc activities. However, note that data is not returned to the process instance until it re-activates.
4. Check **Automatic Invocation**.
5. **Access:** in addition to defining whether users have the ability to cancel an ad-hoc activity using a system action, optionally it is possible to define whether a user can cancel a specific ad-hoc activity by specifying Organization privileges that the user must possess.
6. **Enablement and Execution:** Define the condition under which the activity is enabled and executed automatically.
 - **Initializer Activities:** (optional) Select one or more Initializer Activities which are activities which are in the same process. This activity must be completed before the precondition is evaluated. This can be useful for ensuring that the precondition is not evaluated until the data referenced is initialized.
 - **Precondition:** Set an optional precondition script that controls enablement of

the ad-hoc activity.

i Note: For Automatic ad-hoc activity, at least one Enablement condition should be specified: that is, either the Initializer Activities or the Precondition script must be specified (both can be specified).

i Note: The condition must be a Boolean expression which evaluates to true. The condition has to go from true > false > true in order to retrigger so you must reset the result of the expression to false before it can become true again.

Configuring a Task as Manual Ad-Hoc

The manual ad-hoc activity allows you to define one or more standalone tasks (with no incoming / outgoing flow) in a business process that are executed zero or more times during the lifetime of a process instance. An ad-hoc activity only becomes available for manual invocation when it is enabled via the preconditions and the user holds the necessary privileges to start and cancel that activity.

i Note: This feature is not available for pageflows.

You can have a number of ad-hoc activities in a process. Their preconditions determine when they become available for execution by permitted users.

Procedure

1. Check the **Ad-hoc** Activity Marker on the **General** tab of the properties on a User task or Call Sub-Process task.
2. Click **Ad-Hoc Configuration Properties**. The **Ad-Hoc Configuration** tab is displayed.
3. **General:** optionally select **Interrupt main process until completion**.

This pauses the main process flow, resuming on completion of the ad-hoc activity. It does not stop you processing currently active work items or sub-processes, event sub-processes or other ad-hoc activities. However note that data is not returned to the process instance until it re-activates.

4. Check **Manual Invocation**.

5. Optionally check **Allow multiple invocations**. If you check this, you can manually invoke the activity more than once after it has been enabled.
6. **Access:** in addition to defining whether users have the ability to cancel an ad-hoc activity using a system action (see the "System Actions Reference" topic in the *TIBCO® BPM Enterprise Concepts Guide*), optionally it is possible to define whether a user can cancel a specific ad-hoc activity by specifying Organization privileges that the user must possess.
7. **Enablement:** Define the condition under which the activity is enabled and executed automatically.
 - **Initializer Activities:** (optional) Select one or more Initializer Activities which are activities which are in the same process. The precondition is not evaluated until one of these has been completed.
 - **Precondition:** Set an optional precondition script that controls enablement of the ad-hoc activity. The automatic ad-hoc activity is invoked every time the precondition is changed from false to true. So the activity might be invoked many times.

i Note: For manual ad-hoc activities, specifying enablement is optional, so it not mandatory to specify either the initializer activity or the precondition script.

i Note: The condition must be a Boolean expression which evaluates to true. The condition has to go from true > false > true in order to re-trigger so you must reset the result of the expression to false before it can become true again.

Refactoring Ad-Hoc Activities

You can refactor one or multiple ad-hoc user tasks into a sub-process.

**Note:**

You might want to do this with an ad-hoc user task, as certain actions are not available to you on the ad-hoc user task. However, once you refactor one or multiple user tasks into a reusable sub-process, more actions are available. For example, you can add a deadline to a user task within an ad-hoc reusable sub-process, but not to an ad-hoc user task.

Note that when refactoring a single ad-hoc user task, the reusable sub-process inherits the ad-hoc configuration. When refactoring multiple ad-hoc user tasks the reusable sub-process contains multiple user tasks which retain their ad-hoc configuration.

Procedure

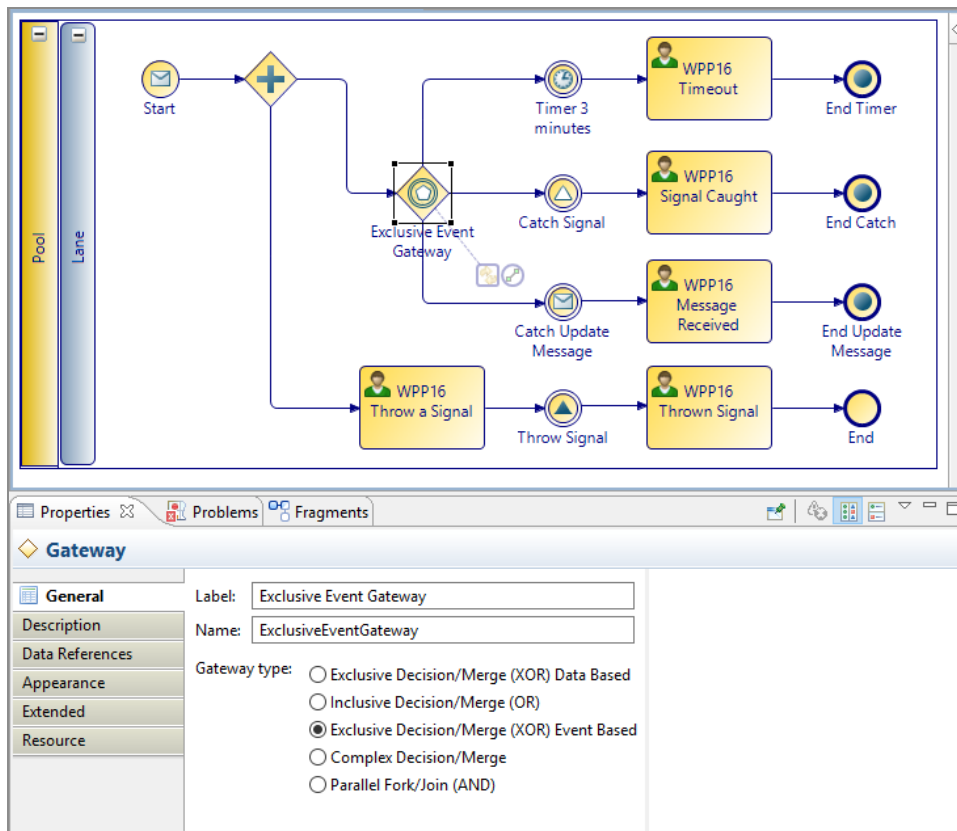
1. Select an ad-hoc user task (or multiple ad-hoc user tasks), right-click and select **Refactor > Extract Into New Sub-Process**.
2. Follow the steps in the wizard to refactor. The ad-hoc user task or tasks now are a reusable sub-process.

Exclusive Event Based Gateway Example

An example of an Exclusive Decision/Merge (XOR) Event Based Gateway that provides an example for workflow process pattern 16, Deferred Choice.

Workflow process pattern 16, Deferred Choice, states:

A point in a process where one of several branches is chosen based on interaction with the operating environment. Prior to the decision, all branches represent possible future courses of execution. The decision is made by initiating the first task in one of the branches i.e. there is no explicit choice but rather a race between different branches. After the decision is made, execution alternatives in branches other than the one selected are withdrawn.



The state remains on the Exclusive Event Based Gateway until a signal or message is caught, or the timer expires.

When the event occurs, the state is pulled from the Gateway to the relevant task, and this removes other paths from possible operation.

Note: It is possible to have multiple paths, with various timers, messages and signals, but the Gateway only actions the first event to occur.

Controlling Flow from an External Application

You can control the flow of a process from an external application using incoming request events and tasks. You can trigger these events or tasks by an external application using the Process sendEvent REST API.

With Incoming request events and tasks, you can perform the following functions:

- Pause a process flow until you receive the incoming request using an in-flow event or task.
- Cancel a long running task such as a user task, or sub-process, that is in progress by attaching the incoming request event to a task boundary.
- Start a parallel event handler flow in the process by creating an event sub-process with an incoming request start event.

The Process REST API triggers an incoming request event or task. With this API, the process instance ID or incoming business data can be used to identify the specific process instance on which you can trigger the incoming request.

Correlating Incoming Requests

You can trigger an incoming request event or task using the Process REST API. You can use the process instance ID or correlation data to identify the specific process instance on which you can trigger the request using your own business data. The incoming request is only triggered on a single process.

i Note: The Business data method of correlation is currently not supported for Incoming Request Event-handlers and Event Sub-Processes.

To use business data for correlating incoming requests with individual process instances, the process must have one or more correlation data fields whose value uniquely identifies an individual process instance. This correlation data can then be associated with an incoming request event or task in its interface property configuration. By default, all correlation data is associated implicitly.

When you invoke the process REST API to trigger an incoming request without a process instance ID, the correlation data field values supplied in the request payload are matched to the correlation data values in an existing process instance. The incoming request is then triggered in that process instance.

You must ensure that the correlation data is set with the appropriate value in each process instance before the incoming request event or task is initialised.

Using Data

Using Process Data

Process Data Overview

Process data is data that can only be used in the process where it is created.

Data Fields and Parameters

Data fields identify the inputs and outputs of an activity. Parameters are input to or output from a source external to the process.

For example:

- Data fields: an activity called "Process Student Course Request" could require a form with the list of courses the student wants to take as input. The availability is checked and a form that lists the courses they are enrolled in is output.
- Parameters: parameters are passed from a process to a sub-process.

Parameters can only be created at the process level. Data fields can be created:

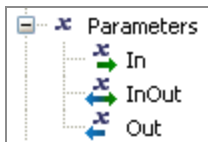
- at the package level, where they can be shared amongst processes defined in that package.
- at the process level, where they can only be used by the parent process.
- at the activity level, where they can only be used by the parent activity.



Tip: You can change a process-level data field to a parameter by right-clicking it, and selecting **Convert Data Field to Parameter**. The default mode for converted parameters is **In/Out**, meaning that they can be specified as either input or output.

Data fields and parameters share the same basic types (Integer, Text, and so on), however they have different properties:

- Parameters can be specified as mandatory (they must be present at runtime).
- Both data fields and parameters can be specified as read only. For data fields, this means that they can only be assigned by their initial value setting. For formal parameters, it means that once input to the process they cannot be re-assigned (for example, in scripts or user tasks).
- Data fields and parameters can be an array of basic types (for example an array of Text).
- Parameters can also be specified as input, output or both by selecting the Mode (**In**, **Out**, or **In/Out**). The mode is indicated by the icon next to the parameter:



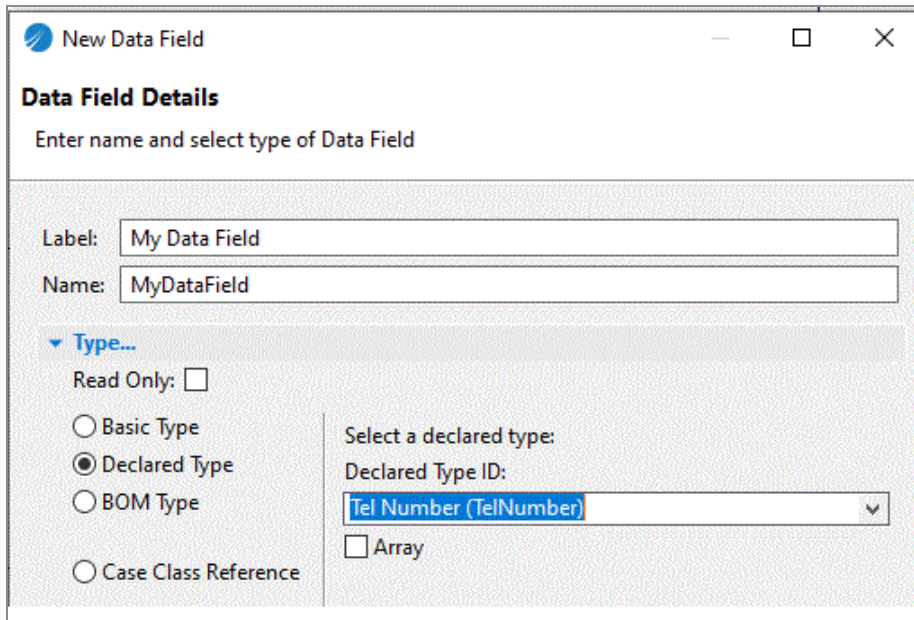
Declared Types

Declared types are used if you want to reuse a definition either when creating a data field or parameter.

For example, you could create a declared type that is text that represents a telephone number:

The screenshot shows the 'Type Declaration' dialog box with the 'General' tab selected. The 'Label' field contains 'Tel Number' and the 'Name' field contains 'TelNumber'. Under the 'Type...' section, the 'Basic Type' radio button is selected. To the right, the 'Set a basic type:' section shows 'Basic Type' set to 'Text', 'Length' set to '50', and 'Decimal Places' set to 'NA'.

This declared type is then available for use in defining data fields or parameters. For example:



Adding Data Fields or Parameters to a Process

You can add data fields or parameters to a process in two different ways, in the Properties view provided when the **Parameters** or **Data Fields** folder is selected in Project Explorer (this method is better if you need to create several data fields or parameters) or using the wizard to create one at a time.

You can add data fields or parameters to a process:

- in the Properties view table provided when the **Parameters** or **Data Fields** folder is selected in Project Explorer (this method is better if you need to create several data fields or parameters).

See [Using the Properties View to Create a New Data Field or Parameter](#)

- using the wizard to create one at a time.

See [Using the Wizards to Create a New Data Field or Parameter](#)

i Note: See [Associating Participants with Activities](#) for information on how to highlight all the tasks in a process that have a particular data field or parameter assigned to them.


Adding Data Fields to an Activity

You can create a new data field for an activity:

Procedure

1. Select the activity for which you want to create a data field.
2. On the Properties view, click **Data Fields**.

i Note: Activities that do not support activity-level data fields (such as gateways) do not have a **Data Fields** tab on their Properties view.
A validation error is displayed if you define a data field for an activity on the **Data Fields** tab.

Click . A new data field is added.

3. Modify the data field's properties as appropriate.

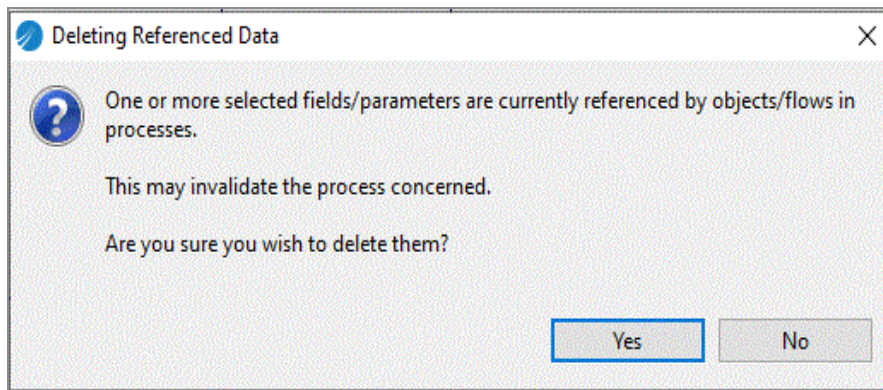
Result

i Note: You can only assign a value to an activity-level data field by using the **Scripts** tab on the activity.

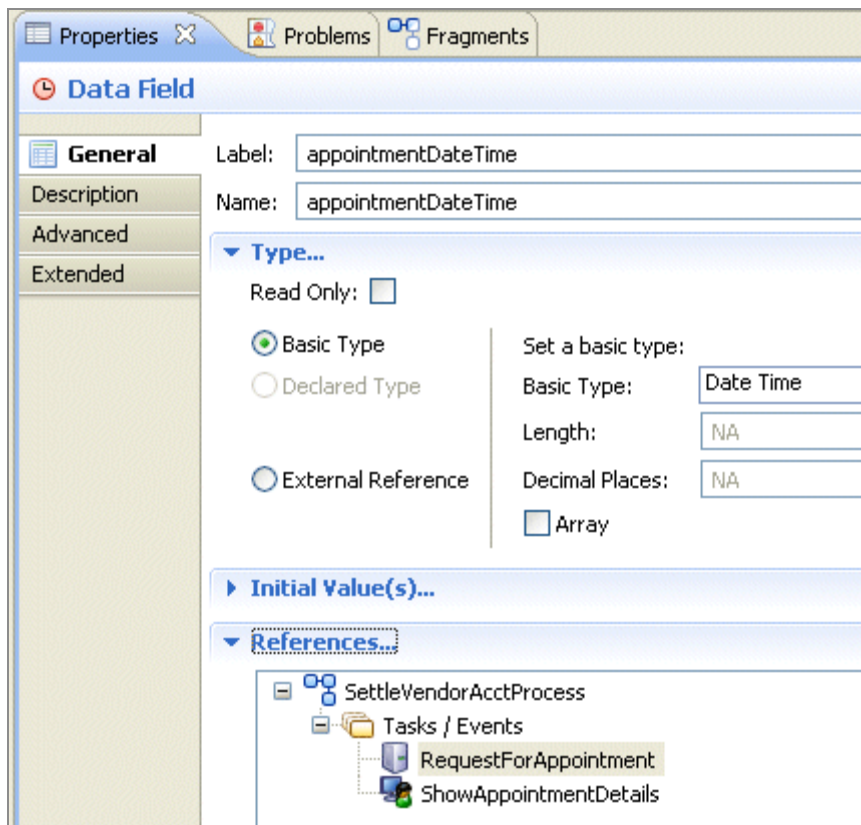
Deletion of Data Fields Parameters Participants and Type Declarations

You can delete a project object such as a data field, parameter, participant, or type declaration by right-clicking it in the Project Explorer and selecting **Delete**.

However, if the project object has been associated with another object such as an activity or a sequence flow, the following message is displayed:



If you click **Yes**, the project object (in this example, a data field) is deleted, but it still is referenced in XPDL for the project, which can cause problems. If you click **No**, the project object is not deleted. You can then go to the Properties view and examine the references to that object:



In this example, the data field is referenced in the service task **RequestForAppointment** and in the user task **ShowAppointmentDetails**. To show that task in the Process Editor, click **Go To**.

Using the Properties View to Create a New Data Field or Parameter

You can create a new data field or parameter in the Properties view.

Procedure

1. In the Project Explorer, expand the process where you want to add a data field or parameter.
2. As appropriate, click **Data Fields** or **Parameters** folder in the Project Explorer.
The Properties view is displayed.
3. Click the plus icon.
A new data field or parameter is added. As appropriate, modify the properties.

Using the Wizards to Create a New Data Field or Parameter

You can create a new data field or parameter using the wizards provided:

Procedure





1. In the Project Explorer, expand the process where you want to add a data field or parameter.
2. As appropriate, right-click **Data Fields** or **Parameters** and select **New > Data Field** or **New > Parameter**. The **New Data Field** or New Parameter dialog box is displayed.
3. Specify a label for the data field or parameter; you cannot specify leading or trailing spaces.
4. For parameters, specify the following:

Option	Description
Mandatory	<i>(Parameter only)</i> Select this check box to specify that the parameter

Option	Description
	must be present when the process is started.
Read Only	Select this check box to specify that the value of the data field or parameter cannot be modified after it is created.
Mode	<p>(<i>Parameter only</i>) Select whether the parameter is an input (In) output (Out) or both (In/Out).</p> <p>Note: The parameters specified on the Interface tab for a user task are from the perspective of the <i>form</i>, not the <i>user</i>. This means that input parameters are sent to the form by the process, not the user. Output Parameters are sent to the form by the user.</p> <p>Note: You must provide an input mapping when the process is called as a subprocess.</p>

5. Specify the type of data field or parameter you want to create:

Option	Description
Basic Type	<p>If you select this type, you can choose from Boolean, Date, Date Time and Timezone, Fixed Point Number, Floating Point Number, Performer, Text, Time, URI.</p> <p>Note: If you need to create a Text field of unlimited length, ensure that the Length field is empty. If you wish to limit the number of characters enter the appropriate numeric value.</p>

Option	Description
	<p>Note: The value of unmapped Basic Type parameters has the following defaults:</p> <ul style="list-style-type: none"> • Text: null • Boolean: false • Date / Time / DateTime, DateTime and Timezone : now (time of process instance creation)
Initial Value	For data fields, you can optionally specify an Initial Value by clicking in the provided text entry area and entering a value. For arrays, you can add more than one value. You can add rows by clicking the  button. You can also delete rows by clicking the  button.
Allowed Values	For parameters, you can optionally specify the permitted input values (values that might be supplied by application starting an instance of the process). You can add rows by clicking the  button. You can also delete rows by clicking the  button.
Declared Type	This option allows you to select from the declared types that you have already defined.
External Reference	Allows you to refer to a business object defined in the Business Object Modeler.

6. Select whether you want the data field or parameter to be an array. Selecting the **Array** check box creates an array of the Basic Type that is selected. For example, if you select the **Array** check box and Text, you are defining the data field or parameter as an array of Text values.
7. Click **Finish** to create the data field or parameter, or click **Next** to specify a documentation URL, description, or extended attributes.
8. The **Documentation URL** and **Description** fields allow you to specify supplementary information about the data field or parameter that you have created.
9. The Extended dialog box allows you to specify extended attributes.

The parameter that you created appears in the Project Explorer.

Participants

When a process designer creates a user task, they can define the participant(s) who performs the task at runtime. Participants are used to identify who or what performs an activity. For example, in a hiring process, a person (human participant) interviews the candidate and an email system (system participant) sends out an automatic follow-up reminder.

Participants are defined in the following ways:

- **statically**, by specifying one or more organizational entities - groups, positions, organization units or organizations.
- **dynamically**, by using runtime data to identify the required organizational entities.
- **using expressions**, by building a query that interrogates the organization model to identify the required organizational entities.

This flexibility allows a process designer to handle both simple and complex distribution scenarios without impact on the overall process design. For example:

- offer a user task to all Customer Service Representatives.
- allocate a user task to an accountant if the value to be signed off is less than \$5000, but allocate it to an Accounts Manager if the value is \$5000 or more.
- allocate a user task to a single loss adjuster who holds at least level 2 motor insurance certification and is based in the Chicago office.

There are two types of participant:

- *user task participants* represent the users who perform the work defined in user tasks. These participants must be defined as *external references* in an organization model used by the process, not as *basic types*.
- *system participants* are used to identify a task that is performed by the system.

Creating a Participant

Participants are used to identify who or what performs an activity.

Procedure

1. In the Project Explorer, expand the package where you want to add a participant.
2. Right-click **Participants** and select **New > Participant**. The New Participant dialog box is displayed.
3. Click the **Back** button if you need to change either the name of the **Project** and **Package** where the participant is created. If you want to change either, click the **Project** or **Package** button.

Participants can be created at either the package level or at the process level. Creating them at the package level is recommended as it enables them to be shared amongst processes. Select the **Process** check box and specify a process if you want to create the participant at the process level.

i Note: If the Participants folder is empty at the Process level, it is hidden by default. This is because the preferred usage is to define Participants at the Package level.

4. Click **Next**.
5. Specify the **Label** and **Name** of the participant (either a basic type, or an external reference as described previously in this section) and click **Finish**.
 - To create a basic type, select **Basic Type** and choose from **System** and **Organization Model Query**, and click **Finish**.

If you select the **Organization Model Query** button, you can then enter the Organization Model Query Script in Resource Query Language (RQL) using a script or expression in the **General** tab of the Properties view. This is evaluated when a referencing task is executed at runtime, so the actual participant is resolved and the activity dispatched and offered to the participant. A query could resolve to a participant in the package/process or to an entity in the organizational model.

- To create an external reference to an organization model, select **External Reference**, and click the picker to select a type from the organization model.

Choose a type from those shown in Matching Items, or key in the first few characters of the name you are looking for in the field under **Select type(s) (? = any character, * = any string)** and choose from those shown. Click **OK**.

The participant that you created appears in the Project Explorer.

i Note: All external references to participants from within the same project must be to the same **major version** of the organization model. However, you can reference different minor or micro versions of the model. For example, if you have included a reference to a participant in version 1.0.0.qualifier of the organization model, and the model subsequently changes, you could reference a participant in version 1.1.0.qualifier, but not version 2.0.0.qualifier.

Associating Participants with Activities

You can associate a participant with an activity to identify who or what performs the activity. You can do this either by dragging and dropping the participant onto the activity or in the Properties view for the activity.

Using Properties View to associate a participant with an activity

You can associate a participant with an activity.

Procedure

1. In the Process Editor, highlight the desired activity.
2. Either:
 - In the Properties view, click the picker.
 - Right-click the activity, and select **Participant**.The Select Participants dialog box is displayed.
3. Highlight participants you want to select and click **Add** to move them to the **Selection** column. When you have finished selecting Participants, click **OK**.

**Tip:**

- You can select multiple participants by pressing either the **Ctrl** (for single selection) or **Shift** (to select a range) keys while making your selection.
- The wildcard **?** returns all matching participants. Use the ***** wildcard to restrict the results (for example, ***2** to return all Participants ending in 2. Note that the wildcard ***** by itself does not return any results; it only works in conjunction with a string.
- You can also select as a participant a data field of the type **Performer**.

4. The participants you selected are displayed in the Properties view and also when you hover the pointer over the activity in the Process Editor.

Using drag-and-drop to associate a participant with an activity

You can drag a participant onto an activity.

Procedure

1. Expand participants in the Project Explorer.
2. Click the participant you want to associate with your activity, holding down the mouse button, drag the pointer to the activity and release the mouse button.

**Note:**

- You can select multiple participants for drag-and-drop operations using the **Ctrl** (for single selection) or **Shift** (to select a range) keys.
- You can also select as a participant a data field or formal parameter of the type **Performer**.

3. A menu is displayed with two options:

Option	Description
Add Task	Selecting this option adds the participants to any existing participants


Option	Description
Participant(s)	for the activity.
Set Task Participant(s)	Selecting this option clears any existing participants associated with the activity, before setting the participants to those you selected.

Highlighting Participant References

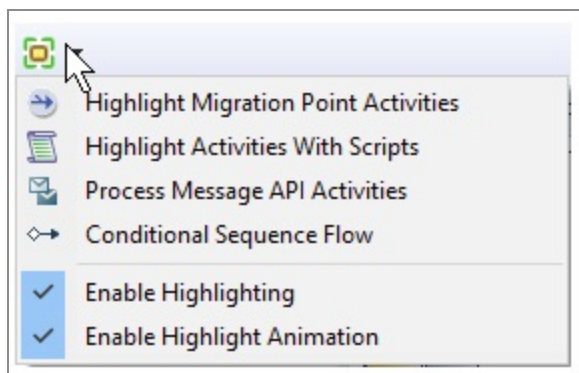
You can see what tasks a participant is assigned to by using highlighting.

Procedure

1. Click in the Process Editor for the process.

On the toolbar, you can see a button: 

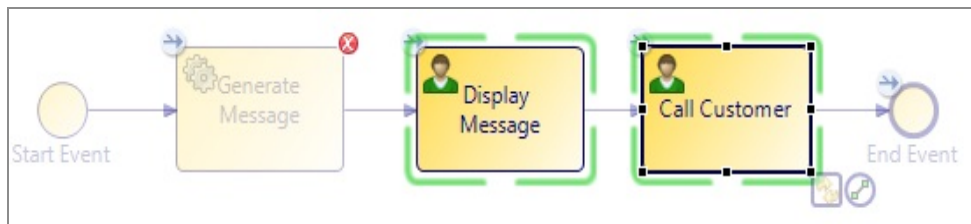
2. Click this to see the dropdown, and select **Enable Highlighting**.



Alternatively, you can select **Diagram > Highlighting > Enable Highlighting**.

3. Select the participant in Process Explorer, and in the Process Editor the tasks which use this participant are highlighted (framed with green lines). Other tasks in the process are disabled.

This behavior also applies to data fields and parameters.



Result

Highlights are cumulative.

- If you select more than one participant, data field, or parameter in Project Explorer, only tasks which reference all of those selected are highlighted.
- If you select "Activities With Scripts" followed by "Process Message API Activities", then the diagram objects highlighted are process message API activities that have scripts.

The current highlight filter is shown when you use the Process Editor mouse-over tooltip.

Highlights are removed when anything other than the highlighted diagram objects are selected.

Participants to Organization Model Mapping

The following table shows how participants that *are not* referenced in a process appear in the exported organization model.

Process Participant Type	Organization Model Object
Human	Position in default organization model unit
Organization unit	Top-level organization unit
Role	Top-level group
System	Ignored

The following table shows how participants that *are* referenced in the process (for example, in a user task) appear in the exported organization model.

Process Participant Type	Organization Model Object
Human participant	Position in organization unit (unit is named after the lane in the original process)

Process Participant Type	Organization Model Object
Organization unit	Organization unit within an organization unit (the containing unit is named after the lane in the original process)
Role	Top-level group
System	Ignored

Using Participants to Create Tasks

If you have participants, you can drag them to a blank area of a process to automatically create several types of task:

Procedure

1. In the Project Explorer, click the participant, and holding down the mouse button drag the pointer to a blank part of the process and release the mouse button.

i Note: You can select multiple participants for drag-and-drop operations using the **Ctrl** (for single selection) or **Shift** (to select a range) keys.

2. A menu is displayed with the following options:
 - **Create User Task For Participant(s):** Selecting this option creates a user task with the selected participants.
 - **Create Service Task For Participant(s):** Selecting this option creates a service task with the selected participants.
 - **Other Task Types:** This submenu contains more task types that you can create from a participant.

Organization Model Entities as Process Participants

Every user task in a process that is to be executed must have at least one participant. The participant can be defined in various ways.

The participant can be defined as the following:

- as a particular organizational entity in an organization model used by the project. See [Assigning Participants to a User Task](#) for more details.
- as an **organization model query**, a statement which uses Resource Query Language to create an expression that locates the required participant within the organization. See [Using a Participant Expression to Define a Participant](#) for details of how to do this.

To be available for use by a process, an organization model must be defined either in the same project as the process, or in a referenced project.

Assigning Participants to a User Task

Participants define who might receive work items generated from a user task.

A user task must have at least one participant, and can have as many as are required by the process. (A validation error is flagged on a user task that does not have at least one participant defined.)

Participants can only be assigned to a user task if they are defined in the process package (at process or package level) and are valid entities in an organizational model used by the process - see [Participants](#).

You can use a combination of the following methods to define a participant for a user task:

- selecting a named participant.
- using a performer data field or parameter. See [Using a Performer Data Field or Parameter to Dynamically Define a Participant](#)
- using a participant expression. See [Using a Participant Expression to Define a Participant](#).
- using organization entities in performer data fields/parameters to allow you to deliver work dynamically to an organizational entity

Using a Participant Expression to Define a Participant

You can use the **Organization Model Query** option to create an *expression* to define a participant. The expression sets out a definition that the participant in question must meet. This defines the participant in terms of organization model entities.

For example, this enables you to:

- allocate a work item to the manager of the person who carried out a particular named task.
- allocate a work item to one named position if the value of the data field *x* is >2000, and allocate it to a different named position otherwise.

Participant expressions enable dynamic definitions of participants, since the participant who fits the criteria on one occasion might not be the same on another occasion.

Expressions can be used to describe concepts such as the following:

- The manager of a given organization unit
- A member of a given organization unit
- A given group
- The manager approving a given task

i Note: A push destination assigned to an organization entity (group, position, organization unit, and so on.) only works when the organization entity is explicitly identified as the participant, and not when it is defined as part of a participant reference.

These expressions can be made up of references to organization model entities:

- Lists of all members of a given organization unit
- Lists of all members in a named position in a given organization unit
- Lists of all members of a given named group
- List of all resources in a named position
- A specific named resource

For instance, you can select a group named "HealthSafety" this way:

```
group(name="HealthSafety")
```

Using a Performer Data Field or Parameter to Dynamically Define a Participant

A performer data field/parameter is a special type of data field/parameter that you can select as a participant for a user task. By assigning a value to the performer data field/parameter earlier in the process, you can dynamically define a participant for a user task. You can populate a performer data field/parameter with one or more organization entity GUIDs, or a single valid non-array RQL expression.

i Note: A parameter can only be defined at the process level.

Using a performer field, you can deliver work dynamically to an organizational entity, so that the work items appear in managed work lists. For example:

- Dynamically deliver to a group by name (e.g. using an organization entity GUID:
`bpm.process.getOrgModel().groupByName('MyGroup')`)
- Dynamically deliver to a position within an organization unit by name (for example:
`orgunit(name='KEYTeam').position(name='AdditionalStaff') union orgunit
(name='Agency').position(name='Contractor')`)

i Note: You can also use non-RQL scripting to identify dynamic performers. For example:

```
var groups = bpm.process.getOrgModel().groupByName('MyGroup');
if (groups != null && groups.length == 1) {
  // If you find the group that you want, then use its GUID as
  the performer field value.
  data.performer = groups[0].getGuid();
} else {
  // If no groups match the given name OR there is more than one
  group with the same name,
  // then take appropriate application-specific remedial action
  here.
}
```

i Note: If you use the presentation channel settings (push destinations) to deliver notification of work items via email, on the Work Resource tab for the user task, you must set the Distribution Strategy to **Allocate to One** rather than **Offer to All**. For example, if you have a performer field set to: `resource(name='tibco-admin')`, `tibco-admin` receives an email notification of a work item **only** if the Distribution Strategy is **Allocate to One**.

Procedure

1. Add the performer data field/parameter as a participant to the user task, using the Properties View.
2. Make sure that the process logic assigns a suitable value to the parameter before the user task is executed - for example, by using a script.

You can assign a script.

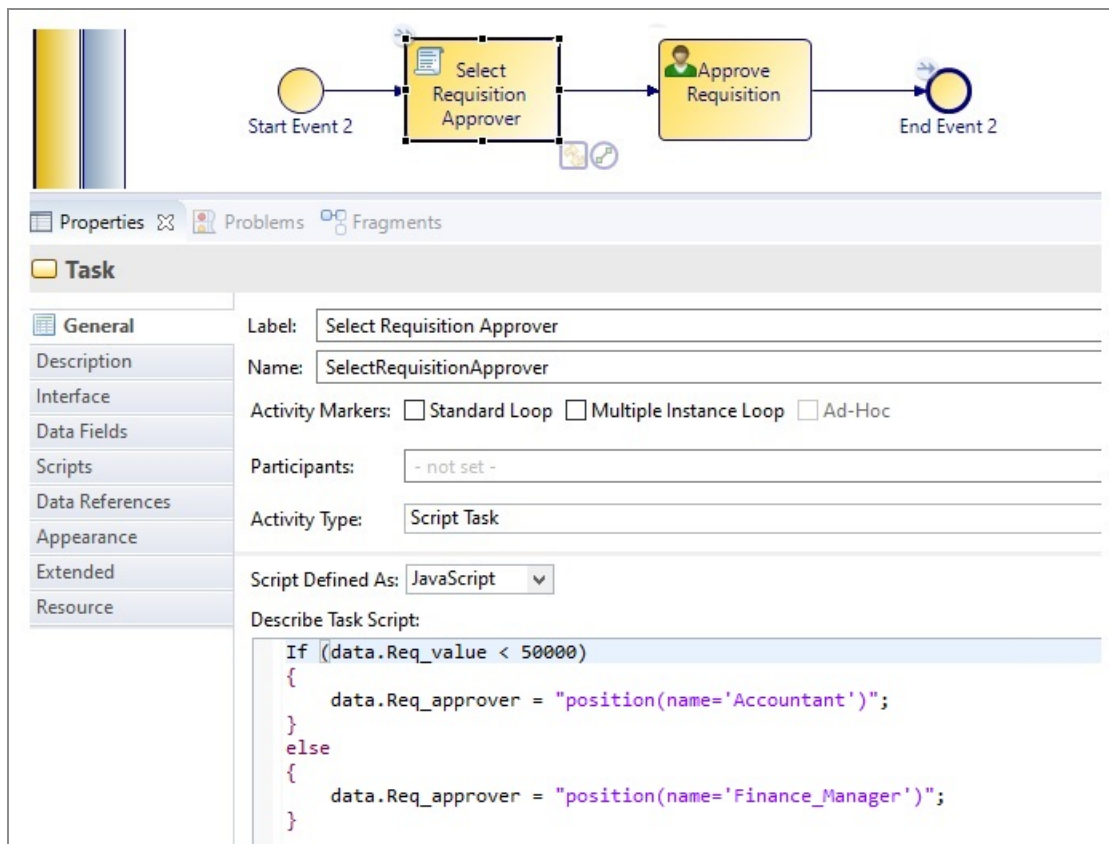
You can populate a performer field with one of the following:

- an organization entity GUID
- multiple organization entity GUIDs (using a performer array field that contains multiple GUIDs - each of the performer fields in the array contains an organization entity GUID)
- a single valid non-array RQL expression (that resolves to one or more organization entities).
- a resource GUID. For example:

```
var users= bpm.process.getOrgModel().resourceByName
('JohnSmith');
if (users!= null && users.length == 1) {
    // If you find the user that you want, then use its GUID as
    the performer field value.
    data.performer = users[0].getGuid();
} else {
    // If no users match the given name OR there is more than
    one user with the same name,
    // then take appropriate application-specific remedial
    action here.
}
```

Note: Using an organization entity GUID or multiple organization entity GUIDs means that dynamic performers allow references to specific organizational entities, so that they appear in the relevant managed work lists. You should only need to use RQL if you want to offer work based on capabilities, privileges or intersections of organization entities. In most cases, writing a script to identify the organization entity GUIDs you need and populating the performer fields with these are much more efficient than using RQL.

For example, the annotated process extract below shows part of a purchasing process that contains a user task to approve a requisition. The process requires that if the requisition value is less than \$50000, this task can be performed by an Accountant. If it is more than or equal to \$50000, it must be performed by the Finance Manager.



The Select Requisition Approver script task assigns a value to the Req_approver performer data field, based on the value of the requisition which is held in Req_value

(and which we assume to have been defined earlier in the process).

The Approve Requisition task assigns the work item to the Req_approver performer data field - the value of which is either "Accountant" or "Finance_Manager" (both of which are also defined as participants for the process).

What to do next

Using a performer field, you can deliver work dynamically to an organizational entity, so that the work items appear in managed work lists.

Dynamic Organization Participants

Dynamic Organization Participants can be created as external references to an organization entity within a Dynamic Organization. These participants can be assigned to a task like any other participant. Dynamic Organization Participants allow the creation of references to dynamic organization entities, that are completed when the organization model is deployed.

A Process Task can reference multiple participants. However, if any of those participants are Dynamic Organization Participants, then all Dynamic Participants of that task must reference entities within the same Dynamic Organization.

When a Dynamic Organization Participant is assigned to a task you need to identify the correct instance of the Dynamic Organization to use. You do this using **Dynamic Organization Identifiers**. The process data fields and parameters that provide these values have to be mapped to these identifiers.

The User Task component passes a reference (Organization Entity Reference) to the Organization Participant for work allocation.

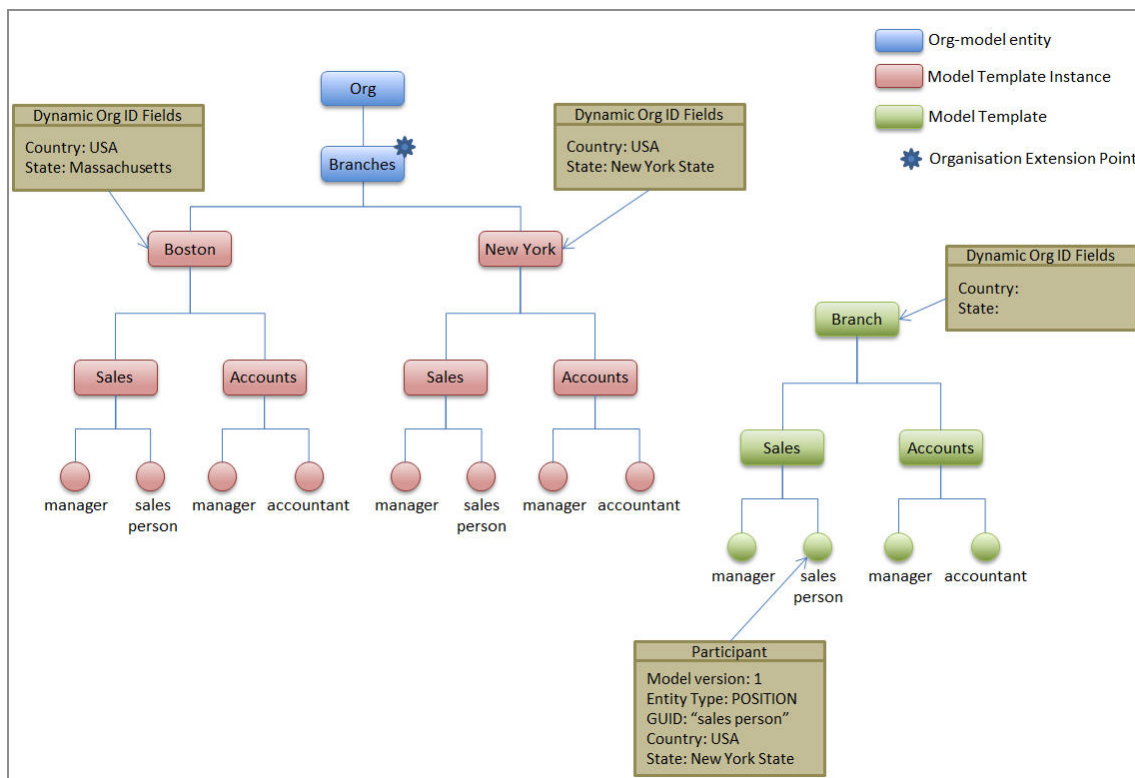
Organization Entity References consist of:

- an Organization Model Major Version - the major version of the Organization Model in which the entity resides.
- an Entity Type - the type of Organization Entity expected; for example, GROUP, POSITION, ORGANIZATION_UNIT.
- a GUID - the unique identifier of the Organization Entity (only unique within a given major version of the Organization Model).
- The value of the **Dynamic Organization Identifier** fields.

The Organization Entity Reference is used to locate the Dynamic Organization Model, and use the Dynamic Organization Identifier field values to identify the Dynamic Organization Model instance, under which it locates the corresponding entity.

As an example; the figure below shows an Organization Model with a Dynamic Organization Model, the root node of which is named **Branch**. **Branches** is the name of the extension point. There are two Dynamic Organization Model instances, named **Boston**, and **New York**. A Dynamic Organization Participant references the **sales person** Position within the Dynamic Organization Model (the GUID would actually be a generated value, but the name has been used here for clarity).

At runtime, the values of Dynamic Organization Identifiers **Country** and **State** identify the particular instance of the Dynamic Organization Model as the **New York** instance. It is from that instance that the Position **sales person** is selected for the work allocation.



Creating a Dynamic Organization Participant

The Organization Units and Positions within a Dynamic Organization might be assigned to Process Participants - known as Dynamic Organization Participants.

Procedure

1. In the Project Explorer, expand the package where you want to add a participant.
2. Right-click **Participants** and select **New > Participant**. The New Participant dialog box is displayed.
3. Specify the **Label** and **Name** of the participant.
4. To create an external reference to a dynamic organization model, select **External Reference**, and click the picker to select a type from the dynamic organization model. Choose a type from those shown in Matching Items, or key in the first few characters of the name you are looking for in the field under **Select type(s)(? = any character, * = any string** and choose from those shown. Click **OK**. The dynamic organization participant that you created appears in the Project Explorer.
5. Click **Finish**

What to do next

When a Dynamic Organization Participant is assigned to a task you need to provide information to identify the correct instance of the Dynamic Organization to use to resolve this participant. See the "Dynamic Organization Identifier Mapping" topic in the *Organization Model Guide* to identify the correct instance of the Dynamic Organization in which the participant can be found at runtime.

Auditing Process Data

By default, process data does not appear in the audit trail (that is, in Audit) for a process instance. You can configure a process so that certain data fields appear in the audit trail.

This capability makes use of the process data to work item attributes mapping facility. For example, if a *work list facade* has been defined, and display labels were assigned to Attributes 3 and 8, the labels would be shown in Audit rather than "Attribute3" and "Attribute8".

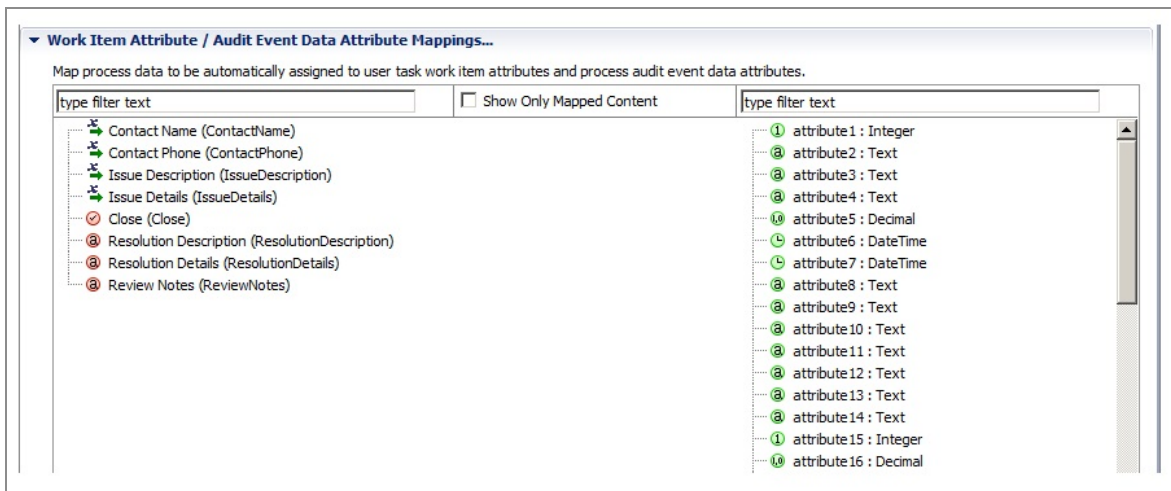
Note: Audit displays audit entries across all processes on the system, and therefore it is usual that specific attributes are used for the same data items across all processes. For example, you might want to consistently use Attribute2 for the value of 'Order Id'. Creating and sharing the same work list facade project across all BPM applications on the system can help to achieve this consistency.

To configure your system to show process data in Audit, you must map process data fields to the desired work item attributes. This mapping process is the same as the process for mapping data fields to work item attributes when creating a work list facade. If you have defined a work list facade, and you map process data fields to attributes for which you have specified a display label in the facade, those custom labels also appear in Audit for the process data entries.

Procedure

1. Open your process and select it in the Process Editor.
2. In the **Properties** view, select the **Work Resource** tab, then expand **Work Item Attribute / Audit Event Data Attribute Mappings**.

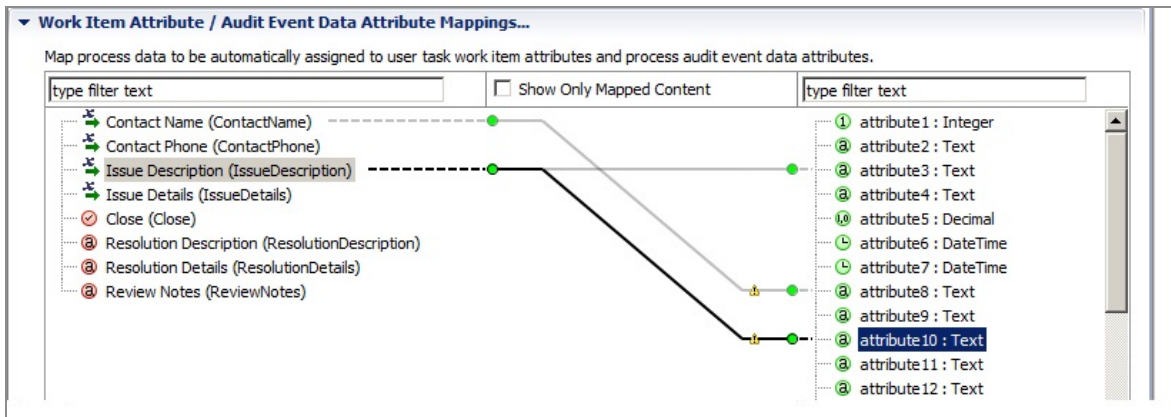
This displays all of the data fields defined for the process (in the left column) and all of the work item attributes (in the right column). For example:



If you have a work list facade defined, the list of attributes in the right column shows the display labels for those attributes that were mapped to data fields in the facade.

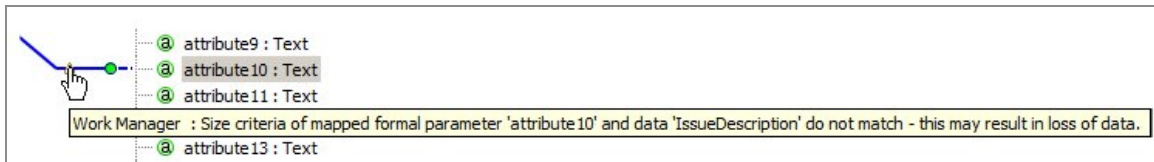
3. Map the data fields that you want to appear in Audit to attributes that are of the same data type.

For example:



As shown in this illustration, data fields can be mapped to multiple attributes, if desired.

Note that if you map a data field to an attribute of a smaller size, a warning marker is shown that indicates this:



This might result in data shown in Audit being truncated to the length of the attribute.

Correlation Data

Correlation data must be used to ensure that each global signal event is received by the process instances to which it applies. Global signals are designed to be caught by multiple process instances. If there are multiple process instances with matching correlation data, then the signal is triggered in each one.

See [Sending Signals Between Processes](#) for more details.

Correlation data can also be used instead of the process instance ID to ensure that an incoming request event or task is triggered in the correct process instance. See [Controlling Flow From an External Application](#) for more details.

To work with correlation data:

- Decide whether elements within the incoming data can be used to uniquely identify instances of this process. If the incoming data can be used to identify the process, create a correlation data field in the process and the global signal payload definition. In the catch global signal event map these correlation fields together.
- In a throw global signal map, the data that identifies the required process instances to the correlation field in the signal payload.

Creating Correlation Data

Correlation data must be used to ensure that each global signal is received by the process instance to which it applies.

You can create correlation data fields in the following ways:

- using the Properties view table provided when the **Correlation Data** folder is selected in Project Explorer (this method is recommended if you need to create several data fields or parameters).
- using the wizard available either from the Project Explorer or from the **File** menu.
- refactoring an existing data field into a correlation data field (right-click the data field and select **Convert to Correlation Data**).
- copying a data field and pasting it under the **Correlation Data** folder in the Project Explorer.

Creating Correlation Data Using the Wizard

This is applicable to the use of global signals only.

Procedure

1. In the Project Explorer, expand the process where you want to add a correlation data field.
2. Right-click **Correlation Data** and select **New > Correlation Data**.

The New Correlation Data Field dialog box is displayed.

For information about how to complete the remainder of the fields in the wizard pages, see [Using the Wizards to Create a New Data Field or Parameter](#) (the properties of a correlation data field are exactly like those of a standard data field).

Result

The correlation data that you created appears in the Project Explorer.

- ✓ **Tip:** If a correlation data field has data that you want to continue to use in your process, but that you no longer want to use for correlation, you can convert the correlation data field to a "standard" data field, by right-clicking it and selecting **Convert Correlation Data to Data Field**.

Creating Correlation Data Using the Properties View

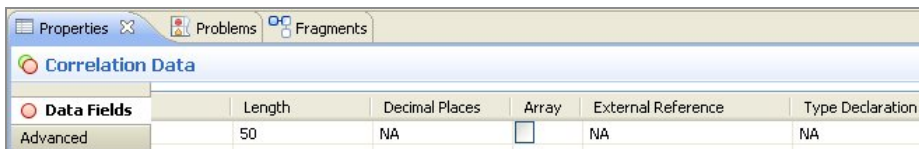
This is applicable to the use of global signals only.

Procedure

1. In the Project Explorer, expand the process where you want to add a correlation data field.
2. Click **Correlation Data** in the Project Explorer.

The Properties view is displayed.

Click . A new correlation data field is added.



Data Fields	Length	Decimal Places	Array	External Reference	Type Declaration
Advanced	50	NA	<input checked="" type="checkbox"/>	NA	NA

As appropriate, modify the properties.

- i Note:** To prevent accidental reassignment of correlation data fields, they are set to read-only by default. If you want to assign a value to correlation data that does not come from an incoming signal, deselect the **Read Only** check box.

Associating Process Data with Events and Tasks

There are many reasons to associate process data with tasks or events. For example, data field needs to be displayed in a form or a process cannot be started unless a particular parameter is passed to a start event.

Using drag-and-drop to Associate a Data Field or Parameter with an Activity

i Note: You can drop process data onto a blank part of the process to create a new user task with the associated process data. For more information, see [Using Participants to Create Tasks](#).

Procedure

1. In the Project Explorer, click the data field or parameter you want to associate with your activity, holding down the mouse button, drag the pointer to the user task and release the mouse button.

i Note: You can select multiple data fields or parameters for drag-and-drop operations using the Ctrl (for single selection) or Shift (to select a range) keys.

2. A menu is displayed with the following options:

- **Add Data To View And Assign**

Selecting this option adds the process data as an **In/Out** parameter on the **Interface** tab. This means that in the runtime environment, users can display the associated form to view the field and can also assign new values to it.

- **Add Data To View**

Selecting this option adds the process data as an **In** parameter on the **Interface** tab. This means that in the runtime environment, users can display the associated form to view the field. However, they cannot assign new values to it.

- **Add Data To Assign**

Selecting this option adds the process data as an **Out** parameter on the **Interface** tab. This means that in the runtime environment, users can display the associated form to assign new values to the field.

Using the Interface Tab to Associate a Data Field or Parameter with an Activity

Procedure


1. Select the event or task.
2. In the Properties view, click the **Interface** tab.

The screenshot shows the Properties view for a Task in TIBCO Business Studio. The left sidebar lists various tabs: General, Description, Interface (selected), Data Fields, Scripts, Data References, Appearance, Extended, and Resource. The main area shows the 'Interface' tab with a 'Visibility' section containing 'Private' (selected) and 'Public' radio buttons. Below this is a 'Parameters' section with a dropdown arrow. A text label reads 'Select a subset of data that is accessible for this activity.' followed by a checkbox 'No interface data association required.' which is currently unchecked. Below the checkbox is a table with four columns: 'Process Data Name', 'Mode', 'Mandatory', and 'Description'. The first row of the table contains '[All Process Data]' in the 'Process Data Name' column, and the other three columns are empty. There are two additional empty rows below it.

Process Data Name	Mode	Mandatory	Description
[All Process Data]			

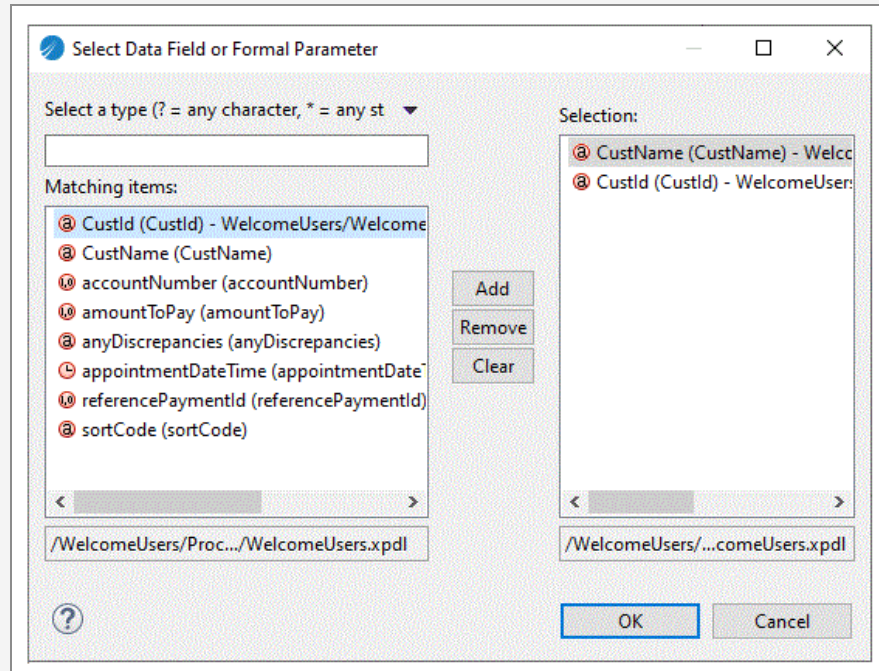
3. Select whether you want the visibility of the event or task to be private or public - see [Setting Event and Task Visibility \(Private and Public\)](#).
4. Choose between the following:
 - Select the check box **No interface data association required** if you do not want to associate process data with the event or task. This deletes any associations that already exist.

i Note: Usually if no data is explicitly associated with an event or task, then all available data is implicitly associated. Selecting this option prevents this, which can be useful for example when you wish to define an incoming message event with no parameters, even though the process already has formal parameters for other reasons.

- Click  to select the process data that you want to associate with the event or task. The Select Data Field or Formal Parameter dialog box displays the list of available process data:

Note:

- By default, all process data is available to a task. When you explicitly associate process data with an event or task, only the process data you associate with the event or task can be used by that task.



- The process data displayed depends on what type of event or task is selected. Most events and tasks can have both data fields and formal parameters associated with them, however receive tasks and events of type None can have only formal parameters associated with them.

- The process data you select is added to the table of data. Select whether you want the data to be mandatory.

- Note:** The mandatory setting on the Interface tab for a formal parameter overrides the mandatory setting in the Properties view for the formal parameter. This allows complete freedom in designing the process - you can define a formal parameter as mandatory in one place in a process, and optional in another.

Use the space provided if you want to add an optional usage description of the process data. Selected parameters also display their mode (In, Out, or In/Out). You can change the mode by selecting from the drop-down list.

Using Process Data to Create Tasks

If you have created data fields or parameters, you can drag them to a blank area of a process to automatically create a user task.

Procedure

1. In the Project Explorer, click the data field or parameter, and holding down the mouse button drag the pointer to a blank part of the process and release the mouse button.



Note: You can select multiple data fields or parameters for drag-and-drop operations using the **Ctrl** (for single selection) or **Shift** (to select a range) keys.

2. A menu is displayed with three options:
 - **Create User Task To View And Assign Data:** Selecting this option creates a user task with the process data as an **In/Out** parameter on the **Interface** tab. This means that in the runtime environment, users can display the associated form to view the field and can also assign new values to it.
 - **Create User Task To View Data:** Selecting this option creates a user task with the process data as an **In** parameter on the **Interface** tab. This means that in the runtime environment, users can display the associated form to view the field. However, they cannot assign new values to it.
 - **Create User Task To Assign Data:** Selecting this option creates a user task with the process data as an **Out** parameter on the **Interface** tab. This means that in the runtime environment, users can display the associated form to assign new values to the field.

Using Business Data

Business Object Models

A business object model is a set of business terms and relationships specific to your corporate environment (for example, in a financial environment, broker, counterparty, and so on).

The Eclipse editor called the Business Object Model Editor helps you construct your business object model. In object-oriented terms, when you create a business object model, you are creating a class diagram using UML.

The advantage of creating or importing a business object model is that you can use it:

- for analysis purposes,
- for documentation purposes,
- to incorporate business data in your business processes.

You can create a Data Field in a Process that corresponds to a Class that you have defined in the business object model. If you create a data field in a process and set the field type by an external reference to a Class in the business object model, the field has sub-fields that correspond to the attributes defined for that Class.

Diagram Nodes

Business object model Diagram Nodes consist of BOM diagrams, BOM classes, primitive types, and attributes.

See [BOM Diagrams](#), [BOM Classes](#), [Primitive Types](#), and [Attributes](#).

When creating diagram nodes, note that:

- When you create a business object model in the Business Object Modeler, the Business Object Modeler automatically creates a package of the same name in which to place your diagram nodes. Therefore, there is no need to create a package in which to place all your diagram nodes, unless you want to create extra packages in which to group diagram nodes. For example, you might wish to create a package to contain all your Primitive Types.

- The Business Object Modeler uses the following Java programming conventions.
 - Business object names must not be a Java keyword.
 - Business object names should follow the Java Naming Convention.

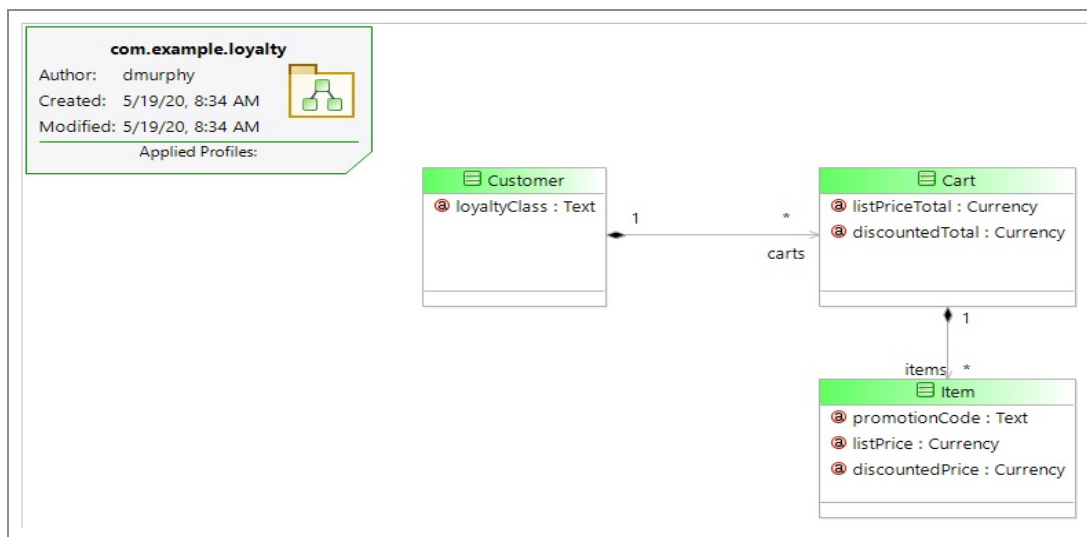
See <http://java.sun.com/docs/codeconv/> for more information about these.

These Java Programming Conventions are options under Preferences, so you can choose to disable them if you require.

BOM Diagrams

You use the BOM diagram editor to define your application data in the form of classes (representing complex data types), enumerations (predefined sets of values) and primitive types (specialized simple data types).

For example:



BOM Classes

A class is a description of a set of properties that, when grouped together, create a meaningful unit.

Classes can be organized in a hierarchical structure.

For example: *Department*, *employee*, *purchase order*, and *inventory item* are all classes.

Primitive Types

A Primitive Type is a data type. By defining a Primitive Type, you can define your own data types and then specify how data of that type is interpreted.

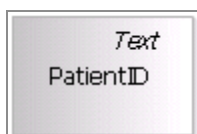
You can specify what values the data can have or any constraints on the data, for example. You can refine your data to your specific business domain. You can create a Primitive Type called Patient ID and specify that Patient IDs should always be a maximum of 7 characters long, and consist of 2 letters followed by 5 numbers, for example. See [Setting Restrictions on Primitive Types and Attributes](#) on page 10 for more information on the restrictions you can set.


All the Primitive Types that you define must be based on the following standard business object model Primitive Types that are available in the business object model Editor. Some types have subtypes. The types are described in the following table.

Primitive Type	Description
Boolean	A value of True or False.
Date	A date in the format dd/mm/yy.
Date Time and Timezone	A date (in the format dd/mm/yy hh:mm) and time zone.
Number	Any whole number, positive or negative.
Text	Any characters can be entered up to the length you specify.
Time	A time in the format hh:mm (24-hour clock).
URI	A Uniform Resource Identifier.

For information on constraints on field or parameter values, see [Value Spaces for BOM Native Types](#)

When you create a Primitive Type, it defaults to Text. The type is displayed above the name of the Primitive Type in italics as shown below.



You can change the standard type on which the Primitive Type is based by clicking on the  button in the **Superclass** field, on the General tab in the Properties View. Additional information required by the Primitive Type is specified on the Advanced tab; see [To Set Restrictions on Primitive Types and Attributes](#).

To Set Restrictions on Primitive Types and Attributes

You can specify restrictions for Primitive Types and Attributes. To do this, in the **Properties View** for the Primitive Type or Attribute, select the **Resource** tab and expand the **Restrictions**.

The restrictions you can specify depend on the type (including subtype where appropriate) of your Attribute or Primitive Type. The following table describes the restrictions you can set for each type.

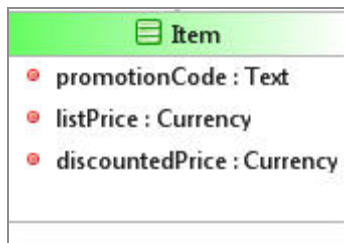
Type	Property	Description
Boolean	Default Value:	A default value of either True or False.
Date	Default Value:	You can specify a value that should be automatically supplied for the type.
Date Time and Time Zone	Default Value:	Enables a value to be automatically supplied.
Number	Default Value:	You can specify a value that should be automatically supplied for the type.
	Lower Limit:	You can specify the smallest number the type should allow.
	Lower Limit inclusive:	You can specify a lower limit within a range of numbers.
	Sub Type:	Either Floating Point (the default) or Fixed Point.
	Upper Limit:	You can specify the highest number the type should allow.

Type	Property	Description
	Upper Limit Inclusive:	You can specify a higher limit within a range of numbers.
Text	Default Value:	You can specify a value that should be automatically supplied for the type.
	Maximum Length:	The maximum length of the value allowed.
Time	Default Value:	You can specify a value that should be automatically supplied for the type.
URI	None	There are no restrictions for these types.

Attributes

Attributes describe the information stored or maintained about a Class.

For example:



The Item Class has the Attributes promotionCode, listPrice, and discountedPrice.

Attributes, like Primitive Types, must be based on one of the standard business object model Primitive Types that are available in the business object model Editor.

Relationships

Relationships are used to show the relationships between objects and consist of associations and composition.

i Note: The terms described in this section are identical in meaning to the same terms described in UML.

Relationships in the business object model can be grouped into two types:

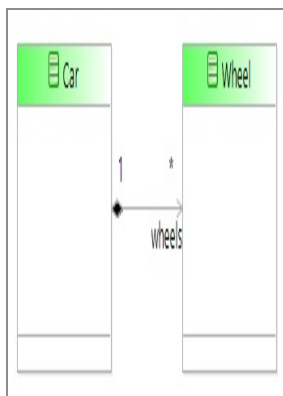
- **Associations**, and **Composition**. Composition is a more specific type of the general relationship type Association.



These types can be further classified with regards to:

- End ownership (Composition only).
- Navigability. These relationships might be unidirectional or bidirectional.

All relationships can have various labels when they are created in a business object model. A label name for the relationship is not displayed by default but can be added. They also display the names of the Classes and the multiplicity allowed for the Classes.

This is shown below:

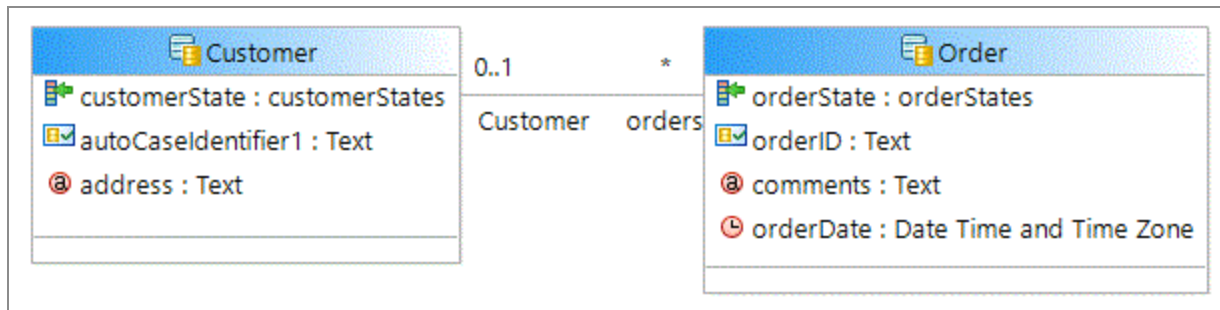


✓ Tip: You can choose to hide these labels by clicking  or you can delete the ones that you do not require from the actual Model. If you delete them from the Model, you are not actually deleting the labels, you are hiding them. You can display them again by clicking .

Associations

Case classes are used for data that is stored and persisted outside of individual processes.

Use association relationships between case classes to relate two types of separate case classes with each other. For example, an Order case type might have an association to a Customer case class.



This then represents the relationship that an Order has a **reference** to a **Customer**.

Use the multiplicity on each end of the association to define the limitations of the relationships. For example, a Customer can be referenced from many Orders but an Order only has one customer.

Also see [Navigating Association Links to find Related Case Objects](#).

Composition

Composition is a specific type of Association used when a Class is a collection or container of other Classes, but the relationship is such that if the Class that functions as the container is destroyed, the Classes representing the contents are destroyed as well.

In the Business Object Modeler, Composition is represented as a filled diamond shape connected to the containing Class.

Compositions can be bi-directional, or can be navigable only in one direction. When you create a composition in Business Object Modeler, by default its navigability is from the source class to the target class.

Note: Bi-directional compositions are not supported.

Business Object Modeler Tasks

The business object model is where you define in business terms the Classes, Attributes, Primitive Types, Associations, and so on that describe your business.

There are a number of common tasks that you perform using the Business Object Modeler.

i Note: You can use quick-find (Ctrl+F) in the project explorer to find existing business object model entities and select them in the project explorer.

When you have created a business object model, you can search for business object model diagram elements within it using quick-find (Ctrl-F) within the diagram, and entering the initial characters of the name you are searching for. Double-click the element you are shown in the search to go to its location in the diagram.

Creating a Business Object Model in an Existing Project

You can create a business object model in an existing project.

Procedure

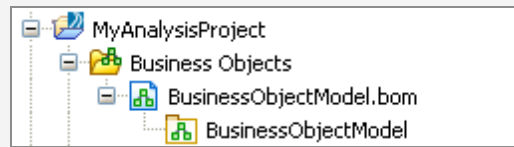
1. In the Project Explorer, right-click the **Business Objects** folder and select **New > Business Object Model**.
2. Name the business object model.

! Warning: If you rename a business object model that has not been saved, any changes you have made are lost.

✓ Tip: Click **Finish** on this dialog box to create a business object model with default settings. Click **Next** to modify the default project settings and create a business object model using a template. Select a template from those available and click **Finish**.

3. In the Project Explorer you can confirm that the business object model has been created.

Note: By default, the Business Object Model name is prefixed by the domain name, as set in **Window > Preferences > User Profile**.



4. To apply a UML Profile to a business object model, see [Applying a UML Profile to a Business Object Model](#).
5. To apply stereotypes to a Model, see [Applying Stereotypes to Business Objects](#).

Business Object Model Editor

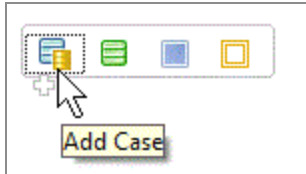
You can add objects to your business object model, and edit them, using the Business Object Model Editor.

Adding Diagram Nodes to a Business Object Model

To create a business object model, you add diagram nodes (Package, Class, Attribute, and Primitive Type) and connect them with Connections (Association and Composition).

You can add diagram nodes and connections in any of the following ways:

- Using the tools on the diagram editor's palette. The palette contains tools that enable you to add a diagram node or a connection either by selecting the required tool in the palette and dragging and dropping on to the diagram, or by clicking on the tool in the palette and then clicking on the diagram.
If there is a stereotype already applied to your business object mode, the palette contains additional tools to create business objects with the stereotype already applied to them.
- Using the pop-up icons. If you hover your mouse over the empty area of the diagram, or over a Package, a pop-up containing icons appears. Move your cursor onto one of these icons to display the label as shown in the following illustration, and then click to create a new node of that type in the diagram.



Similarly, hovering the mouse over a Class displays an icon for an Attribute.

- Using the Project Explorer. You can add new diagram nodes (such as adding a new Package, Class, or Attribute) using right-click menus in the Project Explorer. For example, to add a new Class, you can expand the Project Explorer, select the business object model, and right-click **Add Child > Class**.

The Project Explorer menu lists the objects appropriate to the context. For example, if you select a Class and right-click **Add Child**, the following menu is displayed:



Adding Classes Attributes

You can add classes and attributes to your Business Object Model.

Procedure

1. Using the Class tool, place a Class on the model and name it.
2. Using the Attribute tool, place Attributes within the Class.
3. In the Properties view for each Attribute, select the Type and specify whether there can be multiple values for the Attribute (whether it is an array). (See [Attributes](#)). When you specify multiplicity values in the Properties view for Attributes, you can use content assistance. Press **Ctrl + space** in the field and the available multiplicity values are displayed.

To apply restrictions to your Primitive Type, click the **Resource** tab and expand **Restrictions**.

4. In the **Properties View** for each Class or Attribute, click the **Stereotypes** tab to apply

any stereotypes you want to apply. See [Applying Stereotypes to Business Objects](#) for more information.

Adding Primitive Types

To add a Primitive Type to your business object model, use the Primitive Type tool, drag-and-drop a Primitive Type on the model and name it.



When you create a Primitive Type, it is always created with a standard type of Text. To select another standard type for the Primitive Type:

Procedure

1. In the **Properties View** for the Primitive Type, select the **General** tab.
2. Click the picker to display the Select Type dialog box. To display a list of the available types you can set, type ? in the **Select Type(s)** field. A list of the available standard types is displayed.
3. From the **Matching Items** box, select a standard type and click **OK**.

Result

To apply stereotypes to the Primitive Type, click the **Stereotypes** tab. See [Applying Stereotypes to Business Objects](#) for more information.

To apply restrictions to your Primitive Type, click the **Resource** tab and expand **Restrictions**.

Connections

Connections indicate the relationships between Classes and include Association and Composition.

See [Creating an Association between Two Classes](#)

See [Composition](#)

Creating an Association between Two Classes

You can create an association between two classes. An association is a relationship between two classes.

Procedure

1. Click the Association tool on the Palette.
2. Drag from one Class to the other. This creates a bi-directional Association. You can change the direction of the Association by selecting from the **Navigability** drop-down list in the Properties view. You can also change the **Source** or **Destination** of the Association in the Properties view.

Note: A bi-directional Association shows both the **Source** and **Destination** on the business object model. A unidirectional Association shows only the **Destination**.

You can also set the Multiplicity of each end of the Association by selecting from the **Source Role Multiplicity** and **Target Role Multiplicity** drop-down lists in the **Properties** view.

3. In the Properties View for each Association, click the **Stereotypes** tab to apply any stereotypes you want to apply. See [Applying Stereotypes to Business Objects](#) for more information.

Composition

A Composition is created in the same way as an Association, and shares the same properties. The only difference is the meaning and appearance of the Connection.

See [Creating an Association between Two Classes](#).

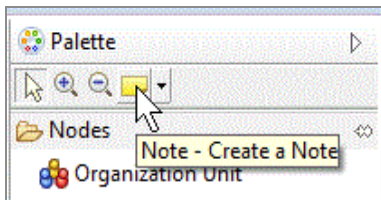
Adding Text to a Model

You can add a note or text to your business object model to describe the business object model or to add any supporting explanations to the diagram nodes.

Note: When adding text or a note, press Ctrl+Enter to start a new line.

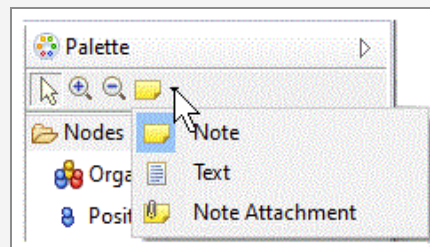
To add a note

- Right-click the Business Object Model Editor and select **Add > Note**.
- In the Business Object Modeler palette, select the Note tool in the upper part of the palette.



Then click in the Business Object Model Editor.

- i Note:** You can also select Text or Note Attachment from this menu. The icon for the tool changes to show the item you have selected.



A Note is displayed where you can enter any text you require. Use the **Note Attachment** option to draw a line connecting a Note to the Class it comments on.

To add text

- Right-click in the Business Object Model Editor and select **Add > Text**.
- In the Business Object Modeler palette, select the Text tool in the upper part of the palette and then click in the Business Object Model Editor.

A Text box is displayed where you can enter any text you require.

Adding a Child Diagram to a Business Object Model

You can add a child diagram to a Business Object Model, to allow you to reuse part of a parent diagram, and focus on that part of the diagram.

Procedure

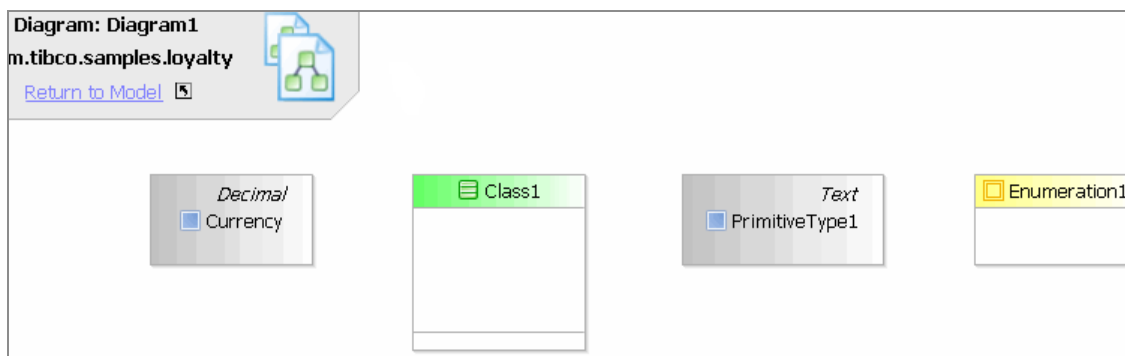
1. From the Diagrams folder under the Business Object Model file in Project Explorer, select **Add Child > Diagram**.

You can also make this selection from the Business Objects folder (**New > Other > Other > Diagram**, and select which Business Object Model to use), or from the .bom file: **New > Diagram** or **New > Other > Other > Diagram**.

2. From the New Business Object Model Diagram wizard, name the child diagram, and select the elements of the Business Object Model which you want to include in your new diagram.

Note: The elements you can include in your diagram are Class, Primitive Type and Enumeration. **Packages are not available for you to include.**

3. Click **Finish**. Using the elements selected above, your child diagram looks like this:



Adding Elements to the Child Diagram

To add elements to a child diagram after you have created it, you can drag-and-drop Class, Primitive Type or Enumeration elements from the parent Business Object Model in Project Explorer (you can drag-and-drop from other Business Object Models but then you can only create copies or subclasses).

Drag the element, and when you drop it, you can see a dropdown menu:

- To create a view of the element in this diagram, select **Create View**. This means that you can see the element in this diagram, as well as in the main (parent) Business Object Model diagram.
- To make a new copy of the element in the diagram select **Copy Elements**. In this case, a new copy of the element appears in this diagram, and also in the main

(parent) Business Object Model diagram

- (when dragging and dropping a Class) To create a subclass of the selected class, select **Create Subclasses**. In this case, the subclass appears in this diagram, and also in the main (parent) Business Object Model diagram.

You can also use the Palette to add Elements, Children, and Relationships to the diagram, by dragging and dropping the elements from the palette onto the child diagram. When you do this, the elements are new elements, and appear in the child diagram and also in the main (parent) Business Object Model diagram.

Deleting Elements from the Child Diagram

You can delete elements in the child diagram from either the diagram or the model.

Procedure

1. Select the element, and select **Delete from Diagram** or **Delete from Model**.
 - If you select **Delete from Diagram**, the elements are only deleted from this child diagram.
 - If you select **Delete from Model**, the element is deleted from the child diagram and also from the main (parent) Business Object Model diagram.

Result

You can also delete the element using the Delete key. This has the same action as **Delete from Diagram**, that is, only the graphical view is deleted.

Business Data Project Deployment

When you create a Business Data Project it contains one or more BOM which you then deploy as part of the project.

The error you receive is similar to the following:

You must deploy data projects before you deploy the process projects that depend upon them. If you change the BOM models in a data project, you must re-deploy it before deploying changed process projects that depend upon it, otherwise you receive an error when deploying the process project.

Business Data

Business data is structured data that contains information about real-world entities that an organization deals with, for example Customer, Order, and Orderline.

Each of these entities have a number of attributes. For example, name, address, and date. These entities are also related to each other in different relationships and with different multiplicities.

TIBCO Business Studio™ - BPM Edition provides a tool, Business Object Modeler, that allows you to build up a description of the business data that the process manipulates. You can interact with business data (any BOM-based data) from any activity in a Process (for example, a user task, script task, and so on). The resulting Business Object Model (BOM) contains the different types of objects that the business uses, their attributes, and their relationships to each other. For more information on using Business Object Modeler, see *TIBCO Business Studio Modeling Processes Guide*.

i Note: TIBCO Business Studio™ - BPM Edition supports two sorts of business data - normal business data and case data. This guide describes the use of normal business data. Additional information about case data is provided in the *TIBCO Business Studio Case Data User's Guide*.

Case Data Manager (CDM) is the component responsible for handling business data in TIBCO Business Studio - BPM Edition.

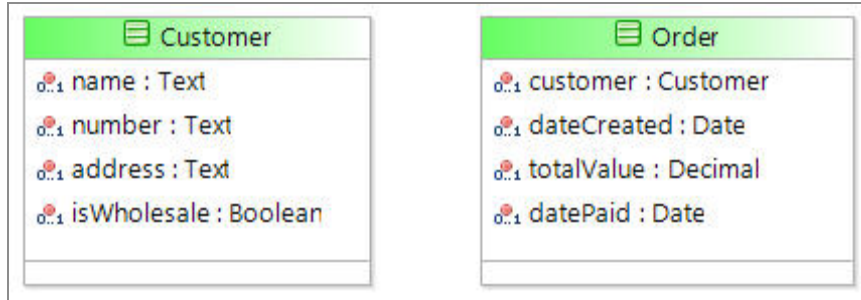
Object Orientation

The BPM runtime is an Object Oriented (OO) system, as is the scripting environment that supports it. The topics in this Introduction describe important concepts that stem from the use of this OO design. It is important to understand these concepts to achieve the desired behavior when using scripts to manipulate business data.

i Note: If you have difficulty understanding any of these concepts, see the examples described in [Business Data Scripting](#), then return to this section. One of the best ways to understand these concepts is to learn by example.

BOM Class

As the name implies, the whole focus of OO is on objects. Objects are created to represent real world things such as a customer or an order.



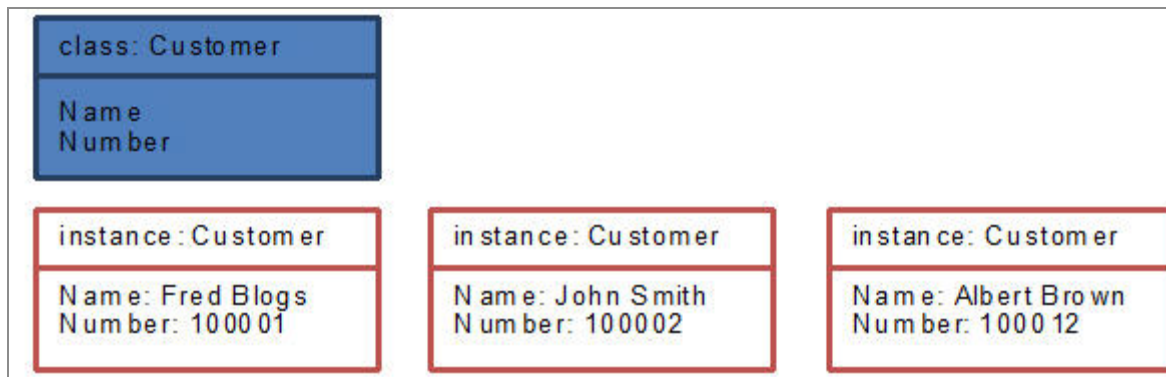
However, before we can have objects we must create a template, or pattern, for each type of object we want to process. This template is called a *BOM class*, and defines what the different objects are like. For example, you can define a Customer BOM class and an Order BOM class, which would model real world customers and orders. Creating BOM classes for your application is a design-time activity.

Business Objects

At runtime, instances of these BOM classes are created to represent particular instances of the generic BOM class. These instances are referred to as Business Objects. For example, the Customer BOM class can have two Business Objects to represent two actual customers, John Smith and Fred Blogs.

Each of these BOM classes and Business Objects have attributes. For example, a Customer might have attributes that include a name and number, and an Order might have attributes that include Customer and DateCreated. Although Customer attributes differ from Order attributes, all Customers have the same set of attributes.

The diagram below shows a simple Customer class and three Business Objects, which are instances of the class:



Business Object Scope

The lifespan of a Business Object is typically bounded by the Process Instance in which it is created. This is called Process Local Scope.

It is possible in a service task to write information out to, or read from, a database. The structure of the database table might match one of the normal BOM classes, or the data in one of the normal BOM classes might have to be mapped into a Business Object that matches the database table structure depending on the implementation.

Business Object Creation by Factory

Business Objects are created by factories. There is a factory method for each class (for example, createCustomer, createOrder, and so on) within the BOM.

The factory methods have to be used when a Business Object is created. This happens *implicitly* when objects are created within User Tasks, but has to be done *explicitly* when an object is created in a script. See [Business Data Scripting](#) for more information about factories.

Retrieving and Setting Business Object Attributes

Using Business Data Scripting capabilities, retrieving Business Object attributes, for example the attributes of a customer instance, is as simple as running a script:

```
var custName = data.customerInstance.name;
```

You can set the value of a Business Object attribute as follows:

```
data.customerInstance.name = "Clint Hill";
```

Invoking Operations on Business Object Attributes

As well as attributes, some classes have methods which perform operations on the object.

For example, the String class is used to represent a Text attribute's value, such as the name of a customer. Two of its methods are toUpperCase() and toLowerCase().

```
lowercaseName = data.customer.name.toLowerCase();  
uppercaseName = data.customer.name.toUpperCase();
```

In this example, if the Text attribute `data.customer.name` was "Fred Blogs", then the first assignment would set the `lowercaseName` variable to the value "fred blogs", and the second assignment would set the `uppercaseName` variable to the value "FRED BLOGS". In some cases, such as the example cited above, methods can return values. In other cases, methods can alter some of the attributes of the instance. It is important to know how the methods behave when you are using them.

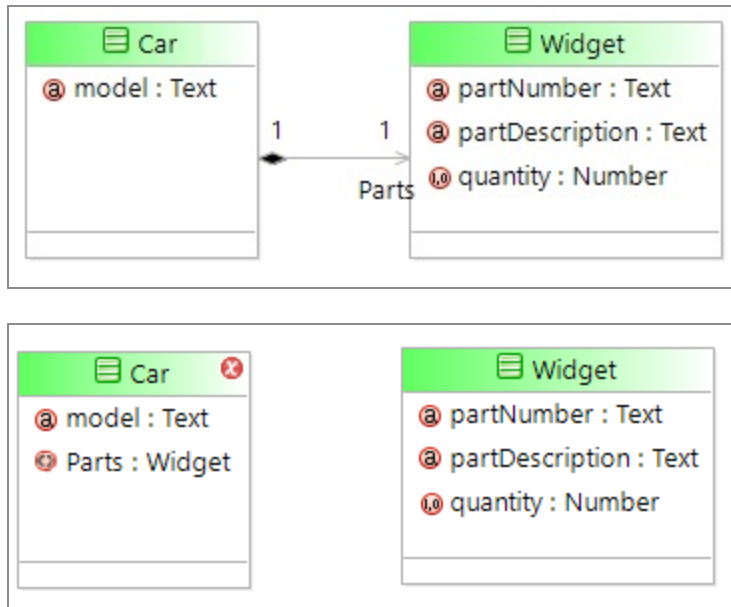
BOM Relationships and Process Data

This section describes how BOM relationships are honored by process data.

Composition

The Composition relationship is used to model the concept that "X is composed of Y". For example, a Car is made up of a number of Widgets.

This can be drawn in either of the following ways in the Business Object Modeler:



- In the first example, there is a line drawn between the Car and Widgets, which is labeled "parts".
- In the second example, there is a "parts" attribute in the Car class.



Note: At runtime, these two approaches are treated identically. It does not matter which is used.

The Composition relationship is also known as the Containment relationship.

Assignment by Value and by Reference

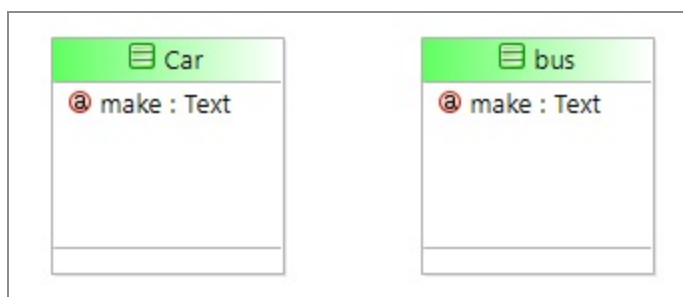
Assignments can be made either by reference or by value. For an assignment *by reference*, the entity to which the assignment is made refers to the entity being assigned. In other words, subsequent changes to the entity by either the new or existing reference is reflected in both places (there is only one entity; it isn't copied). However, for a *byvalue* assignment, a copy of the assigned entity is made, and that is what is applied to the entity receiving the assignment. This results in two independent objects. Therefore, changes in one place do not affect the other.

Assignment Conventions

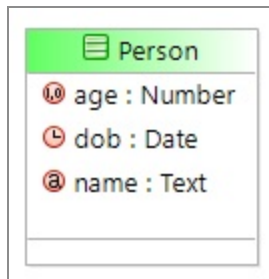
Assignment to...	...of type...	...behaves as follows:
Business Object attribute or composition	BOM Native or Primitive Type (except Date, Time, DateTime)	Effectively by value. For efficiency's sake, objects are only copied where they are mutable (where their internal value can be changed). In other words, by reference behavior is sometimes used, but always behaves like by value. See BOM Native Type or Primitive Type Object to Business Object Attribute .
	BOM Class	By reference. Important note: Two BOM attributes can be programmatically set to refer to the same object instance, and the changes are reflected in both places. However, once the script is complete, the data and single value stored thereafter behave as two independent objects.
Process Data Field	BOM Class	By reference. See Assigning a Business Object .
	Basic Type (except Date, Time, DateTime)	By value. See Assigning a Basic Type Object to a Process Data Field .

BOM Native Type or Primitive Type Object to Business Object Attribute

Assigning a BOM Native Type or Primitive Type object to a Business Object attribute is by value:



```
data.car.make = data.bus.make;
data.bus.make = "Ford" // does not affect car.make
```

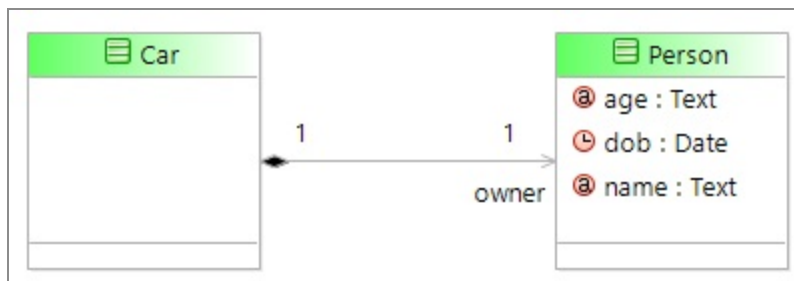


```
data.person1.age = data.person2.age;
data.person2.age = data.person2.age + 1;
```

```
data.person1.dob = new Date("1968-01-04");
data.person2.dob = data.person1.dob;
data.person2.dob.setFullYear(1970); // person1.dob is now 1970-01-04;
```

Assigning a Business Object

Assigning a Business Object is by reference.



```
data.personDataField = data.car.owner; // Business Object assigned to data
                                     // field by reference
data.personDataField.age = 25;        // Also affects car.owner.age
```

```
var tempPerson = data.personDataField; // Business Object assigned to
// local variable by reference
data.tempPerson.name = "Bob";          // Also affects personDataField.name
```

```
var owner = factory.com_example_refval.createPerson();
data.car.owner = owner;
data.owner.name = "Ludwig"; // Also affects car.owner.name;
```

i Note: You can make a copy of the object first using the `scriptUtil.copy(...)` utility.

In the next example, although all assignments are by reference (there is only ever one Address), the final line of the script removes the Address from Customer, leaving Customer with no Address:



```
var address = factory.com_example_refval.createAddress();
data.customer.address = address;
data.account.address = customer.address; // They both point to the same
object
```

If this script is modified to use `scriptUtil.copy(...)`, account and customer end up with independent copies of the address:

```
var address = factory.com_example_refval.createAddress();
```

```
data.customer.address = address;
data.account.address = bpm.scriptUtil.copy(customer.address);
// Creates an independent Address
```

Assigning a Basic Type Object to a Process Data Field

Assigning a Basic Type object to a process Data Field is by value.

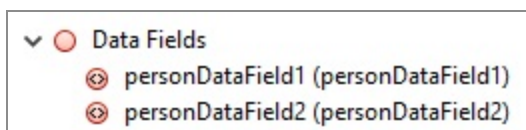
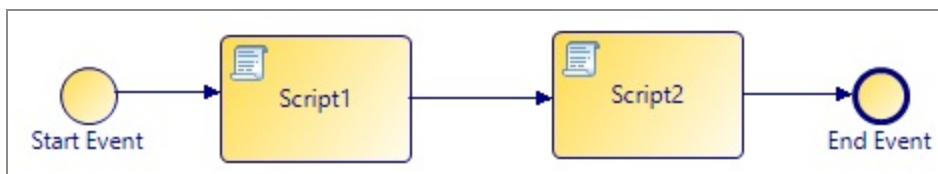


```
var greeting="Hello";
data.greetingDataField = greeting;
data.greetingDataField = "Goodbye"; // does not affect local variable
// greeting
```

Significance of the Script Boundary

When a script completes, all process Data Fields are independently converted to a form that can be stored in a database. Therefore, in a later script, modifying a value in one place never affects the other, regardless of whether a by reference assignment occurred in an earlier script.

This is illustrated by the following two scripts, which can be assumed to run one after the other, operating on the same process Data Fields:



Script 1:

```
data.personDataField1.age = 20;  
data.personDataField2 = data.personDataField1; // Both data fields now //  
refer to same object  
data.personDataField1.age = 35; // This affects both data fields
```

Script 2:

```
// Each data field now refers to an independent Person object  
data.personDataField1.age = 40; // do not affect  
// personDataField2.age (still 35)
```

Business Data Definition

This section describes how you define business data in the Object Model Editor, and the types of data it supports.

Business Data Projects

Business Data projects are used to store business data models that can be referenced by process projects.

A Business Object Model (BOM) can contain two types of data:

- **Case data** - Case data classes describe the primary data objects that are persisted outside of individual processes.
- **Normal (Local) data** - Local data classes describe data that either describes secondary data content of case classes data that is persisted only in individual process instances.

Business Data Project Versioning

Version numbers are set on the **Properties > Lifecycle** dialog box of the Business Data project.

The default format for a Business Data project's version number is:

major.minor

where:

- *major* defines the major version number of the project.
- *minor* defines the minor version number of the project. This is incremented automatically after a project has been 'locked for production' and subsequently a new draft is created.

You can only make backwards compatible changes to a BOM that has already been deployed. You cannot remove data types for example. If you want to make changes like this to a deployed BOM, you must increment the BOM's major version, re-deploy it, and then redeploy all projects that reference that BOM. This can be done from the **Properties > Lifecycle** dialog box of the Business Data project.

BOM Native Types

A number of BOM Native Types are supported by BOM Editor.

- Boolean
- Date
- Date Time and Time Zone
- Fixed point number. A fixed point number allows you to specify the number of decimal places. This is defined in TIBCO Business Studio - Cloud BPM Edition.
- Floating point number. A floating point number does not allow you to define decimal places.

Both number native types have additional methods like add/subtract/multiply/divide. TIBCO recommends that wherever arithmetic precision is important (for example, financial calculations) these operations are used instead of $+/-/*//$ operators. See [Working with Numeric Types](#).

- Text
- Time
- URI

Using Business Object Modeler, these BOM Native Types can be used to generate Primitive Types that have some application-specific restrictions, for example, a range must match.

Value Spaces for BOM Native Types

The value space of the different BOM Native Types is shown in the following table.

Value Spaces for BOM Native Types

BOM Native Type	Value Space
Boolean	true, false
Date	Year in range [-999,999,999 – 999,99,999] Month in range [1 – 12] Day in range [1 - 31] (dependent on month & year) For example: "2011-12-31"
Date Time Timezone	Date and Time fields according to Datetime type above, but timezone is mandatory For example: "2011-12-31T23:59:59Z" or "2011-12-31T23:59:59+05:00"
Floating Point number	Any decimal number that fits into 15 digits.
Fixed Point number	Any decimal number that respects number length and decimal places defined in TIBCO Business Studio - Cloud BPM Edition.
Text	Arbitrary length of long text string of Unicode characters For example: Fred Blogs
Time	Hour in range [0 – 24] (for value 24 minutes and seconds must be 0) Minute in range [0 – 59] Second in range [0 – 60] (NB 60 only allowed for a leap second) For example: 12:34:56
URI	Any valid URI. For example: http://www.tibco.com

BOM Design-time Model

The BOM Design-time model is used to define your business object model, and is created using a set of tools in the Business Object Modeler.

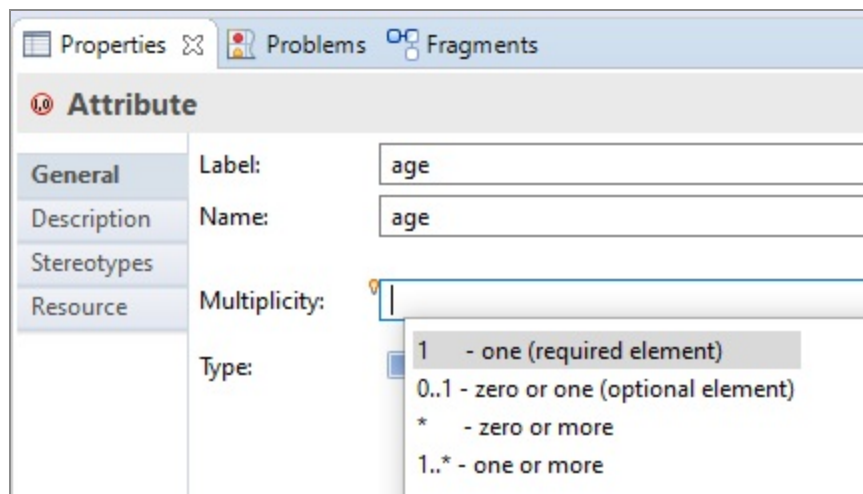
Primitive Types

In a BOM, it is possible to define your own Primitive Types, which are specializations of BOM Native Types. Usually, the value space of a user-defined Primitive Type is restricted in some way, for example, a type called Hour can be defined, based on the Number type with the values restricted to numbers between 0 and 23. The sub-type and restrictions are defined on the Property tab for Primitive Types.

Multiplicity

When BOM class attributes and compositions are defined, the developer can define how many values the attribute can or must have.

When defining the multiplicity for an attribute, content assist is available by typing Ctrl-SPACE. You can then select one of the following options:



It should be noted, however, that multiplicity can have other values, such as the following:

Multiplicity	Meaning
2..3	There must be 2 or 3
4	There must be 4
4..*	There must be 4 or more

If there can be more than one element, you use a JavaScript array.

Size Restrictions

When Text attributes are defined, they have a maximum size defined. This can be changed in the attribute's Advanced Property tab.

When sub-types are defined for Floating Point Number, Fixed Point Number and Text attributes, the length for the attribute is specified in the Advanced Property sheet.

For all Number sub-types, upper and lower bounds can be set on the value that the attribute can take. Again, this is done through the attribute's Advanced Property tab.

Default Values

Default values can be used when defining attributes for BOM classes. This is done through the **Resource** tab in the Properties.

If an attribute has a default value, then as soon as an instance of the class that it is an attribute of is created, the attribute has that value. For example, if a class has a Number type attribute called quantity, it might be given a default value of 1 in the **Resource** Property tab as shown below:

Attribute		
General	Property	Value
Description	Restrictions	
Stereotypes	Default value:	1
Resource	Lower limit:	0
	Lower limit inclusive:	true
	Sub Type:	Floating Point
	Upper limit:	- not set -
	Upper limit inclusive:	true

When an instance of this class is created, the quantity attribute is 1.

If you attempt to set a default on an optional attribute, for example, one with a potential multiplicity of zero, a

Default value for an optional attribute will always apply

warning is generated.

If you attempt to set a default on an attribute with a multiplicity greater than one, a

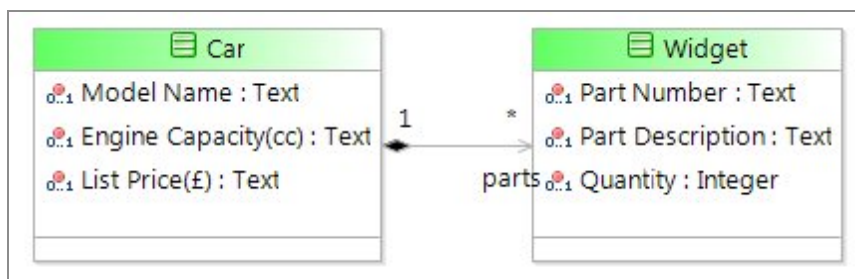
Default values are ignored for attributes with a multiplicity greater than one

warning is generated.

BOM Labels and Names

BOM packages, BOM classes, and BOM attributes have a Label as well as a Name.

A *Label* is a free format field that is designed to be a user-friendly description of the attribute or class. The label text is displayed by the BOM editor as shown below:



And by the default Forms:

User Task

Car Details **parts**

Part Number	Part Description	Quantity
partNumber1	partDescription1	0
partNumber2	partDescription2	0
partNumber3	partDescription3	0
partNumber4	partDescription4	0
partNumber5	partDescription5	0
partNumber6	partDescription6	0
partNumber7	partDescription7	0
partNumber8	partDescription8	0
partNumber9	partDescription9	0

New Delete

Model Name modelName

Engine Capacity(cc) engineCapacitycc

List Price(£) listPrice

However, the Label cannot be used with the entire product, as in scripting. For example, the names that can be used to refer to attributes are considerably constrained. Therefore, a name is defined for each class and attribute. The default name is generated by using the characters from the label that are allowed in names. Space and most punctuation are removed. The image of the form above shows the names as well as the labels for the different fields. You can see that the spaces, brackets, and currency symbol have all been ignored when the names were generated. In scripting, the attribute name is used. For example:

```
var data.engineSize = data.car.engineCapacitycc;
var data.carPrice = data.car.listPrice + delivery + tax;
```

Label to Name Algorithm

All characters in the label that are not valid characters in the name are removed when the label is converted to the default name.

Only the characters A-Z (excluding accented characters), a-z, and the underscore character "_" are valid for use as a name's first character.

For subsequent characters in the name, the same characters are valid as for the initial character, with the addition of the digits 0-9.

Spaces and punctuation characters are not allowed in names, including the following:

Character	Character Description
.	Dot
,	Comma
-	Hyphen

BOM Class Label and Name

Any characters are valid in a label. As the label is converted to the class name, any illegal name characters are ignored.

In the following example, you can see the quotation marks and spaces in the Label are removed in the Name:

Label:	"Test" Class Name
Name:	TestClassName

The Label is used in BOM diagrams and the Name is used in scripts. Because the Name is used in scripts, it is easier to read if initial capital letters are used for each separate word, as in the example above. The following example shows a Label and Name pair where only lowercase letters are used

Label:	another test (class) name
Name:	anothertestclassname

Compared that to the following:

Label:	Another Test (Class) Name
Name:	AnotherTestClassName

The Name field is generated from the Label field whenever the label is changed unless the Name field has been manually set, in which case the name must then be manually changed. For example:

Label:	another test (class) name2
Name:	TestClassName

BOM Attribute Label and Name

For the names of class attributes, there is a requirement that the first two characters of the name must use the same case.

To meet this requirement, the software automatically uses lowercase letters for the first two characters if they are of different cases, as shown in the following examples:

Attributes		
Label	Name	Type
@ A field	afield	<input type="checkbox"/> Bom Primitive Types::Text
@ A long Text Attribute	alongTextAttribute	<input type="checkbox"/> Bom Primitive Types::Text
1.0 a Numeric Attribute	anumericAttribute	<input type="checkbox"/> Bom Primitive Types::Number
✓ a boolean	aboollean	<input type="checkbox"/> Bom Primitive Types::Boolean
✓ BOOLEAN Attribute	BOOLEANAttribute	<input type="checkbox"/> Bom Primitive Types::Boolean
@ An Attribute	anAttribute	<input type="checkbox"/> Bom Primitive Types::Text
1.0 iINTEGER	inTEGER	<input type="checkbox"/> Bom Primitive Types::Number
@ attributeName	attributeName	<input type="checkbox"/> Bom Primitive Types::Text

When creating attributes for BOM classes, it is recommended that you use uppercase letters for the initial letter of each word in the label, and if possible, do not use a single letter word as the first word. The names follow the "Camel case" convention (Camel case names have lots of "humps").

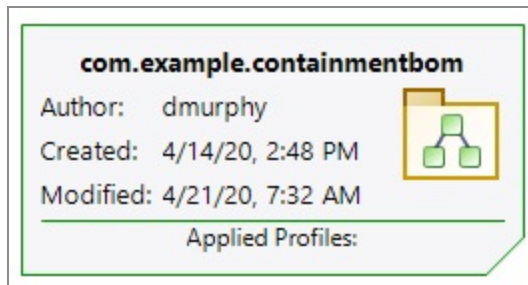
BOM Package Label and Name

The BOM top-level package or model name must be made up of dot-separated segments, each beginning with a letter or underscore and containing only letters, underscores, and numbers, and must avoid reserved words in a similar manner as BOM class names.

Each BOM must have a unique name across all other projects in the TIBCO Business Studio workspace.

The BOM package name must differ from the project lifecycle ID, which is also a dot-separated name in the same format. The project lifecycle ID can be viewed by right-clicking on the Project and selecting **Properties > Lifecycle**.

The BOM Label is free format text, meaning it can be any text that you want to display. The Label is displayed in the BOM editor, and by default is the same as the name of the BOM, as shown in the following example:



Reserved Words

There are certain reserved words that cannot be used as Names, but there are no such restrictions for Labels.

Words that are keywords in JavaScript cannot be used as Class or Attribute names, as they would cause problems when running the JavaScript that referenced the class or attribute. A full list of reserved words can be found in [Reserved Words in Scripts](#).

Using Business Data in Tasks

General

All Tasks have an **Interface** tab that can be used to restrict which fields the task has access to. By default, no fields are listed, which means there are no restrictions and the Task has access to all the fields in the process.

i Note: If the **Interface** tab is used, and a new field is added to the process, the new field is not available in the Task until it is added to the Interface.

Each field that is specified in the **Interface** tab is specified to be one of the following:

- In
- Out
- In / Out

These specifications define whether the value is input or output to the task. There is also a Mandatory flag that can be specified, which controls whether the field has to have a value.

User Task

BOM fields can be displayed and updated in User Task steps. After a User Task has been completed, all the BOM fields that are In / Out or Out fields are initialized.

Script Task

Any BOM fields that have not been initialized by a previous task have a null value, and therefore need to be initialized using the factory method before any of the attributes can be referenced. See [Creating a New Business Object](#) for more details on using BOM fields in Script Tasks, and using Scripts in general.

Forms

If the form generated from a task includes a sequence, choice, or group with multiplicity assigned to it, that multiplicity is not reflected in the form.

Process Migration

If you want to migrate a process instance from one version of a process template to a later one, you must make sure that the data in the first process template is compatible with the second process template version. The Case Data Manager (CDM) component does a compatibility check for all types of classes (Local or Case) whenever a new version of Business Data Project is deployed.

As much of the structure of the data comes from the BOM, you have to ensure that the BOM used by the two versions of the project is compatible. This means that the BOM used by the process template that you are migrating from must be a subset of the BOM used by the process template that you are migrating to.

Therefore, if you want to be able to migrate process instances from the old process template to the new version of the process template, you can only make compatible changes to the BOM. If incompatible changes are made to the BOM, there is a possibility that process instances are not migrated to the new version of the process templates.

A compatible change adds a new entity to a BOM, or makes an existing entity less restrictive, for example, the addition of a new class, or increasing the length of a text

attribute from 50 to 60 characters. Examples of ‘incompatible changes’ include removal of a class, making an optional attribute mandatory, or adding a maximum value to an integer attribute that was previously unrestricted.

The following changes in the process template are considered as compatible when migrating a process instance from one version of a process template to a later one.

General Changes

Any BOM entity’s label can be changed (as long as the name remains the same).

Diagrams can be rearranged, annotated, and so on.

Changes to a Class

- Addition of new attributes and composition attributes, as long as they are optional (for example, they must have multiplicity with a lower bound of zero, such as 0..1 or "*").

Changes to Class Attributes and Composition Relationships

The multiplicity of an attribute or composition relationship might be changed, as long as it makes it less restrictive (for example, it either increases the upper bound or decreases the lower bound) and it does not change between having a maximum multiplicity of 1 and greater than 1. Examples are given in the following table.

From	To	Valid?
1..5	1..8	Yes - increase in multiplicity
1	0..1	Yes – made optional
0..1	*	No (cannot change from single to many)
1	1..*	No (cannot change from single to many)
*	1..*	No (might have zero)

From	To	Valid?
0..1	1	No (might have zero)
*	4..*	No (might have less than 4)

The attribute type cannot be changed. If an attribute's type remains the same, its restrictions might be altered, as long as they are less restrictive than the old restrictions.

Restriction	Permitted Change
Default value	might be changed
Lower limit	might be decreased or removed if a former lower limit was set
Upper limit	might be increased or removed if a former upper limit was set
Lower limit inclusive	might be changed from false to true
Upper limit inclusive	might be changed from false to true
Maximum text length	might be increased
Number length	might be increased
Decimal places	might be increased (if length increased by the same amount or more)

Changes to an Enumeration

Addition of new enumeration literals.

Change to a Primitive Type

When the type has a BOM Native Type as its superclass, it might be altered subject to the compatible change rules described in [Changes to Class Attributes and Composition Relationships](#).

Business Data Scripting

The data objects that are passed around the TIBCO Business Studio - BPM Edition system, both within and between processes, can sometimes be mapped as whole objects from one process to another, or attributes of one object can be mapped onto attributes of another object in graphical ways using the mappers. However, there are some places where the processes require custom processing of the data beyond the direct mapping of attributes. In these cases, the scripting capabilities can be used.

TIBCO Business Studio - BPM Edition scripting can be used in a lot of places within BPM processes by selecting **JavaScript** as the script grammar, for example:

- Script tasks within processes
- Action Scripts that are run on particular events related to tasks (for example, initiate, complete, timeout, cancel, open, submit and close)
- Timer Scripts - used to calculate a deadline or duration of a task within a process
- Condition Scripts – used to determine which direction flow should take within a process
- Loop Scripts – control how many times loops are executed within processes

The TIBCO Business Studio - BPM Edition Script is based on the JavaScript language, but has some unique restrictions and extensions, all of which are described later in this guide.

Factories

At runtime, when a new object needs to be created to represent a particular Business Object (instance of a BOM class) or another variable, a factory method needs to be used to create the object instead of using the `new` keyword that is usually used in JavaScript.

The factory methods for BOM classes can be found in factory classes whose names match the package name of the BOM, for example, for a BOM with a package name `com.example.ordermodel` the factory class would be called `factory.com_example_ordermodel`, and there would be methods in the factory called `createClassname` for each class in the BOM. For example, if the BOM contained Classes called `Order`, `OrderLine`, and `Customer`, there would be the following factory methods:

```
factory.com_example_ordermodel.createOrder()  
factory.com_example_ordermodel.createOrderLine()  
factory.com_example_ordermodel.createCustomer()
```

Creating a New Business Object

You can either create a new Business Object directly, or use a copy of an existing object.

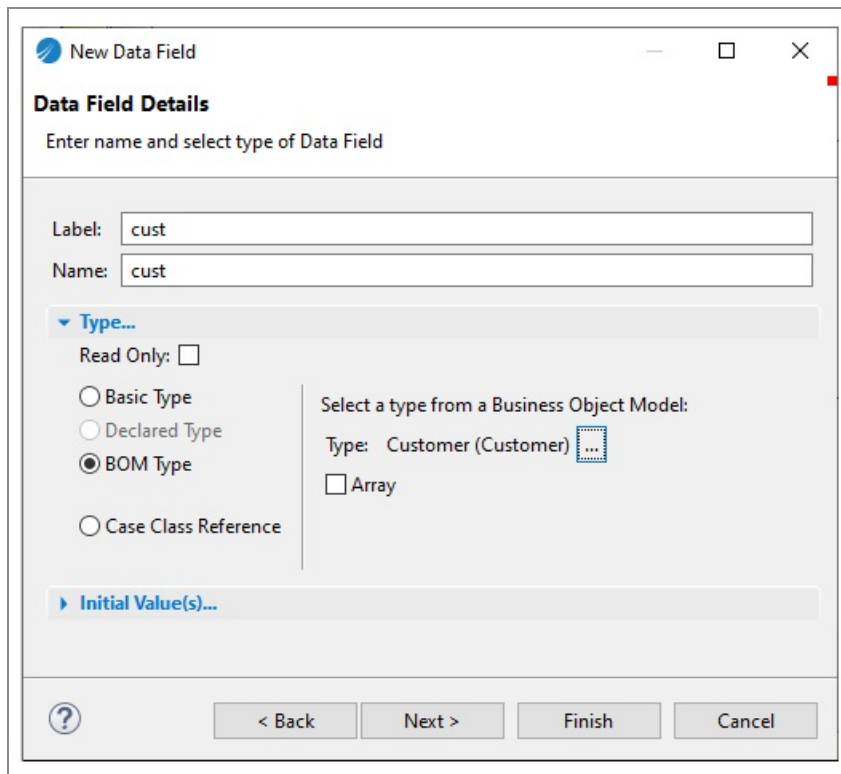
Create an Instance of a Class

You can create a new Business Object directly.

Taking the example of the Person/Customer/Employee BOM:

A screenshot of a software interface showing a class definition for 'Customer'. The class name 'Customer' is at the top in a green header. Below it, there are three attributes, each preceded by a red '@' symbol: 'name : Text', 'email : Text', and 'phone : Text'. The attributes are listed in a white box with a thin border.

In order to create a customer instance, we first need a data field to hold the instance. This is done by creating a new Data Field in a process, and setting the Type to be an External Reference to a type in the BOM. Here we have created a field called `cust` that can hold instances of the Customer class instances:



New Data Field

Data Field Details
Enter name and select type of Data Field

Label:

Name:

Type...

Read Only: ☐

☐ Basic Type
☐ Declared Type
☒ BOM Type
☐ Case Class Reference

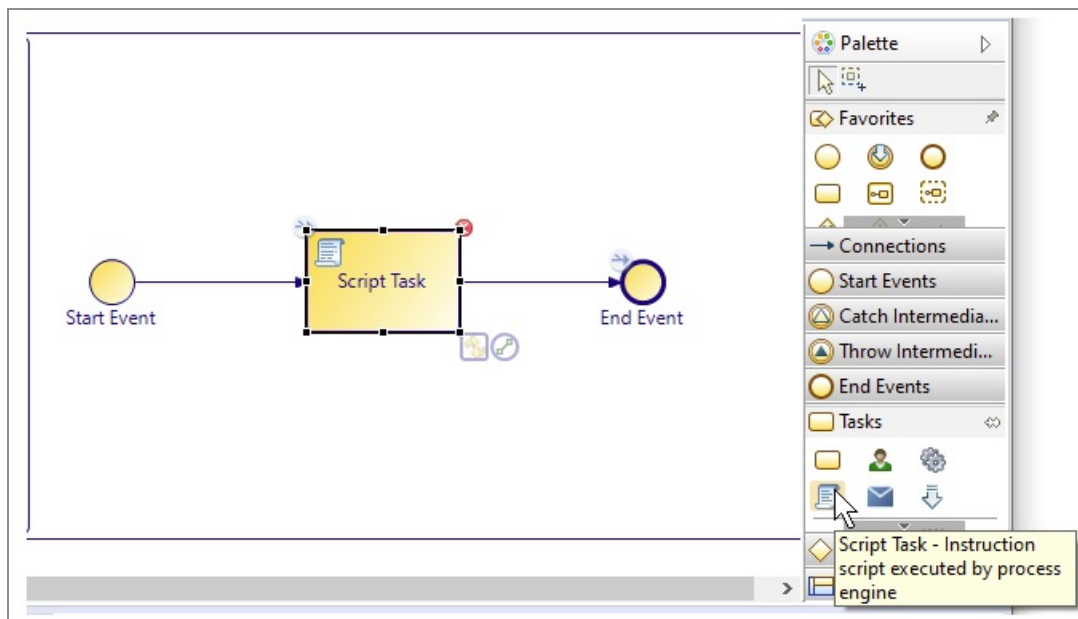
Select a type from a Business Object Model:

Type: ...

☐ Array

[Initial Value\(s\)...](#)

Then a Script Task can be dragged onto the process diagram from the Tasks palette:



View the Script Task properties and from **Script Defined As** menu, select **JavaScript**:

The Describe Task Script dialog box is displayed. Type the script.

It is very important to understand that when a process starts, the `cust` process data field does not contain or refer to an instance of the Customer class, it just has the ability to. So, the first thing to do before attempting to access the attributes of the process data field is create an instance of the Customer object and assign it to the `cust` process data field. The instances of the Customer class are created by the "Customer Factory Method", the name of the factory that creates Customer instances is based on the name of the BOM package that contains the Customer class as described in the previous section.

```
data.cust = factory.com_example_data.createCustomer();
```

Create a Copy of a Business Object

One way of creating a new Business Object is to create a copy of an existing object.

The `scriptUtil` utility method, as shown below, is provided for doing this:

```
data.cust1 = factory.com_example_data.createCustomer();
data.cust2 = bpm.scriptUtil.copy(cust1);
data.cust3 = cust1;
```

The script in the example above, sets the `cust2` process data field to refer to a copy of the Business Object that the `cust1` process data field refers to, and `cust3` to refer to the same Business Object that the `cust1` process data field refers to.

The `bpm.scriptUtil.copy()` method performs a "deep" copy, which means that it copies all the objects contained by the Business Object being copied, as well as the Business

Object itself. It is only for copying whole Business Objects, not for just copying BOM Primitive values.

To copy an array of business objects, use `bpm.scriptUtil.copyAll(array)`

Using the Special Value Null

The special value written as null can be used in several different ways.

Checking for Null Attributes

This section is intended chiefly for those readers not familiar with JavaScript.

The default value of the `cust` variable when the process starts is a special value written as null. If our script was running later on in the process, and there was a possibility that an earlier script might have set the `cust` variable to refer to a Customer, but it could still be null, then this can be checked in the script before calling the factory method, as shown below:

```
if (null == cust)
{
    data.cust = factory.com_example_data.createCustomer();
}
else
{
    // cust was assigned in an earlier script
}
```

There are several things to note here:

- `if (CONDITION) {IF-BLOCK} else {ELSE-BLOCK}` is used for testing and conditionally executing statements. Between the `()` (parenthesis mark), there should be a condition that results in a `true` or `false` result. If the value results in `true`, then the IF-BLOCK statements between the first `"{}"` (curly braces) are processed. If the value results in `false` the statements between the second curly braces in the ELSE-BLOCK are processed. There can be multiple statements between the curly braces. These are referred to as a block of statements. In BPM Script, curly braces are mandatory. In JavaScript, they are only required if there is more than one

statement to be processed.

- The "==" operator is used to test for equality. Here, it is being used to test if the `cust` variable has the value `null`. Writing `null == cust` instead of `cust == null` can help if you forget to use "==" and use "=" instead, since `cust = null` is valid in some places. However, `null = cust` is never valid, so the syntax checker would help you in this case.
- The "/*" in the `else`-block is used to introduce a comment. The rest of the line following "/*" is ignored by the script processing engine.
- If a `UserTask` is processed that has a BOM field as an Out or In / Out parameter in its Interface (the default for all fields is In / Out), then after the `UserTask` is complete the BOM field always refers to an object, so it is not necessary to initialize the BOM field from the Factory method in any scripts that follow the `UserTask` that outputs the field.
- This is also true for any other task that has a mandatory Out or In / Out parameter (the difference between `UserTasks` and `Forms` is that it always creates objects, even for Optional parameters).

Once we know that the `cust` field has a value, we can then set the name. We can check to see if attributes have been previously assigned by comparing them against `null`, although this only works for attributes that do not have a default value. For example:

```
if (data.cust == null)
{
    data.cust = factory.com_example_data.createCustomer();
}
/*
Set the cust.name if not already set
*/
if (data.cust.name == null)
{
    data.cust.name = data.customerName;
}
```

The example above shows how to use a multi-line comment. The comment is opened with a "/*", then all text until a matching "*/" is ignored by the script engine.

Similarly, you should check that an attribute is not `null` before using any methods on an object, as shown below:

```

if (data.cust.dateOfBirth != null)
{
    data.year = data.cust.dateOfBirth.getFullYear();
}

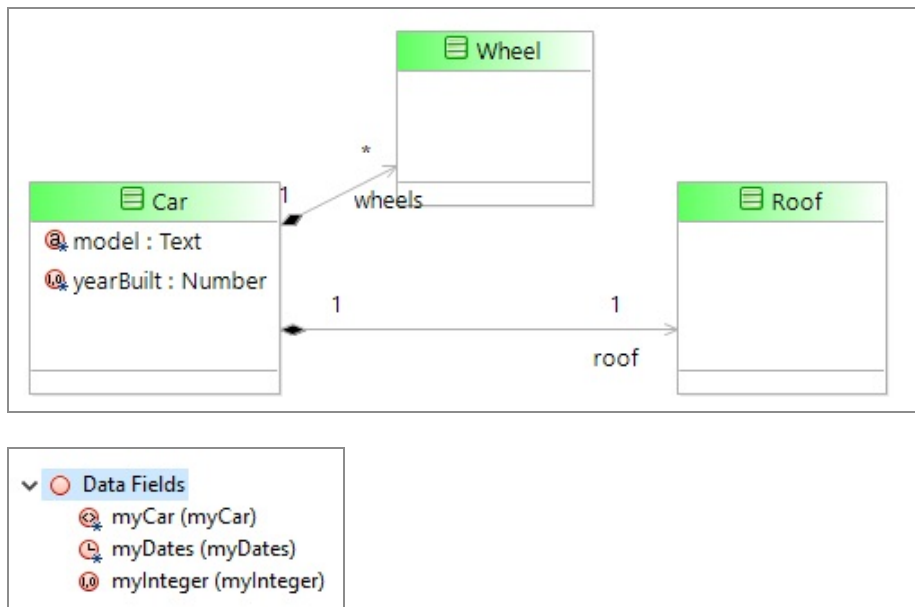
```

Otherwise, you might get a runtime exception.

Assigning a Null Value

The value of single value Data Fields and Business Objects' attributes and compositions can be cleared by assigning them the special value, null.

The following diagram and script illustrate this:



```

// Clear car's model value (attribute)
data.myCar.model = null;
// Remove car's roof value (composition)
data.myCar.roof = null;
// Clear myCar Data Field (Business Object)
data.myCar = null;
// Clear myNumber Data Field (Number)
myNumber = null;

```

For Data Fields or Business Object attributes and compositions that have a multiplicity greater than one, the assignment of null is not possible. Instead, values can be removed using the appropriate Array methods. For example, setting the Array length to 0, for the removal of all values. In the above example, this applies to Car's wheels composition and the myDates data field.

Using Content Assist

TIBCO Business Studio can provide some helpful assistance when entering scripts.

If you cannot remember whether you had called the field `data.cust` or `data.customer` you can type the first few letters and press `Ctrl+Space`. You are prompted with a list of words, variables, methods and so on, that are appropriate for where you are in the script.

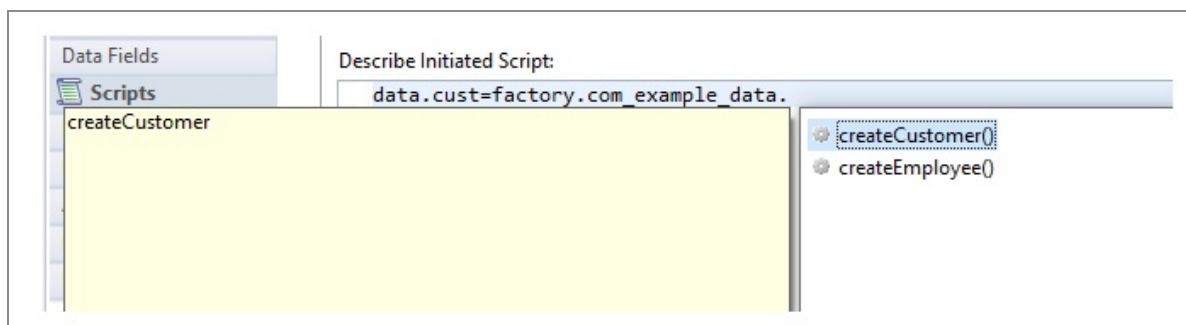
To insert `data.cust` in the script, you can:

- Select `data.cust` and press `ENTER`. A list of words, variables, methods and so on, associated with `data.cust` is displayed.
- Type `u`. Only items beginning with "cu" are displayed.
- Press `ENTER`.
- Double-click `cust`.

Next, type `=co` and press `Ctrl+Space`. Only the content assist that matches "co" in our example is displayed. Press `ENTER` to insert the factory name into the script, as shown below:

```
data.cust = factory.com_example_data
```

Next, type `."` to give a list of the factory methods. This allows you to choose the type of Business Object to create, as shown below:



Since the Business Object we want is already selected, press ENTER to cause the text `createCustomer()` to be added to the script. Press ENTER to complete the line.

```
data.cust = factory.com_example_data.createCustomer();
```

Working with Single Instance Attributes of Business Objects

To add contact details to a Customer instance, you can write a script.

```
if (data.cust != null)
{
    data.cust.phone      = phoneNumber;
    data.cust.email      = emailAddress;
    data.cust.address    = postalAddress;
}
```

Ensure that no variables used to assign attributes are `null`. Otherwise, the script causes a runtime exception that can cause the process to fail when the script is run.

If the address attribute of the Customer class is an attribute of an Address type rather than a Text type, the address attribute needs to be set to refer to an instance of the Address class before the attributes of the `customer.address` can be set. For example, the following can be done:

```
if (data.cust != null)
{
    data.cust.phone      = phoneNumber;
    data.cust.email      = emailAddress;
    if (data.cust.address == null)
    {
        data.cust.address = factory.com_example_data.createAddress();
    }
    data.cust.address.street = streetAddress;
}
```

```

data.cust.address.district = districtAddress;
data.cust.address.city     = cityAddress;
data.cust.address.country  = countryAddress;
data.cust.address.postcode = postCode;
}

```

Multiple Instances of a BOM Class

It is possible for the process data field to refer to multiple instances.

Multiple Instances of a BOM Class in a Process Data Field

If a process data field is flagged as being an Array Field when the process data field is created (or its properties are changed later), then instead of referring to a single Business Object, the process data field refers to multiple instances of the Business Object. This is done using a JavaScript array.

Let us consider a process data field that holds an Array of Customer objects called `custList`. The properties sheet for `custList` has **Array** selected and is of type Customer:

The screenshot shows the 'Data Field' properties sheet for a field named 'custList'. The 'General' tab is selected. The 'Label' and 'Name' fields both contain 'custList'. Under the 'Type...' section, the 'BOM Type' radio button is selected. To the right, a dropdown menu shows 'Customer (Customer)' as the selected type. Below the dropdown, the 'Array' checkbox is checked. Other options like 'Basic Type', 'Declared Type', and 'Case Class Reference' are unselected. The 'Read Only' checkbox is also unselected.

Array fields with a multiplicity greater than 1 are implemented using a JavaScript array.

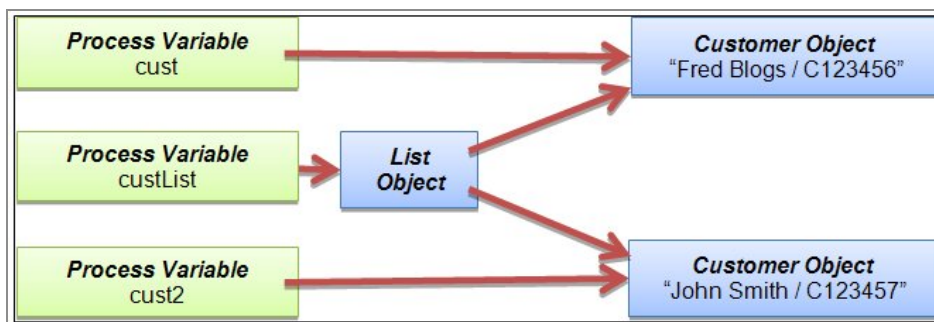
The Array objects do not need to be created. They are created by default as empty Arrays. If you want to associate a particular Customer with the `custList` variable, you can assign the `cust` field with a single instance field. This is shown below.

```
data.cust = factory.com_example_data.createCustomer();  
data.cust.name = "Fred Blogs";  
data.cust.custNumber = "C123456";
```

Another way is to add the new customer to the `custList`. We can add multiple customers to an Array as well, as shown below:

```
// Using cust variable created above (Fred Blogs / C123456) is  
// added to the Array custList:  
data.custList.push(data.cust);  
// Now add a second customer to the Array (John Smith):  
data.cust2 = factory.com_example_data.createCustomer();  
data.cust2.name = "John Smith";  
data.cust2.custNumber = "C123457";  
data.custList.push(data.cust2);
```

This can be pictured as follows:

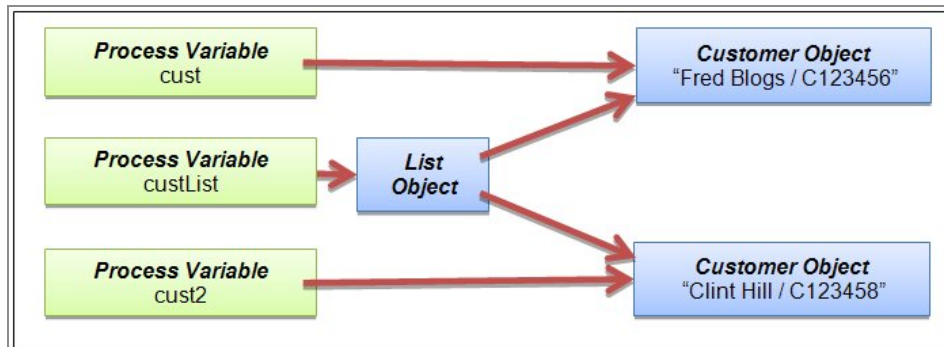


WARNING: If, after you used the script above, you then used the following script to add a third customer to the list, this would go wrong on two accounts.

```
data.cust2.name = "Clint Hill";  
data.cust2.custNumber = "C123458";  
data.custList.push(data.cust2);
```

First, a new Customer instance has not been created for Clint Hill, so the first two lines above modify the John Smith Customer instance to refer to Clint Hill. Then when the `push()` method is called for the third time, it attempts to add a second reference to the same

Customer instance. However, adding a second reference fails because the array type used does not allow the same object to be included more than once. So, the list ends up containing the Fred Blogs and Clint Hill Customer instances but not John Smith:



Instead, a Customer instance must be created using the factory method for each Customer to be added to the list. If references to the individual customer `cust` are not required outside of the script then local script variables can be used in place of the process variable `cust`. The example below shows two script local variables `c1` and `c2` being used to correctly add two Customers to a `custList`:

```
// Create first customer instance
var c1 = factory.com_example_data_createCustomer();
c1.name = "Fred Blogs";
c1.custNumber = "C123456";
data.custList.push(c1);
// Create second customer instance
var c2 = factory.com_example_data_createCustomer();
c2.name = "John Smith";
c2.custNumber = "C567890";
data.custList.push(c2);
```

It is not necessary to use different variable names for the two Customer instances, variable `c` could have been used throughout the script in place of `c1` and `c2`, but then the word `var` would have to be removed from the line that contains the second call to the `createCustomer()`:

```
var c = factory.com_example_data_createCustomer();
c.name = "Fred Blogs";
```

```
c.custNumber = "C123456";
data.custList.push(c);
c = factory.com_example_data_createCustomer();
c.name = "John Smith";
c.custNumber = "C567890";
data.custList.push(c);
```

Another way of creating the second and subsequent instances would be to make copies of the first. This can be useful if there are a lot of attributes with the same value, for example:

```
// Create first customer instance
var c1 = factory.com_example_data_createCustomer();
c1.name = "Fred Blogs";
c1.custNumber = "C123456";
c1.isRetail = true;
c1.dateAdded = new Date.now();
data.custList.push(c1);
// Create second customer instance by copying the first
var c2 = bpm.scriptUtil.copy(c1);
c2.name = "John Smith";
c2.custNumber = "C567890";
data.custList.push(c2);
```

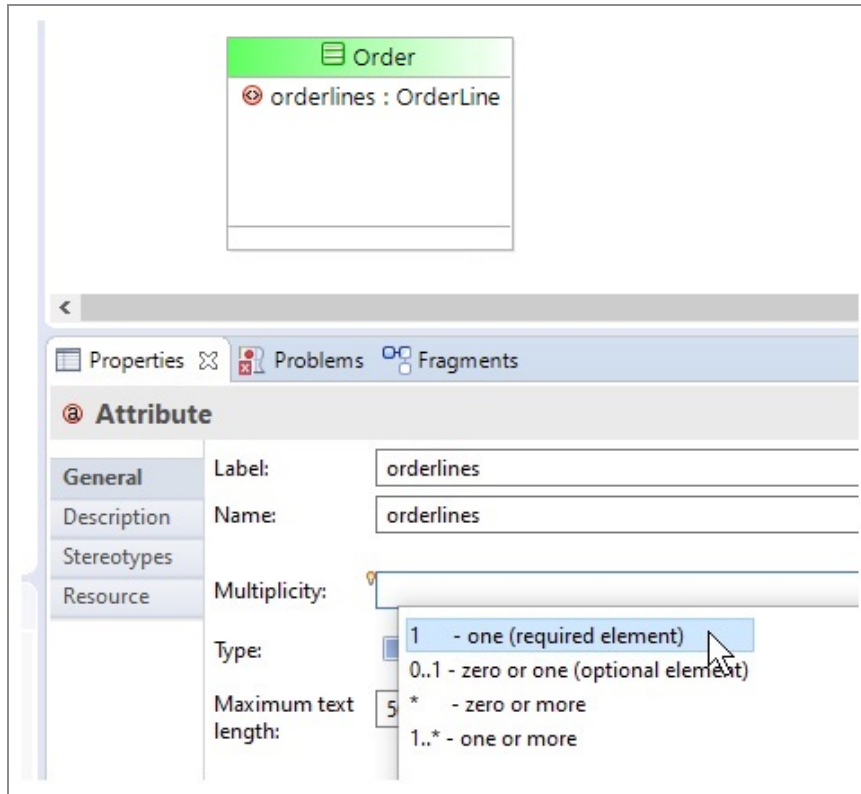
We can use the same var to store the new customer once the first customer has been added to the Array. We do not need to keep a reference to it any longer:

```
// Create first customer instance
var cust = factory.com_example_data_createCustomer();
cust.name = "Fred Blogs";
cust.custNumber = "C123456";
cust.isRetail = true;
cust.dateAdded = new Date.createDate();
data.custList.push(cust);
// Create second customer instance by copying the first
cust = bpm.scriptUtil.copy(cust);
cust.name = "John Smith";
cust.custNumber = "C567890";
data.custList.push(data.cust);
```

Multiple Instances of a BOM Class in a BOM Class Attribute

Just as we defined Process Fields that contained Multiple Instances of Customer Business Objects in the previous section, we can also specify that an attribute of a BOM Class can have and must have multiple instances, for example, an Order might contain multiple OrderLine objects.

In BOM Editor, this is configured by setting the multiplicity of the attribute, as shown here:



The above screen capture shows some example values that can be used to specify the multiplicity, however, other values can also be used, for example, 3..6 would mean between 3 and 6 instances must be added to the field.

If the multiplicity is greater than one (e.g. "*" or "1..*" or "3..6"), an Array is used to manage the field, and values must be added to the field using Arrays. Otherwise, if the multiplicity is 1 (for example, multiplicity is "1" or "0..1"), a straightforward assignment can be used.

When the multiplicity of an attribute is greater than one, an Array is used to manage the data at runtime, in the same way Process Fields are managed when the Array check box is set in the Field Properties. To manage the multiple orderlines associated with an Order, the following script can be written.

```
var orderline = data.com_example_data_createOrderLine();
orderline.partNumber = 10023;
orderline.quantity = 3;
data.order.orderlines.push(orderline);
orderline = factory.com_example_data_createOrderLine();
orderline.partNumber = 10056;
orderline.quantity = 1;
data.order.orderlines.push(orderline);
```

or using the `bpm.scriptUtil.copy()` method:

```
var orderline = factory.com_example_data_Factory.createOrderLine();
orderline.partNumber = 10023;
orderline.quantity = 3;
data.order.orderlines.push(orderline);
orderline = bpm.scriptUtil.copy(orderline);
orderline.partNumber = 10056;
orderline.quantity = 1;
data.order.orderlines.push(orderline);
```

Arrays provide methods for finding out how many items there are in the array, accessing particular entries in the array, and enumerating the array.

Working with Temporary Variables and Types

When writing scripts, the need for temporary variables often arises. In JavaScript, a temporary variable is declared using the `var` keyword. This saves having to add all variables as Process Data Fields, which is especially useful if the variables are only used within a single script. Adding them to the list of Data Fields would just end up complicating the process.

For example, to declare (that is, to tell the script engine about) two temporary variables called `x` and `carName` you can write:

```
var x;
var carName;
```

The variables can also be initialized (given initial values) when they are declared like this:

```
var x = 5;
var carName = "Herbie";
```

When writing BPM scripts, it is always best to initialize variables when they are declared so that TIBCO Business Studio's Script Validation and Content Assist code knows what are the types of variables. If you do not initialize a variable, you get the following warnings:

Unable to determine type, operation might not be supported and content assist might not be available

The content assist does not work. So, using the example from the previous section, it would not be a good idea to write:

```
var c1;
var c2;
c1 = factory.com_example_data.createCustomer();
data.c1.name = "Fred Blogs";
data.c1.custNumber = "C123456";
custList.push(c1);
c2 = factory.com_example_data.createCustomer();
data.c2.name = "John Smith";
data.c2.custNumber = "C123457";
custList.push(c2);
```

Instead, the variables should be initialized when they are declared:

```
var c1 = factory.com_example_data.createCustomer();
data.c1.name = "Fred Blogs";
data.c1.custNumber = "C123456";
custList.push(c1);
var c2 = factory.com_example_data.createCustomer();
data.c2.name = "John Smith";
data.c2.custNumber = "C123457";
custList.push(c2);
```

Loops Within Scripts

This section is intended chiefly for those readers not familiar with JavaScript.

There are three different JavaScript loop syntaxes that can be used in TIBCO Cloud BPM Script, and one that cannot be used. The ones that are supported are:

```
while (CONDITION) { BLOCK }  
do { BLOCK } while (CONDITION);  
for (INITIALIZER; CONDITION; INCREMENT) { BLOCK }
```

One that is not supported is:

```
for (FIELD in ARRAYFIELD) { BLOCK }
```

Here is a simple `while` loop that loops until the `ix` variable becomes less than 0:

```
var result = "";  
var ix = 10;  
while (ix >= 0)  
{  
    result = result + " " + ix;  
    ix = ix - 1;  
}
```

Things to note:

- `result = result + " " + ix;` can be written `result += " " + ix;` using abbreviations allowed in JavaScript.
- `ix = ix - 1;` can be written `ix--;` or `--ix;` using abbreviations allowed in JavaScript.
- `ix++;` or `++ix;` can be used similarly instead of `ix = ix + 1;`.
- The curly braces are required for loops in TIBCO Business Studio - Cloud BPM Edition JavaScripts just as they are for if/else statements.

The do-while loop is very similar, but the condition is evaluated at the end of the loop, so the loop is always executed at least once:

```
var result = "";
var ix = 10;
do
{
    result += " " + ix;
    ix--;
}
while (ix >= 0);
```

The for loop is similar to the while loop, but has two extra expressions in it. One is executed before the loop starts and the other is executed at the end of each loop. It results in a more compact script. Here is the equivalent for loop to the above while loop:

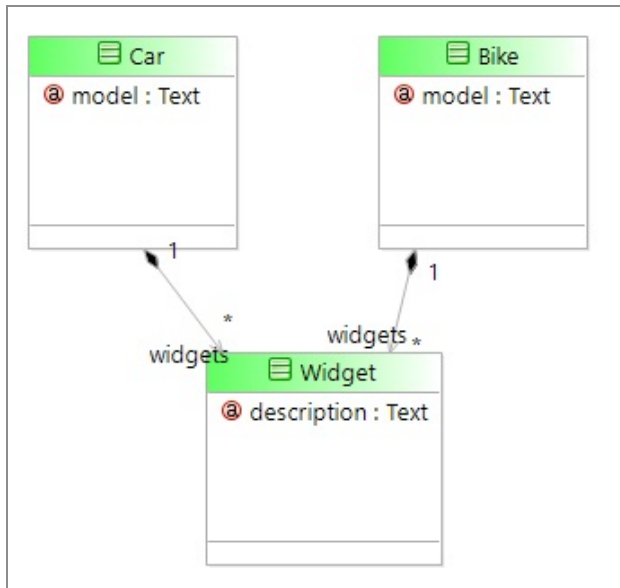
```
var result = "";
for (var ix = 10; ix >= 0; ix--);
{
    result += " " + ix;
}
```

Scripting Containment Relationships

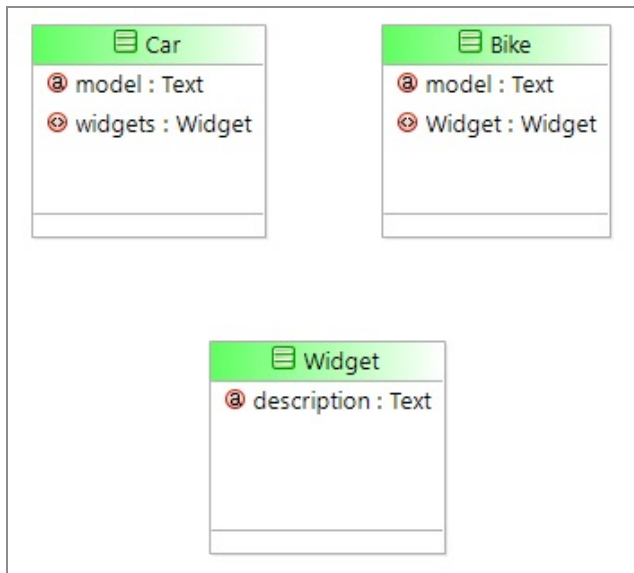
The BOM editor allows you to say that instances of one class are contained within instances of another. For example, if there were classes for Car and Widget, instances of the Widget class can be said to be contained by a Car instance. For the contained relationship, the contained objects are affected by the container's lifecycle events. So, when the container (Car in our example) is destroyed, all the contained (Widget in our example) objects are destroyed too.

The diagram below shows that Widget objects can be contained by a Car or a Bike. However, for an individual Widget object instance, it can only belong to a Car or a Bike instance, not both at once.

There are two ways to model this in the BOM editor. First, a Composition link can be drawn between the two classes, like this:



Alternatively, the Car and Bike class can be given an attribute called widgets of type Widget, as shown below:



Both of these two Car/Widget relationships appear the same when scripting.

There is an attribute of the Car object called widgets, which is an Array, that can contain Widget objects. This would be processed in a similar way to the Array processing examples already covered.

For example, to create a Car and add two Widgets, we can write:

```

var car = factory.com_example_data_containment_createCar();
data.car.model = "Saloon";
var widget = factory.com_example_data_containment_createWidget();
data.widget.description = "M8 Bolt";
data.car.widgets.push(widget);
widget = factory.com_example_data_containment_createWidget ();
data.widget.description = "M8 Nut";
car.widgets.push(widget);

```

Working with Numeric Types

CDM supports the data types Floating Point Number and Fixed Point Number.

There are two types of number:

- Floating point number. A floating point number does not allow you to define a fixed number of decimal places.
- Fixed point number. A fixed point number allows you to specify the number of decimal places. This is defined in TIBCO Business Studio.

See [BOM Native Types](#)

TIBCO Business Studio - BPM Edition provides 4 functions for use with numbers. You should use these functions when mathematical precision is important as they return more accurate results.

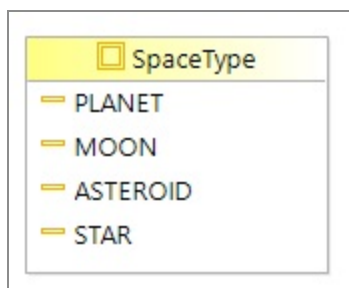
Attribute/Method	Description
add(Number) : Number	Adds one JavaScript Number to another with fixed point precision. For example, a.add(b)
subtract(Number) : Number	Subtracts one JavaScript Number from another with fixed point precision. For example, a.subtract(b)
divide(Number) : Number	Divides one JavaScript Number by another with fixed point precision. For example, a.divide(b)
multiply(Number) : Number	Multiplies one JavaScript Number by another with fixed point precision. For example, a.multiply(b)

Working with Enumerated Types (ENUMs)

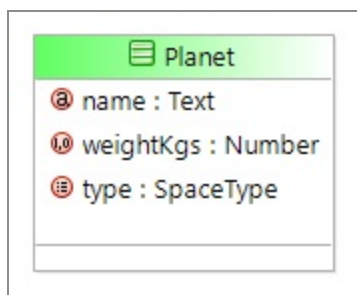
If you want to categorize objects as different types, instead of using a number or a free format string, use an Enumerated Type (ENUM). Enumerated Types provide a better solution because they are restricted, in that they can only take a fixed limited number of values. The names of the values can be made meaningful.

The qualified name of enumerations to be used in script is similar to the Factory names, with the qualified name formatted to replace dot '.' by '_' an underscore character. The qualified name must be wrapped in a pkg object. For example, `com.example.shared.ColorEnum` is used as `pkg.com_example_shared_ColorEnum` in script.

An ENUM is created in the BOM editor by selecting the Enumeration type from the Elements section of the Palette. Having selected the Enumeration Element, it can be named, and values can be added to it. The following is an example of an Enumerated type called `SpaceType`, with `PLANET`, `MOON`, `ASTEROID`, and `STAR` values:



Having defined the Enumeration type, a class attribute can be set to that type:



A Business Object attribute that is configured to be of a particular Enumeration type can only be assigned with values of that enumeration type. Either constants of that type, such as:

```
data.planet.type = pkg.SpaceType.PLANET;
```

Additional JavaScript Global Functions

This topic describes some additional JavaScript global functions you can use.

escape() and unescape()

The JavaScript `escape()` and `unescape()` functions can be used to encode strings that contain characters with special meanings or outside the ASCII character set.

The `escape()` function encodes a string. This function makes a string portable, so it can be transmitted across any network to any computer that supports ASCII characters.

The function does not encode A-Z, a-z, and 0-9. Additionally, the function encodes all other characters with the exception of: `"*", "@", "-", "_", "+", ".", "/"`.

The `escape()` function maps:

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
```

To:

```
%20%21%22%23%24%25%26%27%28%29*+%2C-./0123456789%3A%3B%3C%3D%3E%3F@ABCDEFGH
IJKLMNOPQRSTUVWXYZ%5B%5C%5D%5E_%60abcdefghijklmnopqrstuvwxyz%7B%7C%7D%7E
```

encodeURIComponent() and decodeURI()

The `encodeURIComponent()` function is similar to the `escape()` function but encodes fewer characters. Encoding the string:

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
```

Produces:

```
%20!%22#$%25&'
() *+, - . / 0 1 2 3 4 5 6 7 8 9 : ; % 3 c = % 3 e ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z % 5 b % 5 c % 5 d % 5 e _ % 6 0 a b c
d e f g h i j k l m n o p q r s t u v w x y z % 7 b % 7 c % 7 d ~
```

So only the following characters are encoded:

SPACE, "\"", "&", "<", ">", "[", "\", "]", "^", "`", "{", "|", "}"

encodeURIComponent() and decodeURIComponent()

These functions are similar to the `encodeURIComponent()` and `decodeURIComponent()` methods, but they additionally encode and decode the following characters:

, / ? : @ & = + \$ #

Business Data Modeling Best Practice

This section gives some brief guidance on best practice.

Choose Appropriate Data Types

Take care when selecting attribute types. For example, with Datetime types if a timezone is required, use Date, Time, and Timezone.

Process Data Field Granularity

If a number of related parameters are commonly passed to Tasks, it is a good idea to create a BOM class that contains all the parameters. Then, all the values can be passed to the tasks as a single parameter.

However, one thing to be aware of is that if you have parallel paths within your process that are both processing a Business Object, then the changes made by the first branch to complete might be overwritten by data from the second branch if it is all stored in a single Business Object. To get around this problem, each branch should only be given the data it needs. Then, the data should be merged back into the single Business Object after the branches have merged together.

BOM Class Attribute and Variable Names

It is recommended that BOM class names begin with an uppercase letter and that variable and attribute names begin with a lowercase letter so that they can easily be distinguished. This is the convention used by Java, and what is done by the label to name mapping if Labels contain words with initial capital letters.

In order to read variable names that are made up of several words, it is recommended that you use "camelcase", where the initial letter of every word (apart from the initial word in a data field or attribute name) is capitalized. A data field holding a customer account should be written `customerAccount`. Similarly, you can have data fields called `headOfficeName`. This naming convention is used in the factory methods, so if there is a `CustomerAccount` class, then there is a `createCustomerAccount()` method in the factory. If you enter Labels with initial capitals for each word, then this is achieved. Therefore, a label written as "Customer Account" is converted to a class name of `CustomerAccount`, or an attribute name of `customerAccount`. If the label is written as "Customer account", the class name and attribute name are `Customeraccount` and `customeraccount`, both of which are not so readable.

**Note:**

If you use certain reserved keywords for BOM attribute names and assign them a value using JavaScript, at runtime you get a scripting error while evaluating the JavaScript expression. Avoid using the following names for BOM attributes if you intend using such attributes in scripting:

- - `notify`
- - `equals`
- - `wait`
- - `finalize`
- - `hashCode`
- - `toString`

Do Not Split a Namespace Across Projects

Classes from the same package must not be defined in separate BOM files.

For example, the following classes must both be defined in the package that defines the `com.example.claimmodel.customerdetails` package:

```
com.example.claimmodel.customerdetails.Address  
com.example.claimmodel.customerdetails.Customer
```

Business Data Scripting Best Practice

This section contains some suggestions for Business Data scripting.

Keep the Scripts Small

It is recommended that you keep scripts small. You can do this by breaking potentially large chunks of logic into separate scripts.

Ensure Business Objects Are Valid Before Scripts Complete

Remember that the length, limits, and multiplicity are checked at the end of every script, so ensure that all the Out and In / Out fields for the task have valid values before the script task is completed.

Check for Nulls

It is very important to check that fields and attributes are not null before attempting to get their values.

Use Comments in Scripts

It is good practice to comment code to make it easier for others who follow you to understand what the scripts are doing.

Use Constant First when Comparing a Constant with a Variable

Using:

```
constant == variable
```

is safer than:

```
variable == constant
```

If "=" is used instead of "==" by mistake, the former construction results in a syntax error.

Business Data Services Glossary

The following proxy can be used to generate a list of all glossary entries in your project.

Optional Tasks: You can select whether alphabetized headings should be automatically included at the top of each glossary section in the output. You can also select a style to affect the look of the entire glossary. To do either of these, right-click on the proxy below and select **Edit Glossary Proxy**. Then choose the appropriate option and/or style. If you need help, press **F1** when the dialog is open. When you are ready, you can delete this paragraph or replace it with your own text. **Note:** If you cannot see the proxy below, make sure your markers are turned on. To do this, in the local toolbar click the down arrow next to the **Show Tags** button and select **Show Markers**.

Advanced Business Data Concepts

The following sections provide information about advanced topics:

[UML Profiles and Stereotypes](#)

[Setting Diagram Preferences](#)

UML Profiles

You can apply UML profiles that have been created elsewhere to your business object model. Once your UML profile is available to your business object model, you can use it to apply stereotypes to the various business objects in your business object model



Note: Your business object model must already exist before you can apply UML profiles and stereotypes to it.



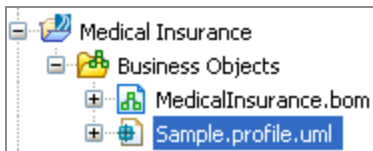
Warning: Once you have applied a UML profile to your business object model, you should not change it. This is because any changes you make to the UML profile could adversely affect your business object model.

Applying a UML Profile to a Business Object Model

You can apply a UML profile to a Business Object Model.

Procedure

1. To use an existing UML profile, copy the file that contains the UML profile to your **Business Objects** folder in the Project Explorer. The UML profile is displayed as follows:



2. Select the **Properties View** for the Model or Package that you want to apply a UML profile to.
3. Select the **Profiles** tab.
4. Click the picker to display the Select Type dialog box. To display a list of the available profiles you can apply to the Package, type ? in the **Select Type(s)** field. A list of the available profiles is displayed.
5. From the **Matching Items** box, select a profile and click **Add**.
6. Click **OK** to close the dialog box.

Applying Stereotypes to Business Objects

You can apply stereotypes to all the business objects in your business object model. You can apply as many UML profiles to a model or Package as you require.

You can apply the UML profile to the business object model or Package first, or it can be applied automatically when you apply a stereotype to an object.


The UML profile must exist in the BOM Special Folder (normally the **Business Objects** folder for your project). The Select Type(s) dialog box used to apply stereotypes then displays all the available stereotypes in that profile. If you select a stereotype from a profile that has not yet been applied to the business object model or Package, the profile automatically is applied when you click OK to apply a stereotype.


Procedure

1. In your business object model, select the **Properties View** for the business object that you want to apply stereotypes to.
2. Click the **Stereotypes** tab.
3. Click the picker to display the Select Type(s) dialog box. To display a list of the available stereotypes that you can apply, type ? in the **Select Type(s)** field.
4. From the **Matching Items** box, select a stereotype and click **Add**.
5. Click **OK** to close the dialog box.
6. To edit your stereotypes, select the **Properties View** for the diagram node which has the stereotype applied to it and click the **Resource** tab. The Stereotype is displayed in the **Resource** tab.

UML Profiles and Stereotypes

By using a UML profile, you can specify stereotypes. Stereotypes provide a method of extending your business object model.

 **Note:** Your business object model must already exist before you can apply UML profiles and stereotypes to it.

 **Warning:** Once you have applied a UML profile to your business object model, you should not change it. This is because any changes you make to the UML profile could adversely affect your business object model.

You can apply UML profiles that have been created elsewhere to your business object model. Once your UML profile is available to your business object model, you can use it to apply stereotypes to the various business objects in your business object model

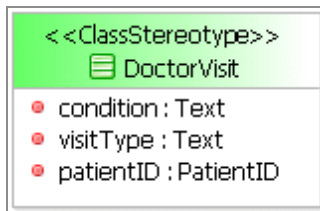
You might have a particular domain within your business that you want to include in your business object model. You can have a UML profile that creates new model elements, derived from your existing model elements but that have specific properties for your business domain. The UML profile tailors the language in your business object model to your specific business domain.

If you are using your UML model in a database for example, you might want to specify which database table a Class should be stored in. You can have a UML profile that contains the database table names as stereotypes. Once you have created your UML profile you can

apply it to your business object model and then apply those Stereotypes to your Classes as required.

Stereotypes can be applied to all business objects in a business object model, including packages, classes, attributes, primitive types, and associations.

A stereotype is displayed as a name enclosed by guillemets and placed above the name of another business object, as shown below:



You can apply existing UML profiles that have been created elsewhere to your business object model.

You can apply as many profiles as you like to your business object model.

Refer to the following Eclipse documentation for a description of how to create UML2 profiles and define stereotypes.

http://www.eclipse.org/modeling/mdt/uml2/docs/articles/Introduction_to_UML2_Profiles/article.html

Unified Modeling Language (UML)

Unified Modeling Language (UML) is an Object Management Group (OMG) specification that helps you specify, visualize, and document models of software systems or business systems, including their structure and design.

The Business Object Modeler uses terms and notation similar to UML, so an understanding of UML can be useful.

The latest major version of UML (and the one employed within the Business Object Modeler) is Version 2, commonly referred to as UML2. For more information see <http://www.uml.org/>.

Importing Existing UML Models into the Business Object Modeler

You can import existing UML models into the Business Object modeler.

Procedure

1. Start the import wizard using one of the following methods:
 - Right-click the **Business Objects** folder, and select **Import > UML Model**.
 - Select **File > Import > Business Object Modeler > UML Model**.
2. The Import UML Model dialog box is displayed. Click the **Browse** button to select the **From Directory** where you have stored the UML model. Any UML models that are found are displayed.
3. Browse to select the **Business Objects** folder into which you want to import the model.

Select the **Overwrite existing resources without warning** check box if you know you want to replace an existing model with the one you are importing; otherwise you are prompted to confirm whether to overwrite the existing model.
4. Click **Finish**. The Model is imported into the folder that you specified.

Setting Diagram Preferences

You can customize the appearance of the Business Object Modeler.

Procedure

1. Select **Window > Preferences**.
2. Expand **Diagram** and **Business Object Model Diagram**.
 - Clicking **Appearance** displays a dialog box that allows you to change the color of fonts, backgrounds, fill colors, and so on.
 - Clicking **Connections** displays a dialog box that allows you to change the line style of the Connections in the Business Object Modeler (for example, Associations, and so on).
 - Clicking **Pathmaps** displays a dialog box that allows you to specify path

variables to modeling artifacts that you might want to use in your business object model.



Note: There are two more options **Printing** and **Ruler and Grid**. Even though these options are not displayed directly under **BOM Diagram**, they apply only to the Business Object Modeler, and not to the Process Modeler.

- Clicking **Printing** displays a dialog box that allows you to specify the print settings for your business object models (for example, orientation, page size and the margins).
- Clicking **Ruler and Grid** displays a dialog box that allows you to specify whether or not the ruler or grid is displayed when defining business object models in the Business Object Modeler.

Result

Any changes you apply affect new objects only. To change the appearance of existing objects, select the object then use the **Appearance** tab in the **Properties** view or the options on the **Diagram** menu.

Using Case Data

Case Data Overview

Case data is business data that is centrally managed and can therefore be accessed and updated by multiple BPM process applications.

Business data is structured data that contains information about real-world entities or business concepts that an organization needs to manipulate, for example: Customer, Order, Orderline, Claim or Policy.

Two sorts of business data are supported - normal (or local) business data, and case data. In contrast to normal business data, case data:

- has a unique identifier.
- has a state.

- can be referenced by a set of processes and/or work items that are responsible for making state changes to it.
- has its own audit history, independent of any processes or work items that operate on it.

Case data can be modeled at design-time as [case classes](#) in a [case data model](#), then represented at runtime as [case objects](#), which can be referenced by corresponding [case references](#).

Any BPM application can interact with a case data model to create and use case data. A BPM application, in this context, is either:

- a BPM process, defined in TIBCO Business Studio and deployed to the BPM runtime.
- a custom client application that uses the BPM public API to invoke BPM services provided by the BPM runtime.

Case Data Features

A broad range of features and capabilities allow you to manage and manipulate case data in the following ways:

- create, deploy, upgrade and delete case data models.
- create, update and delete case objects in a process.
- display and update case objects in forms or pageflows.
- use case data to find in-progress work items and/or process instances that are associated with a particular case.
- locate a particular case and perform [case state-specific actions](#) on it.
- update case objects on an ad-hoc basis, independent of any enterprise process update - for example, a customer reporting a change of address.
- generate standalone pageflows and forms that you can use to create, update or delete case objects in a particular case model.

Case Data Terminology

BOMs

A Business Object Model (BOM) must contain a case class or a class. It can also contain, as required, additional case classes or classes. A BOM can only be created in or added to a Business Data project.

Case Actions

A case action is a special type of business service, associated with a case class, that defines an action a user can perform that is related to a case. The availability of case actions can be restricted based on the current case state and/or the user's permissions. Case actions cannot be made available in terminal states, because terminal states indicate that the case is complete.

Case Classes

A case class is a template for a case object. Case classes must have a single case identifier attribute.

You can create a case object from a case class from within any BPM process application that references that case class.

Unlike local objects, which exist only over the lifespan of a process, case objects are persisted - each case object is stored in the case data tables that represent the case model from which the case object was instantiated.

Case Data Models (or Case Models)

A case data model is the BPM representation of a particular set of case data.

At design-time, a case data model is represented as one or more BOMs in a Business Data project. Cases are modeled as case classes. The project must be deployed to the BPM runtime to make the case data model available to BPM applications.

Case Data Tables

Tables in which case data objects are stored. Case data tables represent the case model from which the case object was instantiated.

Case Identifiers

A case identifier (CID) is a special type of attribute that can be used to uniquely identify an instance of a case class. It can be used in processes, scripts, or API calls to create or to find a particular case.

Case Objects

A case object is an instance of a case class. Data for a case object is stored in the case data tables associated with the case data model to which the case class belongs. You can create case objects from within a process by using a Case Data service task.

Case References

A case reference is a unique reference (or pointer) to a case object, created by BPM when that case object is created.

Creating a case object creates and returns a case reference. That reference can be used by any process to find and manipulate that case object.

You can use a case reference within a process to find, display, update, or delete an existing case object.

Case States

A case state is a special type of attribute, available only on case classes, that can be used to uniquely identify a set of business-specific states that a case can be in. Case states can be used to control the availability of case actions to users.

i Note: Each case class must have one or more *terminal case states*, which indicate that a case is complete and can be deleted. When a case enters a terminal state, it is no longer visible in the process. At runtime, while creating a case, the case state must be assigned some value.

Case Summary Attributes

A case summary is a specified subset of the attributes defined for a case class. The case summary defines the initial set of information that is returned about a case object when that object is returned as part of a search result. When a user performs a search on a case class, the case summary information is displayed for each matching case object.

Searchable Attributes

A searchable attribute is one that is designed to be used in case data search criteria. Searchable attributes can be used in process scripts or BPM public API calls to find cases that match specified attribute values.

Creating and Deploying a Case Data Model

Before you can use case data in a business process, you must create a model of that data and deploy it as a separate application.

Procedure

1. Click **File > New > Business Data Project**.
2. In the New Business Data Project wizard:
 - a. Enter a suitable **Project name**.
 - b. Click **Finish**.A new project is created, containing an empty BOM.
3. [Model your case data](#) as required.
4. Deploy the Business Data project. See [Project Lifecycle](#).

Result

Once deployed, you can now use it in business processes, by referencing the defined case classes to create, read, update, delete and search for case objects. See [Using a Case Data Model in a Business Process](#).

Modeling Case Data in a BOM


Use the BOM Editor to model your case data.


Procedure

1. Add at least one [case class](#) to the BOM.
2. Add a [case identifier](#) to each case class.
3. Add a [case state](#) attribute to each case class.
4. Select the case state attribute, then on the **General** tab of the **Properties** view choose the enumeration type that lists the possible states.
5. Select the case state attribute, then on the **Terminal States** tab of the **Properties** view, select one or more terminal states for the case.
6. Specify the attributes that you want in the [case summary](#).
 - a. In the BOM Editor, select the attribute.
 - b. On the **General** tab of the **Properties** view, select or clear the **Summary** option.
7. Specify the attributes that you want to be [searchable attributes](#) in case classes:
 - a. In the BOM Editor, select the attribute.
 - b. On the **General** tab of the **Properties** view, select or clear **Searchable** option.
8. Define required [BOM relationships](#).
9. Save the BOM.

Case Classes and Local Classes

A BOM must contain a case class. It can also contain further case classes and local classes.

Type	Icon	Description
Case		<p>A case class is a template for a case object. A case class:</p> <ul style="list-style-type: none"> • must have a case identifier.

Type	Icon	Description
		<ul style="list-style-type: none"> • must have a case state attribute. • can have summary attributes. • can have searchable attributes. <p>You can create case objects from case classes using either a case data service task in a process or a public API. Creating a case object returns a case reference. That reference can be used by any process to find and manipulate that case object.</p> <p>Note: Unlike local objects, which exist only over the lifespan of a process instance, case objects are persisted - each case object is stored in the case data tables that represent the case model from which the case object was instantiated.</p>
Local		<p>A local class is a template for a local object. A local class:</p> <ul style="list-style-type: none"> • cannot have a case identifier. • cannot have a case state attribute. • cannot have summary attributes. • cannot have searchable attributes. <p>Note: A business object created from a local class only has local scope, even though the class exists in a BOM. The local object is not persisted to the case data store and can only be accessed by the process that creates it.</p>





Note: You can convert a class from its existing type to the other type. For example, right-click the case class and select the **Convert to Local** menu option.

Case Identifiers

A case identifier (CID) is a special type of attribute that can be used to uniquely identify a case class. It can be used in processes, scripts or API calls to delete or to find a particular case.

A case class must have exactly one case identifier.

There are the following types of case identifier:

Type	Icon	Description
Auto		<p>A unique text string that is automatically generated when an instance of the case class is created. This is the default option. You can define a prefix and suffix for the auto identifier.</p> <p>For example, XYZ/123456-CP(Car policy id for XYZ insurance)</p>
Manual		<p>A single attribute of the class that has unique values within the business domain. For example, a policy name could be used as a manual case identifier for a policy class.</p> <p>Manual identifiers must be populated by the process, script or API call that creates an instance of the case class.</p>

Adding a Case Identifier to a Case Class

A case class must have at least one case identifier.

Procedure

1. In the BOM Editor, drag a **Case Identifier** from the palette to the case class.
An automatic case identifier is created by default.
2. On the **General** tab of the **Properties** view, change the **Label** to a suitable name to identify the case identifier.

What to do next

You can change this case identifier to be a manual type.

i Note: You can also refactor an existing attribute as a case identifier. Right-click the attribute and select **Convert to Case Identifier**. Similarly, you can refactor an existing case identifier as an attribute. Right-click the case identifier and select **Convert to Attribute**.

Configuring a Case Identifier

The default case identifier type is automatic (a unique text string that is automatically generated when an instance of the case class is created). You can instead use manual identifier types, which use values that your application supplies.

Procedure

1. In the BOM Editor, select the **Case Identifier**.
2. Select the **General** tab of the **Properties** view.
3. Set the **Case Identifier** types as follows:
 - **Auto** - The value of the case identifier is auto-generated when an instance of the case class is created. If set to **Auto**, you can also specify:
 - **Minimum Digits** - The case identifier is at least this length, padded with leading zeros, as required,
 - **Prefix** - The case identifier includes this prefix.
 - **Suffix** - The case identifier includes this suffix.
 - **Manual** - The case identifier value is assigned by the application. If set to **Manual**, you can also specify:
 - **Maximum text length** - The maximum length to which the application can set the case identifier value.

Case States and Case Actions

Case states and case actions provide a powerful case management tool, allowing a user to choose the most appropriate actions to perform on a case at a particular point, according to the business context of the case.

Cases (for example, claims or orders) flow through states that are particular and specific to the type of case itself. For example, an order might be in one of the states "submitted", "fulfilled", "dispatched" and "delivered". Based on the state of the case, there might be one or more applicable actions that a user might need or want to perform. What actions are required, in what order they are performed or indeed whether an action is performed more than once is at the discretion of the user working on the case. Implementing case states and case actions for a case class gives users the ability to make these decisions and perform the required actions.

At design time, you must:

1. define a case state for the case class.
2. define the set of case actions that are required to support the case.
3. for each case action, define the case states in which it is available, and the permissions a user needs to be able to use the case action.

At runtime, when a user has located a particular case, the case actions are displayed that are available to them and appropriate for the current case state. They can then decide which, if any, of these actions they want to perform based on the particular case circumstances.

Case States

A case state is a special type of attribute, available only on case classes, that can be used to uniquely identify a set of business-specific states that a case can be in.

A case class has only a single case state attribute, which must be a text property type whose enumeration type lists the possible states of the case object.

i Note: As cases live outside of individual processes, at some point the data becomes redundant (for example, a case representing the tracking of an invoice might not be required after a 1-year retention period after the invoice is paid). Therefore all case types must define one or more states in which a case can be considered redundant and therefore can be deleted permanently. Select one or more terminal case states (which indicate that a case is complete and can be deleted.) These are selected on the case state and that there must be at least one terminal state (so that the case can ultimately be deleted).

Case Actions

A case action is a special type of business service, associated with a case class, that defines an action a user can perform on a case. You can either [create a new case action](#) or [generate one directly from a case class](#). Template case action processes are provided that allow you to view or update the contents of a case object, but you can modify these templates to provide whatever functionality you need for a particular case class.

You can define as many case actions as you need for a particular case class. The set of case actions define all the actions that a user could perform on a case. You can then restrict the availability of these actions by case state and user privileges:

- **Case state:** You can specify that a case action is only available when a case object is in a specific case state (or states). For example, if you have defined a case action to update delivery details for an order, you might want to restrict this to be available only for orders that are "submitted".

i Note: Case actions cannot be made available in terminal states, because terminal states indicate that the case is complete.

- **Privileges:** You can specify that a case action is only available to users who have a specific set of privileges. For example, in a call center, you might want to make certain actions available only to supervisors.

At runtime, a case action can only be invoked for a particular case object, either from the Case Manager or, in a custom client, by using the BusinessService API service.

Configuring a Case State for a Case Class

Procedure

1. From the palette, add an Enumeration to the BOM, and give it an appropriate name that represents the case states (for example, orderStates).
2. To the enumeration, add a number of Enum Literals, one for each possible case state, and name them appropriately (for example, orderTaken, picked, packed, and shipped).
3. Add a Case State attribute to the case class, and name it appropriately (for example, orderState).
4. With the case state selected, on the **General** tab, select the browse button to the right of the **Enumeration Type** field, then select the enumeration you added in step 1.
5. On the **Terminal States** tab, select one or more of the states for which the case is considered complete.

i Note: You can convert an existing case state attribute back to a *regular* attribute. Right-click the case state attribute, then select **Convert to Attribute**. You can also convert a case class attribute to a case state attribute or case identifier - right click the case class attribute, then select **Convert to Case State** or **Convert to Case Identifier**, respectively.

Creating a Case Action

Before you begin

- The Business Data project containing the case data model that you want to use must be open in the same Workspace.
- The case class that you want to associate with this [case action](#) should contain a case state attribute that defines the required [case states](#). (If it does not, you can still create the case class but a problem marker is displayed.)
- If you want to assign privileges to control the availability of the case action, the Organization project in which those privileges are defined must be open in the same Workspace.

Procedure

1. Open the project in which you want to create the case action.
2. Right-click the process package in which you want to create the case action, or the **Processes** folder within that package, then select **New > Case Action**.
3. In the Case Action wizard:
 - a. Enter a suitable descriptive **Label** for the case action.
 - b. Select the **Case Class** with which you want to associate this case action.
 - c. Select one of the **Case Action Process Templates** to use as the basis for the process - either **Update Case Action Process** (to generate a process that displays a form showing the contents of the case object, then updates the case object with any changes when the form is submitted), or **View Case Action Process** (to generate a process that displays a read-only form showing the contents of the case object).
 - d. Click **Finish**.

The process is displayed in the Process Editor.

4. On the **General** tab of the **Properties** view for the process:
 - a. (Optional) Select the specific case state (or states) which the case must be in for this case action to be available to a user at runtime.
By default, the case action is always available, whatever case state the case is in.
 - b. (Optional) Select the privilege (or privileges) that a user must hold to be able to use this case action.
5. Modify the template process as required so that it performs the required action (or actions) when it is used.
6. Save the process.

Generating a Case Action From a Case Class

Before you begin

- The BPM Process project in which you want to save the generated [case action](#) must be open in the same Workspace.
- The case class that you want to associate with this case action must contain a [case states](#) attribute that defines the required case states.
- If you want to assign privileges to control the availability of the case action, the Organization project in which those privileges are defined must be open in the same Workspace.

Procedure

1. In the BOM Editor, right-click the case class from which you want to generate a case action, then select either:
 - **Generate Case Action > To Update Case Data**, to generate a process that displays a form showing the contents of the case object, then updates the case object with any changes when the form is submitted.
 - **Generate Case Action > To View Case Data**, to generate a process that displays a read-only form showing the contents of the case object.
2. In the Generate Case Action dialog box, select the appropriate existing process

package.xpd1 file (or select a process packages folder and enter the name of a new *.xpd1* file to create) in which you want to save the generated case action process, then click **Finish**.

The generated process is displayed in the Process Editor.

3. On the **General** tab of the **Properties** view for the process:
 - a. (Optional) Select the specific case state (or states) which the case must be in for this case action to be available to a user at runtime.
By default, the case action is always available, whatever case state the case is in.
 - b. (Optional) Select the privilege (or privileges) that a user must hold to be able to use this case action.
4. Modify the template process as required so that it performs the required action (or actions) when it is used.
5. Save the process.

Case Summaries

A case summary is a specified subset of the attributes defined for a case class. The case summary defines the initial set of information that is returned about a case object when that object is returned as part of a search result.

At design time, you define the attributes that make up a case summary in the BOM Editor.

At runtime, when a user searches the case class for matching case objects, the case summary information is displayed for each matching case object. Each attribute is displayed as a column. The attribute's **Label** is used as the column heading.

A case summary includes, by default, any attributes that are defined as case identifiers or case states. You cannot remove these attributes from the case summary. You can add any of the other attributes defined for the case class to the case summary provided that the attribute:

- has multiplicity of either 0..1 or 1.
- is either a primitive type or an enumeration.

Attributes are listed in the case summary in the same order in which they are defined in the case class.

Specifying Attributes in a Case Summary

You can add attributes of a case class to its case summary so that the attribute information is returned in search results for matching case objects.

Procedure

1. In the BOM Editor, select the case class attribute.
2. On the **General** tab of the **Properties** view, select the **Summary** option to include the attribute in the case summary.

The screenshot shows the 'Properties' view in the BOM Editor. The 'Attribute' tab is selected, and the 'General' sub-tab is active. The following fields are visible:

- Label:** orderDate
- Name:** orderDate
- Multiplicity:** 0..1
- Type:** Bom Primitive Types::Date Time and Time Zone (with a dropdown arrow and a 'Clear' button)
- Searchable:** ☐
- Summary:** ☒

All the attributes that have been chosen as summary attributes are shown on the case class **Summary** tab.

Note: Boolean and URI attributes cannot be used in the Case Summary.

Searchable Attributes

A searchable attribute is one that is intended to be used in case data search criteria. Searchable attributes can be used in process scripts, or BPM public API calls to find cases that match specified attribute values.

For example, find all policies whose IDs begin with the string PN-456.

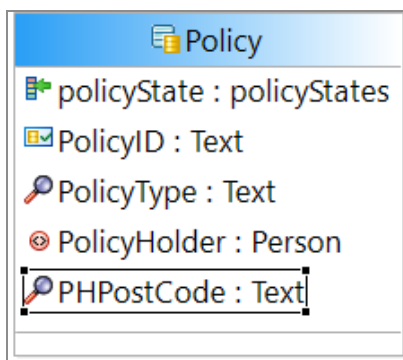
Note: Boolean and URI attributes cannot be made searchable

Searching is possible using searchable attributes. (In the case data tables that represent the case model, indexes are created for all searchable attributes, to facilitate rapid searching of the database.)

Any appropriately typed attribute of a case class can be defined as searchable. Attributes of local classes cannot be made searchable.

Case identifiers are always searchable. (This setting cannot be changed.) Other attributes can be set to being searchable or not searchable (their default value). A searchable attribute is indicated by the 🔍 icon.

For example, in the following case data model fragment, the Policy case class has been optimized to search for policies by a policy ID or type, or by a policy holder's postcode.



BOM Relationships and BOMs

Only certain types of UML relationships are permitted when using case data. Whether or not a relationship is supported depends upon its type, direction, the source and target classes, and whether those classes are both in the same BOM and the same Business Data project.

i Note: Bi-directional composition relationships, and unidirectional association or aggregation relationships, are not supported.

Validation errors are displayed if you try to define an invalid relationship between two classes.

Unidirectional Composition

Source Type	Target Type	
	Case	Class (local)
Case	No	No
Class (Local)	Yes	Yes

Bi-directional Association

Source Type	Target Type	
	Case	Class (local)
Case	Yes, provided that both of the following are true: <ul style="list-style-type: none"> • The relationship is optional (multiplicity = 0..n). Mandatory associations (multiplicity = 1) are not supported. • Both classes are in the same BOM. Cross-BOM or cross-project associations are not supported. 	No
Class (Local)	Yes	No

BOM Versioning

Versioning of the BOM is maintained in TIBCO BPM Enterprise, and it follows version semantics, that is, it has major and minor versions. This depends on whether the new version of the BOM is compatible with the currently deployed version of the BOM or not.

When you create a new version of a BOM in the Business Studio:

1. If this new version is compatible with the existing version of the BOM, you can deploy the new minor version of the same BOM without any problem. So, if the current

version of the BOM is 1.0.0, you can create, and deploy a new minor version, say, 1.1.0.

- a. This minor version must have the following compatibility constraints.
 - Though existing attributes cannot be deleted, new attributes can be added to the BOM.
 - The data type of the existing attributes cannot be changed.
 - Decrease in length for existing attribute is not supported.

i Note: In case the new version created has any change that is not in agreement with the above specified compatibility constraints, the new version is incompatible with the existing version of the BOM, and while deploying in TIBCO BPM Enterprise gives an error.

- b. When a new minor version of BOM is deployed to TIBCO BPM Enterprise, the existing minor version of the BOM is automatically undeployed.

i Note: At any given point in time, the latest minor version exists in the system.

2. As mentioned earlier, if the new version is incompatible with the existing version of the BOM, you might get an error while deploying the new version in BPM Enterprise. In such a case, you must increment the major component of the BOM's version number in Studio. For this, go to **Properties > Lifecycle**. Click **Increment Major Version** and then, click **Apply and Close**. When this new major version of the BOM is deployed to TIBCO BPM Enterprise, the new version of the process application uses the new major version of the BOM.

So, if the current deployed version of the BOM is 1.1.0, and you deploy version 2.0.0, the 1.1.0 version of that BOM moves into the "Undeploying" state, and stays in the "Undeploying" state until there are outstanding 1.1.0 cases present in the system, or any process application referring to this BOM. Once there are no active cases for BOM 1.1.0 and all the referring process applications are also undeployed, the 1.1.0 gets completely undeployed.

Upgrading a Case Data Model

When you upgrade a case data model that has already been deployed, each change is validated by the BPM runtime as part of the deployment process (based on either database rules or the ability to support existing data from an earlier version). Whether changes are valid or invalid determines how you can deploy the updated case model.

If your update contains only non-destructive and supported changes, you can deploy the case data model (it is automatically given a new minor version), which upgrades the existing business data application and case data tables. If you attempt to deploy an updated case data model that contains destructive or unsupported changes as a new minor version, deployment fails.

You can choose to fix destructive changes, or you can:

1. increment the *major* component of the project's version number. Select **Properties > Lifecycle** and then click **Increment Major Version** and apply your change.
2. deploy the case data model as a new major version, which creates a new business data application and a new set of case data tables. The application and case data tables for the existing case data model version are unaffected.

Non-Destructive Changes

The following changes are non-destructive:

- Add a new optional attribute to a class.
- Add a new class.
- Add an optional composition.
- Add an association.
- Change multiplicity upper bounds from 2 or higher to a higher value.
- Change multiplicity lower bounds to a lower value.
- Add a new enumeration literal to an existing enumeration type.
- On an attribute:
 - Increase the length of a text field up to 400 (if it is currently less than 400).
 - Increase the "Integer - Fixed Length" number of digits up to 31.

- Increase the "Decimal - Fixed Length" number of digits up to 31.
- Increase the upper bounds of a numeric type.
- Decrease the lower bounds of a numeric type.
- Remove restrictions such as upper bound, length, lower bound or pattern.
- Make it searchable or not searchable.

Destructive Changes

Any change other than those specifically listed as non-destructive is destructive, requiring changing, or an upgrade to a higher major version. The following are specific examples of destructive changes:

- Change the type of an existing attribute.
- Add mandatory compositions.
- Change multiplicity from 1..1 to 1..*.
- Delete mandatory attributes or compositions.
- Add a mandatory attribute to a class.
- Change multiplicity 0..* to 0..1 or 1..1.
- Delete optional attributes, associations, compositions.
- Delete a class.
- On an attribute:
 - Change between different numeric types.
 - Change a pattern that is applied to the attribute.
 - Decrease the length of a text field.
 - Increase the lower bounds of a numeric type.
 - Decrease the upper bounds of a numeric type.

Process Dependencies and Case Data Model Versions

A process that references an element in a case data model has an application dependency on the exact version of the referenced case data model. It is important to understand the

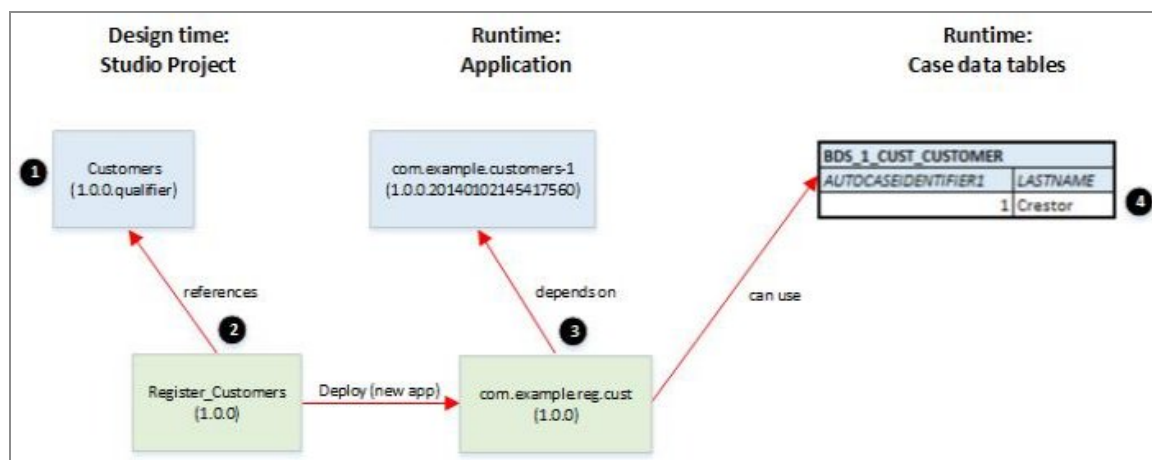
implications of this exact-match relationship, and to manage those implications when upgrading either a case data model or a dependent process. Not doing so could result in deployment errors or runtime issues.

When you deploy (or generate the RASC (Runtime Application Shared Concept) file for) a process project that references a data element in a Business Data project:

- The process application has an application dependency on the referenced business data application. This dependency is recorded against the exact version number of the Business Data project that exists when the process application is deployed (or the RASC generated).
- That version of the case data model must be deployed before you can deploy any process project that references it. Attempting to deploy the process project first results in a deployment error.
- Process instances can only access the case data tables and case data specifically defined by the version of the case data model on which they are dependent.

Consequently, TIBCO recommends that you keep version numbers of case data models and dependent process projects in step with each other. Whenever you upgrade a case data model, you should review the impact on all process projects that reference that case data model, and upgrade process projects as required.

The following example illustrates how successive updates to the very simple Customers case data model could impact an equally simple process project that uses that case data model, and how a solution designer could manage that impact.

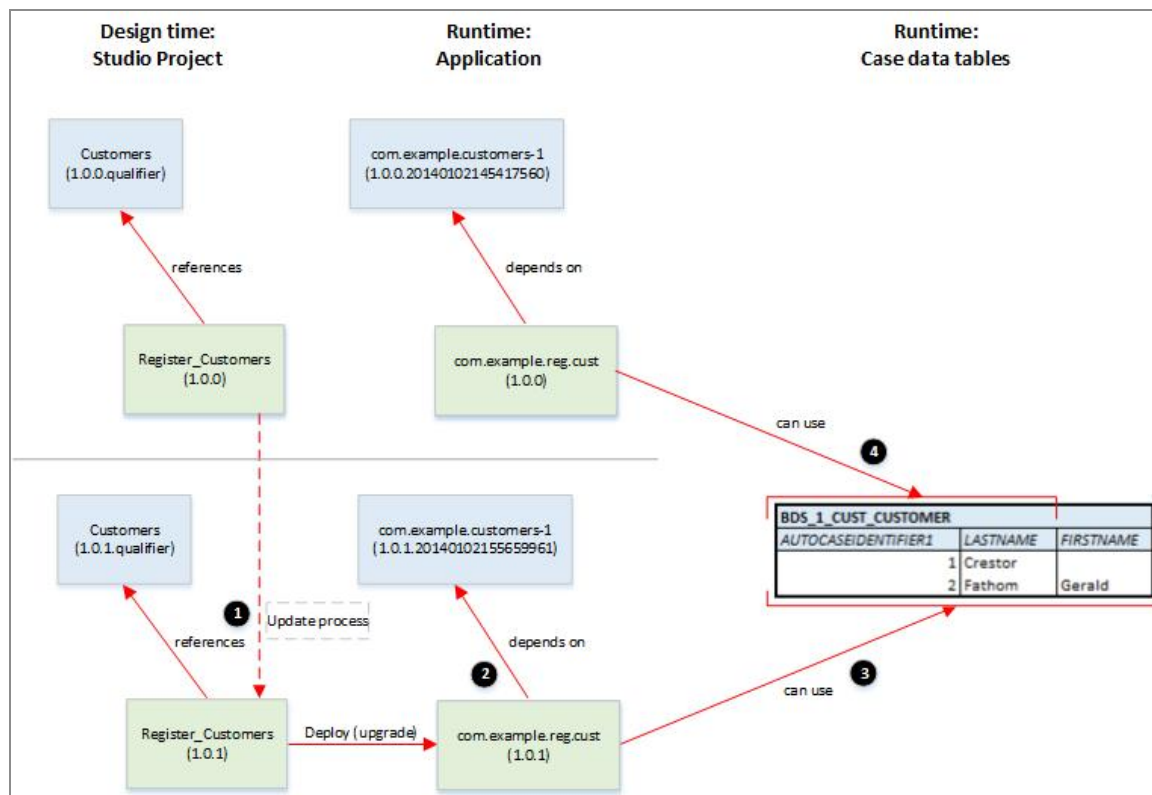


1. The solution designer creates and deploys the Customers Business Data project.
2. He creates a process project, Register_Customers, that uses the Customer case class provided in the Customers Business Data project.

3. He deploys the project, creating the process application `com.example.reg.cust`. Because it uses the Customer case class, this application has a dependency on the `1.0.0.qualifier` version of the `com.example.customers-1` business data application.
4. He runs an instance of the `Register_Customers` process, which creates a customer object in the `BDS_1_CUST_CUSTOMER` case data table.

Upgrading the Process Application Following a Minor Upgrade of the Case Data Model

The solution designer subsequently updates the Customers project, adding an optional `firstName` attribute to the Customer class.



1. The solution designer deploys the Customers project.
2. He then updates the `Register_Customers` process to make use of the new `firstName` attribute on the Customer case class.
3. He deploys the process as an upgrade to the `com.example.reg.cust` application. This version of the application has a dependency on the upgraded version of the `com.example.customers-1` business data application.

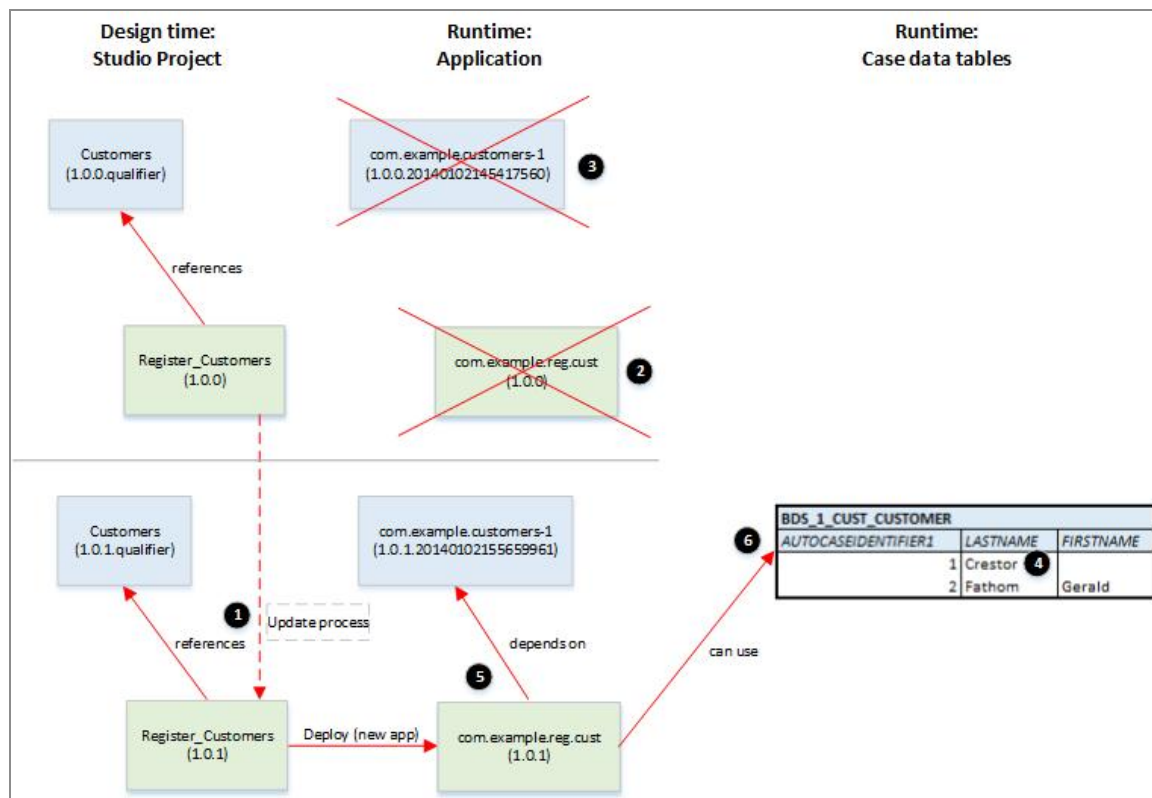
4. He runs an instance of the upgraded Register_Customers process, which creates a customer object in the BDS_1_CUST_CUSTOMER case data table (for the customer Gerald Fathom).
5. Note that existing process instances of the original version of the com.example.reg.cust application continue to run and use the same case data table, but cannot access the FIRSTNAME data values. For example, if a later task in the Register_Customer process returns details of a customer based on search criteria applied to the LASTNAME, the following table shows the different search results that would be returned by each version of the process, even though both versions are using the same search string and accessing the same case data table.

Search string	Register_Customers process version	Data returned
Crestor	1.0.0	Crestor
	1.0.1	Crestor
Fathom	1.0.0	Fathom
	1.0.1	Gerald Fathom



Note: This scenario is only possible if the Register_Customer process uses the firstName attribute in a way that does not affect the process' service interface. If, for example, the process wanted to use the customer's first name as an input parameter, this would change the service interface and application upgrade would be impossible.

Upgrading the Process Application Following a Minor Upgrade of the Case Data Model that Breaks the Process' Service Interface

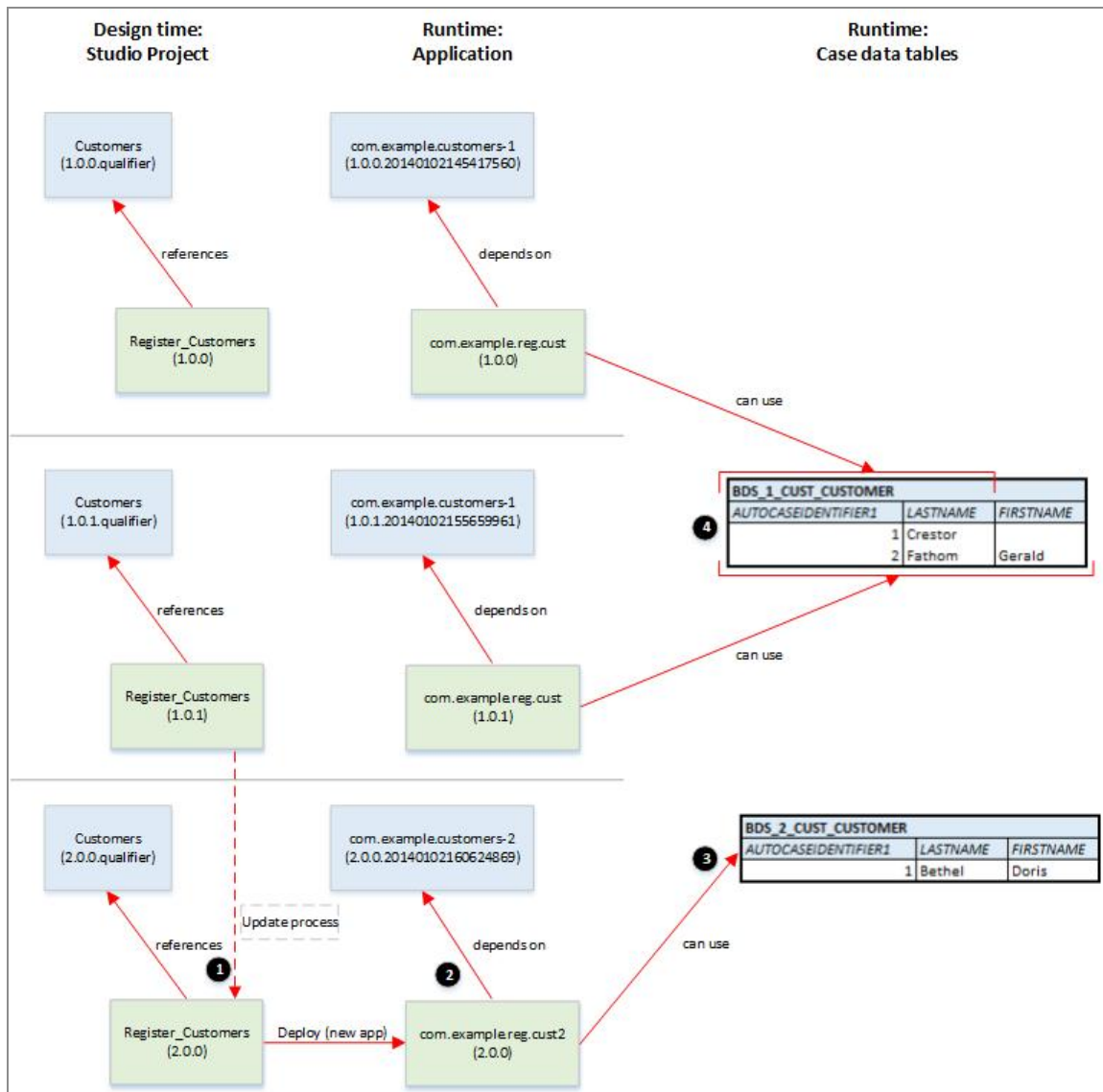


1. The solution designer updates the Register_Customers process to make use of the firstName attribute, using it as an input parameter to the process. However, this changes the process' service interface, which means that the process application cannot be upgraded.
2. He cancels all in-progress instances of the version 1.0.0 application, then undeploys and deletes the application.
3. Deleting the process application removes the dependency on the original version of the com.example.customers-1 business data application. As a later version of this application has already been deployed, the BPM runtime now deletes the 1.0.0 version.
4. Although the 1.0.0 versions of both the process and business data applications have been deleted, the data written by those applications to the BDS_1_CUST_CUSTOMER case data table (the Crestor record) is not deleted.

5. The solution designer can now deploy the Register_Customers process as a new application.
6. He then runs an instance of the Register_Customers process, which creates a customer object in the BDS_1_CUST_CUSTOMER case data table (for the customer Gerald Fathom).

Upgrading the Process Application Following a Major Upgrade of the Case Data Model

The solution designer again updates the Customers project, changing the firstName attribute to be a mandatory attribute of the Customer class. This constitutes a destructive change to the model, so the major component of the project Version number must be updated (to 2). Deploying the project again results in the creation of a new application.



1. The solution designer updates the Register_Customers process to handle the use of the mandatory Customer.firstName attribute.
2. He then deploys the process as a new application, which has a dependency on the com.example.customers-2 application.
3. Running an instance of the Register_Customers process creates a customer object in the BDS_2_CUST_CUSTOMER case data table (for the customer Doris Bethel).

i Note: The `com.example.reg.cust2` application has no access to the data in the `BDS_1_CUST_CUSTOMER` case data table, and the `com.example.reg.cust` application similarly has no access to the data in the `BDS_2_CUST_CUSTOMER` case data table.

Upgrading a Case Data Model to a New Major Version

When you upgrade an already deployed case data model, you can deploy it as a new major version if you have made any destructive changes to the model and do not want to fix them.

You should only redeploy an existing case data model as a new major version if you have made destructive changes to the model.

Before you begin

Make sure that the Business Data project that contains the case data model has the desired version number (with a higher major component of the version number than the currently deployed version).

Procedure

1. Increment the *major* component of the project's version number. Select **Properties > Lifecycle** and then click **Increment Major Version** and apply your change.
2. Deploy the Business Data project.

i Note: TIBCO recommends that you do not change the default application name used for the case data model. This name is *projectId-projectMajorVersionNumber*. For example:

```
com.example.simpleCaseModel-2
```

3. In **Deployment Manager** in Administrator, upload the RASC for the new business data application and deploy it.

Result

You can now deploy and run any process applications that use this version of the case data

model.

Earlier major versions of the case data model are not affected. Process applications that reference an earlier major version of this case data model continue to run against that version, and cannot create or access case objects in this new version. This is because each major version of the case data model is a separate business data application with its own set of case data tables. See [Process Dependencies and Case Data Models](#).

Deleting a Case Data Model

You can delete a case data model that you no longer need by undeploying the appropriate business data application.

Before you begin

- Complete or cancel all outstanding work items and process instances that belong to any process application that references this case data model.
- Undeploy all process applications that have an application dependency on this business data application.

Procedure

1. In **Deployment Manager** in Administrator, select **Action** for the business data application.
2. Click **Undeploy**.
3. To completely undeploy the case data model from the system, delete all of the existing cases (including the terminal state cases).

Using a Case Data Model in a Business Process

Within a process, you can create, read, update and delete case objects created from the case classes defined in a referenced case data model.

A case object is an instance of a case class. Data for a case object is stored in the case data tables associated with the case data model to which the case class belongs. You can create case objects from within a process by using a Case Data service task.

A case reference is a unique reference (or pointer) to a case object, created by BPM when the case object is created. You can use a case reference within a process to find, display, update or delete an existing case object.

Creating a New Case Object

Within a process, you can create a case object by first creating a process data field of the case class type, using a script and other tasks to create the object and set values in it. Then use a **Case Data Operations** service task to create the case object from the local data (by adding the case object to the case data store).

Before you begin

The Business Data project that contains the case class that you want to use must exist in the same workspace as the process from which you want to create the case object.

Procedure

1. Create a new data field, of type **BOM Type**, which references the appropriate case class.
2. In a script, use the appropriate factory method to create an instance of the case class as a local business object and assign it to the data field. (See [Creating a Case Data Model](#).) For example:

```
data.cust = factory.com_example_scriptingguide.createCustomer( );
```

3. Populate the local business object with the appropriate data. You must assign a value for each mandatory attribute in the case class.



Note: If the case class uses an automatic case identifier you cannot (and do not need to) assign a value for it. You must assign values for manual case identifiers.

4. Add a service task to the process.
5. On the **General** tab of the **Properties** view, set **Service Type** to **Case Data Operations**.
6. Select **Create Case Object(s)**.
7. In the **Case Class** field, select the same case class that you used to create the local

business object.

8. In the **Local Data Value(s)** field, enter the name of the data field that holds the local business object.
9. Create a new data field, of type **Case Class Reference**, which references the same case class.
10. In the **Return Case Reference(s)** field, enter the name of this field.

When the service task is executed, the case reference for the newly created case object is returned to this field.

Result

BPM now has two copies available of the same data - the local BOM object and the case object. These objects are separate entities and are not maintained in step:

- The local BOM object is only visible to the process and can only be manipulated in and by that process.
- The case object is visible to and can be manipulated by any process that has access to its case data model.

Any updates to the case object are *not* automatically reflected in the local BOM object, and vice versa.

What to do next

(Optional) Add appropriate error handling to the service task boundary to deal with the following specific error that might be returned by the **Create Case** operation.

Error (Error Code)	Description	Possible solutions
NonUniqueCaseIdentifierError (NonUniqueCIDError)	The manual case identifier specified in the Local Data Value(s) field cannot be used because it is not unique. An existing case object already uses this case identifier.	Amend the case identifier value(s) in the local data object and retry the Create Case operation.

Displaying a Case Object in a User Task - Using a Form

You can display a case object in the form that is displayed when a user performs a user task. You can also, optionally, automatically update the case object with any changes made by the user when the user closes or submits the form.

Before you begin

An already populated data field must exist, containing a case reference to the case object that you want to display.

Procedure

1. Select the user task from which the form is displayed.
2. On the **Interface** tab of the **Properties** view, make sure that the appropriate **Case Reference** data field is included in the **Parameters** list, either explicitly or as part of the **[All Process Data]** setting.
3. If you want to automatically update the associated case object when the form is closed or submitted, set the data field's mode to **IN/OUT** and click **Mandatory**.

Result

At run time, when the form is displayed, the referenced case object is retrieved from the case data store and displayed in the form.



Note: Association and aggregation relationships are ignored when the form is generated.

When the form is closed or submitted, any changes made to fields that are part of the case object are written back to the case data store.

Displaying a Case Object in a User Task - Using a Pageflow

You can display a case object as part of a pageflow that is displayed when a user performs a user task. If the user updates data for the case object, you must manually update this data in a subsequent process step.

Before you begin

An already populated data field must exist in the business process, containing a case reference to the case object that you want to display.

Procedure

1. Select the user task from which the form is displayed.
2. On the **Interface** tab of the **Properties** view, make sure that the appropriate **Case Reference** data field is included in the **Parameters** list, either explicitly or as part of the **[All Process Data]** setting.
3. On the **General** tab of the **Properties** view, click **Pageflow**, select the pageflow process that you want to use, then click **Open PageFlow**.

Make sure that the pageflow process contains the **Case Reference** data field as a formal **Parameter**.

4. Create a new data field, of type **BOM Type**, which references the appropriate case class.
5. Add a script task that uses the read method available on the case reference to read the associated case object data and assign it to the local business object.

For example, the following script reads the data from the case object referenced by `custRef` and assigns it to the `cust` local business object:

```
data.cust = bpm.caseData.read(data.custRef);
```

This creates a local business object that contains a copy of the case object and is scoped to the pageflow process.

6. Add a user task that displays the local business object in a form, allowing the user to view or update the local business object as required.

Association and aggregation relationships in the case object are ignored when the form is generated.

7. If necessary, add a service task (and, optionally, appropriate error handling) to update the case object with the changes to the local business object - see [Updating a Case Object from a ServiceTask](#).

Updating a Case Object

From within a process, you can use scripting to update a local BOM object of the appropriate case class, then use a case data service task to update the case object from the local data.



Note:

You do not need to follow this procedure if you use a case object as an **IN/OUT** parameter on a form in a business process. In that case, changes made by the user are automatically written back to the case data store when the user closes or submits the form. See [Displaying a Case Object in a User Task - Using a Form](#).

However, you do need to follow this procedure if you display a case object as part of a pageflow process. See [Displaying a Case Object in a User Task - Using a Pageflow](#).

Before you begin

The process must already contain:

- A data field containing a case reference to the case object that you want to update.
- A local business object created from the appropriate case class, containing the data that you want to write to the case object.

Procedure

1. Add a service task at an appropriate point in the process.
2. On the **General** tab of the **Properties** view, set **Service Type** to **Case Data Operations**.
3. Select **Change or Delete Case Object(s) Using Case Reference Field**.
4. In the **Case Reference** field, select the data field that contains the case reference to the case object that you want to update.

Content assistance is available on the **Case Reference** and **Local Data Value(s)** fields. Click the lightbulb icon or press Ctrl+Space to display a list of valid field names that are available to the task and that begin with the characters to the left of the cursor.

5. In the **Operation** field, select **Update Case Object(s) From Local Data**.

6. In the **Local Data Value(s)** field, enter the name of the data field that holds the local business object.

If the selected **Case Reference** field is an array then the selected Local Data Value field must also be an array, with an element corresponding to each case reference array element.

What to do next

1. (Optional) Add appropriate error handling to the service task boundary to deal with the following specific errors that might be returned by the **Update Case Object(s) From Local Data** operation.

Error (Error Code)	Description	Possible solutions
CaseOutOfSyncError (CaseOutOfSyncError)	<p>The case object referenced from the Case Reference field has been modified in some way since the field was last populated. That is, the case reference is stale.</p> <p>This could be either a change to the data held in the case object, and/or a change in the object's internal version number.</p>	<ol style="list-style-type: none"> a. Re-read the case object. b. If necessary, examine the case data to check that no significant changes have occurred. c. Retry the Update Case Object(s) From Local Data operation.

2. (Optional) Modify any process that might need to do something if this case object is updated:
 - a. Add a [case data signal event](#) to the process, so that the process is notified when the case object is updated.
 - b. Add suitable business logic to perform any actions that the process needs to take as a result of the update to the case object.

Notifying a Process That a Case Object It Uses Has Been Modified

You can add a case data signal event handler to a process so that it can subscribe to a particular case object that the process uses. The process then is notified if that case object is modified, and can take appropriate action to respond to the change.

At runtime, the case data signal is:

- initialized when the case reference field is first assigned.
- re-initialized if the case reference field is changed to reference a different case object.
- uninitialized if the case reference field is set to null.

While the case data signal is initialized, the event is triggered whenever the referenced case object is either updated, linked to or unlinked from other case objects, or (in some circumstances) deleted.

Before you begin

The process must already contain a data field of type **Case Class Reference** that identifies the case object to which the case data signal event handler subscribes.

Procedure

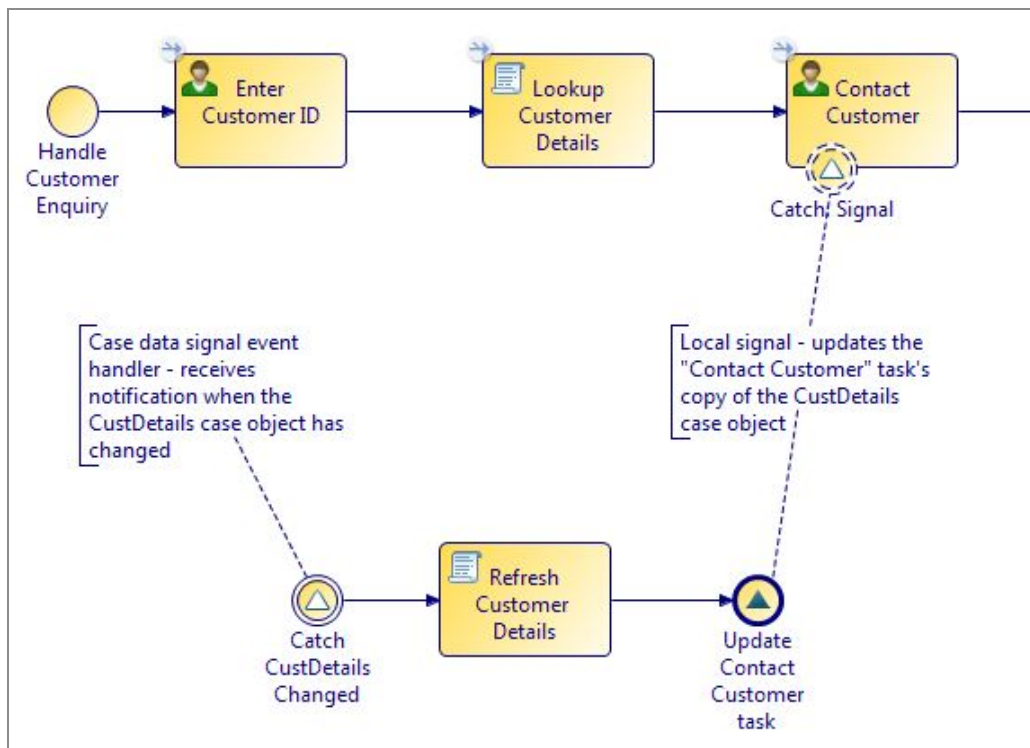
1. Either:
 - Add a catch signal intermediate event to the process (if you want to use a signal event handler).
 - Add an event sub-process to the process, then add a signal start event to the sub-process (if you want to use a signal event sub-process).
2. On the **General** tab of the signal event's **Properties** view, click **Case Data** as the **Signal Type**, then enter the name of the process data field that contains the case reference for the required case object.

i Note: Selecting **Case Data** also sets the **Serialize concurrent flows for this event handler** option. Concurrent execution of the event handler flow is disabled. (This behavior would be undesirable if local copies of the referenced case object are being updated as a result of the event, which is likely.)

3. Add a script task that uses the `bpm.caseSignal` class and methods to determine how the referenced case object has been modified - whether it has been updated, linked to or unlinked from other case objects, or (in some circumstances) deleted.
4. Add appropriate business logic to the process to deal with the change to the case object.

Example

The following process fragment is part of a business process that handles a customer enquiry.



The main **Handle Customer Enquiry** process flow works like this:

1. A customer support representative (CSR) dealing with the enquiry enters the customer's ID, which is then used in the **Lookup Customer Details** script task to retrieve a case object containing the customer's details:

```
// Get the case reference to the CustDetails class that matches the
// identifier provided in the Customer ID (custID) field.
data.custRef = bpm.caseData.findByCaseIdentifier(data.custId,
'com.example.caseclass.CustomerDetails');
// Read the case reference to get the corresponding CustomerDetails
// case data object.
data.custDetails = bpm.caseData.read(data.custRef);
```

2. The customer's details are displayed to the CSR in the **Contact Customer** task, which is offered to the CSR pool of workers.

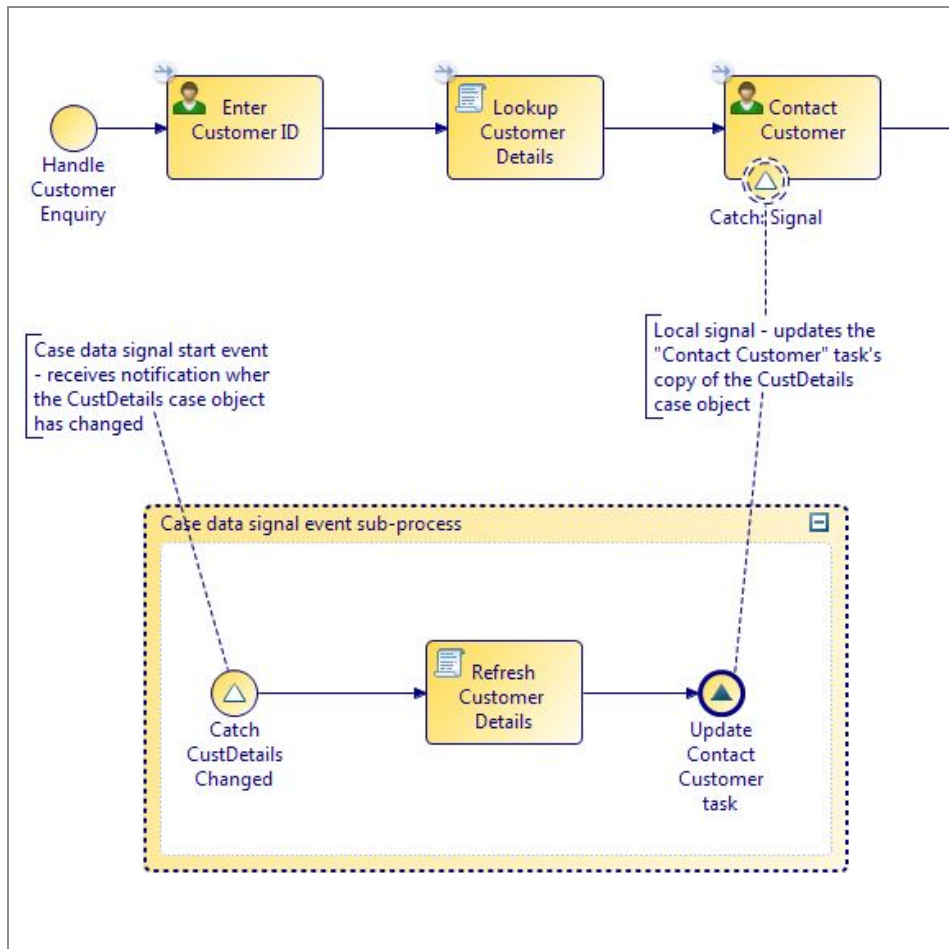
To cater for the possibility that a customer's details could change in between the time that a **Contact Customer** work item is scheduled and a CSR opening and completing that work item, a case data signal event handler is implemented:

1. The **Catch CustDetails Changed** event subscribes to the custRef case reference. It is triggered whenever the referenced CustDetails case object changes.
2. The **Refresh Customer Details** script task interrogates the received event to find out if it was caused by an update. If it was, it re-reads the customer's details from the referenced case object and updates the local CustDetails data field.

```
if
(CaseSignalAttributes.isCaseUpdate())
{
data.custDetails = bpm.caseData.read(data.custRef);
}
else
{
// Optionally perform other activities to handle the event if the
// case object has been
// linked to or unlinked from other case objects, or deleted
}
```

3. The **Update Contact Customer task** throw signal event sends the updated CustDetails data field to its corresponding **Catch Signal** event, which is configured to update that data in the **Contact Customer** work item.

Note: You can use a signal event sub-process instead of a signal event handler, as shown below.



bpm.caseSignal Class and Methods

As part of a case data signal event handler flow, you can use the methods available on the `bpm.caseSignal` class to determine whether the referenced case object has been updated, linked to or unlinked from other case objects, or (in some circumstances) deleted.

The `bpm.caseSignal` class is available in any script in a case data signal event handler flow (or in a case data signal event sub-process).

Method	Description
<code>isCaseUpdate()</code>	Returns <code>true</code> if one or more of the referenced case object's properties has been updated.
<code>isCaseDelete()</code>	Returns <code>true</code> if the referenced case object has been deleted. See Reasons to Avoid Deleting Case Objects .

Deleting Case Objects

You should set a case to a terminal state and allow auto case purge to purge cases. You delete directly from a process if you want to remove a case from the system immediately.

Cases are automatically deleted ('auto-purged') at a set interval (for example, 90 days) after they enter a terminal state, as long as they are not referenced by processes. The interval at which auto-purging happens is configurable via a global configuration setting.

For example, if an Order transitioned to a 'COMPLETED' state (a terminal state) at 8am on 1st January, then, assuming the system has been configured to auto-purge cases after 28 days, this case is removed at 8am on 29th January (or in the next auto-purge cycle that occurs after that point in time).

A case is not purged if any processes remain that reference it.

Within a process, you can use a case data service task to delete one or more case objects. You can identify the case objects to be deleted using either case references or case identifiers.

- [Reasons to Avoid Deleting Case Objects](#) The best practice is to not delete case objects as part of a normal operation. If you do delete case objects, the best practice is to only delete using a single case reference from within a service task in a business process.
- [Deleting Case Objects by Case Reference](#) Within a process, you can delete one or more case objects for which you have the identifying case reference(s).

Reasons to Avoid Deleting Case Objects

The best practice is to *not* delete case objects as part of a normal operation. If you *do* delete case objects, the best practice is to only delete using a single case reference from

within a service task in a business process.

The primary reason to not delete case objects is that if there are other processes (other than the process performing the deletion) that have a reference to the deleted case object, those processes become halted and cannot proceed. In addition, case history (audit trail) is constructed using case objects; if those objects are deleted, the history is no longer available.

Therefore, case objects should not be deleted until it is known for certain that no other processes are referencing the object, and that the case history is no longer needed.

If you delete a case object using a single case reference from within a service task in a business process, built-in checking is provided for other processes that are referencing the case object. (You can catch the `UnsafeToDeleteCaseError` error code.)



Note: It is also possible to use the following methods to delete case objects, but these methods do not provide any error checking for other processes that are referencing the case object(s) that can be deleted. Using any of these methods could result in processes being halted because they are referencing a case object that no longer exists.

- Using the "deleteCase" operations in the BusinessDataServices API.
- Using a service task in a business process to delete:
 - multiple case objects using an array of case references.
 - a single case object using a case identifier.
- Using a service task in any type of process other than a business process - for example, a pageflow process or service process - to delete case objects (by any method).

Deletion of case objects is controlled by a system action (Delete Case Data) that defaults to deny, therefore you must be explicitly granted the permission to delete case objects.

Deleting Case Objects by Case Reference

Within a process, you can delete one or more case objects for which you have the identifying case reference(s).

Before you begin

The process must already contain a data field containing either:

- a single case reference to the case object that you want to delete.
- an array of case references to a set of case objects that you want to delete.

If a case object that you want to delete has any [association links](#) to other case objects, you must [delete those association links](#) before attempting to delete that case object. If any case object has any active links to other case objects, the entire delete operation fails - no case objects is deleted.

i Note: Best practice is to avoid deleting case objects at all, but if you must do so, to delete only a single case object using a single case reference. For more information, see [Reasons to Avoid Deleting Case Objects](#).

Procedure

1. Add a service task at an appropriate point in the process.
2. On the **General** tab of the **Properties** view, set **Service Type** to **Case Data Operations**.
3. Select **Change or Delete Case Object(s) Using Case Reference Field**.
4. In the **Case Reference** field, select the data field that contains the case reference(s) that you want to delete.
5. In the **Operation** field, select **Delete Case Object(s)**.

What to do next

1. (Optional) Add appropriate error handling to the service task boundary to deal with the following specific errors that might be returned by the **Delete Case Object(s)** operation.

Error (Error Code)	Description	Possible solutions
CaseOutOfSyncError (CaseOutOfSyncError)	A case object referenced from the Case Reference Field has been modified in some way since the field was last populated. That is,	<ol style="list-style-type: none"> a. Re-read the case object. b. If necessary, examine the

Error (Error Code)	Description	Possible solutions
	<p>the case reference is stale.</p> <p>This could be either a change to the data held in the case object, and/or a change in the object's internal version number.</p>	<p>case data to check that no significant changes have occurred.</p> <p>c. Retry the Delete Case Object(s) operation.</p>
UnsafeToDeleteCaseError (UnsafeToDeleteCaseError)	<p>The case object is referenced by processes other than the one that is attempting to delete the case object, therefore it is unsafe to delete the object.</p> <p>(This error is catchable only if you have specified a single case reference, not an array of case references.)</p>	Do not delete the case object.

2. (Optional) Modify any process that might need to do something if this case object is deleted:
 - a. Add a [case data signal event](#) to the process, so that the process is notified if the case object is deleted.
 - b. Add suitable business logic to perform any actions that the process needs to take as a result of the case object being deleted.

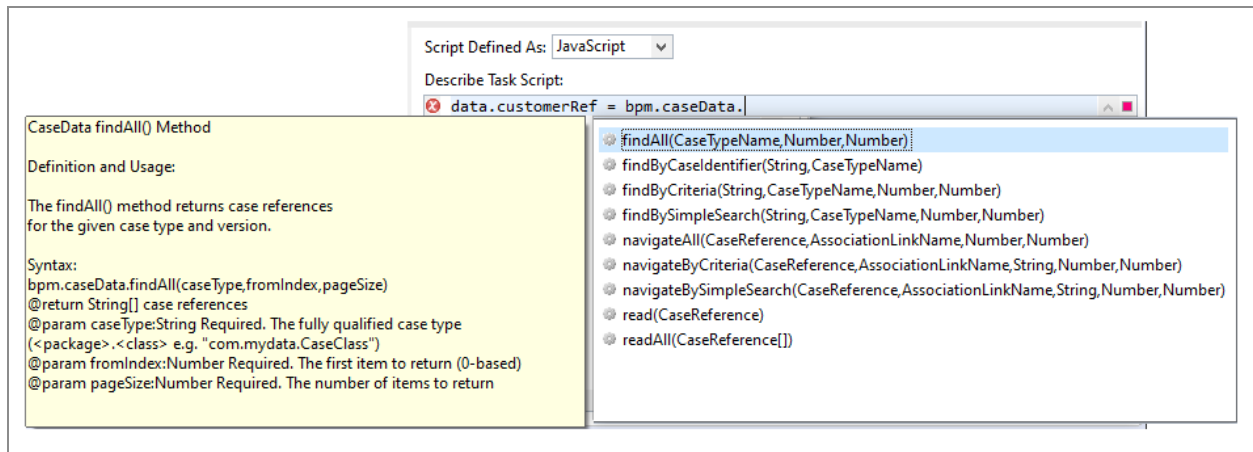
Searching For and Reading Case Objects

An API is available that can be used in scripts to find, navigate to, and read case objects.

This API takes the form:

```
bpm.caseData.<function>
```

where *<function>* is one of the available find, navigate, or read functions. Details of each of the available functions are shown in content assist:



The following sections provide examples of using some of these functions.

Finding a Case Object by its Case Identifier

From within a script, use the `findByCaseIdentifier` method to return the case reference to the case object that matches the specified case identifier value(s).

Method syntax	Description
<code>findByCaseIdentifier(<i>caseId</i>, <i>caseType</i>)</code>	Returns the case reference to the case object that matches the specified <i>caseId</i> case identifier value of type <i>caseType</i> .

where:

- *caseId* is the case identifier of the case object.
- *caseType* is the parent ID followed by the case class name. For example, 'com.example.carapplication.Customer'.

The method returns null if there is no case object that matches the specified *caseId* value(s).

Example

This example finds a customer's ID using the value in a `custID` field, which is assumed to have been populated earlier in the process (for example, by a customer service

representative filling in a form when the customer telephones).

```
// Get the case reference to the Customer class that matches the
// identifier provided in the customerID field.
data.customerRef = bpm.caseData.findByCaseIdentifier
(data.customerID, 'com.example.carapplication.Customer');

// You can now read the reference to get the corresponding Customer
// case data object.
data.customer = bpm.caseData.read(data.customerRef);
```

Finding All Case Objects of a Particular Type

From within a script, use the `findAll` method on the appropriate case access class to return a list of all case references for that case class.

Method syntax	Description
<code>findAll(caseType, index, pageSize)</code>	Returns a paginated list of all case references for the case class.

where:

- *caseType* is the parent ID followed by the case class name. For example, 'com.example.carapplication.Customer'.
- *index* is the (zero-based) number of the first record to return in a paginated list.
- *pageSize* is the number of records to return in each page of a paginated list. A value of -1 means return all records.

Examples

This example shows how to get a list of all Customer case references.

```
// Get a list of all Customer case references
data.customerReferenceList = bpm.caseData.findAll
('com.example.ordermodel.Customer', 0, 100);

// You can now read the list of case references to obtain a list of
// corresponding Customer case data objects.
```

```

var customerList = bpm.caseData.read(data.customerReferenceList);

// Then process those customers.
for (var ix = 0; ix < customerList.size(); ix++)
{
    var customer = customerList.get(ix);

    // process customer objects...
    // ...
}

```

This example shows how to use a paginated list to iterate through the returned list of customers and find the one with the largest number of orders.

```

var maxOrderCount = 0;
var name = "";
var index = 0;
var pageSize = 100;

while (true) {
    var custRefs = bpm.caseData.findAll('com.example.ordermodel.customer',
index, pageSize);
    for (var ix = 0; ix < custRefs.size(); ix++) {
        var customerRef = custRefs.get(ix);
        var count = customerRef.getOrdersRefs().size();
        var customer = customerRef.readCustomer();
        if (count > maxOrderCount) {
            maxOrderCount = count;
            name = customer.name;
        }
    }
    if (custRefs.hasMoreResults) {
        index += pageSize;
    } else {
        break;
    }
}

```

Finding Case Objects by Criteria

From within a script, use the `findByCriteria` method to return a list of case references that match the criteria defined in a query string.

The query must be expressed in Data Query Language (DQL).

Method syntax	Description
<code>findByCriteria (DQLString, caseType, index, pageSize)</code>	Returns a paginated list of case references that match the criteria specified in the <i>DQLString</i> .

where:

- *DQLString* is the query to be executed. See [Case Data Query Language \(DQL\)](#).
- *caseType* is the parent ID followed by the case class name. For example, 'com.example.carapplication.Customer'.
- *index* is the (zero-based) number of the first record to return in a paginated list.
- *pageSize* is the number of records to return in each page of a paginated list. A value of -1 means return all records.

Examples

The following example shows how to use a query string.

```
//Find orders from customers whose name begins with "EasyAs" (returning
references to the first 20).
var ordRefs = bpm.caseData.findByCriteria("customer.name='EasyAs'",
'com.example.ordermodel.Customer', 0, 20);
```

Case Data Query Language (DQL)

DQL is used to define query expressions that can be used as parameters in the `findByCriteria` methods to search for case references that match the criteria defined by the query expression.

DQL expressions use the syntax:

attributeNameoperatorvalue

where:

- *attributeName* can be specified as a simple name
- *operator* refers to =(equal to). Currently, only this is supported. For details, refer [operators](#)
- *value* is the value to match against. It can use [constants](#).

i Note: Only those attributes which are made searchable at design time can be used to search cases through DQL. Case State and Case Identifier are by default searchable.

For example:

```
name = 'Tony Pulis'
```

i Note: This release of BPME currently does not support attributes paths, and values as parameters. Also, it does not support sort, size, between, wildcard characters in DQL. Further, in compound expressions, only AND operator is supported.

Compound Expressions

Simple expressions can be combined with AND to make more complicated expressions. For example, to match records that contain 'London' in the address attribute where the name starts with 'Fred':

```
address = 'London' AND name = 'Fred'
```

i Note: AND is case-insensitive.

Reserved Words

The following are reserved words in DQL:

- and
- asc
- between
- by
- desc

- lower
- not
- of
- or
- order
- size
- type
- upper

Operators

Operator	Description	Notes	Notes/Examples
=	Equal to	<p>Can be used to test for null - for example, to test for attributes and associations that have not been set.</p> <p>Can be used with the following BOM types: Text, ID, URI, Date, Time, Datetimetz, Duration, Integer, Decimal, Boolean, Enumeration.</p>	<pre>name = 'John'</pre>

See Also

[Case Data Query Language \(DQL\)](#)

Values

Values in DQL expressions use constants.

Constants

Type	Notes	Examples
String	<p>Must be single-quoted only.</p> <p>Single quote characters in single-quoted strings must be escaped using a backslash character ('\').</p> <p>A backslash character must be escaped using a second backslash character.</p>	<pre>'bill' '\ 'fred' '\\fred'</pre>
Numeric		<pre>0 -123 123.456</pre>
Date	<p>Must be specified as: <i>YYYY-MM-DD</i></p> <p>A timezone can be specified by appending the date with either "Z", "+HH:MM" or "-HH:MM".</p>	<pre>2013-12-25 2013-12-25Z 2013-12+08:00 2013-05:00</pre> <p>The following expression searches for a date of 2013 or later.</p>

Type	Notes	Examples
	When doing date range checking you can abbreviate dates, leaving off the non-significant parts.	<code>dispatchNote.dateOfDispatch > 2012</code>
Time	<p>Must be specified as: <i>HH:MM:SS.SSS</i></p> <p>When doing time range checking you can abbreviate times, leaving off the non-significant parts.</p>	<p><code>14:23:56.123</code></p> <p>The following expression searches for a time of 09:00:00 or later.</p> <p><code>openingTime >= 09:00</code></p>
Datetimetz	<p>Must be specified as: <i>YYYY-MM-DDTHH:MM:SS.SSS</i></p> <p>A timezone must be specified by appending the date with either "Z", "+HH:MM" or "-HH:MM".</p> <p>When doing range checking you can leave off the non-significant parts.</p>	<p><code>2013-12-25T00:00:01Z</code> <code>2013-12</code> <code>2013-12-25T09+05:00</code> <code>2013-12-25T00:00:01-08:00</code></p>
Enum	Enum names must be quoted with either single or double quotes.	<p><code>orderLines.orderItem.itemEvents.event = 'RECEIVED'</code> <code>orderLines.orderItem.itemEvents.event IN ("RECEIVED", "SHIPPED")</code></p>

Navigating Association Links to find Related Case Objects

From within a script, methods are provided that you can use to navigate association links from the referenced case object to find related case objects.

i Note: You can only use these methods to navigate existing [association links](#). If you try to navigate using a link that either has not been created yet or has been deleted, execution of the script fails.

The following shows the navigation methods that are available.

Method	Description
<code>navigateAll(caseRef, associationLinkName, fromIndex, pageSize)</code>	Returns a list of case references linked to the supplied case.
<code>navigateByCriteria(caseRef, associationLinkName, DQL, fromIndex, pageSize)</code>	Returns a list of case references linked to the supplied case that match the DQL search string.
<code>navigateBySimpleSearch(caseRef, associationLinkName, searchString, fromIndex, pageSize)</code>	Returns a list of case references linked to the supplied case that match the search string.

where:

- *caseRef* - The case reference to search from.
- *associationLinkName* - The field name linking the cases to return.
- *DQL* - The [Data Query Language](#) (DQL) search string.
- *searchString* - The search string.
- *fromIndex* - The first item to return -- zero based.
- *pageSize* - The number of items to return.

Association Links and Association Relationships

Association links allow you to navigate from one case object to another, using predefined association relationships between the corresponding case classes.

An association relationship exists between two case classes. You [create association relationships](#) when you create a case data model.

An association link exists between two case objects:

- An association link does not exist automatically because there is an association relationship between two case classes. You must [create an association link](#) from the source case object to the destination case object from within a business process using a **Case Data Operations** service task.
- An association link can only be created between case objects where there is an association relationship between the source and destination case classes.
- Once created, an association link exists until you [delete it](#), which you must do from within a business process using a **Case Data Operations** service task.

Note: When using script methods to [navigate association links](#) from within a particular process, it is important to realize that association links can be created, navigated or deleted by any process that has access to the case reference for the source case object of the link. The process design might therefore need to account for the fact that a particular association link might or might not exist at runtime.

Creating an Association Relationship Between Two Case Classes

Association relationships define potential relationships between two case classes - for example, customers and orders. From within a business process, you can use these relationships to create links between case objects, and to navigate those links when searching for case objects.

An association relationship must be bi-directional, and optional in each direction. (Neither end of the relationship can have a multiplicity value of "one" or "one or more".) You can define multiple associations between the same two classes, if appropriate.

Before you begin

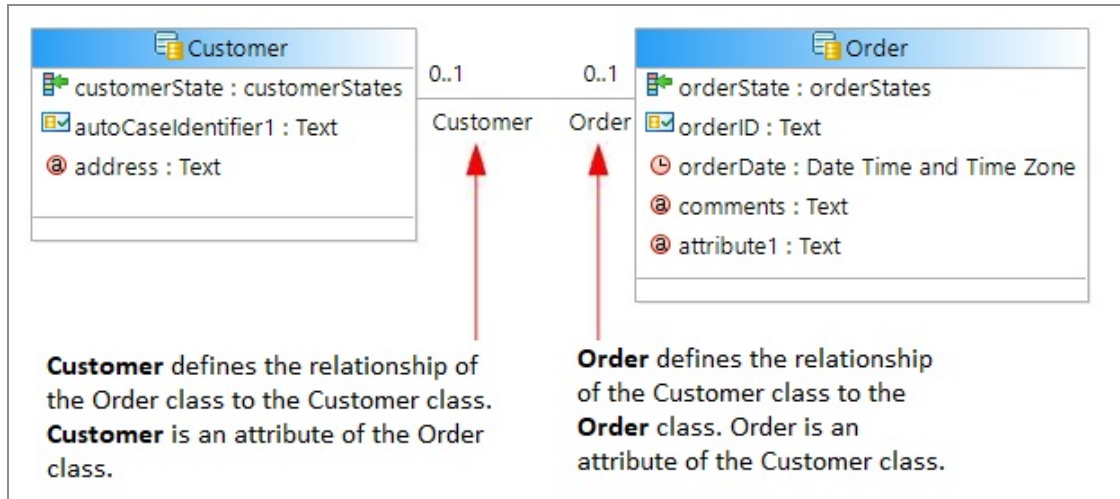
The case classes that you want to associate must already exist.

Procedure

1. In the BOM Editor, click **Association** in the Relationships section of the palette.

- Click one case class and drag the relationship to the other case class.

This creates a bi-directional association relationship between the two case classes. By default, each end of the relationship has the name of the destination class, with a multiplicity of 0..1. For example:



- For each end of the relationship, edit the **Multiplicity Value** and identifying **Label/Name** as needed to define the intended relationship. For example:

You can either select and edit the relationship's **Label** and/or **Multiplicity** directly on the canvas...

... or edit them from the **Attributes** tab of the **Properties** view of the parent class.

In this example, the values for the Customer to Order relationship have been changed to show that a single Customer can have zero to many Orders.

Label	Name	Type	Multiplicity
customerState	customerState	com.example.association::customerStates	1
autoCaselIdentifier1	autoCaselIdentifier1	Bom Primitive Types::Text	1
address	address	Bom Primitive Types::Text	0..1
orders	orders	com.example.association::Order	*

Note: It is good practice to keep a relationship's **Label** and **Name** the same. This makes it easier to avoid mistakes when constructing, for example, an attribute path in the DQL query on a `navigateByCriteria` method.

What to do next

From within a process that references this case data model, you can use association relationships to:

1. [create association links between case objects](#) of the related case classes. You can also [delete association links](#) when they are no longer required.
2. [navigate those links](#) when searching for case objects.

Using the relationship shown above, this would enable you to, for example, find all orders placed by a particular customer on a particular date.

Creating an Association Link

From within a process, use a **Case Data Operations** service task to link one case object to another. Association links allow you to navigate from one case object to another, from any process that has access to the case references of the source and destination case objects.

Before you begin

An [association relationship](#) must exist between the source and destination case classes. Local data fields containing case references to the source and destination case objects must already exist.

Procedure

1. Add a service task at an appropriate point in the process.
2. On the **General** tab of the **Properties** view, set **Service Type** to **Case Data Operations**.
3. Select **Change or Delete Case Object(s) Using Case Reference Field**.
4. In the **Case Reference Field**, enter the name of the data field that references the source case object from which you want to add the link.
5. In the **Operation** field, select **Add Link(s) To Other Case Object(s)**.
6. In the **Case Class Association** field, enter the name of the association link that connects the source case class to the destination case class.

Content assist displays the association links that are available from the case class referenced in the **Case Reference Field**.
7. In the **Case Object Reference(s)** field, enter the name of the data field that references the destination case object for the specified **Case Class Association** field.

What to do next

1. (Optional) Add appropriate error handling to the service task boundary to deal with the following specific errors that might be returned by the **Add Link(s) To Other Case Object(s)** operation.

Error (Error Code)	Description	Possible solutions
CaseOutOfSyncError (CaseOutOfSyncError)	<p>The source case object referenced from the Case Reference Field has been modified in some way since the field was last populated. That is, the case reference is stale.</p> <p>This could be either a change to the data held in the case object, and/or a change in the object's internal version number.</p>	<ol style="list-style-type: none"> Re-read the case object. If necessary, examine the case data to check that no significant changes have occurred. Retry the Add Link(s) To Other Case Object(s) operation.
UserApplicationError (CaseNotFoundError)	The source case object referenced from the Case Reference Field does not exist.	
UserApplicationError (AlreadyLinkedError)	The link to the destination case object cannot be created because it already exists.	
UserApplicationError (NoTargetsError)	The link cannot be created because the case reference specified in the Case Object Reference(s) field has a null value.	

2. (Optional) Modify any process that might need to do something if this association link is created:
 - Add a [case data signal event](#) to the process, so that the process is notified when the association link is created.

- b. Add suitable business logic to perform any actions that the process needs to take as a result of the association link being created.

Deleting an Association Link

From within a process, use a **Case Data Operations** service task to delete an existing association link between two case objects. Association links allow you to navigate from one case object to another, from any process that has access to the case references of the source and destination case objects.

You cannot delete a case object that has an association link to another case object.

Before you begin

An [association link](#) must exist between the source and destination case objects. Local data fields containing case references to the source and destination case objects must already exist.

Procedure

1. Add a service task at an appropriate point in the process.
2. On the **General** tab of the **Properties** view, set **Service Type** to **Case Data Operations**.
3. Select **Change or Delete Case Object(s) Using Case Reference Field**.
4. In the **Case Reference Field**, enter the name of the data field that references the source case object from which you want to add the link.
5. In the **Operation** field, select **Remove Link(s) To Other Case Object(s)**.
6. In the **Case Class Association** field, enter the name of the association link that connects the source case class to the destination case class.
Content assist displays the association links that are available from the case class referenced in the **Case Reference Field**.
7. In the **Case Object Reference(s)** field, enter the name of the data field that references the destination case object for the specified **Case Class Association** field.

What to do next

1. (Optional) Add appropriate error handling to the service task boundary to deal with

the following specific errors that might be returned by the **Remove Link(s) To Other Case Object(s)** operation.

Error (Error Code)	Description	Possible solutions
CaseOutOfSyncError (CaseOutOfSyncError)	<p>The source case object referenced from the Case Reference Field has been modified in some way since the field was last populated. That is, the case reference is stale.</p> <p>This could be either a change to the data held in the case object, and/or a change in the object's internal version number.</p>	<ul style="list-style-type: none"> a. Re-read the case object. b. If necessary, examine the case data to check that no significant changes have occurred. c. Retry the Remove Link(s) To Other Case Object(s) operation.
UserApplicationError (CaseNotFoundError)	The source case object referenced from the Case Reference Field does not exist.	
UserApplicationError (NotLinkedError)	The link to the destination case object cannot be deleted because it does not exist.	
UserApplicationError (NoTargetsError)	The link cannot be deleted because the case reference specified in the Case Object Reference(s) field has a null value.	

2. (Optional) Modify any process that might need to do something if this association link is deleted:

- a. Add a [case data signal event](#) to the process, so that the process is notified when the association link is deleted.
- b. Add suitable business logic to perform any actions that the process needs to take as a result of the association link being deleted.

Reading Case Objects

From within a script, you can read case objects by using the `bpm.caseData.read()` or `bpm.caseData.readAll()` methods.

Method	Description
<code>read(caseRef)</code>	Returns the case object for the given case reference.
<code>readAll(caseRefs)</code>	Returns a list of case object for the given case references.

Processing a Paginated List of Case References

When a paginated list of case references is returned by a method on a case class or case reference, the list object has a `hasMoreResults` attribute that can be used to determine whether there are more pages of results that could be returned.

Paginated lists can be returned by the following the `findAll()` method:

Example

This example shows how to use a paginated list to iterate through a returned list of customers.

```
var maxOrderCount = 0;
var name = "";
var index = 0;
var pageSize = 100;

while (true) {
```

```

        var custRefs = bpm.caseData.findAll('com.example.ordermodel.Customer',
index, pageSize);
        for (var ix = 0; ix < custRefs.size(); ix++) {

            // Process the case reference...

        }
    }
    if (custRefs.hasMoreResults) {
        index += pageSize;
    } else {
        break;
    }
}

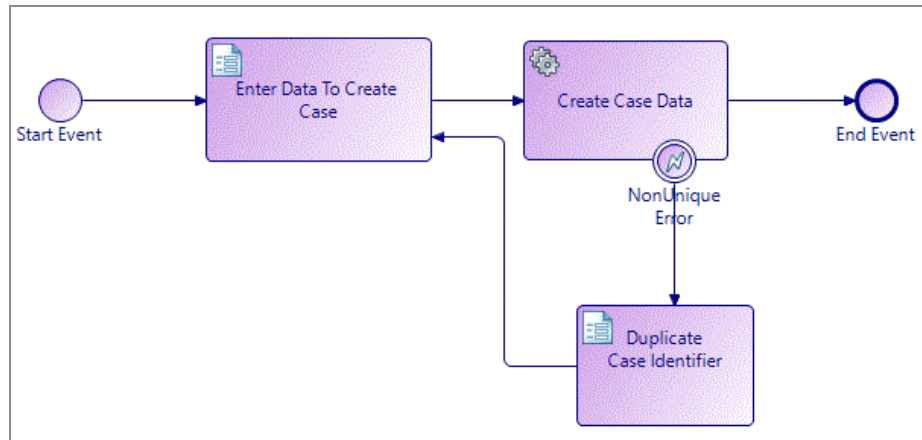
```

Generating a Business Service to Create/Update/Delete Case Data

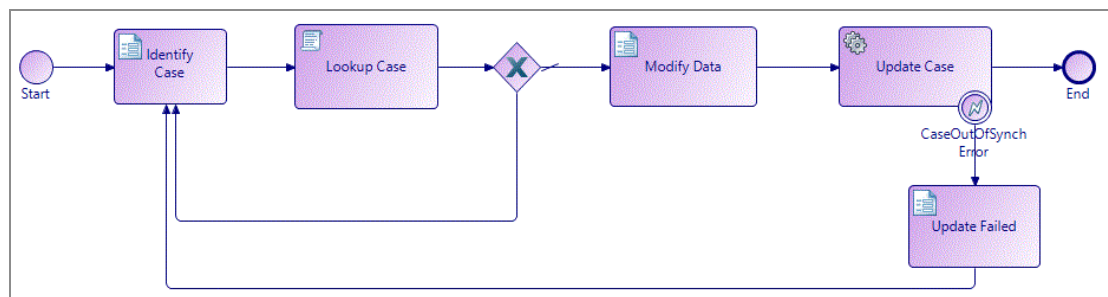
You can generate a business service from Case Class. The business service generated contains the relevant tasks to create, update or delete case data, depending on which option you selected when generating the business service.

Procedure

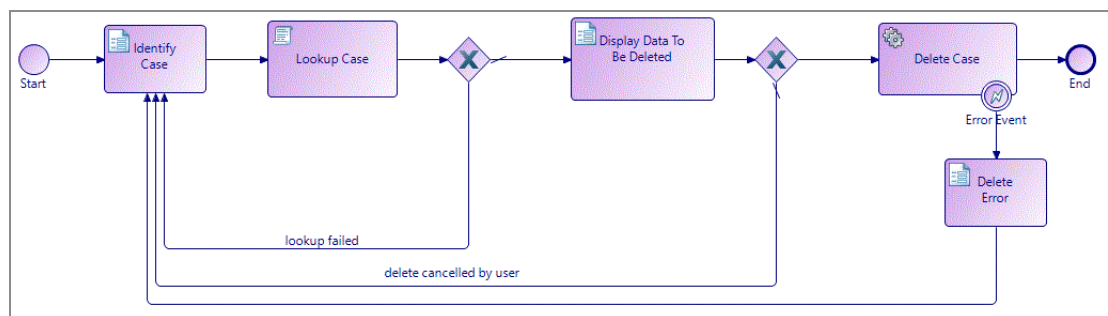
1. Right-click a Case Class in a BOM, and select **Generate Business Service**.
2. Select one of the following options:
 - **To Create Case Data:** This creates a case data business service designed to allow you to create case data with appropriate runtime error handling, should a case object with the given case identification already exist.



- **To Update Case Data:** This creates a case data business service designed to allow you to update case data with appropriate runtime error handling, should a case object with the given case identification not exist or have been updated since the lookup was performed.



- **To Delete Case Data:** This creates a case data business service designed to allow you to delete case data with appropriate runtime error handling, should a case object with the given case identification not exist or have been updated since the lookup was performed.



3. From the Generate Business Service dialog box, open the project and Process Packages where you want to place the xpdL. You can also select an existing xpdL.

Click **Finish**.

What to do next

When you have generated a case data business service, you can use it to edit case data. You are most likely to do this when you have an ad-hoc request not related to one of your normal business processes (for example, a customer has notified you of a change of address, and you want to make this change independent of any other changes to their order information. In this example you could use the Update Case Data business service to make the relevant changes to their data).

Configuring Events

Events

An Event in a process is something that happens that affects the sequence or timing of activities in a process.

There are three main types of Event: Start, Intermediate and End.

- Start events are used to indicate the start of a process or to control how a process is started (or triggered).
- Intermediate events can throw or catch events with a specified trigger type after a process instance has started. In-flow Catch signals halt the flow until the event is triggered. Task boundary signals affect the task they are attached to if the event is triggered while the task is in progress.

Event handler flows / Event Sub-Processes are separate flows that can be triggered and processed any number of times during the lifetime of a process instance.

- End events are optional, and indicate the end of a flow or branch.

Start Events

Start events can be used to initiate the start of a process or event sub-process (they are optional). Different types of start events control how a process is started (or triggered). All start events are "catch" events:

Start events are available in the main process, in embedded sub-processes, and in event sub-processes.

- **Start Request Event** - There is no specific trigger to start the Process. **Start Request Event** is used for start events in the main process and in embedded sub-processes. In the main process this indicates where the process starts when initiated manually by a user, via a sub-process invocation or triggered via the public REST API from a third-party application. In an event sub-process, the start event is triggered through the

public REST API.

- **Signal** - The start of the process is triggered by the receipt of a signal. Signal Start is only possible for an event sub-process.

Intermediate Events

Intermediate events can throw or catch events with a specified trigger type after a process instance has started.

Intermediate events can be used as follows:

- Intermediate events can be used "in-flow", that is, between two other activities.
- Most catch-type intermediate events can be attached to a task boundary to catch a triggered event only while that task is in-progress.
- You can define an event handler by starting a flow with an intermediate event. See [Event Handlers](#) (trigger type support is destination specific).

You can use the following types of intermediate events:

i Note: There are some restrictions on the placement of intermediate events. For example:

- Intermediate events of type **Link** cannot be placed on the boundary of a task.
- Intermediate events of type **Cancel** cannot be placed in sequence flow.

Restrictions are placed upon the ways that you can use events; these are enforced via problem markers.

- **Incoming Request Event** - Event that is triggered via the public REST API.
- **Catch Timer** - The event is triggered at a specific date/time or at a regular interval (time cycle). When placed on the boundary of a task, a timer event defines a deadline for the task. In the Properties view for the event, **Use as activity deadline** is preselected. If more than one timer event is attached to a task, only one of them can be selected as the timer deadline. If there is a canceling timer, then it must be selected as the deadline.

There are two options you can select from to decide how the task is treated if the

event times out (**Withdraw Task on Timeout** and **Continue Task on Timeout**).

- **Throw/Catch Link** - Indicates a connection from one or more throw link intermediate events to a catch link intermediate event in the same parent process/parent embedded sub-process. This can be thought of as a "go to" or "off page connector" that you can use to break up a process for better legibility.
- **Throw/Catch Signal** - Broadcasts or catches a signal. A throw signal event is assigned a default signal name (**signaln**). You can use a catch signal attached to a user task boundary to resend data to the outstanding work item for that task
- **Throw Error** - Attached to a task boundary to end a sub-process with an error.
- **Catch Error** - Attached to a task boundary to catch an application error that occurs during the processing of that task. Either catches the specified error, or catches any error if no specific error is specified.
- **Catch Cancel** - Catches a sub-process cancel.
- **Throw/Catch Compensation** - Used to process compensating activities for previously executed tasks:
 - If located in a sequence flow of the process, this event throws a call for compensation.
 - If attached to the boundary of an activity, this event catches a named compensation call.

End Events

An end event indicates when the process has been completed. They are optional, however if a process contains a start event, it must contain an end event.

End events have different types that indicate different results upon completion of the process. All end events are "throw" events:

- **None** - There is no specific end result to the process.
- **Error** - Ends all activities in the process immediately without compensation or events, and appears in the Event log as a failed process instance.
- **Compensation** - Indicates that a compensation is necessary. For more information, see the BPMN specification at the [BPMN Website](#).
- **Signal** - Indicates that a signal is broadcast at the end of the process. A signal end

event broadcasts a default signal name (**signalIn**).

- **Cancel** - Used within a transaction sub-process to trigger a cancel intermediate event attached to the sub-process boundary.
- **Terminate** - Ends all activities in the process immediately without compensation or events.

Setting Event and Task Visibility (Private and Public)

The visibility of an event or task (whether it is private or public) controls whether process information (such as required parameters) is available to an external process or application.


Setting the visibility to public results in a subset of events or steps that are then available to external processes or applications.

Note: The runtime effect of this setting is entirely destination specific.

For each event or task, you can specify a list of expected input fields (on the **Interface** tab), and define the URL of the document describing the purpose of the public event or task (on the **Description** tab).

By default, events and tasks in a process are private; in a process interface, they are public. If you want to change the visibility of an event or task:

Procedure

1. Select the event or task.
2. On the **Interface** tab, select the visibility (**Private** or **Public**).
3. Click  to add parameters to the event or task.
Use the **Mandatory** check box to specify parameters that must be present.
4. (Optional) Select the **Description** tab, then enter a URL that provides documentation describing how the event or task is used.

Throw and Catch Events

An event that is located in a sequence flow can "throw" an event that can be "caught" by a catch event.

Catch intermediate events can be placed in flow, or on a task border. An in-flow catch event halts the flow until the event is triggered. A catch event attached to a task border usually cancels the task when the event is triggered.

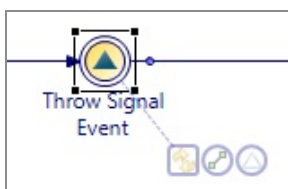
Creating References Between Throw and Catch Events

As an alternative to creating references between throw and catch events in the Properties view, graphical tools called gadgets that allow you to easily create references.

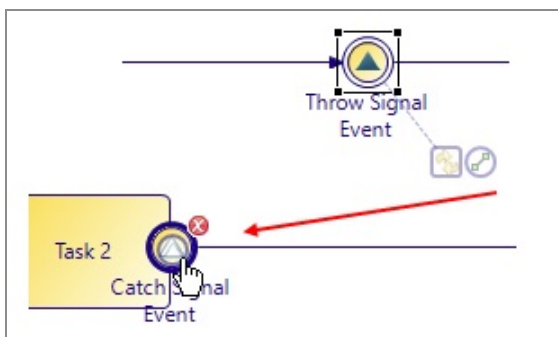
The example in this section shows throw and catch signal events, but the same procedure applies to all throw and catch events.

Procedure

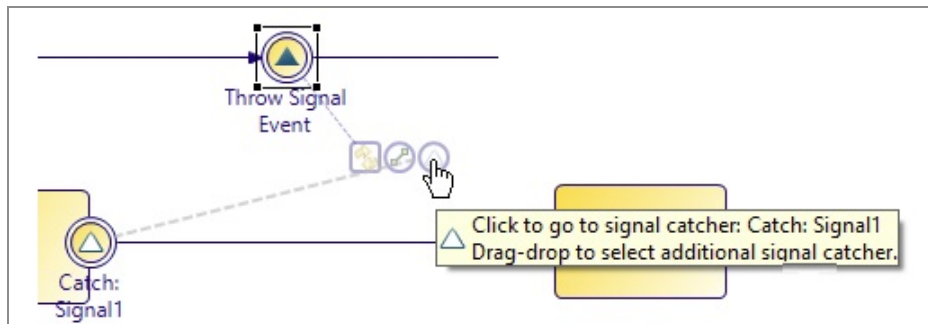
1. Place the throw and catch events on the diagram.
2. Select the throw event.



3. Drag the event reference gadget to the desired catch event.



- Once you link a throw and catch event, you can click the event reference gadget to go to the catch event, or drag the event reference gadget to select additional catch events:



Throw and Catch Signal Events

With throw/catch signal events, an *in-flow* signal event broadcasts a signal, and if there is an active task with a signal event on the task boundary, that event "catches" the signal, and generally follows the exception path.

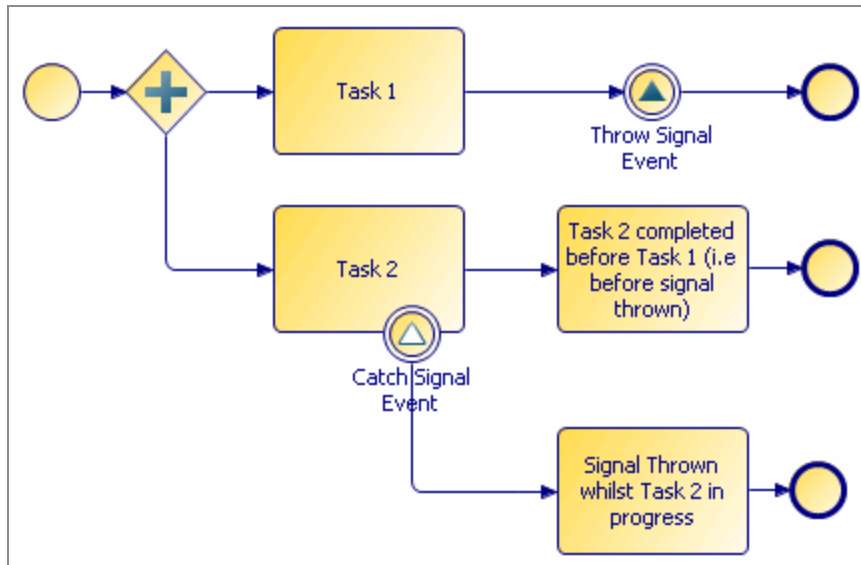
Signal events can be local or global.

Local signal events can be used for triggering event handlers, task cancellations and to halt and resume parallel flows. When attached to a user task they can be used to resend data to an outstanding work item for that task.

Global signal events are thrown and caught across process instances, allowing processes to collaborate with each other. They are only available for an event sub-process/handler. See [Signals](#) for information about defining signal correlation data that identifies the process instance waiting for the global signal.

- You can drag-and-drop a global signal event onto a process to create throw / catch events.
- A catch signal event can cancel the task it is attached to or, for user tasks, can allow the task to continue. If they are set to "continue", they cannot have an outgoing flow but they can map data into the task that they are attached to and also reschedule timers attached to the same task.

A signal is transient: when it is thrown, if there is nothing to catch it at that moment, it is discarded. It is not stored for later use.



You might specify a signal name with a throw signal event. With a catch signal event, a signal name is optional:

- If a catch signal event has no signal name specified, it catches any thrown signals in the process (while the task it is attached to is active).
- If a catch signal event has a signal name specified, it only catches signal events that throw the named signal.

When you specify signal names in the Properties view for signal events, you can use content assist. This means, for example, that if you have entered SIGNAL1 for a throw signal event, when you specify the signal name for the catch signal event you can press **Ctrl+Space** and the available signal names are displayed from which you can select one.

You can define whether you want to update a scheduled work item from a non-canceling signal event. Select from a pair of radio buttons directly beneath the signal name from the Properties view of a catch signal event attached to a task boundary:

- Cancel task when signal is caught
- Continue task when signal is caught

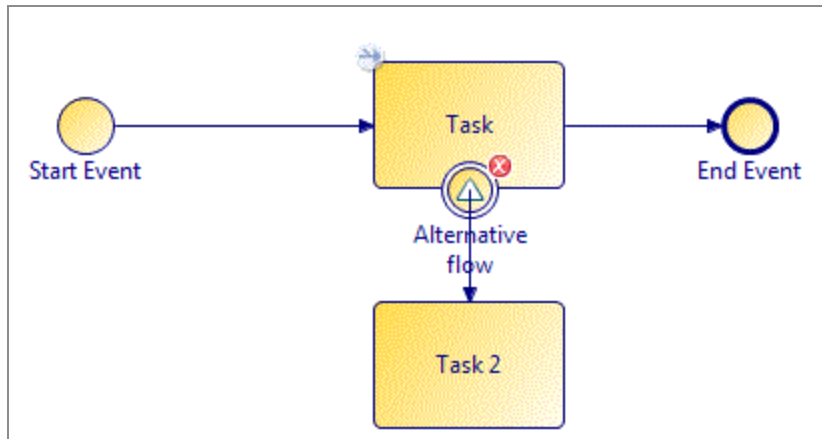
Use the **Map from Signal** tab to map from the throw-signal's payload data (listed in its **Interface** tab) to the data associated with the attached user task.

A **signal end event** functions like any other inflow signal event, except that when one fires, if there are more tasks that need to complete as the result of an exception flow, the outstanding tasks complete; if there are no more tasks that need to be completed, the process ends.

Catch Signal Events

In BPMN, when an event on the boundary of a task is triggered, the task is canceled and the alternative flow from that event is followed.

For example:

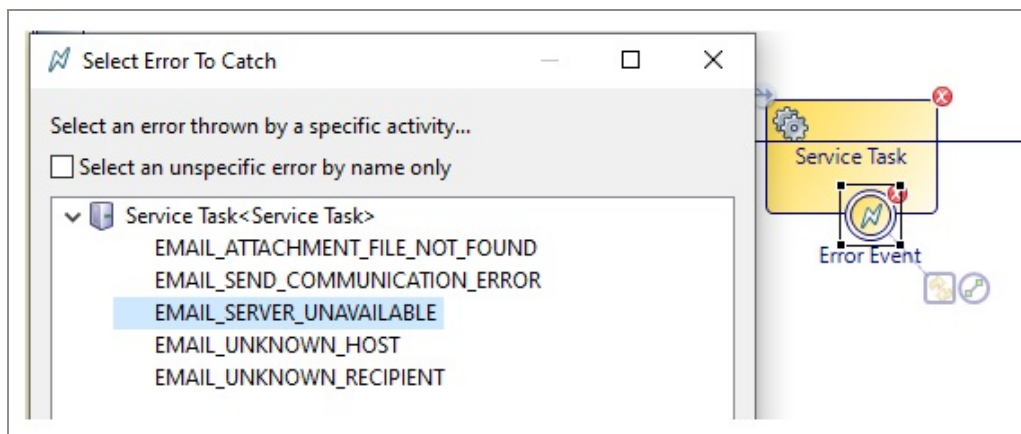


In this example, if the signal event is triggered, the task to which it is attached is canceled and the sequence flow to Task 2 is followed.

Throw and Catch Error Events

An error event attached to a task boundary can be set to catch any error, or errors, thrown by the task to which it is attached.

For example, when attached to an email service task, you can configure the catch error event to catch emails when the email server is unavailable:



When attached to a call sub-process activity or an embedded sub-process task, the event can catch errors thrown by any of the following:

- Activities within the sub-process.
- End error events
- Process interface error events (if the call sub-process activity references a process interface rather than a process)
- Any other error throwing activity executed within the sub-process (including its sub-process tasks). This includes activities whose errors cannot be caught directly by attaching the error event.

Configuring Error Events

Error events are configured in the Properties view.

For a catch error event, you can do the following:

- **Catch All Errors**

(default) Catches any error thrown by any event.

- **Catch Named Errors**

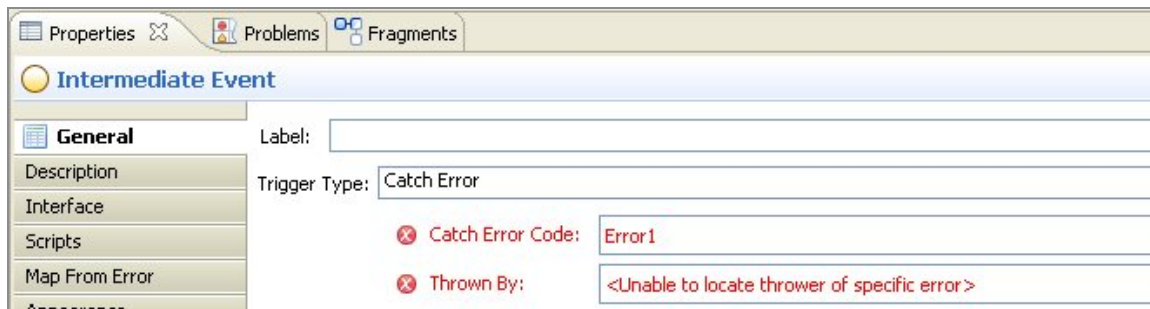
To do this, click **Select Error** and select the **Select an unspecific error by name only** check box. The dialog box lists the catchable error codes. Select the error codes that you want to catch. This configures the event to catch the selected error from any activity.

- **Catch Specific Errors from Specific Activities**

Click **Select Error** and deselect the **Select an unspecific error by name only** check box. The dialog box lists events and activities that generate error codes. Expand the desired event or activity to catch a specific error thrown by a specific activity.

Fixing Invalid Error Events

After configuring a catch error event, the event that throws the error code is deleted, invalidating the catch error event as shown in the Properties view.



Other causes of invalid error events include:

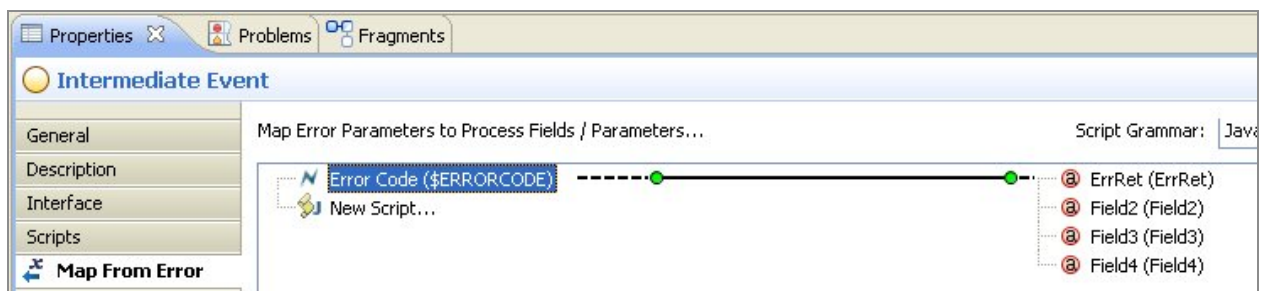
- The event is detached from the task.
- The activity no longer throws the given error.
- The activity that throws a specific selected error is no longer processed as a result of the task to which the event is attached.

To fix the error event, click **Select Error**. If possible, the closest match to the originally selected error code is identified. Select an error code and click **OK**.

Error Data Mapping

You can map error parameters to process data from the **Map From Error** tab.

For example:



The process data available on the right side of the tab is either all data or a subset of data as specified on the selection on **Interface** tab.

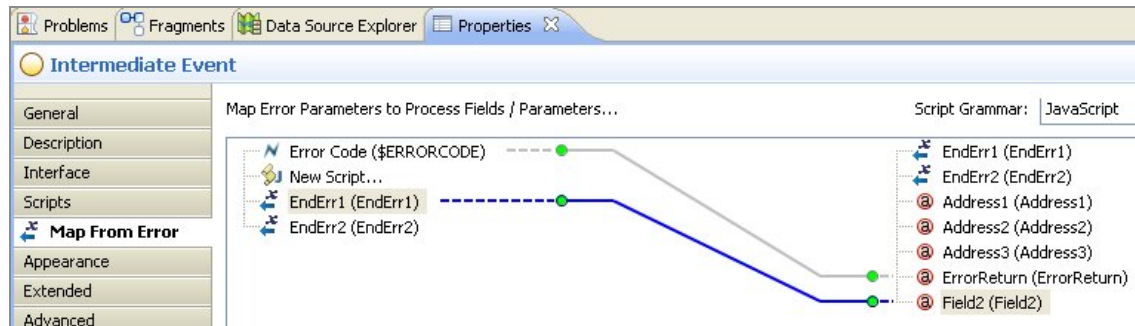
The content of the left side of the **Map From Error** tab depends on the configuration of the **General** tab:

- **Catch All / Catch Named Errors**

Only the automatically provided error code can be used for mapping as shown in the previous example. Typically, this error code is mapped to a process text data field or parameter for display to the user. At run time, the text data field or parameters is populated with the error code name when the error is caught.

- **Catch Specific Error Thrown By Sub-Process End Error Event**

All parameters with a mode of **Out** or **In/Out** that are associated with the throw error end event (on the **Interface** tab) are displayed for mapping:



Event Handlers

Event Handlers are supported in Business processes and Pageflow processes. You can use event handlers to execute a flow that is separate from the main flow of the process (for instance to update process data used by the main flow).

Event handlers can be triggered **zero** or more times during the life of a process instance.

✓ **Tip:** The flow from an event handler cannot be joined to the main flow (the flow from a start activity) or another event handler flows.

While an event handler allows you to do something multiple times during a process, it does not have to happen for the process to complete.

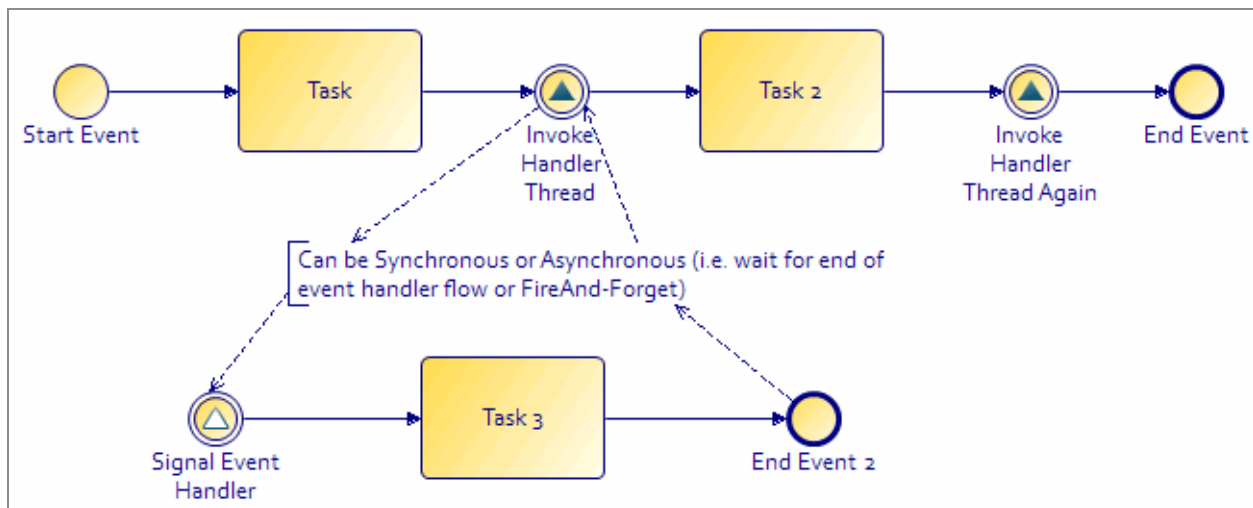
An event handler is a catch intermediate event with **no** incoming flow. Event handlers with no specific trigger type normally are triggered through a destination-specific API or utility.

You can execute a throw/catch cancellation event handler flow (only one per process) to manage the cancellation of a process and its sub-processes. When you cancel a sub-process call activity, the sub-processes that the parent process created are canceled first. The sub-processes can be designed to execute compensation events to close down gracefully.

Signal Event Handlers

An event handler flow can be triggered from within the same process instance using signal events. The signal event handler flow can be executed asynchronously or synchronously with the invoking flow (the invoking flow generally waits at the throw-signal event for the completion of the event handler flow before continuing). This allows process designers to define and re-use complex sets of repeatable activities.

i Note: This could be especially useful when considering the additional tasks and exception handling that are required by case data updates because each update on a single piece of case data could involve several tasks (one to do the update, plus exception handling if the update fails). Therefore if many updates are performed on case data, then the process could soon become very cluttered with case data handling activities reducing the visibility of the actual business process tasks. Use of the signal event handler could greatly reduce this type of problem by allowing the process designer to implement the set of tasks to update a case data item and handle exceptions only once and re-use that from many places in the process.



The catch signal event Properties tab contains Event Handler configuration controls (which are shown only when the catch signal is an event handler (with no incoming flow)).

The default setting is **Wait at the invoking signal until event handler flow is complete**. This setting controls whether the throw signal's outgoing flow is processed after the event handler flow. If you unset this flag it can be processed without waiting.

Note that:

- Signal event handlers can also be used within embedded sub-processes. Place the signal event handler inside the embedded sub-process. This ensures that the embedded sub-process local data is in scope of the signal event handler flow.
- Signal event handlers can be used within pageflow processes.
- Case data signal event handlers allow a process to subscribe to a particular case object that the process uses. The process then is notified if that case object is modified, and can take appropriate action to respond to the change. See [Notifying a Process That a Case Object it Uses Has Been Modified](#).


Using the Cancellation Event Handler


You can execute a cancellation event handler flow (only one per process) to manage the cancellation of a process and its sub-processes.


Procedure

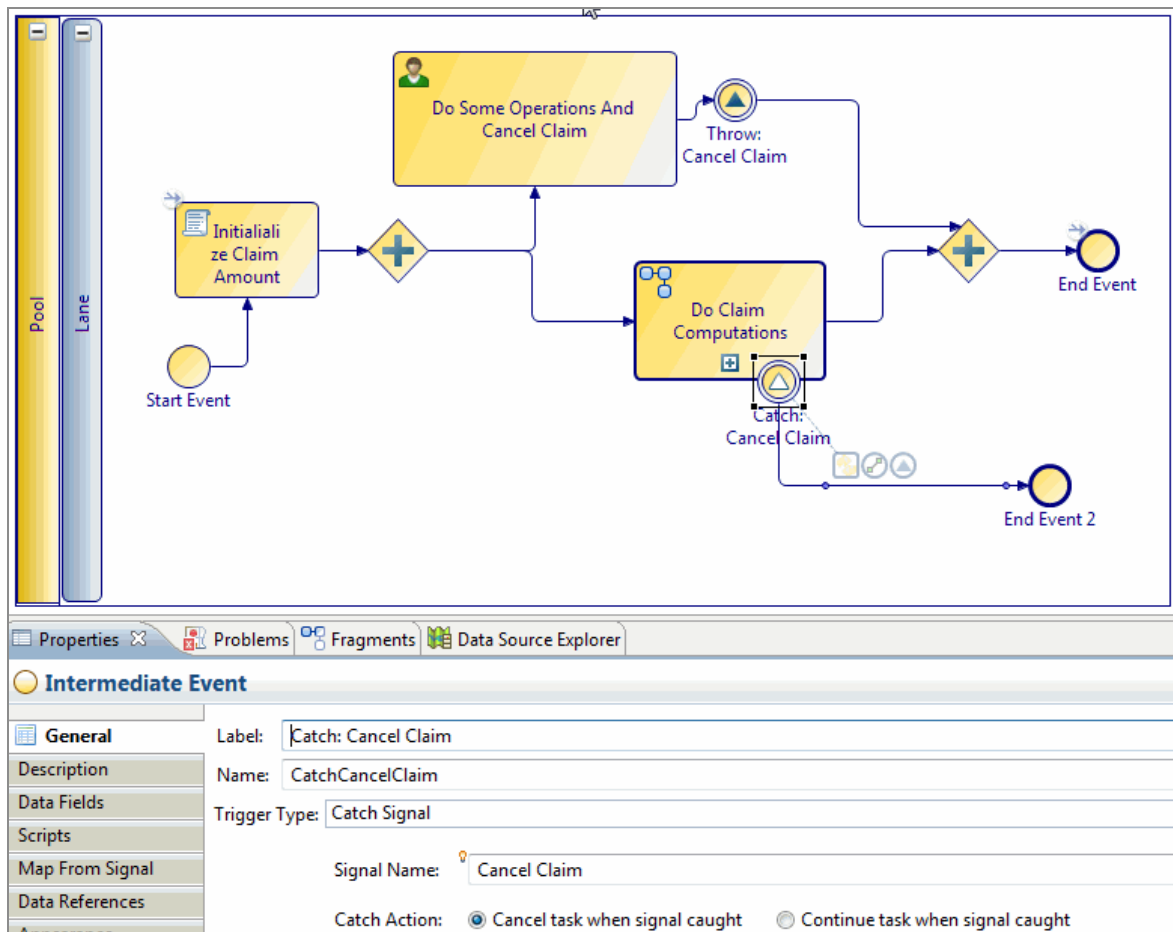
1. **How to cancel a sub-process task:** Attach a catch intermediate event to a call sub-process task in your main process, with a trigger type of Catch Signal. The Catch Action should be **Cancel task when signal is caught**.

Alternatively, you cancel a process sub-process task with a task boundary event also with the following event types:

Incoming request event: 

Timer: 

Signal: 

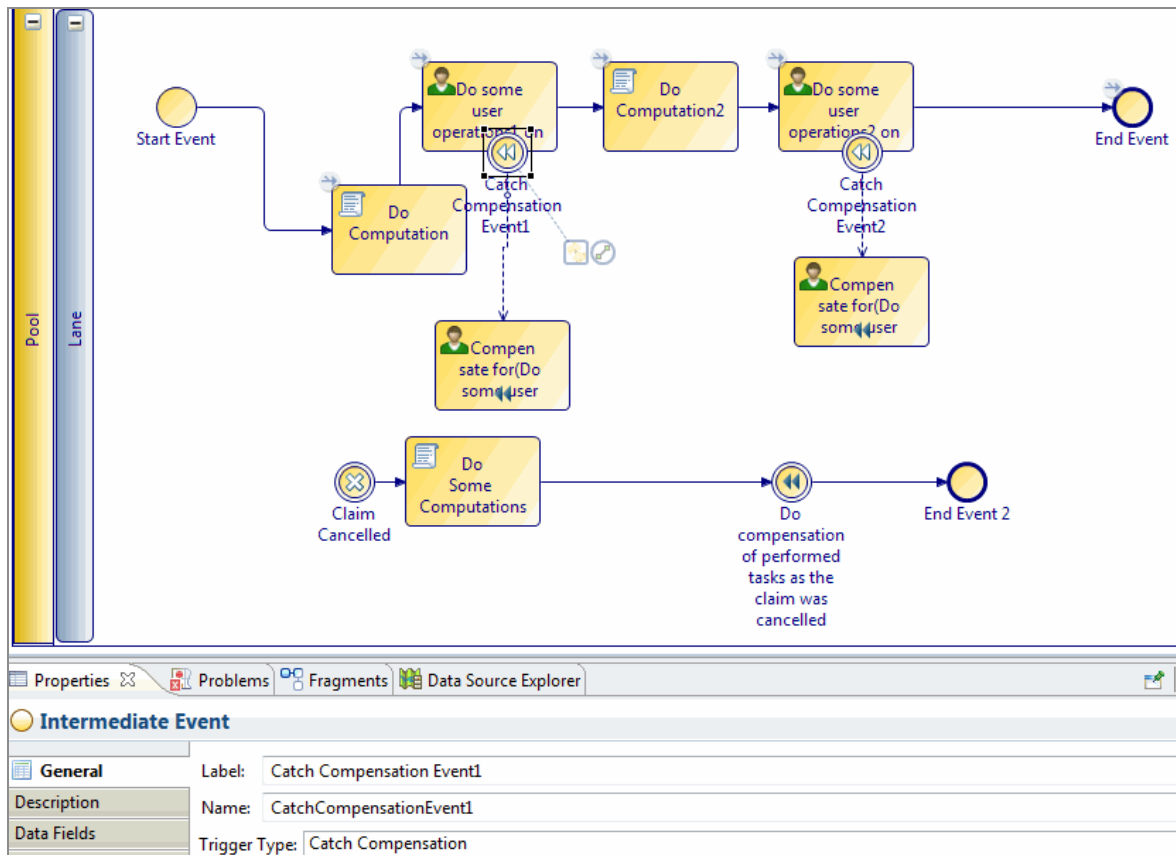


The tree of subprocess instances that were instantiated from a sub-process task are canceled from the bottom up. If the sub-processes need to execute compensation actions to close down gracefully, they can do so using Cancellation Event handlers.

2. **The Sub-Process Cancellation event handler:** Add activities to perform any actions required prior to the final sub-process cancellation. A common pattern would be to use throw-catch compensation as in the example below, where there is an intermediate catch event with trigger type Catch Compensation on each task in the sub-process which requires compensation events. Compensation events execute their actions if only when the task they're attached to has been executed, thus there is not a need to manually conditionalize compensating tasks, depending on the progress through the main process flow.

Note: Cancel event handlers are only executed for process-driven cancellation (not for user/API-driven process cancellations).

Note: The cancel event handler does not have to trigger compensation event handlers. It might either generate signal events into the main part of the process or just perform the compensatory actions directly.

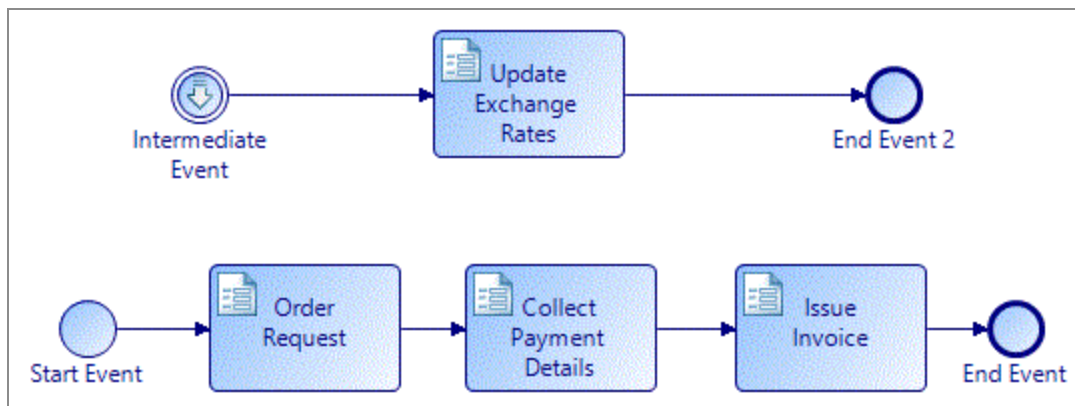


Note:

- The sub-process can also invoke another sub-process (and so on) with more cancel event handlers.
- There does not have to be a cancel event handler in each sub-process if it is not required.
- Cancellation event handlers are not executed when a process instance is canceled through the Process Management Service API.
- If a cancellation event handler fails, then you receive an audit event and can recover from a halted state.

Event Handlers in Pageflows

You could use the event handler to update the process data. Another way to use it would be to add to the runtime identifier array and associated input mapping arrays to add instances to a dynamic sub-process invocation.

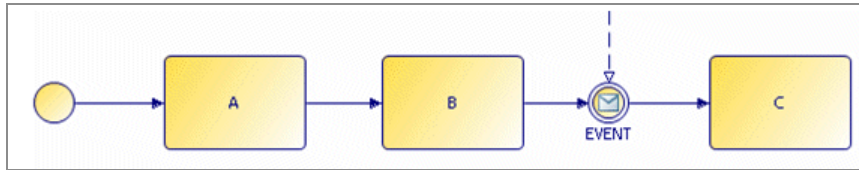


In this example, a pageflow process is used to manage an order request. The event handler is used to update exchange rate information regularly. This updated exchange rate information is then used as input to the task **Issue Invoice**.

Event Persistence

Because a process interface does not specify the intended location of the event within the flow of the process implementation, events are considered "persistent".

As such, they can be triggered at any point in the process flow that is upstream of the event, or when the flow is paused waiting for the event to be triggered. For example:

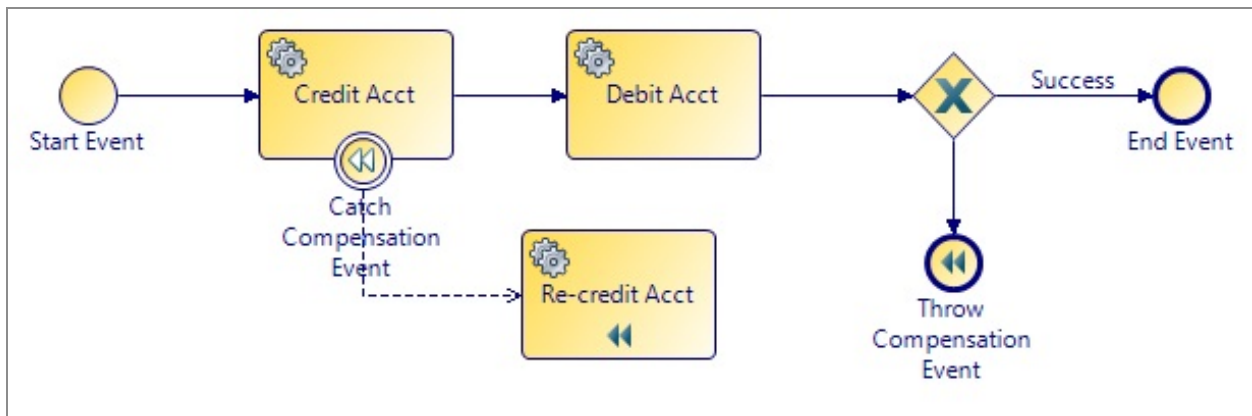


The trigger for the intermediate event can be received while Task A or B is being processed, or when the flow is paused at the intermediate event waiting for the event to be triggered. In either case, Task C is not processed without an event trigger. If the trigger is received during processing of Task A or B, the trigger arrival is persisted, and the event is triggered immediately when it is reached in the flow.

Compensation Events Configuration

Compensation events are used to reverse the effect of previously executed tasks.

For example:



In this example, if the credit/debit fails, a compensation event is thrown. The target of the event is the **Credit Acct** task, which catches the error and proceeds with the **Re-credit Acct** task. In the Properties view for the **Throw Compensation** end event, you can press Ctrl+Space to see a list of tasks from which you can select the target of the compensation. Alternatively, you can compensate all previously executed tasks by not specifying a target for the compensation:

Intermediate Event

General

Label: Throw Compensation Event

Name: ThrowCompensationEvent

Trigger Type: Throw Compensation

Select specific task to compensate (no selection = compensate all):

- Credit Acct
- Debit Acct
- Re-credit Acct

Timer Event Scripts

Scripts can be added to Start events or Catch Intermediate events in the Properties view, if the event is defined as triggered by a timer.

There are several script types available from the **Script Defined As** list:

- **Free Text** - You can use this field to enter text that describes the desired behavior for the script.

Note: On the **General** tab, there might already be text comments describing the required script. These comments are lost if you change the select **JavaScript** from the **Script Defined As** list. If you need to save these comments, copy them before changing the script type.

- **Constant Period** - this allows you to specify the timeout period after the event is initiated using the following time units.

Script Defined As: Constant Period ▾

Specify timeout as offset from event initiation:

Years:	<input type="text" value="0"/>	Hours:	<input type="text" value="0"/>
Months:	<input type="text" value="0"/>	Minutes:	<input type="text" value="0"/>
Weeks:	<input type="text" value="0"/>	Seconds:	<input type="text" value="0"/>
Days:	<input type="text" value="0"/>	Micro Seconds:	<input type="text" value="0"/>

- **JavaScript** - this script type allows you to enter JavaScript statements in the space provided. The script should return a date, time or datetimetz type for an absolute datetime deadline. The script can be as long as you like, but the result of the script must be one of those types. For example:

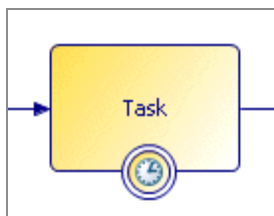
```
var date1 = new Date('December 17, 1995 03:24:00');
```

If only a Date were present, the event would fire immediately when the deadline is created (in the runtime environment) on the Date specified. If only a Time were specified, the event would fire at the specified time on the current date.

Configuring Timer Event Scripts

You can specify that a script is executed for a start timer event or an intermediate timer event (either inflow or on a task boundary).

For example, the following intermediate timer event has been placed on a task boundary:



In the Properties view, you can choose a script type from the **Script Defined As** list:

Select one of the script types:

- **Free Text:** If you want to leave the implementation of the script for the solution designer, you can select Text and use this area provided to describe the desired behavior for the script.
- **Constant Period:** This allows you to specify the timeout period after the event is

initiated using the following time units: Years, Months, Weeks, Days, Hours, Minutes, Seconds, Micro Seconds.

- **JavaScript:** This script type allows you to enter JavaScript statements in the space provided.



Note: The JavaScript script type is not available in the Business Analysis capability.

- **Unspecified:** This means that no description or script is supplied.

Task Scripts on Events

Any event whether timer triggered or not can have a task script assigned to it.

The possible scripts are as follows:

Event Type	Allowed Script Types
Start Event	Complete script
Intermediate Event	Initiate script Complete script Cancel script
End Event	Initiate script

You define task scripts for events on the **Scripts** tab of the Properties view for the event, in the same way that you define the corresponding scripts for tasks.

Timer Event Rescheduling

You can reschedule the timeouts of timer events on user tasks (including the work item deadline). This can be achieved by selecting timers to reschedule from a non-canceling signal event on the same user task.

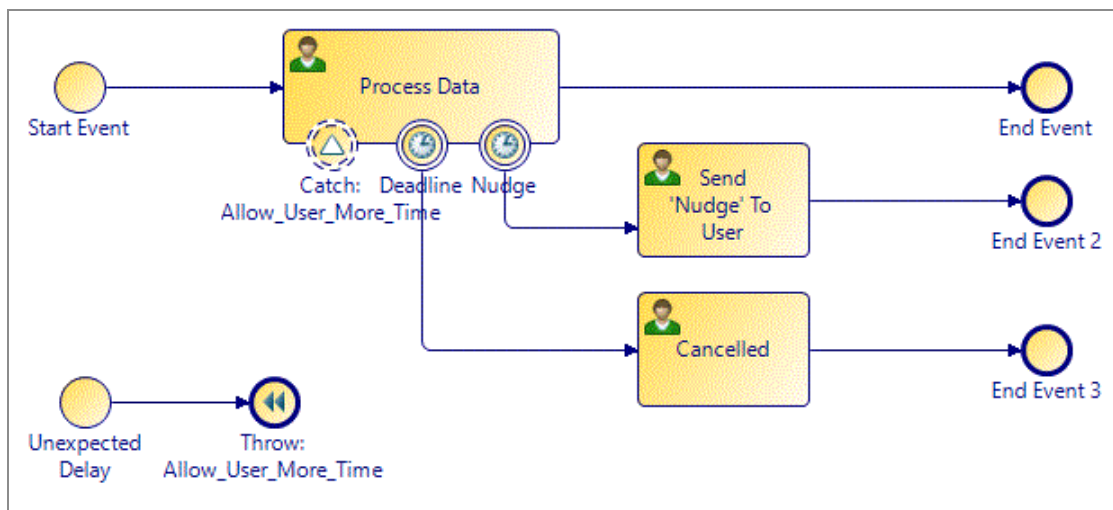
Note: Timer events that have already timed-out are not rescheduled.

A catch signal event attached to a user task, that is configured as **Continue task when signal is caught** allows selection of timer events to reschedule one of the following:

- all
- The Timer currently set as Activity deadline
- one or more explicitly selected timers

When a timer event is selected for reschedule in any one of the above ways then the user **must** define a reschedule timer script on the timer event's General properties tab.

In the example below, the process has a user task with two timers, Deadline and Nudge. When "Unexpected Delay" is triggered, the Allow_User_More_Time signal is thrown and this explicitly reschedules the Deadline timer by adding an extra day to the current Deadline timeout.



The Properties page for Catch: Allow User More Time looks like this:

Label:

Name:

Trigger Type:

Signal Type

☒ Local:

☐ Global:

☐ Case Data:

Catch Action: ☐ Cancel task when signal caught ☒ Continue task when signal caught

Reschedule Task Timer Events

☐ None

☐ All

☐ Deadline

☒ Selected Timer Events...

The Properties page for Deadline looks like this:

Label:

Name:

Trigger Type:

☒ Use as activity deadline

☒ Withdraw Task On Timeout ☐ Continue Task On Timeout

☒ Date and/or Time ☐ Cycle

Description:

[Hide Implementation Details](#)

Initial Timer Script

Script Defined As:

Specify timeout as offset from event initiation:

Years: Hours:

Months: Minutes:

Weeks: Seconds:


Days: Micro Seconds:

Signals

Signals can be local or global. Local signals are thrown and caught within a single process instance. Independent processes might need to collaborate through swapping events and data when you have more than one interrelated process with an interest in the same thing. Global signaling provides an inter process version of the signaling capability, with the capability to pass data as a payload.

Local signals are thrown and caught within a single process instance. They can be used for triggering event handlers, task cancellations and to halt and resume parallel flows.

Global signals are thrown and caught across process instances allowing processes to collaborate with each other.

 **Note:** Global signals are defined separately in Global Signal Definition Projects. See [Creating a Global Signal Definition Project](#).

You can generate (publish) a global signal and it can be caught by subscribing events in more than one process. You can also generate a global signal which is available for a defined period of time. During that time, any processes that subscribe to the signal with the required correlation data, catch the signal. However, the signal does not "go away" if it is caught by a single process during that period of time.

A global signal can be published from anywhere and consumed by multiple processes so you need to identify the process instances that are waiting for a particular instance of a global signal. You define signal correlation data that identifies the process instance waiting for the global signal. Data and parameters from the signal that are used for correlation are defined in the signal specification.

As global signals do not specifically belong to either the publisher or consumer of the signal (signals might be published and consumed from multiple locations), they are defined separately from where they are used or referenced (in the Global Signal Definition project).

Global signals have a data payload. Payload data is defined in the Global Signal Definition project.

Catching of global signals is supported from event handlers and event sub-processes, and is supported from within business processes. Throwing of global signals is supported from business processes, service processes, pageflows and business services.

Correlation parameters are defined in the global signal specification and all subscribers to a particular signal use these parameters to identify the global signal instances of interest. Correlation mappings in the catch global signal specify the process instances that should receive the event (those whose mapped correlation data value matches the signal payload correlation value at runtime). Global signals might correlate immediately or correlate with process instances for a period of time after they have been generated.

- A transitory global signal only exists for the period of time it takes to identify the set of process instances that are waiting/attempting to correlate with this signal instance. As soon as the currently waiting process instances have received the global signal it is destroyed.
- For persistent signals, after the signal has been generated it exists for a defined

period of time and correlate with any global catch signals that try to correlate with it during that period, in addition to those waiting when the signal was generated. When the period expires, the signal is purged.

i Note: Correlation data payload changes are not permitted on Global Signal project upgrade. Therefore, signal correlation parameters cannot be changed unless the major version is changed.

i Note: The number of process instances pending on a particular global signal with the same correlation data values is limited to 100. If the number of attempts to correlate to the global signal using the same correlation data values exceed this number, then an error is generated for each of the further attempts, and the attempt to initialize the signal handler for the defined correlation data in that process instance fails.

Creating a Global Signal Definition Project

You define a Global Signal Definition project to define global signals, which can be referenced from multiple projects.

You can define more than one Global Signal Definition project if desired.

Procedure

1. Select **File > New > Global Signal Definition Project**.
2. Enter the project name. then click **Next**.
3. Accept the default settings (including the selection of **Create initial Global Signal Definition** and click **Finish**.

The Global Signal Definition project is created, with an initial Global Signal. You can then define the payload data to be associated with the global signal. See [Creating Payload Data for Global Signals](#).

On the Properties tab for the **Global Signal**, define the correlation timeout setting. The default setting is **Correlate Immediately**. You can change this to **Timeout Signal After: *n* seconds**.

i Note: It is possible to deploy a process with global signal events without deploying the Global Signal Definition project. However, if you do this, then the process halts with an error at runtime. You should deploy the Global Signal Definition Project before you deploy processes which contain global signal events.

Creating Payload Data for Global Signals

You can define either *normal* or *correlation* payload data for global signals.

Normal payload data is simply process data that is passed from the throw signal event to the catch signal event.

Correlation payload data is used to match waiting catch signal event process instances to specified process instances. For example, a catch global signal might have a correlation payload parameter of 'Invoice Number'. When the catch is initiated, the current value of 'Invoice Number' for that process instance becomes the 'key' on which the signals are matched. The throw global signal maps a value to the 'Invoice Number' correlation payload parameter, and the system finds the corresponding process instance for the catch global signal that was initiated with that same value.

After defining the payload data in your global signal definition project using the procedure below, the global signal throw and catch events in your process map to this payload data. On the throw signal event, use **Map To Signal** to map process data to the defined payload data. On the catch signal event, use **Map From Signal** to map from the defined payload data to process data. For more information, see [Throw and Catch Signal Events](#).

i Note: Correlation data payload changes are not permitted on global signal definition project upgrade. Therefore, signal correlation parameters cannot be changed unless the major version is changed. You must take care to ensure that only valid changes are made. If you upgrade a global signal definition project, any changes you have made to a BOM class referenced by a payload data parameter are not validated against.

Procedure

1. On the **Global Signals** panel, select the global signal you want to add payload data

for.

2. On the **Payload Data** panel, select **PayloadData**.
3. On the **Properties** pane **General** tab, use the **Use for Signal Correlation** selection to specify whether you are defining normal or correlation payload data, as follows:
 - Select **Use for Signal Correlation** to define correlation payload data.
 - deselect **Use for Signal Correlation** to define normal payload data.
4. If you are defining normal payload data, specify whether the payload data is mandatory or optional by selecting or deselecting the **Mandatory** check box. (The **Mandatory** check box is not visible if you are defining correlation payload data, as it is always mandatory.)
5. In the **Type** section, specify the appropriate type information for the payload data, depending on the process data it is mapped to.

Sending signals between processes

Creating a Global Signal Definition Project

You define a Global Signal Definition project to define global signals, which can be referenced from multiple projects.

You can define more than one Global Signal Definition project if desired.

Procedure

1. Select **File > New > Global Signal Definition Project**.
2. Enter the project name. then click **Next**.
3. Accept the default settings (including the selection of **Create initial Global Signal Definition** and click **Finish**.

The Global Signal Definition project is created, with an initial Global Signal. You can then define the payload data to be associated with the global signal. See [Creating Payload Data for Global Signals](#).

On the Properties tab for the **Global Signal**, define the correlation timeout setting. The default setting is **Correlate Immediately**. You can change this to **Timeout Signal After: *n* seconds**.

i Note: It is possible to deploy a process with global signal events without deploying the Global Signal Definition project. However, if you do this, then the process halts with an error at runtime. You should deploy the Global Signal Definition Project before you deploy processes which contain global signal events.

Creating Payload Data for Global Signals

You can define either *normal* or *correlation* payload data for global signals.

Normal payload data is simply process data that is passed from the throw signal event to the catch signal event.

Correlation payload data is used to match waiting catch signal event process instances to specified process instances. For example, a catch global signal might have a correlation payload parameter of 'Invoice Number'. When the catch is initiated, the current value of 'Invoice Number' for that process instance becomes the 'key' on which the signals are matched. The throw global signal maps a value to the 'Invoice Number' correlation payload parameter, and the system finds the corresponding process instance for the catch global signal that was initiated with that same value.

After defining the payload data in your global signal definition project using the procedure below, the global signal throw and catch events in your process map to this payload data. On the throw signal event, use **Map To Signal** to map process data to the defined payload data. On the catch signal event, use **Map From Signal** to map from the defined payload data to process data. For more information, see [Throw and Catch Signal Events](#).

i Note: Correlation data payload changes are not permitted on global signal definition project upgrade. Therefore, signal correlation parameters cannot be changed unless the major version is changed. You must take care to ensure that only valid changes are made. If you upgrade a global signal definition project, any changes you have made to a BOM class referenced by a payload data parameter are not validated against.

Procedure

1. On the **Global Signals** panel, select the global signal you want to add payload data for.
2. On the **Payload Data** panel, select **PayloadData**.
3. On the **Properties** pane **General** tab, use the **Use for Signal Correlation** selection to specify whether you are defining normal or correlation payload data, as follows:
 - Select **Use for Signal Correlation** to define correlation payload data.
 - deselect **Use for Signal Correlation** to define normal payload data.
4. If you are defining normal payload data, specify whether the payload data is mandatory or optional by selecting or deselecting the **Mandatory** check box. (The **Mandatory** check box is not visible if you are defining correlation payload data, as it is always mandatory.)

5. In the **Type** section, specify the appropriate type information for the payload data, depending on the process data it is mapped to.

Correlation Data

Correlation data must be used to ensure that each global signal event is received by the process instances to which it applies. Global signals are designed to be caught by multiple process instances. If there are multiple process instances with matching correlation data, then the signal is triggered in each one.

In a process, a catch global signal event waits for an incoming signal to arrive. In the runtime environment, there might be many instances of the process (each with different data), and many incoming signals (each with different data).

To determine whether a global signal applies to a process instance, the signal's correlation data is compared to the correlation data it is mapped to in each process instance that catches the global signal. Only the catch global signal events in processes that have matching correlation data are triggered. You must initialize the correlation data with a value, before the catch global signal event is initialized, you must configure the catch signal event to initialize after the event or task that initializes the correlation data.

To work with correlation data:

- Decide whether elements within the incoming data can be used to uniquely identify instances of this process. If the incoming data can be used to identify the process, create a correlation data field in the process and the global signal payload definition. In the catch global signal event map these correlation fields together.
- In a throw global signal map, the data that identifies the required process instances to the correlation field in the signal payload.

Using Scripts

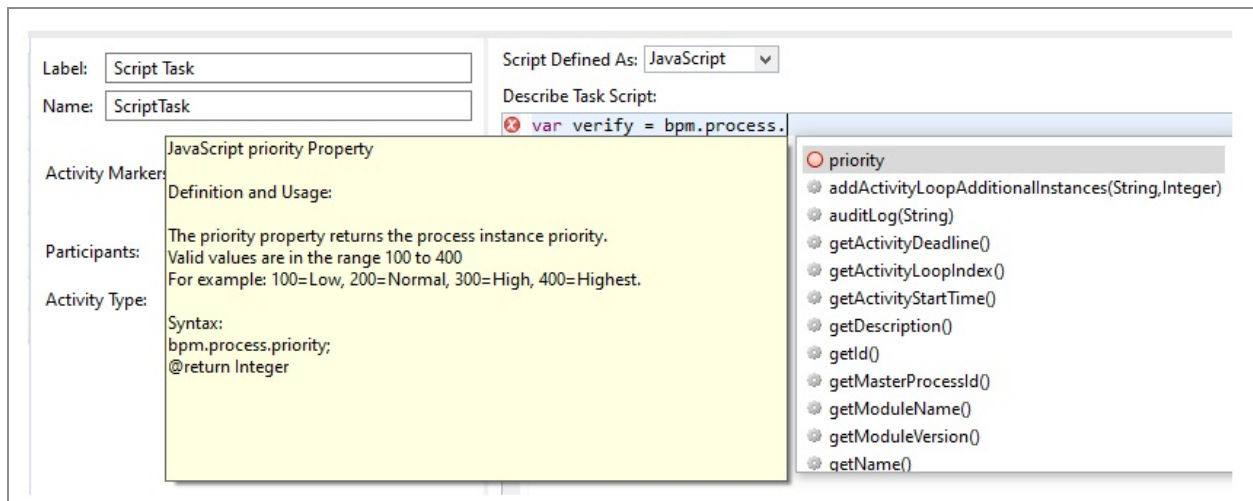
Using Scripts

Except in the very simplest of processes, the sequence of forms that a user is presented with, and the outcome of the work item, depends on the conditional processing written into the application. That conditionality is governed by scripts. For example, a script could take information from fields completed in one work item and pass it to another, depending on the value of those fields.

In addition, scripts can include operations to create, update or delete data objects. Examples would include creating or updating a customer record, or a new insurance claim.

- **Plain text.** When you create a process, you might include a plain text description of the desired behavior of any scripts that form part of the process. These plain text entries are not intended to be executable. The plain text script appears as an error in the process and the process cannot be deployed whilst there is a script defined in this way within it.
- **JavaScript.** Executable scripts are written in JavaScript. Some additional methods compatible with standard JavaScript are provided with ActiveMatrix BPM and can be used in the scripts that you write within TIBCO Business Studio. See [Additional JavaScript Global Functions](#).
- **Data Mapper.** With Data Mapper, you can graphically map data across data fields and parameters to create complex BOM objects from a combination of process data fields and parameters. This script grammar is available for Script Tasks, Task scripts, Call Sub-Processes, global and local signals, and Catch error events. For more information, see [Data Mapping](#).

Use Ctrl+ SPACE to access content assist in all the areas in which you can enter scripts. You are prompted with a list of words, variables, methods and so on, that are appropriate for where you are in the script.



Implementing Scripts

You can specify scripts for use with ActiveMatrix BPM processes in several ways.

Script Type	Description	Location
Script task	Scripts performed individually as part of a process flow.	In the Properties view for the task, select the General tab. Enter the script itself in the Describe Task Script: area.
Conditional Flow	Scripts to define the conditions that determine whether a conditional sequence flow is followed. At runtime, this causes the Sequence Flow to be followed only if the condition is met.	In the Properties view for the conditional flow, select the General tab. Enter the script in the Describe Sequence Flow Condition area.
Loop	Scripts associated with a processing loop in the Properties view for the task to which a loop has been applied.	On the General tab of the Properties view for the task, select either Standard Loop or Multiinstance Loop . From the Loop tab, select JavaScript from the Script Defined As: field and enter the script itself in the area provided.

Script Type	Description	Location
Events	Scripts added to start events or catch intermediate events in the Properties view. You can also add scripts to events that have a timer defined.	In the Properties view for the start event or catch intermediate event, select the General tab. Enter the script itself in the Scripts tab.
Process Manager Scripts	<p>Scripts that are run in the context of the background execution of the process (as opposed to the processing of individual user task work items).</p> <ul style="list-style-type: none"> • Initiate - executes when the task/event activity is instantiated. • Complete - executes when the task/event activity is completed. • Timeout - executes when the task/event activity times out. • Cancel - executes when the task/event activity is canceled. 	In the Properties view for the task, select the Scripts tab. Select from Process Manager Scripts to display the different sets of scripts available.
Work Manager scripts	<p>Scripts that are run in the context of an individual work item. These scripts can also refer to entities in the organization model.</p> <ul style="list-style-type: none"> • Schedule - When the work item is scheduled from a user task, after all standard validations have run but before the task completion message is returned, the Schedule script executes. • Reschedule - Executes when a work item is updated via a non-canceling signal event on task boundary. • Open - When a user opens a work item, the Open script executes. If 	In the Properties view for the task, select the Scripts tab. Select from Work Manager Scripts to display the different sets of scripts available.

Script Type	Description	Location
	<p>necessary this script can force a cancellation of the work item open and keep the work item in the work list.</p> <ul style="list-style-type: none"> • Close - When a user closes a work item in a work list (i.e. saves the state of the work item rather than completing it), the Close script executes. • Submit - When the work item is submitted, after all standard validations have run but before the task completion message is returned, the Submit script executes. If necessary this script can force a cancellation of the work item submit and keep the work item open and with the end user. 	

Using Process Data in Scripts

Any process data added to a script can be used by the script both in the design-time interface and at runtime once the process containing the script has been deployed.

Process data is wrapped in the data object. For example:

```
var data.engineSize = data.car.engineCapacitycc;
var data.carPrice = data.car.listPrice + data.delivery + data.tax;
```

By default, all of a process's data fields are available in a script. If specific data is associated with a task that has an interface, then it might restrict the fields that are available, depending on the activity and script type.

Using Structured Data Types

A range of factory methods are supplied for creating instances of structured data types within scripts.

You use these methods for creating objects and performing operations on objects of specific types in scripts. See [Business Data Scripting](#).

When creating BOM objects, you must use the Create factory methods described in [Business Data Scripting](#). For standard JavaScript objects like Date, you can use `new Date()`.

Factory Methods


In addition, TIBCO Business Studio - BPM Edition scripting supports the use of factory methods to create new Business Object Model objects.

You can call:

```
field=factory.BOM_name.createClass();
```

where:

- *field* is the name of the data field to be populated, and must correspond to a process field defined as an External Reference to *class* in the Business Object Model,
- *BOM_name* is the Business Object Model name, with any dots replaced by underscores (so **com.example.BOM** would become **com_example_BOM**),

 **Note:** Note that this is the Name of the model, not the Label.

- *Class* is the name of the Business Object Model Class object,

For example, assume that there is a Business Object Model called **com.myorg.customermodel**, and that it contains a Class called **Customer**. To use a script to create an instance of this Class in the field **cust**, you would call the corresponding factory method as follows:

```
data.customer = factory.com_myorg_customermodel.createCustomer();
```

As noted above this only needs to be done if the **cust** field has not already been initialized. You could check to see whether it has been created before invoking this line of the script, for example:

```
if (data.customer == null)
{
    data.customer = factory.com_myorg_customermodel.createCustomer();
}
```

The Business Object Model object needs to have been created before any attributes defined in the Business Object Model can be assigned.

WorkManager and Process Classes

WorkManager and process classes provide additional methods that you can use when working with work items, processes and organization models in your scripts. For example, you could set process priority, set work item attributes and so on.

workManager and process classes are wrapped in a bpm object.

Example 1

```
var theOfferSet = bpm.workManager.getWorkItem().getWorkItemOffers();
```

Example 2

```
var myActivityLoopIndex = bpm.process.getActivityLoopIndex();
```

For a list of the attributes and methods provided, see [Process Manager and Work Manager Scripting](#).

Enumerations

If you want to categorize business objects as different types with a predefined set of allowed values, instead of using a number or a free format string, you can use an Enumerated Type (ENUM).

You must use a fully-qualified name (qualified by the package name) for the enumerations in the script. The qualified name of enumerations to be used in scripts is similar to the

factory names, with the qualified name formatted to replace dot '.' by '_' an underscore character. Secondly, the fully-qualified name must be wrapped in a `pkg` object. For example:

```
pkg.com_example_data_Colour.GREEN
```

For more information, see [Working with Enumerated Types \(Enums\)](#).

JavaScript Exclusions

Certain facilities of standard JavaScript are not supported.

These are:

- You cannot define functions in scripts: that is, the JavaScript **function()** method is not supported. An error saying **Local method definition is not allowed** is generated.
- It is not regarded as good practice for scripts to be too large or to provide behavior which would be better and more clearly provided by the diagrammed business process. Scripts should provide only the necessary connections between the process, the services it uses, and work items.
- The **switch(){case: default:}** statement is not supported.
- JavaScript regular expressions are not supported.
- The **valueOf()** method is not supported. You can achieve the same results by using **toString()** instead.
- The Try/Catch statement is not supported.
- The **===** operator is not supported
- **"in"** is a reserved keyword in JavaScript and is not supported.



Note: Note that braces, **{..}**, are required in an **if** statement and in **for** and **while** loops by the script editor, although they are only optional in JavaScript.

Although braces are not compulsory in JavaScript **if**, **for** and **while** statements their use is in any case good practice.

Data Mapping

Data Mapper is a script grammar available for script tasks, task scripts, sub-processes, global signals and local signals and so on.

With Data Mapper, you can graphically map data across data fields and parameters to create complex BOM objects from a combination of process data fields and parameters.



Note: Data mapper employs a create-or-merge strategy at runtime. If the parent-tree for the target element does not exist, it is created prior to assigning the mapped element.

The process data can include data fields, parameters, user-defined scripts, process information, work item information and attributes from the JavaScript classes, `Process` and `WorkManager` (user task only).

For scripts where Data Mapper is valid, it is as available on Script Tasks as an option in the **Script Defined As:** list on the **General** tab of the Properties view. The left-hand side of the data mapper shows the source list, and the right-hand side shows the target list. The mappings defined using the process data mapper are applied to the target data fields and parameters at runtime.

Array Mapping Strategies

Specific array handling strategies can be selected for mappings to multi-instance target data.

Overwrite, append, and merge are the three mapping strategies that are applicable for array types. By default, the target list is overwritten with the source list.

Each mapping strategy affects the target list differently:

Overwrite

When mapping directly between arrays, clears the target array and copies the elements from the source array.

When mapping between children of arrays, clears the target array and creates a new target element for each element in the source array. The assignments implied by the child mappings are then applied to parent elements in the same location in the source and target array.

Append

When mapping directly between arrays, copies elements in the source array to the end of the target array.

When mapping between children of arrays, creates a new element for each element in the source array and appends to the target array. The assignments implied by the child mappings are then applied to new target elements from the source elements.

Merge

When mapping directly between arrays, overwrites each element in the target array for which there is an element at the same location in the source array by a copy of the source element. If there are further elements in the source array then they are appended to the target array.

When mapping between children of arrays, each element in the target array for which there is an element at the same location in the source array has the assignments implied by the child mappings applied from that source element.

If there are further elements in the source array then new target elements are created and appended to the target array, and the assignments are applied to the children of these new elements.

Nested Array Handling

In the case where it is desired to map to an array nested within another array, either directly or mapping into the child content, then you must do it from equivalently nested source arrays.

For example, if your target content is an `Orders` array, with a child array of `OrderLines` then in order to map into `Orders[]->OrderLines[]->LineId` then the mapping must be from an element within a second level nesting source array (`MyConfirmations[]->ConfirmationLine[]->LineConfirmId`).

|

To configure an array mapping strategy, drag an element from the source list or LHS and drop it on a corresponding element in the target list or RHS. Define one of the following mapping strategies by right-clicking the target tree array items:

- **Overwrite Target List**
- **Append to Target List**
- **Merge List Element Content**


Like Mapping

Like mapping is useful in automatically mapping complex elements that are of different types but contain equivalent content.

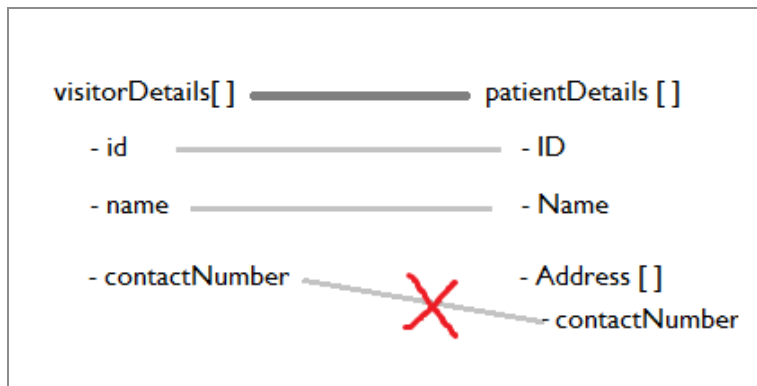
For instance, an application might have two different types `VisitorDetails` and `PatientDetails` that share some commonly named children. Performing a like mapping between these two objects is the equivalent of manually mapping all the same named simple type content from the source list to the target list.

It is possible to perform like mapping between arrays of complex types. In this case the equivalently-named child content mappings implied by the like mapping are applied to each element in the array according to the chosen array mapping strategy. If there are nested child arrays that would be like-mapped then the array mapping strategy is selected separately for each.

A like mapping scans the target element tree looking for the same named content in the source tree. Initially, all equivalently-named child objects get mapped regardless of their type. If the objects are of different types, the mapping shows an error decoration. You can fix the problem in the BOM definition by excluding certain target child elements from the mapping. The overall strategy for handling mappings into the children of complex type target data is to create the target element if it does not exist prior to performing assignment to the child element. If the target parent element already exists then the assignment is made to the child of the existing element. For mappings from children of source items, the assignment is only performed if the parent itself has a value. If the parent element is not assigned, then the target element remains unchanged.

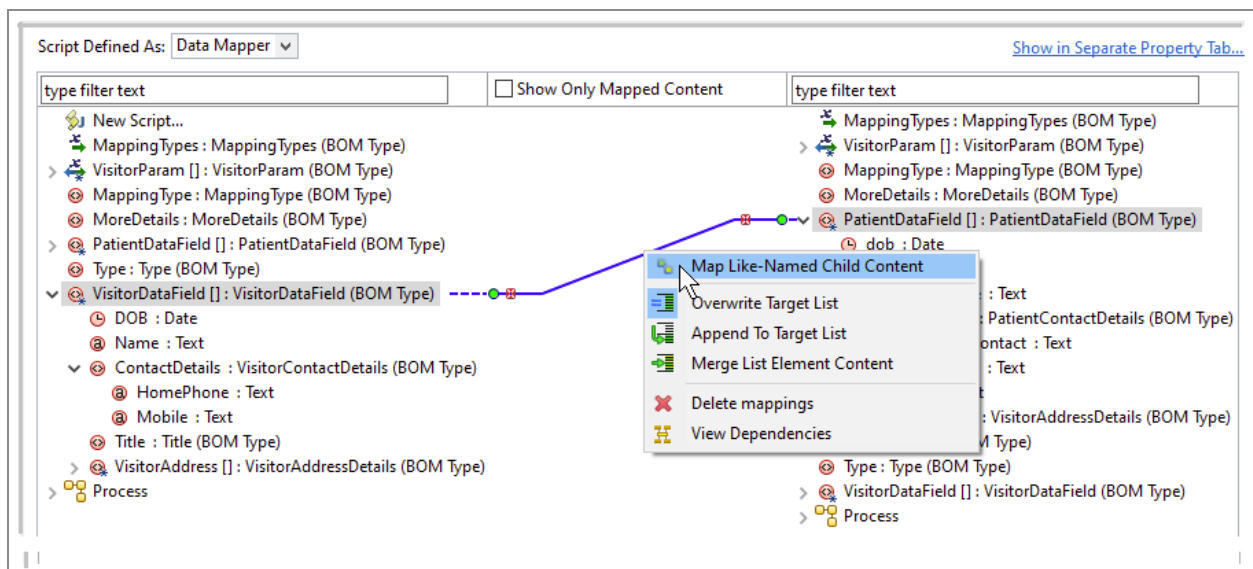
 **Note:** Like mapping is case-insensitive for the names of child objects.

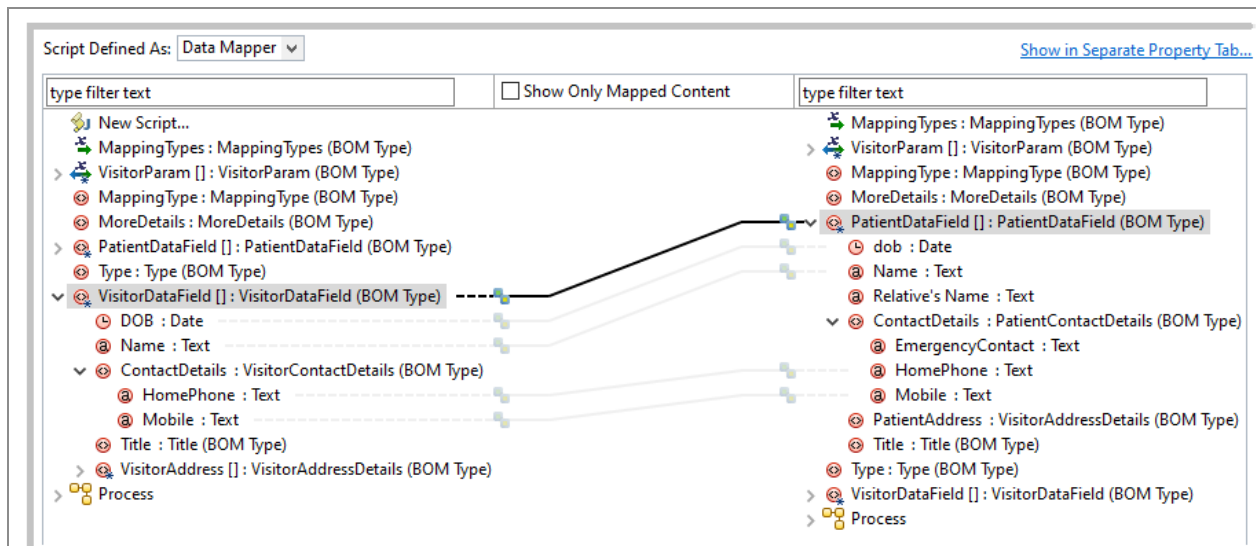
Like mappings are fully recursive, mapping similarly named composite objects that are within a particular target being mapped. However, only the child objects that are at the same level are included in the like mapping.



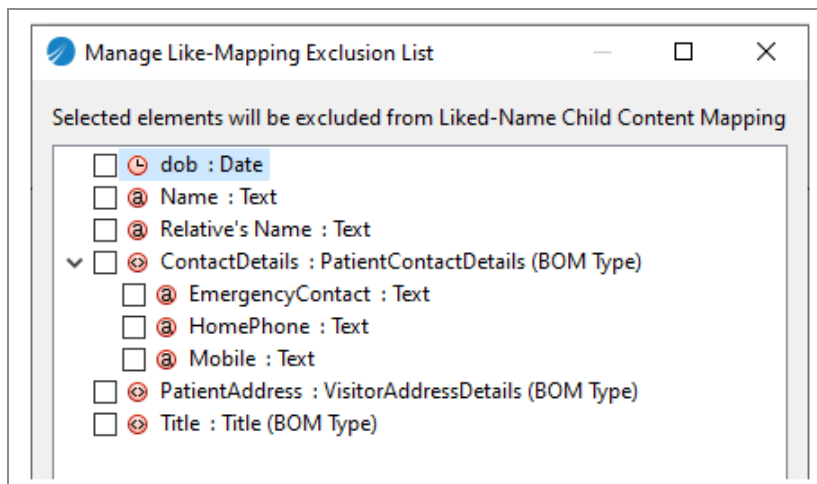
If there are nested child arrays that would be like-mapped then the array mapping strategy is selected separately for each array.

To configure a like mapping, drag an element from the source list or LHS and drop it on a corresponding element in the target list or RHS and right-click the mapping and select **Map Like-Named Child Content** from the context menu. The child elements with the same name and object type are automatically mapped.





If you do not want a few child elements mapped with like-named child elements, right-click the mapping at the parent level, and select **Open Like-Mapping Exclusion List**. Manage Like-Mapping Exclusion List dialog box opens, which lists all the like-mapped elements.



Sample Scripts

This section contains samples of work manager scripting and process manager scripting.

Example of Work Manager Scripting:

The JavaScript below is defined on the 'Schedule' Script, meaning that when the work item is scheduled, this script executes. The script is using the `WorkManagerFactory.getWorkitem`

`()`.getWorkItemOffers() method to return the list of resources to which the work item is offered.

WorkManagerFactory methods are defined in [Process Manager and Work Manager Scripting](#).

```
var theOfferSet = bpm.workManager.getWorkItem().getWorkItemOffers();
var i = 0;
data.DFOfferSet.length(0);
while (i<theOfferSet.length()){
//loop through resources in offer set
    var resourcesFromOfferSet = theOfferSet.get(i).getName();
    data.DFOfferSet.push(resourcesFromOfferSet);
//add resource to OfferSet array field
    i++;
}
```

Where 'DFOfferSet' is a data field defined in the process and added as an interface to the user task.

The `WorkManagerFactory.getOrganizationModel` method can be used to navigate around the organization model to identify a particular organizational entity, or are source mapped to such an entity. For example:

```
var useResource =
bpm.workManager.getOrganizationModel().getAssociatedResources
(data.EasyAs.ClaimsDept.Manager)
```

Resources can be located using RQL queries, using the `bpm.workManager.getOrganizationModel().getResourcesByQuery` method.

Examples of Process Manager Scripting:

The JavaScript below is defined on the 'Initiate' Script meaning that when this user activity is initiated, this script is executed. The script is using the `Process.getActivityLoopIndex()` method to return the loop index for a multiple instance user task.

```
var myActivityLoopIndex = bpm.process.getActivityLoopIndex();
if (myActivityLoopIndex != null)
{
    data.UTActivityLoopIndex = myActivityLoopIndex;
}
```

Where 'UTActivityLoopIndex' is a data field defined in the process and added as an interface to the user task.

The following example shows extracts from a Process Manager script being used to get information about the organization model:

```
var orgModel = bpm.process.getOrgModel();
var unit1 = orgModel.ouByGuid("_xI4ikMpPEd64gM7QE8RwxA");
var position1 = orgModel.positionByName("clerk");
```

Supplemental Information

This topic includes reference material to support your use of Business Data Services.

Data Type Mappings

This section contains tables showing data type mappings.

Process Primitive Data Type Mapping

TIBCO Business Studio supports the following process primitive data types:

Process Primitive Type	Java Type Representation	Comments
Boolean	Boolean	
Fixed Point Number	Number	Constrained to <=15 digits. TIBCO Business Studio validates that the upper limit is not exceeded.
Floating Point Number	Number	A 64-bit floating point number.
Text	String	

Process Primitive Type	Java Type Representation	Comments
Date	XMLGregorianCalendar	Date, without timezone offset.
Date Time Timezone	XMLGregorianCalendar	Date and time, with optional timezone offset.

JavaScript Features not Supported in TIBCO BPM Scripting

Certain JavaScript features are not supported.

New Operator

The new operator from JavaScript is not supported. Instead, the factory methods must be used to create new instances of classes.

Switch Statement

The switch statement from JavaScript is not supported:

```
switch (value) {
    case constant:
        BLOCK;
        break;
    ...
    default:
        BLOCK;
        break
}
```

Instead, an if () {} else if () {} ... statement should be used.

try/catch Statement

The try/catch statement is not supported:

```
try {  
    BLOCK;  
}  
catch (error) {  
    BLOCK;  
}  
Finally {  
    BLOCK;  
}
```

The only way to catch an exception in a script is to catch the error on ScriptTask or another task.

JavaScript Functions

JavaScript functions are not supported. Code must be written out in full.

JavaScript RegExp Object

The JavaScript RegExp object is not supported. For example, the Text field `replace()` method only supports string substitution, not the substitution of a pattern matched by a regular expression.

"==="operator

The `===` operator (same value and type) is not supported.

JavaScript Arrays

Using `[]` for array indices is not supported, as there is no way to create JavaScript arrays in TIBCO Cloud BPM. Instead, the List type is used. Consequently, the following loop is not supported:

```
for (FIELD in ARRAYFIELD) {  
    BLOCK;  
}
```

Using If For and While Expressions

The curly braces are required in TIBCO Business Studio scripts for if, while, and for expressions.

Reserved Words in Scripts

Array	Boolean	Date	Math	Number	Object
RegExp	String	abstract	assert	bdsId(1)	bdsVersion1
boolean	break	byte	case	catch	char
class	const	continue	debugger	default	delete
do	double	else	enum	export	extends
false	final	finally	float	for	function
goto	if	implements	import	in	instanceof
int	interface	long	native	new	null
package	private	protected	public	return	short
static	strictfp	super	switch	synchronized	this
throw	throws	transient	true	try	typeof
upper	var	void	volatile	while	with

(1) Cannot be used, regardless of case. Other reserved words cannot be used in the case shown, but can be used by changing the case (for example, while is prohibited, but WHILE is acceptable).

bpm.scriptUtil

The bpm.scriptUtil provides methods to copy business objects.

The bpm.scriptUtil performs a "deep" copy, which means that it copies all the objects contained by the Business Object being copied, as well as the Business Object itself.

The bpm.scriptUtil also supports simple type and simple type arrays.

Methods

Method	Comments
copy(Object)	Copies a simple value or a business object. For example: account.address = bpm.scriptUtil.copy (customer.address);
copy(Simple[])	
copy(SimpleArray[])	
copyAll(Object[])	Copies all the simple values or business objects in an array.
copyAll(Simple[])	
copyAll(SimpleArray[])	

Process Manager and Work Manager Scripting

Attributes and methods are provided to help with process scripting, work item and organization model scripting.

Attributes and methods are provided for the following:

- Process instances
- Work items
- Work Manager
- Organization models

Use Ctrl + Space to access content assist to find all the words, variables, attributes and methods, that are appropriate for where you are in the script.

Process Instance Attributes and Methods

This section provides additional information (that is not available in content assist) about some of the attributes and methods that are available for accessing information about process instances using the `bpm.process` class.

Attribute/Method	Example	Description
Datetimetz <code>getActivityStartTime()</code>	<code>startTime = bpm.process.getActivityStartTime()</code>	Returns the time the current task was started.
Datetimetz <code>getActivityDeadline()</code>	<code>deadlineTime = bpm.process.getActivityDeadline()</code>	Returns the current task deadline time.
<code>getActivityLoopIndex()</code> : Number	<code>index = bpm.process.getActivityLoopIndex()</code>	<ul style="list-style-type: none"> Retrieves the most local loop index. Index starts from 0 (zero) for first loop/multi-instance instance. In a parallel loop/multi-instance task or embedded sub-process, all loop instances run concurrently. To pass the loop index of a particular loop instance to a task: <ul style="list-style-type: none"> Use the input mapping functionality if this is available. If it is not, use a local data field.

Attribute/Method	Example	Description
		<ul style="list-style-type: none"> Use a process-level data field only as a last resort. Each loop instance tries to set the field with the index of that particular loop instance; but the field is common to all loop instances in the process and can only contain a single value. A task can successfully get and set the index in the same transaction, but if there is any delay in getting the index the field might have been overwritten by another instance.

Attribute/Method	Example	Description
		<ul style="list-style-type: none"> For a non-multi-instance task, this is the loop index for the nearest multi-instance embedded sub-process ancestor (defaulting to 0 if no multi-instance ancestor found). In nested multi-instance situations, the user can transfer an embedded sub-process loop index into an embedded sub-process local data field.
getActivityAttribute (String, String) : String	<pre>workItemId = bpm.process.getActivityAttribute ('UserTask2','WorkItemId'); completer = bpm.process.getActivityAttribute ('UserTask2','Completer');</pre>	<p>Returns value of attribute, for example, 430</p> <p>Only WorkItemId and Completer are supported as attribute names.</p>
Priority : Number	priority = bpm.process.priority;	<p>Once a process instance has been created, it can change its own priority.</p> <p>The default value is 200. Valid entries are 100, 200, 300 and 400.</p>
auditLog (String message)	bpm.process.auditLog ("Unknown Cargo Type " + cargoType.name) ;	Adds a simple text entry to the audit log (process-

Attribute/Method	Example	Description
		instance related event log entry).
		Use of the bpm.process.auditLog() method is not supported for in-memory process instances (service processes, pageflow processes or business services). If the bpm.process.auditLog() method is used on one of these processes (whether started from within a parent business process or independently), no audit entries are generated.
Long getWorkItemId()	workitemID = bpm.process.getWorkItemId()	Returns the work item for the current user task. Returns null if not called from a user task Complete script.
String getWorkItemCompleter()	WCompleter = bpm.process.getWorkItemCompleter()	Returns the GUID of the user that completed the work item. Returns null if not called from a user task Complete script.
List<String> getWorkItemParticipants()	participants = bpm.process.getWorkItemParticipants()	Returns an array of GUIDs for the participants for the work item. Returns null if not called from a user task Complete, Cancel, or Timeout script.

Organization Model Attributes and Methods

Process Manager scripting supports all the organization model methods that are supported in Work Manager scripts.

[WorkManager](#) describes the additional functions that are provided to help with work item and organization model scripting.

Attribute / Method	Comments
OrgModel	Accessed via bpm.process.getOrgModel() in process manager scripts / script tasks or bpm.workManager.getOrgModel() in user task Work Manager scripts
ouByGuid(guid:Text) : EntityDetail	Returns the single EntityDetail that describes the Organizational Unit identified by its GUID. If no such Organizational Unit exists, the return value is null.
ouByName(name:Text) : List<EntityDetail>	Returns the list of EntityDetails that describe the Organizational Units identified by the given name. If no such named Organizational Units exist, the return value is an empty list.
groupByGuid(guid:Text) : EntityDetail	Returns the single EntityDetail that describes the Group identified by its GUID. If no such Group exists, the return value is null.
groupByName(name:Text) : List<EntityDetail>	Returns the list of EntityDetails that describe the Groups identified by the given name. If no such named Groups exist, the return value is an empty list.
resourceByGuid(guid:Text) : EntityDetail	Returns the single EntityDetail that describes the Human Resource identified by its GUID. If no such Human Resource exists, the return value is null.
resourceByName(name:Text) : List<EntityDetail>	Returns the list of EntityDetails that describe the Human Resources identified by the given name. If no such named Human Resources exist, the return value is an empty list.
resourceByLdapDN(ldapDN:Text): List<EntityDetail>	Returns the collection of Resources identified by the given LDAP DN, or an empty list if none can be found. Ideally, there should be only one such Resource for a given DN.

Attribute / Method	Comments
positionByGuid (guid:Text) : EntityDetail	Returns the single EntityDetail that describes the Position identified by its GUID. If no such Position exists, the return value is null.
positionByName (name:Text) : List<EntityDetail>	Returns the list of EntityDetails that describe the Positions identified by the given name. If no such named Positions exist, the return value is an empty list.
orgByGuid(guid:Text) : EntityDetail	Returns the single EntityDetail that describes the Organization identified by its GUID. If no such Organization exists, the return value is null.
orgByName(name:Text) : List<EntityDetail>	Returns the list of EntityDetails that describe the Organizations identified by the given name. If no such named Organizations exist, the return value is an empty list.
EntityDetail	
getEntityType() : Text	<p>Returns the type identifier for this organizational model entity. Example values are:</p> <ul style="list-style-type: none"> • ORGANIZATION • ORGANIZATIONAL_UNIT • GROUP • POSITION • RESOURCE
getGroups() : List<EntityDetail>	For Human Resource entities, this returns the EntityDetails that describe the Groups to which the Resource is associated.
getPositions() : List<EntityDetail>	For Human Resource entities, this returns the EntityDetails that describe the Positions to which the Resource is associated.
getName() : Text	The name of the organizational model entity.
getGuid() : Text	The GUID that uniquely identifies the organizational model entity.

Attribute / Method	Comments
getAlias() : Text	For Human Resource entities, this is the Alias of the LDAP Source from which the Resource is derived.
getDn() : Text	For Human Resource entities this is the Distinguishing Name (DN) of the LDAP entry from which the Resource is derived.
getResourceType() : Text	<p>For entities of Entity Type RESOURCE, this identifies the type of Resource:</p> <ul style="list-style-type: none"> • "DURABLE" • "CONSUMABLE" • "HUMAN" <p>Currently, only HUMAN Resources are supported.</p>
getResources() : List<EntityDetail>	For non-Resource entity types (such as Positions and Groups), this returns the Resource entities associated with that entity. For example, for a Position, it is all the Resources that hold that Position.
getAttributeValue(attrName:Text) :List<Text>	For Resource entity types, this returns the value of the named Resource Attribute held by that Resource entity.
getAttributeType(attrName:Text) : Text	For Resource entity types, this returns the data type of the named Resource Attribute.
getMajorVersion() : Integer	The major version number of the organization model.
getAuthenticatedUser()	This returns the username of the authenticated user in the current login session for the pageflow instance.

WorkManager

This table lists the WorkManager attributes and methods provided.

Attribute / Method	Comments
getWorkItem() : WorkItem	Returns WorkItem object. See WorkItem for properties & methods.
getOrgModel() : OrgModel	Returns an OrgModel object. See Organization Model Attributes and Methods for methods.
getOrgModelByVersion(version: Integer) : OrgModel	Returns an OrgModel object. The parameter specifies which major version of the model is required.

WorkItem

This table lists the WorkItem attributes and methods provided.

Attribute / Method	Comments
WorkItem	
cancel : Boolean	Cancels an API that exists in BRM.
description : Text	The description of the work item.
priority : Number	The specified priority of the work item.
getId() : Number	Returns the work item's unique ID.
getVersion() : Number	Returns the version number of the work item.
getWorkItemResource() : EntityDetail	Returns a resource that has this work item. It contains all the organizational entities whose work list currently contains this work item.
getWorkItemOffers() : List<EntityDetail>	Returns a work item resource object for the work item. If the item was originally offered to more than one organizational entity and is now allocated, this contains all the entities to which the item was originally offered.

Attribute / Method	Comments
getContext() : ItemContext	<p>Returns the work item's context information and provides read only methods to access the following information:</p> <ul style="list-style-type: none"> • activity ID • activity name • application name • application instance • application ID • application instance description
getSchedule() : ItemSchedule	<p>Returns the work item's schedule information and provides read only methods to access the start date and target date.</p>
workItemAttributes.attribute1: Number	<p>These work item attributes (attribute1 and the others listed below) can be used to contain data associated with a work item and to sort and filter your work list. They are available where the WorkManager object can be accessed (for example, on a user task schedule script). For example, attribute2 can be used to hold a customer name, and attribute1 a customer reference number to aid work list sort and filter choices.</p> <p>The attribute1 work item attribute can be assigned integer values in the range -2,147,483,648 to 2,147,483,647.</p>
workItemAttributes.attribute2: Text	<p>Limited to 64 characters in length.</p> <p>See description above.</p>
workItemAttributes.attribute3: Text	
workItemAttributes.attribute4: Text	

Attribute / Method	Comments
workItemAttributes.attribute5: Number	See description above. This can be assigned Decimal or BigDecimal values.
workItemAttributes.attribute6: Date	See description above.
workItemAttributes.attribute7: Date	See description above.
workItemAttributes.attribute8: Text	Limited to a maximum of 20 characters - anything larger is truncated.
workItemAttributes.attribute9: Text	See description above.
workItemAttributes.attribute10: Text	
workItemAttributes.attribute11: Text	
workItemAttributes.attribute12: Text	
workItemAttributes.attribute13: Text	
workItemAttributes.attribute14: Text	
workItemAttributes.attribute15: Number	The attribute15 work item attribute can be assigned integer values in the range -2,147,483,648 to 2,147,483,647.
workItemAttributes.attribute16: Number	See description above. This can be assigned Decimal or BigDecimal values.
workItemAttributes.attribute17: Number	See description above.

Attribute / Method	Comments
	This can be assigned Decimal or BigDecimal values.
workItemAttributes.attribute18: Number	See description above. This can be assigned Decimal or BigDecimal values.
workItemAttributes.attribute19: Date	See description above.
workItemAttributes.attribute20: Date	See description above.
workItemAttributes.attribute21: Text	Limited to a maximum of 20 characters - anything larger is truncated.
workItemAttributes.attribute22: Text	See description above.
workItemAttributes.attribute23: Text	
workItemAttributes.attribute24: Text	
workItemAttributes.attribute25: Text	
workItemAttributes.attribute26: Text	
workItemAttributes.attribute27: Text	Limited to a maximum of 64 characters - anything larger is truncated.
workItemAttributes.attribute28: Text	See description above.
workItemAttributes.attribute29: Text	
workItemAttributes.attribute30: Text	

Attribute / Method	Comments
workItemAttributes.attribute31: Text	
workItemAttributes.attribute32: Text	
workItemAttributes.attribute33: Text	
workItemAttributes.attribute34: Text	
workItemAttributes.attribute35: Text	
workItemAttributes.attribute36: Text	
workItemAttributes.attribute37: Text	
workItemAttributes.attribute38: Text	
workItemAttributes.attribute39: Text	Limited to a maximum of 255 characters - anything larger is truncated.
workItemAttributes.attribute40: Text	See description above.
ItemContext	
getActivityId() : Text	
getActivityName() : Text	
getAppName() : Text	
getAppInstance() : Text	

Attribute / Method	Comments
getAppld() : Text	
getApplInstanceDescription() : Text	
ItemSchedule	
getStartDate() : Date	
getTargetDate : Date	



Note: When using `getWorkItem()` or `getSchedule()`, note that the following object is displayed:

- A null object when there is a null value.
- An empty object, "", when there is a 0 length value.

Troubleshooting Scripting

Occasionally, a script does not function as planned.

This section provides instruction on how to identify and resolve scripting problems.

Reserved keywords to avoid using for attribute names

If you use certain reserved keywords for BOM attribute names and assign them a value using JavaScript, at runtime you get a scripting error while evaluating the JavaScript expression.

Avoid using the following names for BOM attributes if you intend using such attributes in scripting:

- `notify`
- `equals`
- `wait`

- finalize
- hashCode
- toString

Break Script into Smaller Scripts

Add User Tasks between script tasks to see the field values.

Examine the Server Logs

TIBCO BPM Enterprise components write out logging information as they process work. It can be useful to look at these logs when debugging scripts that are not working.

When using BPM developer server, you must extract the logs from the BPM runtime container (this can be determined using the `docker ps` command and then, extracting the logs using the `docker logs` command for that container).

For the production server, the logs are stored according to the set up and policies of your Kubernetes server.

Checking the log file can help locate the cause of problems. For example, the following error appears in the logs if the value for Text attribute is not given in single-quotes in a DQL:

```
2021-07-28T11:04:57.890Z | WARN | ce_3 | CDM | CaseDataManagerImpl |
readCases | | Bad DQL: [DQL_VALUE_SHOULD_BE_QUOTED: Value for Text
attribute should be single-quoted: name="MyName", DQL_VALUE_SHOULD_BE_
QUOTED: Value for Text attribute should be single-quoted:
product="MyProduct"] (extendedMessage = 'Bad DQL: [DQL_VALUE_SHOULD_BE_
QUOTED: Value for Text attribute should be single-quoted: name="MyName",
DQL_VALUE_SHOULD_BE_QUOTED: Value for Text attribute should be single-
quoted: product="MyProduct"]')
com.tibco.bpm.cdm.api.exception.ArgumentException: Bad DQL: [DQL_VALUE_
SHOULD_BE_QUOTED: Value for Text attribute should be single-quoted:
name="MyName", DQL_VALUE_SHOULD_BE_QUOTED: Value for Text attribute
should be single-quoted: product="MyProduct"]
    at com.tibco.bpm.cdm.api.exception.ArgumentException.newBadDQL
(ArgumentException.java:113)
    at com.tibco.bpm.cdm.core.CaseManager.readCases(CaseManager.java:605)
    at com.tibco.bpm.cdm.core.CaseDataManagerImpl.readCases
(CaseDataManagerImpl.java:208)
```

Write Variable Values and Progress Updates from the Script to the BPM Log File

A facility for writing messages from scripts to the server called the BPM Log file is provided.

For example, the following can be done from a script:

```
Log.write("The deadline for the task is " + data.deadline);
```

This generates a message like the following in the BPM Log file, which can be found by searching for the part of the message.

```
Thu Jul 01 12:11:24 GMT+000 2021 Loading Script Engine Module  
11.0.0.046.ACE20201105_2000/2.5.1 initialize took 86  
The deadline for the task is Sun Jul 11 2021 12:11:24 GMT+0000 (GMT)
```

See [Examine the Server Logs](#) for the location of the log.

However, this function is only useful if you have access to the BPM Log file.

eval()

The `eval()` function provides the ability to execute a dynamic script useful for debugging purposes. This is given as a string, and executes it as if it were part of a script.

For example, to test some expressions, you could enter them into a Text field on a Form. Then, get a script to execute the commands in the Text field.

```
eval (scriptField);
```

Although useful for experimenting with scripts and testing them, TIBCO recommends that you do not use this function in a production environment because it allows the execution of any script text that is provided at runtime.

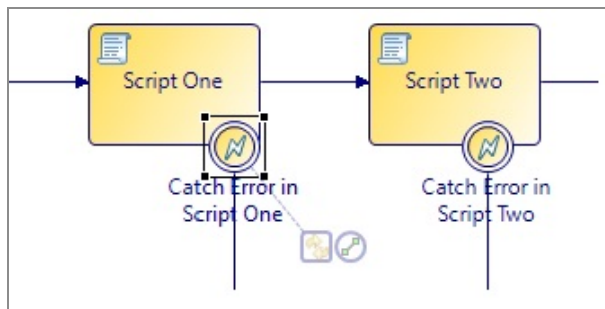
Catch Exceptions

If there is the possibility that a script can generate an exception at runtime, you can catch the exception and take corrective action in the process using a catch event.

For example, the following are some ScriptTasks with IntermediateCatch events attached to them, and the properties of one of them:

The screenshot shows the 'Intermediate Event' properties window. The 'General' tab is selected. The properties are as follows:

Property	Value
Label	Catch Error in Script One
Name	CatchErrorinScriptOne
Trigger Type	Catch Error
Catch Error Code	<Catch All>
Thrown By	<Any Activity>



Calling REST Services

Defining the Interface to an External REST Service

To call a REST Service you need to create a REST Service Descriptor (RSD) file (located in a separate REST Service project). The REST method from the RSD file can then be referenced from REST Service invocation activities.

The purpose of the REST Service Descriptor (RSD) is to define the end-point URL relative resource path (optionally including query and header parameters), the available methods for the resource and the request-response data details for those methods.



Note: REST Service projects do not need to be deployed. The service descriptor can be used by multiple REST service activities.

Procedure

1. Create a REST Services project (**File > New > REST Services Project**).
2. On the **General** tab of the Properties view for the REST Service Descriptor, add the **Context Path** for the REST service.

The context path is used in the URL for the service at runtime. **Context Path** is a common path prefix that is used for all resource URLs in this service. For example, /rest/test/2. The context path is applied to the beginning of all resource paths.

i Note: You should not use URL encoding (for instance %20 for spaces) in the Context Path of the Resource path definition in the REST Service Descriptor. Also, you should not encode values in Path Parameters (that is, the value of data fields mapped to those parameters). This is because the path and parameter values are automatically encoded at runtime. If there is a use case where the path might require URL delimiter characters (such as /, @, ? and so on) that need to be encoded then simply use a path parameter in the required location and then provide the required unencoded values in the data mapped to the parameter.

3. Configure resources (including a Path Template and Path Parameters, which are used in the URL for the method). Path Template might contain path parameter references. Parameter reference is constructed using the name of the parameter enclosed between { and } characters.

Path parameters are variable parts of the path URL that are replaced with process data at runtime (by mapping process data to the parameters in the REST service invocation task).

The URL for the REST invocation method is built up from the following: <baseurl> + context path + template path.

where <baseurl> is http://localhost:8080, context path is /rest/test/2, template path is /issue/{issueId}.

For example:

http://localhost:8080/rest/test/2/issue/{issueId}

The <baseurl> is provided by the HTTP Client shared resource associated with the REST service task participant during project deployment.

4. Add HTTP method/s for each resource, and select the HTTP Method type from GET, PUT, POST or DELETE.

The GET method is already added by default for each resource.

The **Request Overview** provides an idea of what the final URL for the REST invocation method is.

5. Create request and response information using the method in the Request and Response tabs. On each method, you can define Query Parameters/Header Parameters and Payload (which must be a JSON schema type). See [Creating JSON Schemas](#).

The PUT and POST methods have a Request Payload specification.

- **Query Parameters:** for example, `http://localhost:8080/?param1={param1}` where `{param1}` is a query parameter added using the Request tab.
- **Payload Type:** Select from existing JSON objects.



Note: If you cannot create the exact JSON object using the REST service task mapper (or the payload is not in JSON format), you can configure the REST method definition request payload type to be **Unprocessed Text**. In the REST Service task process create a text field whose value is the required JSON string and map that to the request payload.

- **Content Type:** needs to be set to that defined by the service.
- **Header Parameters:** some services require header parameters to be passed, also a service can return header parameters. These can be mapped from and to process data in an invocation task.

If you declare an error, you can also declare a JSON type for the information coming back and map it to process data using a catch error event attached to the REST service task.



Note: HTTP response status codes 300 and above are automatically treated as errors at runtime. You can catch any such error with a 'Catch All' task boundary error event or you can define faults for specific status codes that can be caught individually.

Creating JSON Schemas

You can create JSON schemas using the following procedure or alternatively you can create a JSON schema by deriving it from a JSON string sample (often made available by the documentation for the REST service). See [Creating JSON schema from a JSON sample](#).

A JSON schema declares the types and properties of the complex types used for request / response payload data when invoking REST service methods.

Procedure

1. Right-click a REST Services project in Project Explorer and select **New > New JSON Schema**.
2. On the Create a new JSON Schema file dialog box, click **Finish**.
3. Add Properties to the JSON object.
4. Create and use child types for properties of a complex type.
5. On the General tab on the Properties view for a JSON Object Type, edit the property types (which are 'Text' by default). Alternatively you can drag-drop types onto properties in the main editor. You can also move properties around using drag drop or the controls provided.

For services that require or return arrays, check **Array**.



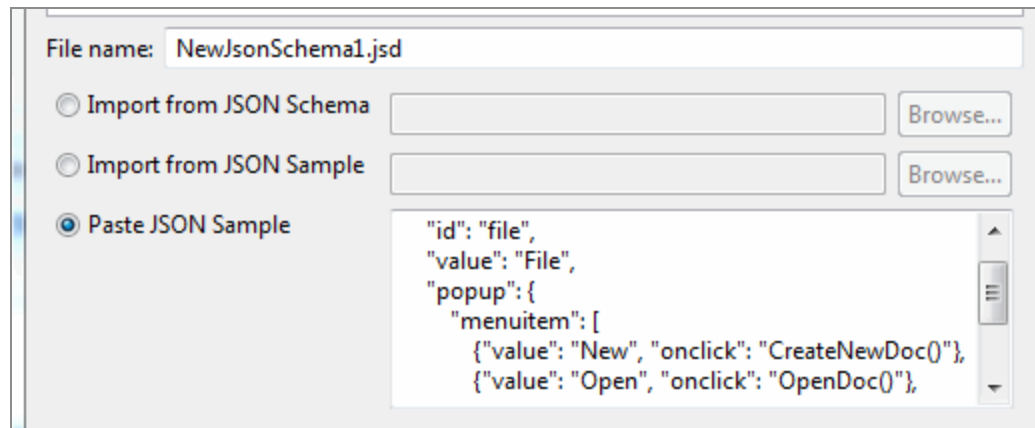
Note: You can also create schemas by importing IETF JSON Schema files (as defined at <http://json-schema.org/>).

Creating JSON Schemas from a JSON Sample

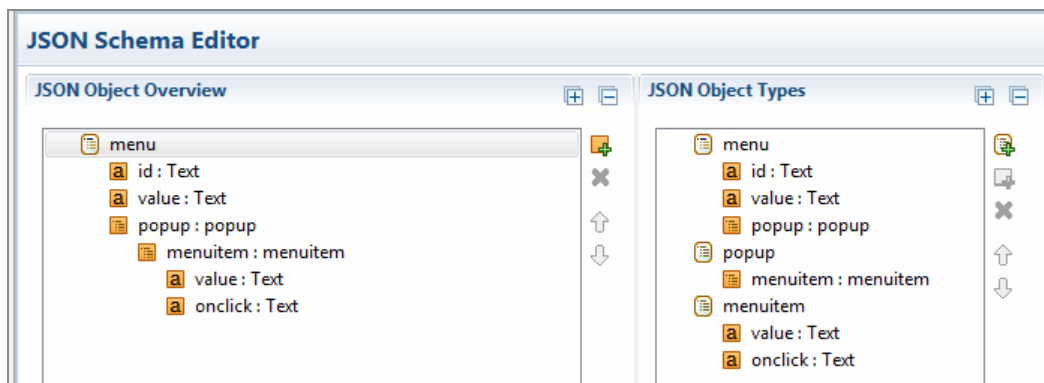
You can use the **Import JSON Schema** wizard to import and save various formats of JSON schema definitions.

Procedure

1. Select **Import > Import JSON Schema** from a REST services project.
2. You can choose to do one of the following:
 - **Import from JSON Schema:** you can import an existing JSON schema.
 - **Import from JSON Sample:** this allows you to import a sample of JSON data, and TIBCO Business Studio - BPM Edition creates a schema that describes the data structure.
 - **Paste from JSON Sample:** this allows you to paste a sample of JSON text directly into the wizard and TIBCO Business Studio - BPM Edition creates a schema that describes the data structure.



3. You can now view and edit the JSON schema you imported or pasted in the JSON schema editor.



Note: If you import a sample that is an array, then you need to import it as a single instance type and specify that it is an array on a specific method call in the REST Service Descriptor file. See [Creating JSON Schemas](#).

Configuring the Process Project from Which you Want to Call the REST Service

You can configure a task (Service Task, Send Task) in a process to call a REST Service.

Procedure

1. Add a Service Task or Send Task to your process, and configure it with a Service Type of **REST Service**.
2. Select Operation: **Select** to select the REST service method. See [Defining the Interface to an External REST Service](#).
3. Define any data fields needed for the call.
4. Map process data to / from the method parameters (see [Defining Input and Output Mappings](#) for more information).

Using a System Participant to Identify the REST Service Shared Resource

A REST service must use a system participant that identifies the shared resource name of the REST service that is to be invoked. This is a reference to a shared resource that must be created in ActiveMatrix BPM Administrator, and holds the runtime configuration for the REST service location and credentials.

When you select a REST service method in a REST service task, a system participant is automatically created using the service name as the shared resource name.

Depending on the name of the administration-defined shared resource, you might need to change the reference to it in the REST service participant. From Properties for the participant, under **Shared Resource > REST Service Consumer**, enter the shared resource name.

Defining Input and Output Mappings

When you are invoking a REST service operation, you must map the appropriate input/output parameters provided by the REST service operation to the appropriate parameters and/or data fields in the process.

On the Properties view for the relevant task or event, the **Input to Service** and **Output from Service** tabs provide a Mapper tool that allows you to easily perform the required mappings.

Data mappings are defined as follows:

- on the call request (the **Input To Service** tab), from process fields/parameters to

path/query/header/payload parameters exposed by the selected Method.

- on the call response (the **Output From Service** tab), from the payload or header parameters exposed by the selected Method to process fields/parameters.

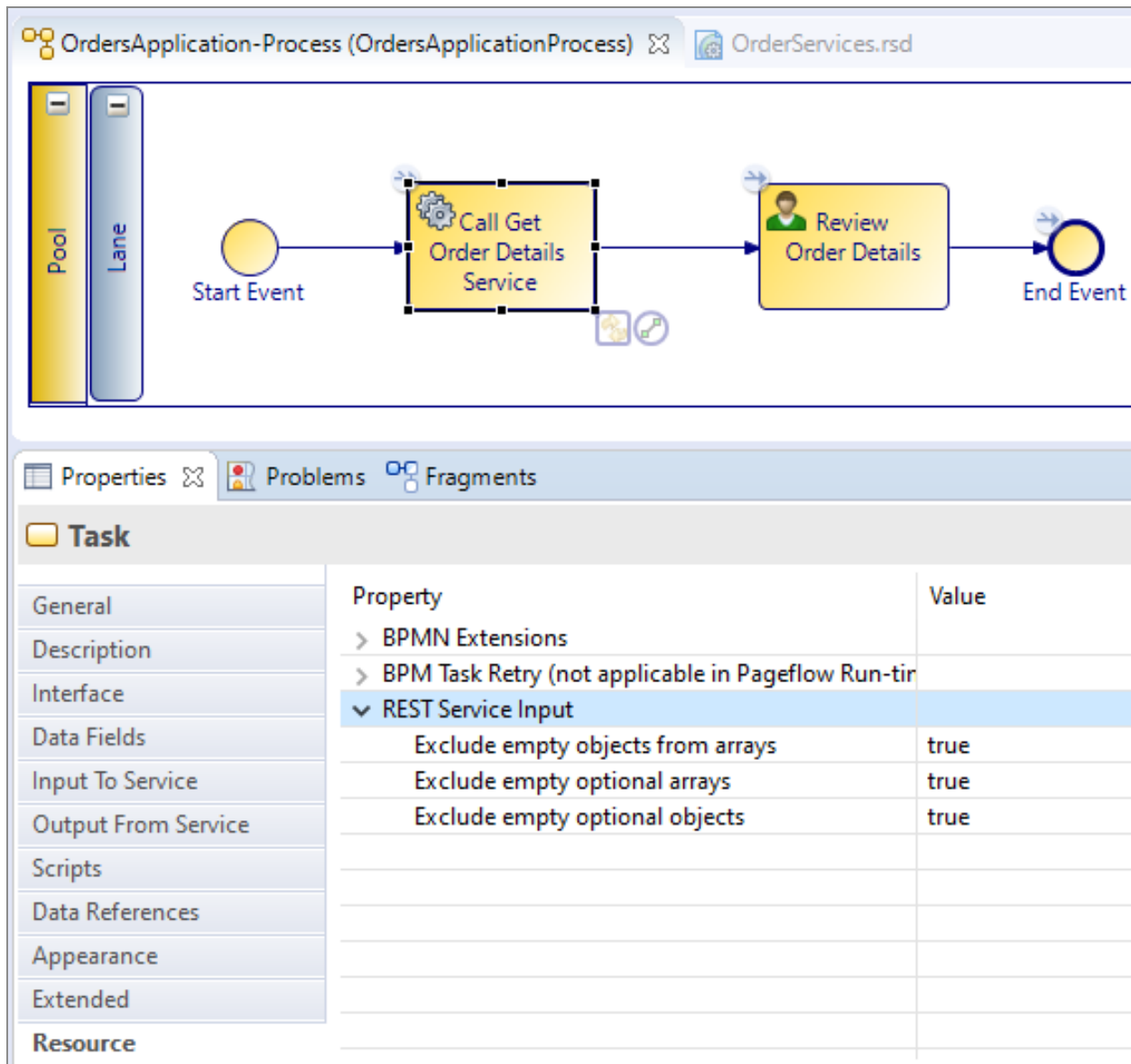
The request / response payload (optional) is JSON (or 'Unprocessed Text' type). The path and query parameters are used to map into variable parts of the request URL from process data values. The header parameters allow mapping process data to/from request/response header.

Some services do not need a payload, but might still need mappings for headers, path, and query parameters.

Exclusion of Empty Content from REST Service Input

You can configure the following options on REST service tasks:

- Exclude empty objects for optional input data
- Exclude empty objects in input array data
- Exclude empty arrays for optional input data



Exclude Empty Optional Objects

Set this option to `true` to exclude optional objects from the REST service input payload if they are empty.

If an optional input object is empty after applying all mappings within that object, then it is completely excluded from the REST input payload. In other words, if all of the source data fields mapped into the optional object are undefined, then the object is not included in the input payload.

If this option is set to `false`, then an empty object is passed in the input payload even if all of the source data fields mapped into an optional object are undefined.

Exclude Empty Optional Arrays

Set this option to `true` to exclude optional arrays from the REST service input payload if they are empty.

If an optional input array is empty after applying all mappings within that array, then it is completely excluded from the REST input payload. In other words, if all of the source data fields and lists mapped into the optional array are undefined or empty, then the array is not included in the input payload.

If this option is set to `false`, then an empty array is passed in the input payload even if all of the source data fields mapped into an optional array are undefined.

Exclude Empty Objects from Arrays

Set this option to `true` to exclude empty objects from arrays in the REST service input payload if the object is empty.

If an input object in an array is empty after applying all mappings within that object, then it is completely excluded from the array in the REST input payload. In other words, if all of the source data fields mapped into the object are undefined, then the object is not included in the input payload array.

If this option is set to `false`, then an empty object is passed in the input payload array even if all of the source data fields mapped into an optional object are undefined.

Workflow Process and Data Pattern Support

Control-flow (or process) patterns capture the various ways in which activities are represented and controlled in process workflows, ranging from basic control patterns to advanced multi-instance patterns.

Data patterns capture the various ways in which data is represented and utilized in workflows.

TIBCO Business Studio - BPM Edition provides in-built support for many of these patterns, giving it the capability to handle the widest range of possible scenarios for modeling and executing processes and process data.

The following sections describe the workflow patterns (Workflow Resource Patterns, Workflow Process Patterns, and Workflow Data Patterns) that are supported in this release of TIBCO Business Studio - BPM Edition.

Workflow Data Patterns Support

Data patterns capture the various ways in which data is represented and utilized in workflows. Implementing these patterns gives TIBCO Business Studio - BPM Edition the capability to handle the widest range of possible scenarios for modeling and executing data.

The table below lists the data patterns that are supported in this release of TIBCO Business Studio - BPM Edition. The pattern numbers, names, and descriptions are those defined by the Workflow Patterns initiative. See:

- <http://www.workflowpatterns.com/patterns/data/index.php>
- N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns: Identification, Representation and Tool Support. (PDF, 281Kb) In Proceedings of the 24th International Conference on Conceptual Modeling (ER 2005), volume 3716 of Lecture Notes in Computer Science, pages 353-368. Springer-Verlag, Berlin, 2005.

Supported Data Patterns

Pattern Number	Pattern Name	Pattern Description
<i>Data Visibility</i>		
1	Task Data	Data elements can be defined by tasks which are accessible only within the context of individual execution instances of that task.
2	Block Data	Block tasks (i.e. tasks which can be described in terms of a corresponding subprocess) are able to define data elements which are accessible by each of the components of the corresponding subprocess.
5	Case Data	Data elements are supported which are specific to a process instance or case. They can be accessed by all components of the process during the execution of the case.
7	Workflow Data	Data elements are supported which are accessible to all components in each and every case of the process and are within the context of the process itself.
8	Environment Data	Data elements which exist in the external operating environment are able to be accessed by components of processes during execution.
<i>Internal Data Interaction</i>		
9	Task to Task	The ability to communicate data elements between one task instance and another within the same case. The communication of data elements between two tasks is specified in a form that is independent of the task definitions themselves.
10	Block Task to	The ability to pass data elements from a block task

Pattern Number	Pattern Name	Pattern Description
	Sub-Workflow Decomposition	instance to the corresponding subprocess that defines its implementation. Any data elements that are available to a block task are able to be passed to (or be accessed) in the associated subprocess although only a specifically nominated subset of those data elements are actually passed to the subprocess.
11	Sub-Workflow Decomposition to Block Task	The ability to pass data elements from the underlying subprocess back to the corresponding block task. Only nominated data elements defined as part of the subprocess are made available to the (parent) block task.
12	Data Interaction - to Multiple Instance Task	The ability to pass data elements from a preceding task instance to a subsequent task which is able to support multiple execution instances. This might involve passing the data elements to all instances of the multiple instance task or distributing them on a selective basis. The data passing occurs when the multiple instance task is enabled.
13	Data Interaction - from Multiple Instance Task	The ability to pass data elements from a task which supports multiple execution instances to a subsequent task. The data passing occurs at the conclusion of the multiple instance task. It involves aggregating data elements from all instances of the task and passing them to a subsequent task.
14	Data Interaction - Case to Case	The passing of data elements from one case of a process during its execution to another case that is executing concurrently.
<i>External Data Interaction</i>		

Pattern Number	Pattern Name	Pattern Description
15	Task to Environment - Push Oriented	The ability of a task to initiate the passing of data elements to a resource or service in the operating environment.
16	Environment to Task - Pull Oriented	The ability of a task to request data elements from resources or services in the operational environment.
19	Data Interaction - Case to Environment - Push-Oriented	<p>The ability of a case to initiate the passing of data elements to a resource or service in the operational environment.</p> <p>Note: Case in this situation means an ActiveMatrix BPM process instance.</p>
20	Data Interaction - Environment to Case - Pull-Oriented	<p>The ability of a case to request data from services or resources in the operational environment.</p> <p>Note: Case in this situation means an ActiveMatrix BPM process instance.</p>
21	Data Interaction - Environment to Case - Push-Oriented	<p>The ability of a case to accept data elements passed to it from services or resources in the operating environment.</p> <p>Note: Case in this situation means an ActiveMatrix BPM process instance.</p>
22	Data Interaction - Case to Environment - Pull-Oriented	<p>The ability of a case to respond to requests for data elements from a service or resource in the operating environment.</p> <p>Note: Case in this situation means an ActiveMatrix BPM process instance.</p>
23	Data Interaction -	The ability of a process environment to pass data

Pattern Number	Pattern Name	Pattern Description
	Workflow to Environment - Push-Oriented	elements to resources or services in the operational environment.
24	Data Interaction - Environment to Workflow - Pull-Oriented	The ability of a process environment to request case data elements from external applications.
25	Data Interaction - Environment to Workflow - Push-Oriented	The ability of services or resources in the operating environment to pass case data to a process.
26	Data Interaction - Workflow to Environment - Pull-Oriented	The ability of the process environment to handle requests for case data from external applications.
<i>Data Transfer</i>		
29	Data Transfer - Copy In/Copy Out	The ability of a process component to copy the values of a set of data elements from an external source (either within or outside the process environment) into its address space at the commencement of execution and to copy their final values back at completion.
30	Data Transfer by Reference - Unlocked	The ability to communicate data elements between process components by utilizing a reference to the location of the data element in some mutually accessible location. No concurrency restrictions apply to the shared data element.

Pattern Number	Pattern Name	Pattern Description
32	Data Transformation - Input	The ability to apply a transformation function to a data element prior to it being passed to a process component. The transformation function has access to the same data elements as the receiving process component.
33	Data Transformation - Output	The ability to apply a transformation function to a data element immediately prior to it being passed out of a process component. The transformation function has access to the same data elements as the process component that initiates it.
<i>Data-based Routing</i>		
38	Event-based Task Trigger	The ability for an external event to initiate a task and to pass data elements to it.
40	Data-Based Routing	Data-based routing provides the ability to alter the control-flow within a case based on the evaluation of data-based expressions. A data-based routing expression is associated with each outgoing arc of an OR-split or XOR-split. It can be composed of any data-values, expressions and functions available in the process environment providing it can be evaluated at the time the split construct with which it is associated completes. Depending on whether the construct is an XOR-split or OR-split, a mechanism is available to select one or several outgoing arcs to which the thread of control should be passed based on the evaluation of the expressions associated with the arcs.

Workflow Process Patterns Support

Control-flow patterns capture the various ways in which activities are represented and controlled in workflows. Implementing these patterns gives TIBCO Business Studio - BPM

Extend the capability to handle the widest range of possible scenarios for modeling and executing processes.

The below table lists the control-flow patterns that are supported in this release of TIBCO Business Studio - BPM Edition. The pattern numbers, names and descriptions are those defined by the Workflow Patterns initiative. See:

- <http://www.workflowpatterns.com/patterns/control/index.php>
- N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, BPMcenter.org, 2006.
- W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. Distributed and Parallel Databases, 14(3), pages 5-51, July 2003

Supported control flow patterns

Pattern Number	Pattern Name	Pattern Description
<i>Basic Control Flow Patterns</i>		
1	Sequence	An activity in a workflow process is enabled after the completion of a preceding activity in the same process.
2	Parallel Split	The divergence of a branch into two or more parallel branches each of which execute concurrently.
3	Synchronization	The convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled.
4	Exclusive Choice	The divergence of a branch into two or more branches. When the incoming branch is enabled, the thread of control is immediately passed to precisely one of the outgoing branches based on the outcome of a logical expression associated with the branch.

Pattern Number	Pattern Name	Pattern Description
5	Simple Merge	The convergence of two or more branches into a single subsequent branch. Each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.
<i>Advanced Branching and Synchronization Patterns</i>		
6	Multi-Choice	The divergence of a branch into two or more branches. When the incoming branch is enabled, the thread of control is passed to one or more of the outgoing branches based on the outcome of distinct logical expressions associated with each of the branches.
7	Structured Synchronizing Merge	The convergence of two or more branches (which diverged earlier in the process at a uniquely identifiable point) into a single subsequent branch. The thread of control is passed to the subsequent branch when each active incoming branch has been enabled.
8	Multi-Merge	The convergence of two or more branches into a single subsequent branch such that each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.
9	Structured Discriminator	The convergence of two or more branches into a single subsequent branch following a corresponding divergence earlier in the process model. The thread of control is passed to the subsequent branch when the first incoming branch has been enabled. Subsequently enabling incoming branching does not result in the thread of control being passed on. The

Pattern Number	Pattern Name	Pattern Description
		discriminator construct resets when all incoming branches have been enabled.
29	Canceling Discriminator	The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when the first active incoming branch has been enabled. Triggering the canceling Discriminator also cancels the execution of all of the other incoming branches and resets the construct.
30	Structured Partial Join	The convergence of two or more branches (say m) into a single subsequent branch following a corresponding divergence earlier in the process model such that the thread of control is passed to the subsequent branch when n of the incoming branches have been enabled where n is less than m. Subsequent enablements of incoming branches do not result in the thread of control being passed on. The join construct resets when all active incoming branches have been enabled. The join occurs in a structured context, i.e. there must be a single Parallel Split construct earlier in the process model with which the join is associated and it must merge all of the branches emanating from the Parallel Split. These branches must either flow from the Parallel Split to the join without any splits or joins or be structured in form (i.e. balanced splits and joins).
32	canceling Partial Join	The convergence of two or more branches (say m) into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to

Pattern Number	Pattern Name	Pattern Description
		the subsequent branch when n of the incoming branches has been enabled where n is less than m. Triggering the join also cancels the execution of all of the other incoming branches and resets the construct.
<i>Multiple Instance Patterns</i>		
12	Multiple Instances without Synchronization	Within a given process instance, multiple instances of an activity can be created. These instances are independent of each other and run concurrently. There is no requirement to synchronize them upon completion.
13	Multiple Instances with <i>a priori</i> Design Time Knowledge	Within a given process instance, multiple instances of an activity can be created. The required number of instances is known at design time. These instances are independent of each other and run concurrently. It is necessary to synchronize the activity instances at completion before any subsequent activities can be triggered.
14	Multiple Instances With <i>a priori</i> Run-Time Knowledge	Within a given process instance, multiple instances of an activity can be created. The required number of instances might depend on a number of runtime factors, including state data, resource availability and inter-process communications, but is known before the activity instances must be created. Once initiated, these instances are independent of each other and run concurrently. It is necessary to synchronize the instances at completion before any subsequent activities can be triggered.
15	Multiple Instances	Within a given process instance, multiple instances

Pattern Number	Pattern Name	Pattern Description
	without a priori Run-Time Knowledge	of a task can be created. The required number of instances might depend on a number of runtime factors, including state data, resource availability and inter-process communications and is not known until the final instance has been completed. Once initiated, these instances are independent of each other and run concurrently. At any time, whilst instances are running, it is possible for additional instances to be initiated. It is necessary to synchronize the instances at completion before any subsequent tasks can be triggered.
<i>State-based Patterns</i>		
16	Deferred Choice	A point in a process where one of several branches is chosen based on interaction with the operating environment. Prior to the decision, all branches represent possible future courses of execution. The decision is made by initiating the first task in one of the branches i.e. there is no explicit choice but rather a race between different branches. After the decision is made, execution alternatives in branches other than the one selected are withdrawn.
18	Milestone	A task is only enabled when the process instance (of which it is part) is in a specific state (typically a parallel branch). The state is assumed to be a specific execution point (also known as a milestone) in the process model. When this execution point is reached the nominated task can be enabled. If the process instance has progressed beyond this state, then the task cannot be enabled now or at any future time (i.e. the deadline has expired). Note that the execution does not influence the state itself, i.e.

Pattern Number	Pattern Name	Pattern Description
		unlike normal control-flow dependencies it is a test rather than a trigger.
<i>Cancellation and Force Completion Patterns</i>		
19	Cancel Task	An enabled task is withdrawn prior to it commencing execution. If the task has started, it is disabled and, where possible, the currently running instance is halted and removed.
20	Cancel Case	A complete process instance is removed. This includes currently executing tasks, those which might execute at some future time and all sub-processes. The process instance is recorded as having completed unsuccessfully.
25	Cancel Region	The ability to disable a set of tasks in a process instance. If any of the tasks are already executing (or are currently enabled), then they are withdrawn. The tasks need not be a connected subset of the overall process model.
<i>Iteration Patterns</i>		
10	Arbitrary Cycles	The ability to represent cycles in a process model that have more than one entry or exit point.
21	Structured Loop	The ability to execute a task or sub-process repeatedly. The loop has either a pre-test or post-test condition associated with it that is either evaluated at the beginning or end of the loop to determine whether it should continue. The looping structure has a single entry and exit point.

Pattern Number	Pattern Name	Pattern Description
22	Recursion	The ability of a task to invoke itself during its execution or an ancestor in terms of the overall decomposition structure with which it is associated.
<i>Termination Patterns</i>		
11	Implicit Termination	A given process (or sub-process) instance should terminate when there are no remaining work items that are able to be done either now or at any time in the future.
43	Explicit Termination	A given process (or sub-process) instance should terminate when it reaches a nominated state. Typically this is denoted by a specific end node. When this end node is reached, any remaining work in the process instance is canceled and the overall process instance is recorded as having completed successfully, regardless of whether there are any tasks in progress or remaining to be executed.
<i>Trigger Patterns</i>		
23	Transient Trigger	The ability for a task instance to be triggered by a signal from another part of the process or from the external environment. These triggers are transient in nature and are lost if not acted on immediately by the receiving task. A trigger can only be utilized if there is a task instance waiting for it at the time it is received.
24	Persistent Trigger	The ability for a task to be triggered by a signal from another part of the process or from the external environment. These triggers are persistent in form and are retained by the process until they can be acted on by the receiving task.

Workflow Resource Patterns Support

Workflow resource patterns capture the various ways in which resources are represented and utilized in workflows. Implementing these patterns gives TIBCO Business Studio - BPM Edition the capability to handle the widest range of possible scenarios for modeling and executing business processes.

The table lists the workflow resource patterns that are supported in this release of TIBCO Business Studio - BPM Edition.

The pattern numbers, names and descriptions are those defined by the Workflow Patterns initiative. See:

- <http://www.workflowpatterns.com/patterns/resource/index.php>
- N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. (PDF, 206 Kb). In Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05), volume 3520 of Lecture Notes in Computer Science, pages 216-232. Springer-Verlag, Berlin, 2005.

Supported workflow resource patterns

Pattern Number	Pattern Name	Pattern Description
<i>Creation Patterns</i>		
1	Direct Distribution	The ability to specify at design time the identity of the resource(s) to which instances of this task are distributed at runtime.
2	Role-Based Allocation	The ability to specify at design time that a task can only be executed by resources which correspond to a given role.
3	Deferred Distribution	The ability to specify at design-time that the identification of the resource(s) to which instances of this task are distributed is deferred until runtime.
4	Authorization	The ability to specify the range of privileges that a

Pattern Number	Pattern Name	Pattern Description
		resource possesses in regard to the execution of a process. In the main, these privileges define the range of actions that a resource can initiate when undertaking work items associated with tasks in a process.
5	Separation of Duties	The ability to specify that two tasks must be executed by different resources in a given case.
7	Retain Familiar	Where several resources are available to undertake a work item, the ability to allocate a work item within a given case to the same resource that undertook a preceding work item.
8	Capability-Based Distribution	The ability to distribute work items to resources based on specific capabilities that they possess. Capabilities (and their associated values) are recorded for individual resources as part of the organization model.
10	Organizational Distribution	The ability to distribute work items to resources based on their position within the organization and their relationship with other resources.
11	Automatic Execution	The ability for an instance of a task to execute without needing to utilize the services of a resource.
<i>Push Patterns</i>		
13	Distribution by Offer – Multiple Resources	The ability to offer a work item to a group of selected resources.
14	Distribution by	The ability to distribute a work item to a specific

Pattern Number	Pattern Name	Pattern Description
	Allocation - Single Resource	resource for execution on a binding basis.
15	Random Allocation	The ability to allocate work items to a selected resource chosen from a group of eligible resources on a random basis.
16	Round Robin Allocation	The ability to allocate a work item to a selected resource chosen from a group of eligible resources on a cyclic basis.
19	Distribution on Enablement	The ability to advertise and allocate work items to resources at the moment they are enabled for execution.
<i>Pull Patterns</i>		
21	Resource-Initiated Allocation	The ability for a resource to commit to undertake a work item without needing to commence working on it immediately.
22	Resource-Initiated Execution - Allocated Work Item	The ability for a resource to commence work on a work item that is allocated to it.
23	Resource-Initiated Execution - Offered Work Item	The ability for a resource to select a work item offered to it and commence work on it immediately.
24	System-Determined Work Queue Content	The ability of the workflow engine to order the content and sequence in which work items are presented to a resource for execution.

Pattern Number	Pattern Name	Pattern Description
25	Resource-Determined Work Queue Content	The ability for resources to specify the format and content of work items listed in the work queue for execution.
26	Selection Autonomy	The ability for resources to select a work item for execution based on its characteristics and their own preferences.
<i>Detour Patterns</i>		
27	Delegation	The ability for a resource to allocate an unstarted work item previously allocated to it (but not yet commenced) to another resource.
28	Escalation	The ability of a system to distribute a work item to a resource or group of resources other than those it has previously been distributed to in an attempt to expedite the completion of the work item.
29	Deallocation	The ability of a resource (or group of resources) to relinquish a work item which is allocated to it (but not yet commenced) and make it available for distribution to another resource or group of resources.
30	Stateful Reallocation	The ability of a resource to allocate a work item that they are currently executing to another resource without loss of state data.
31	Stateless Reallocation	The ability for a resource to reallocate a work item that it is currently executing to another resource without retention of state.
32	Suspension-	The ability for a resource to suspend and resume

Pattern Number	Pattern Name	Pattern Description
	Resumption	execution of a work item.
33	Skip	The ability for a resource to skip a work item allocated to it and mark the work item as complete.
<i>Auto-Start Patterns</i>		
37	Commencement on Allocation	The ability to commence execution on a work item as soon as it is allocated to a resource.
38	Piled Execution	The ability to initiate the next instance of a task (perhaps in a different case) once the previous one has been completed with all associated work items being allocated to the same resource. The transition to Piled Execution mode is at the instigation of an individual resource. Only one resource can be in Piled Execution mode for a given task at any time.
39	Chained Execution	The ability to automatically start the next work item in a case once the previous one has been completed. The transition to Chained Execution mode is at the instigation of the resource.
<i>Visibility Patterns</i>		
41	Configurable Allocated Work Item Visibility	The ability to configure the visibility of allocated work items by process participants.
<i>Multiple Resource Patterns</i>		
42	Simultaneous Execution	The ability for a resource to execute more than one work item simultaneously.

Using Organization Models

Organization Model Concepts

The Organization Modeler allows you to define the organizational structure of your enterprise and the relationships between the different components (for example, organization units and positions) within your organization.

Organization models are useful in the analysis phase of a project, when you are defining the business processes.

Organization Model Deployment

To be used at runtime an organization model must be deployed to a server.

An organization model is seen as part of an application. The application consists of a business process and any supporting material, which can include an organization model. One organization model can be used by multiple applications; your business might have different applications for different business functions, but all of them would need to reference a model of the same organization.

When you deploy an organization model, any Resources that you have defined are not deployed (with the exception of the Human Resource Type, which must exist and is always deployed). All other parts of the organization model are deployed. See [Resources](#) for further details.

Data Types

Schema Type attributes and qualifying information for Capabilities and Privileges can have data types.

The following elements of an Organization Model can have data types:

- Schema Type attributes, see [Attributes](#).

- Qualifying information for Capabilities and Privileges, see [Capabilities and Privileges](#).

The table describes the data types that these elements can have.

Type	Description
Boolean	A value of True or False, or blank.
Date	Any date in the format of the locale on your machine.
DateTime	A date and time in the format of the locale on your machine.
Decimal	Any number, positive or negative, up to the number of decimals you specify.
Enum	An enumerated type. By selecting this type, you can specify an enumerated value. Each Enum is one of the list of values that make up an EnumSet.
EnumSet	A data type that can contain a list of values. By selecting this type, you can specify a set of enumerated values.
Integer	A whole number, including zero and negative numbers.
Set	Allows arrays of arbitrary, unique, string values.
Text	Any combination of alphanumeric characters.
Time	Any time in the format of the locale on your machine.

The Organization Model as an Analysis Tool

In the analysis phase of a project you can use Organization Modeler to visualize the organizational structure that underpins a process.

The structure of an organization is a fundamental aspect of how the organization works.

- It shows how people are organized to achieve the objectives of the business.
- It models the relationships between enterprise systems relevant to the organization.
- It models the relationships between the different departments that describe the day

to day operation of the organization.

- It determines how the work that arises from the business processes is allocated between different departments and positions within your organization.
- It identifies both concrete resources - people and buildings, for instance - and abstract resources such as roles.

The structure of an organization is a key aspect in the operation of information systems like human resources, payroll and accounting and business process/workflow systems. These systems require a consistent view of the organization to operate efficiently.

However, maintaining a consistent view of the organization is difficult for two reasons:

- Modern enterprises are often spread across different locations and have relationships with extended enterprises. People work in cross-functional teams which might be spread across different locations and enterprises. This makes it hard to identify resources when allocating work within the organization.
- Organizations no longer consist of one global scheme. Organizations are split geographically, by product or by markets and these co-exist within one corporate entity.

Organization Modeler allows you to maintain a model of your enterprise's organization structure. It consists of elements that represent the organization's entities, their attributes and the relationships between them.

Organization Modeler does not produce an organization chart; it does not identify named individuals. But, you can model your organization abstractly. Managers need to be able to develop robust models of their organization so that this information can be shared by people and systems.

The Organization Model at Runtime

At runtime, how an end user's position is defined in the organization model can be used to determine what type of work is presented to them.

Customized role-based clients can offer work to users depending on the Position they hold, the Capabilities or Privileges attributed to them, or both. For example, a user with an 'LDAP Administration' privilege could be offered all and only LDAP work.

Elements of Organization Models

Using the Organization Modeler, you can create a robust model of your organization.

You can create the following elements that would make up your Organization Model:

- **Organizations** represent both the organization you are modeling and any other enterprises that your organization might have a relationship with. For example, there might be a company your organization has outsourced part of its operation to.
- **Organization Units** represent sub-divisions of an Organization. They are collections of positions which are associated together because they fulfil a business purpose within the organization. For example, an organization unit can be a department, project or location.
- **Positions** represent a set of responsibilities for a job. Positions are created within Organization Units. For example, an Administrator in the Finance Department has different responsibilities from an Administrator in the Human Resources Department.
- **Groups** represent job types within your organization; for example, Chef, Salesman, Doctor and Pharmacist. This is useful for example, if you want to allocate work to a group of people with a specific set of skills. Groups are equivalent to Roles in the Process Modeler.
- **Capabilities** can be applied to Positions and to Groups. They represent the skills within your organization; for example, ability to speak Spanish or customer care training.
- **Privileges** represent the authority that Groups, Positions and Organization Units can have within your organization. They can have Qualifiers which indicate the level of the privilege. For example, an Approval privilege might have one qualifier to sign off expenses up to \$500, or a higher level of qualifier for approval of budgets up to \$10,000.
- **Locations** represent the physical locations that are used by your organization.
- **Resources** are used to specify items such as people, equipment or buildings.
- **Organization queries** are specified either as strings of text, which is not checked or validated within Organization Modeler; or in Resource Query Language, which is validated.
- **System actions** are actions that users perform at runtime but that need to be authorized. They are not defined within Organization Modeler, but you can associate Privileges with a list of available system actions in order to specify the level of

authorization that a user needs to carry out an action.

- **Types**, in the schema, represent typical elements within your Organization. You can use the schema as a template for the different organizational components and ideas that your Organization contains.

Benefits of Organization Models

By creating an Organization Model, you can provide a number of benefits and capabilities.

Using the Organization Model, you can:

- take advantage of the capabilities to distribute work at runtime to those users who are best suited to carry out a particular task, as determined by their position within the organization model or by privileges and capabilities defined within the organization model.
- when you are defining a process, use expressions to define the participant who carries out a task in terms of the Organization Modeler entities.
- model the way people are organized together within the organization. This is useful as you can see how people work together and how work is allocated.
- model both hierarchical relationships within the organization and also the associations that exist between cross-functional teams.
- model virtual or temporary project teams and positions. This is useful for example, as you can create project teams irrespective of their physical locations or job status.
- specify locations so that you can view how your organization is distributed across its various locations.
- specify privileges so that you can view the chains of authority in your organization. Authority is not always hierarchical; it is often given for different purposes. There might be multiple chains of authority within an organization.
- specify the skills (capabilities) you have within your organization so that you can view what skills are available and what skills your organization might need to acquire.
- specify types for certain elements within the organization. Some extended enterprises refer to organizational concepts differently, for example, region vs. district. By specifying types for particular elements, you create a generic schema for your Organization Model that enables it to be exchanged with other systems. Using types also gives you the ability to add custom attributes and custom elements to

your organization schema.

- specify attributes for types. You can specify an attribute of telephone number or email address for a resource type, for example, and then values for that attribute can be assigned to each resource of that type.

Organization Model Creation

An Organization Model must be contained within an organization project that contains an Organization Model special folder.

You can create an Organization Model in the following ways:


- Create a new Organization Model Project (or Project, which by default creates an Organization Model and corresponding special folder);
- Add an Organization Model and corresponding special folder to an existing project.

Creating a Project Containing an Organization Model

You can create an Organization Model in a new project.

Procedure

1. Select **File > New > Organization Project**.
2. The **New Project** wizard is displayed. Enter a name for your project in the **Project name** field and click **Next**.

 **Note:** This accepts default values for **Location** and **Id**.

3. In the Organization Model dialog box, ensure that **Create initial Organization Model** is checked. In the **file name** field, enter a name for the Special Folder for Organization Models, or use the one provided by default. Click **Finish**.

i Note: Leave the **Create default schema types** box checked to use the delivered schema.

Check the **Apply default Organization Type to Organization** box if you want the Organization Type delivered in the default schema to be applied to the initial organization created. If not, leave it unchecked.

See [Creating a Schema](#) for further details.

4. If you are not already in the **BPM Modeling** (or **Modeling**) perspective, you are prompted to switch to **BPM Modeling**. Click **Yes** to switch perspective.

Adding an Organization Model in an Existing Project

You can add an Organization Model to an existing project but you must first create a special folder.

Procedure

1. Right-click the project where you want to add the Organization Model and select **New > Folder**. The New Folder dialog box is displayed.
2. In the **Folder Name** field, type a relevant folder name. Click **Finish** to close the dialog box.
3. Right-click the **Organization** folder and select **Special Folders > Other > Use as Organization Models Folder**. If you right-click the **Organization** folder now, the option to create a new Organization Model is available.

Creating an Organization Model

The Organization Model is contained in a file called **name.om**, where *name* is typically the name of the entity for which you are creating the Organization Model. An Organization Model can contain more than one Organization.

Procedure

1. In the **Project Explorer**, select the **Organization** folder in the project where you want to create your Organization Model.

2. Right-click the **Organization** folder or the **OrganizationModel.om** file and select **New > Organization Model**.

3. The **Create Organization Model Diagram** wizard is displayed.

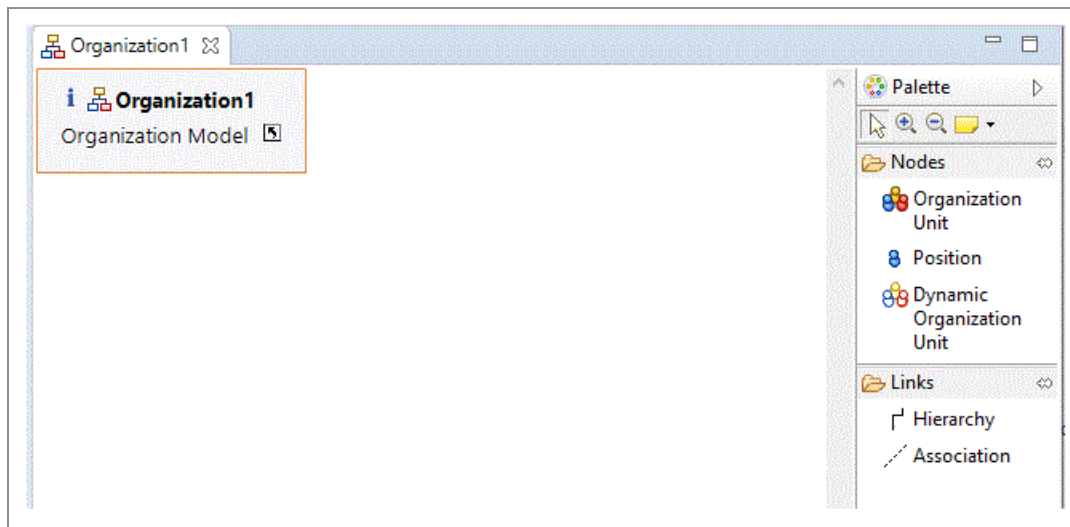
Make sure that the **Create default schema types** box is checked.

The folder you selected should be displayed in the **Create Organization Model Diagram** wizard. You can use the folder you selected or select a different folder, depending on your requirements. However, it must be a special folder of the **Organization** type.

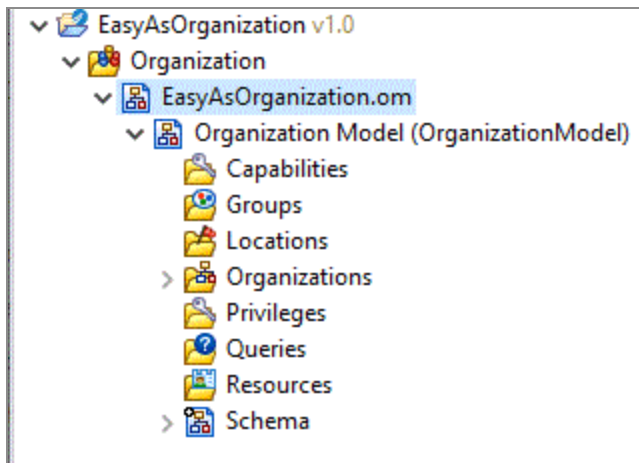
4. In the **File name:** field, type the name you want to apply to your Organization Model, followed by the suffix **.om**. Click **Finish**.

Result

The Organization Editor is displayed, representing the single default Organization created in the Organization Model. This information is displayed in the badge in the top left hand corner of the editor. You can navigate to the parent Organization Model editor by clicking on the shortcut arrow in the badge.



The Organization Model is also displayed in the Project Explorer view.



Note: You can use quick-find (Ctrl+F) in the project explorer to find existing organization model entities and select them in the project explorer.

When you have created an organization model, you can search for organization model diagram elements using quick find (Ctrl-F) within the diagram, and entering the initial characters of the name you are searching for. Double-click the element you are shown in the search to go to its location in the diagram.

Creating a Schema

The Organization Schema is contained in the Organization Model file (named by default *organization.om*, where *organization* is the name of the organization for which you are creating the Organization Model).

Procedure

1. To create the default schema, when you are creating an organization model, either as part of creating a new project or separately, ensure that you leave the **Create default schema types** check box selected.

If you select the **Apply default Organization Type to Organization** check box, the default Organization Type (called **Public Company**) from the default schema is applied to the initial Organization that is automatically created as part of this Organization Model. If you leave this check box empty, the Standard Organization Type is not applied.

Note that if you do not select **Create default schema types**, the **Apply default Organization Type to Organization** check box is not available:

2. Click **Finish**. The default schema is created.

Using Your Own Schema Types

You can use your own schema types by following the procedure described for creating a schema, but without selecting the **Create default schema** check box.

See [Creating a Schema](#).

Procedure

1. To use your own schema type, when you are creating an organization model, either as part of creating a new project or separately, ensure that you uncheck the **Create default schema types** check box.

Note that if you do not select **Create default schema types**, the **Apply default Organization Type to Organization** check box is not available so the Standard Organization Type is not applied:

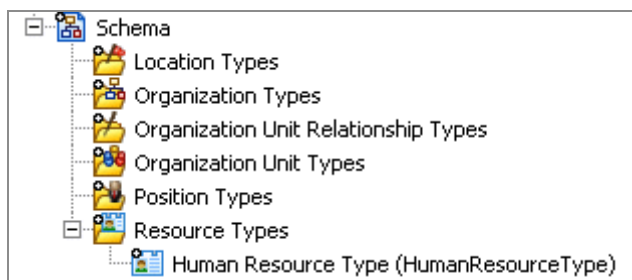
2. Click **Finish**. Your own schema type is created.

The schema created does not include the standard types provided with the default schema. The only type created is one Resource type, the Human Resource Type.

Note: Unlike the Human Resource Type created in the default schema, this one has no attributes defined.

Result

The following illustration shows what is visible in the Project Explorer when you have created a non-default schema:



Creation Types in the Schema

If you are building your own schema, you need to create the types that make it up.

If you are using the default schema provided, you can create new types to add to the schema in the same way as you would for a new schema.


You can also modify the types provided. You can also extend the default schema provided by setting up additional attributes to define more closely the elements that make up the organization you wish to model.

Creating a Location Type

You can create a location type in a schema.

Procedure

1. In the **Project Explorer**, expand the **Schema** folder for your Organization Model.
2. In the **Schema** folder, right-click **Location Types** and select **Add Child > Location Type**. A new Location Type is created.
3. Click the **General** tab in the **Properties** view. In the **Label** box, type the name of the Location Type.

Click the **Attributes** tab. Click  to add an attribute. Type in the name and select a type for the Attribute from the **Type** drop-down list. You can also add a default value for the Attribute, depending on your requirements.



Click  to remove attributes.

Creating an Organization Type


You can create an Organization type in a schema.

Procedure

1. In the **Project Explorer**, expand the **Schema** folder for your Organization Model.
2. In the **Schema** folder, right-click **Organization Types** and select **Add Child > Organization Type**. A new Organization Type is created.

3. Click the **General** tab in the **Properties** view. In the **Label** box, type the name of the Organization Type.
4. To add an Organization Unit as a member to the Organization Type:
 - a. In the **Unit Members** box, click . Type the name of the member in the **Label** field.
 - b. To specify that the member should be of a particular Organization Unit Type, click in the **Type** field and then click the  button that then becomes available.
 - c. The Select Type dialog box is displayed. Click on the Organization Unit Type you require, and then click **OK**.
5. In the **Multiplicity** field, type in the multiplicity you want.

Click  to remove Organization elements.


Click the **Attributes** tab. Click  to add an attribute. Type in the name and select a type for the Attribute from the **Type** drop-down list. You can also add a default value for the Attribute, depending on your requirements.







Click  to remove attributes.

Creating an Organization Unit Type

You can create an Organization Unit type in a schema.

Procedure

1. In the **Project Explorer**, expand the **Schema** folder for your Organization Model.
2. In the **Schema** folder, right-click **Organization Unit Types** and select **Add Child > Organization Unit Type**. A new Organization Unit Type is created.
3. Click the **General** tab in the **Properties** view. In the **Label** box, type the name of the Organization Unit Type.
4. To add another Organization Unit as a member to the Organization Unit Type:
 - a. In the **Unit Members** box, click . Type the name of the member in the **Label** field.


- b. To specify that the member should be of a particular Organization Unit Type, click on the **Type** field and then click the  button that then becomes available.
 - c. The Select Type dialog box is displayed. Click on the Organization Unit Type you require, and then click **OK**.
 5. In the **Multiplicity** field, type in the multiplicity you want.
 6. To add a Position as a member to the Organization Unit Type:
 - a. In the **Position Members** box, click . Type the name of the member in the **Label** field.
 - b. To specify that the member should be of a particular Position Type, click in the **Type** field and then click the  button that then becomes available.
The Select Type dialog box is displayed.
 - c. Click on the Position Type you require, and then click **OK**.
 7. In the **Multiplicity** field, type in the multiplicity you want.
- Click  to remove Position and Unit elements.
- Click the **Attributes** tab. Click  to add an attribute. Type in the name and select a type for the Attribute from the **Type** drop-down list. You can also add a default value for the Attribute, depending on your requirements.
- Click  to remove attributes.


Creating a Position Type

You can create a Position type in a schema.

Procedure

1. In the **Project Explorer**, expand the **Schema** folder for your Organization Model.
2. Right-click **Organization** and select **Add Child > Position Type**. A new Position Type is created.
3. Click the **General** tab in the **Properties** view. In the **Label** box, type the name of the Position Type.

Click the **Attributes** tab. Click  to add an attribute. Type in the name and select a type for the Attribute from the **Type** drop-down list. You can also add a default value for the Attribute, depending on your requirements.


Click  to remove attributes.

Creating an Organization Unit Relationship Type

You can create an Organization unit relationship type in a schema.

Procedure

1. In the **Project Explorer**, expand the **Schema** folder for your Organization Model.
2. In the **Schema** folder, right-click **Organization Unit Relationship Types** and select **Add Child > Organization Unit Relationship Type**. A new Organization Unit Relationship Type is created.
3. Click the **General** tab in the **Properties** view. In the **Label** box, type the name of the Organization Unit Relationship Type.

Click the **Attributes** tab. Click  to add an attribute. Type in the name and select a type for the Attribute from the **Type** drop-down list. You can also add a default value for the Attribute, depending on your requirements.


Click  to remove attributes.

Creating a Resource Type


You can create a resource type in a schema.

Procedure

1. In the **Project Explorer**, expand the **Schema** folder for your Organization Model.
2. In the **Schema** folder, right-click **Resource Types** and select **Add Child > Resource Type**.
A new Resource Type is created.
3. Click the **General** tab in the **Properties** view.
4. In the **Label** box, type the name of the Resource Type.

5. Click the **Attribute** tab.
6. Click  to add an attribute, and type in the name and select a type for the attribute from the **Type** drop-down list.

You can also add a default value for the attribute, depending on your requirements.

Click  to remove attributes.

Using a Schema in an Organization Model

This is an overview of the steps required to create the example shown as a Schema and then use it in an Organization Model.

An example of an organization unit that you might want to model as a Schema is shown below.

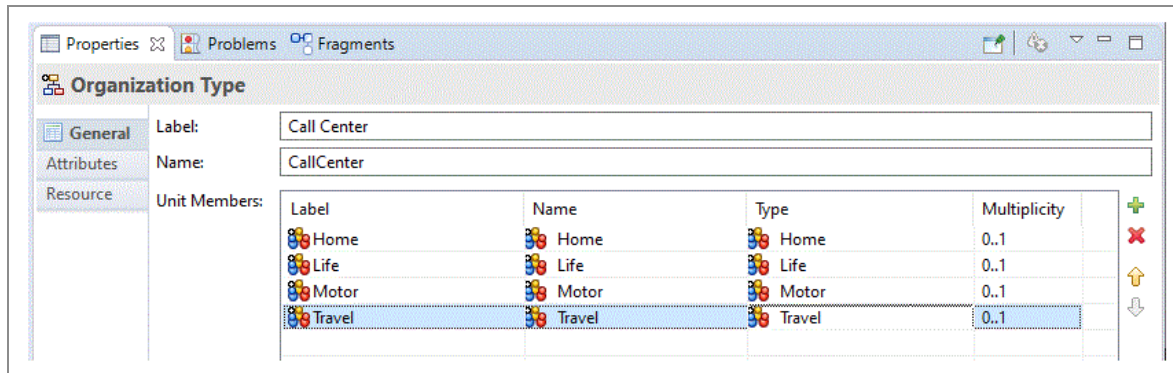
|

Procedure

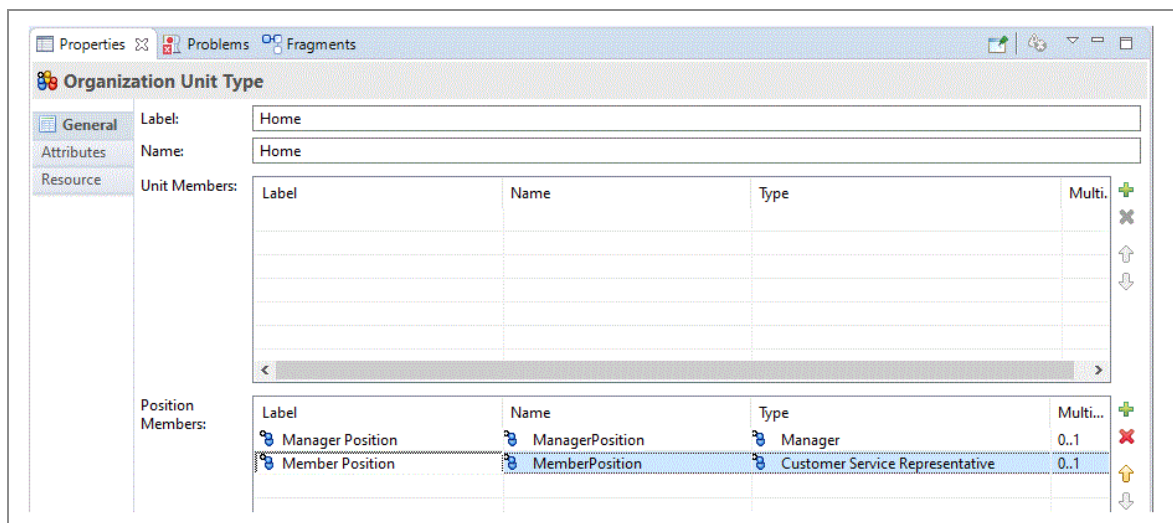
1. Create a project. On the Organization Model dialog box of the **New Project** wizard, uncheck the **Create default schema types** field.

The Schema for this project therefore contains only those types that you define.

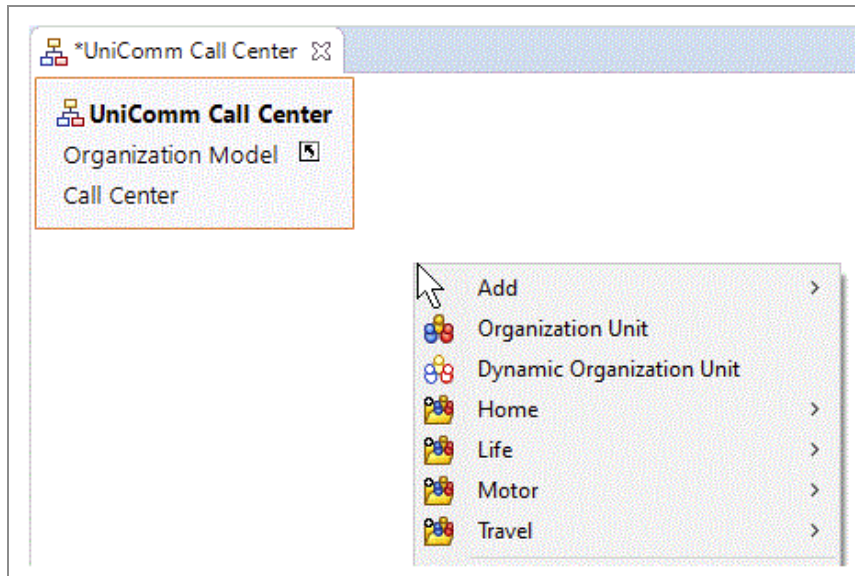
2. Create the following types in the Organization Schema:
 - An Organization Type called **Call Center**. See [Creating an Organization Type](#).
 - Organization Unit Types called **Home**, **Life**, **Travel** and **Motor**. See [Creating an Organization Unit Type](#).
 - Position Types called **Manager** and **Customer Services Representative**. See [Creating a Position Type](#).
3. For the **Call Center** Organization Type, specify the **Home**, **Life**, **Motor** and **Travel** Organization Unit Types as Unit Members.



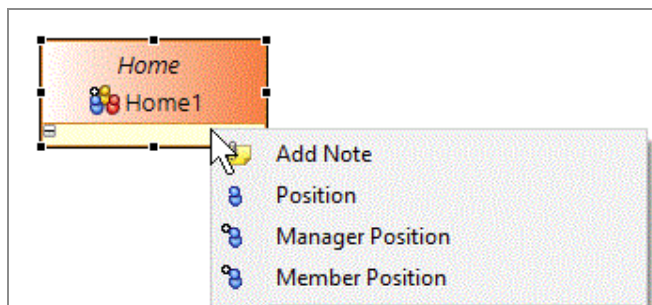
- For the **Home**, **Life**, **Travel** and **Motor** Organization Unit Types, specify the **Manager Position** and **Member Position** as Position Members.



- Create an Organization Model. Ensure that you uncheck the **Create default schema** check box.
- Create an Organization within that Organization Model.
- On the **General** tab in Properties view, change the Type of the Organization you have just created from a **Standard Organization Type** to the **Call Center Type**.
- In the Organization diagram editor, right-click the empty canvas. On the context menu that pops up, you can select **Home**, **Life**, **Motor** and **Travel** Organization Units, as shown in the image. You can also select an untyped Organization Unit.



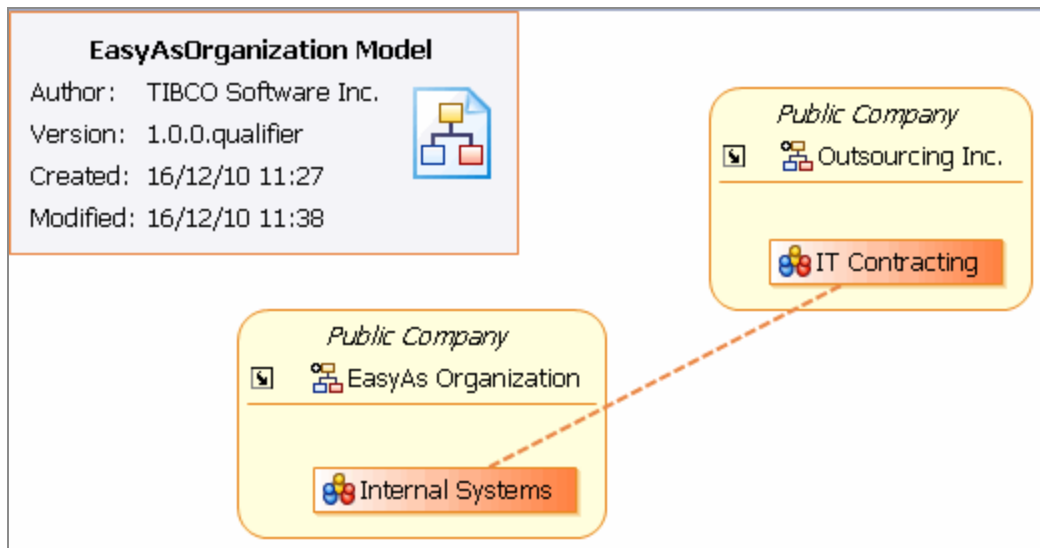
9. Create either a **Home**, **Life**, **Travel** or **Motor** Organization Unit.
10. If you right click on the Organization Unit you created, the **Manager** and **Customer Service Representative** Positions are displayed, as shown below.



Organization Modeler Diagram Editors

Two graphical editors are provided for producing organization diagrams, an Organization Model Editor (for the root Organization Model) and an Organization Editor (for the diagrams of each Organization included in the Organization Model).

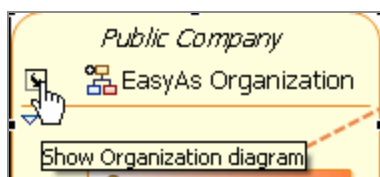
The Organization Model Editor shows a high-level view of the organization or organizations that you have created. The following example shows an Organization Model diagram which includes two organizations, one being your own organization and the second a representation of an external organization with which your organization has dealings, in this case an outsourcing company.



Note that an Association has been created between two organization units, one in each organization, that have a business relationship with each other.

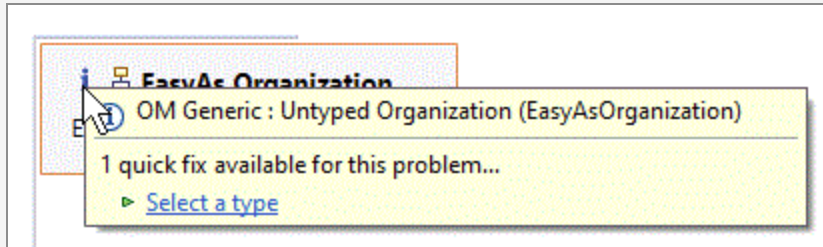
When you create an Organization Model Diagram, a new Organization is automatically created within the Organization Model. The Organization Editor for this default Organization opens automatically after you click **Finish** on the **Create New Organization Model** wizard. To view the parent Organization Model Editor you can navigate to it using the following methods:

- Double-click an existing Organization Model.om file in the Project Explorer.
- Right-click an existing Organization Model.om file in the Project Explorer, and select **Open** or **Open With > Organization Model Diagram Editing**.
- From the Organization Editor displaying any Organization in that Organization Model, click the shortcut arrow that is displayed on the badge.



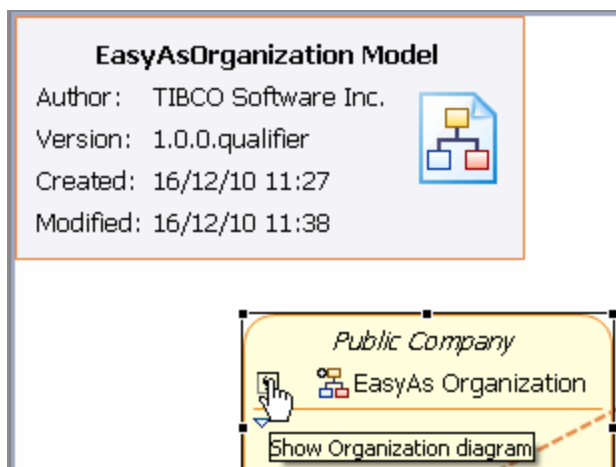
The Organization Editor shows a more detailed view of one Organization, including the Organization Units and Positions that it includes, and the relationships between them.

- i Note:** The small icon showing a letter **i** on the title badge of the Organization in this illustration indicates an information message. In this case it is displayed because the Organization has no Type assigned to it.



The Organization Editor opens:

- Automatically for the default Organization that is created when you create a new Organization Model Diagram.
- When you double-click the representation of an Organization in the Organization Model Editor.
- When you double-click an Organization in the Project Explorer.
- From the Organization Model Editor, when you click the shortcut arrow that is displayed on the representation of each Organization within the model.



Adding Objects in Organization Modeler Diagrams

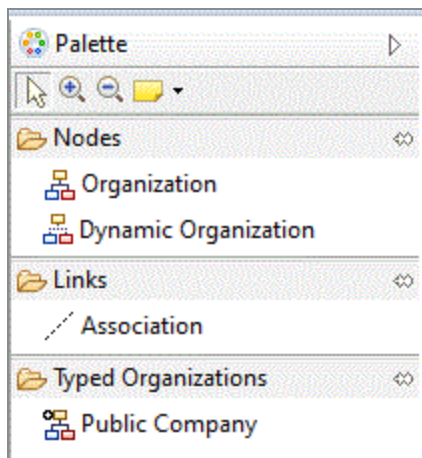
You can add objects to Organization Modeler diagrams in several ways.

- Using the tools on the diagram editor's palette
- Using the pop-up icons
- Using the context menu
- Using the Project Explorer

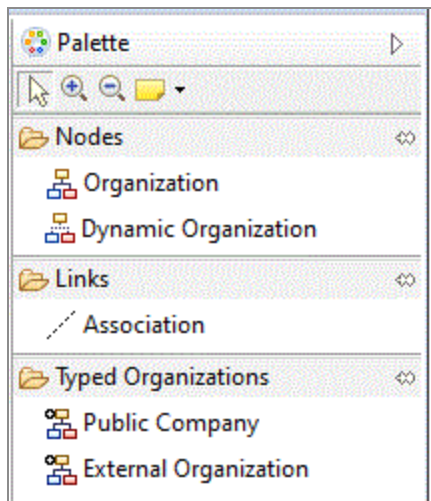
To add Objects in Organization Modeler Diagrams using the Palette

To add an object to an organization diagram, you can use the tools provided on the diagram editor's palette.

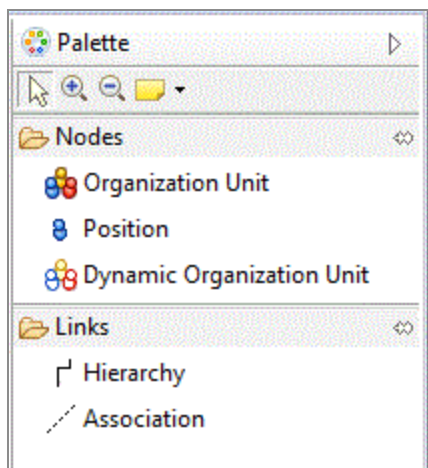
In the Organization Model Editor, the following tools are always available in the palette:



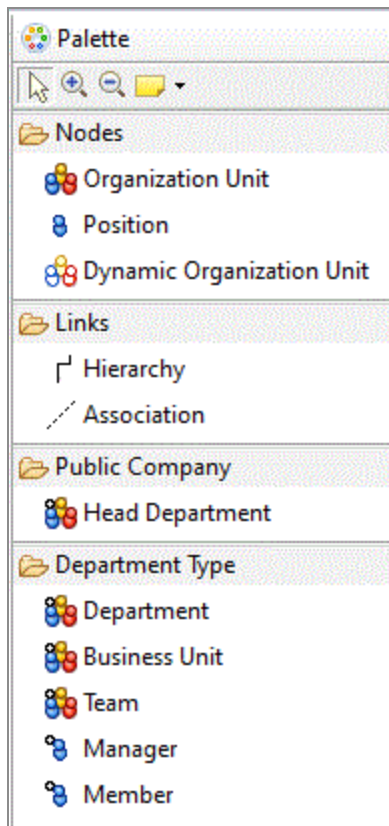
If you have applied a Schema to your organization model, and it contains any Organization Types, tools for adding these are also available. For example, if you have used the standard Schema but added a new Organization Type to it, both the standard Public Company Organization Type and the additional one is available in the palette, as in the following illustration.



In the Organization Editor, the following tools are always available in the palette:

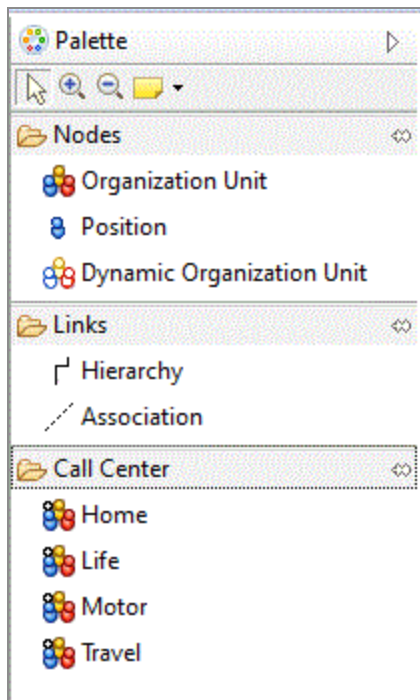


Additional tools are available if you have applied a Schema to your organization model. If you have applied the Standard Organization Type to the Organization, the palette contains the following tools for the elements that are defined in that Standard Organization Type:



i Note: Note that the Sub Unit element is not available from the palette.

Similarly, if the Organization that you are editing has a different or modified Type applied, tools are included in the palette for the elements that are defined for that Type. The following illustration shows an example. Here an Organization is used that has another Organization Unit Type defined.



The availability of these tools on the palette is dynamic; if you add a new Type to the Schema, it is immediately made available on the palette.

To use any of these tools, you can either:

- Select the required tool and drag-and-drop on to the diagram, or
- Click the required tool in the palette to select it, and click on the diagram.

Where you should drop, or click, depends on the object you are adding:

- For Organization and Organization Unit (including Types of Organization Unit), use the empty part of the diagram.
- For Position (including Types of Position), drop into or click the position compartment of an Organization Unit - that is, the empty space below the title bar.
- For Hierarchy or Association, drop into or click the Organization Unit where the connection is to start from, and drag the connection to the Organization Unit where it is to end.

When you add an Organization, Organization Unit, or Position object, the Label field on the title bar is automatically selected. Enter a name for the object.

Adding Objects in Organization Modeler Diagrams using the Project Explorer

You can add objects to the diagram editor from the **Project Explorer**.

1. Select the parent in the Project Explorer.
2. Right-click the parent and select **Add Child**.

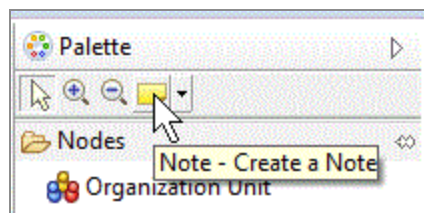
For example, to add an untyped Organization Unit you would right-click the parent Organization and select **Add Child > Organization Unit**. If you are using the Schema, the choices displayed include not just the basic Organization Unit but the Types of Organization Unit available.

Adding Notes and Labels

You can add notes or text labels to the organization diagram.

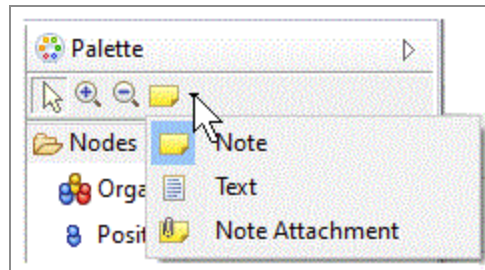
Procedure

1. Do one of the following:
 - Right-click an object in the editor and select **Add Note** from the pop-up context menu,
 - Right-click the background in the editor and select **Add - > Note** or **Add - > Text** from the pop-up context menu,
 - In the Organization Editor palette, select the Note tool in the upper part of the palette.



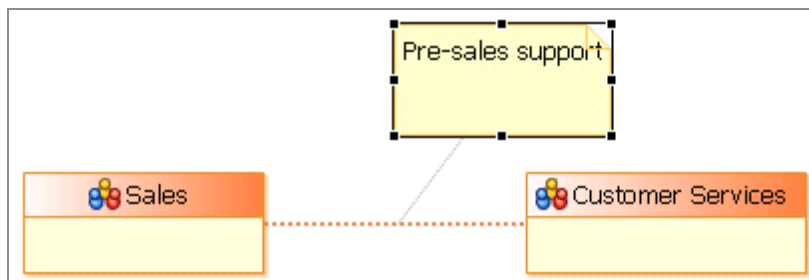
- Click in the Organization Editor.

Note: You can also select Text or Note Attachment from this menu. The icon for the tool changes to show the item you have selected.



Result

The following illustration shows a note being used to add explanation to an Association. A note attachment is drawn between the note and the object with which it is associated.

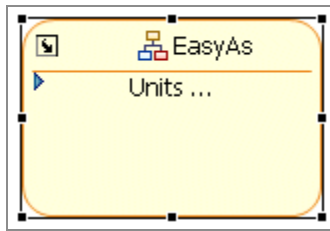


Tailoring the Appearance of Organization Modeler Diagrams

You can change the level of detail that the Organization Modeler diagrams display.

In the overview diagram, each Organization is by default shown including all the Organization Units that make it up, and the Hierarchies and Associations that link them. To hide the detail, click the downward-pointing arrowhead that is shown in the corner when you select the Organization.

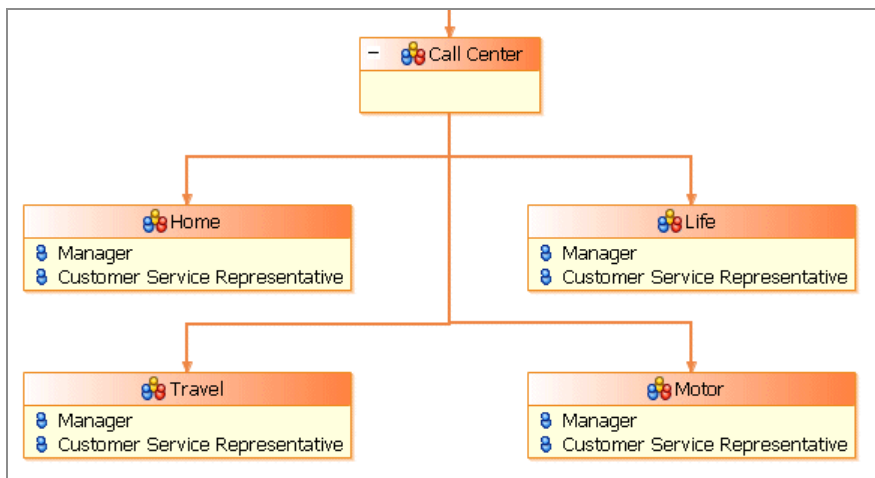
This action hides the contents of the Organization, and displays the collapsed version shown below.



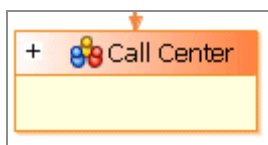
A solid right-pointing arrowhead is now displayed in place of the downward-pointing one when the Organization is selected. Clicking on this restores the display of the Organization's contents.

In the Organization Editor, the same arrowheads appear in Organization Units, and you can use them to hide or to display the Positions defined within those Organization Units.

Also in the Organization Editor, you can choose to display or to hide the hierarchy of Organization Units that depend on the selected unit. The following illustration shows an excerpt from an organization diagram, with several Organization Units dependent on the Call Center unit.



If you click the small minus sign in the title bar of the Call Center, the display changes to the following collapsed view.



Click the plus sign to restore the hierarchical view.

You can also:

- use the options in the **Diagram** menu to arrange the objects on the diagram editor

screen,

- right-click the background of the diagram and select **Arrange All** to make the diagram more ordered.

Organizations

An Organization represents the top layer of your Organization Model. Typically, you would create an organization to represent the relationship of a process to an application, and not to model the entire organizational structure of a company or division of a company. So for example, only certain roles might be involved in interacting with a Human Resources application, so those roles would be modeled in the organization associated with the process for that application.

You can create many Organizations in the Organization Model. This is useful because you might have an enterprise that has relationships with other enterprises. Part of the operation might have been outsourced to another company, for example. In this situation, you can create an Organization in your Organization Model for each enterprise that your Organization has a relationship with.

An Organization does not necessarily have to represent an organization or enterprise. It can represent a department or project. It might make sense for your business model to create a project as an Organization if the project is large enough and it consists of several Organization Units, for example.

An Organization can contain many Organization Units. See [Organization Units](#).

Creating an Organization

You can create an Organization within your Organization Model.

Before you begin

See [Creating an Organization Model](#)

**Note:**

If you want to create an organization that is likely to be replicated elsewhere - for example a Branch which consists of a similar structure in a number of geographical locations - then you might want to consider creating a Dynamic Organization. See [Dynamic Organizations](#).

Procedure

1. Activate the Organization Model Editor for your Organization Model.
2. Select the Organization tool in the Organization Model Editor palette, and click on the empty part of the Organization Model editor. This places an Organization in the Model.




Note: You can alternatively expand the Organization Model in the Project Explorer. Right-click **Organizations** and select **Add Child > Organization**. A new Organization is created.




Note: If you want to use a template of a typical public company as a starting point for your own organization, right-click **Organizations** and select **Add Child > Public Company**. This provides you with a number of sample department types which you can edit and add to your company (Department, Business Unit, Team) and Positions already defined for Manager and Member which you can also edit.


3. At this point the **Label** field of the Organization is automatically selected. Enter the label you require. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces. For example, if you put Head Office in the **Label** field, the **Name** is HeadOffice.
4. If the **Properties View** is not already displayed, right-click the organization you just created and select **Show Properties View**. The **Properties View** is displayed.
5. In the **Purpose** field on the **General** tab, and on the **Description** tab, you can enter some text to describe the purpose of the Organization and/or any supporting explanations, depending on your requirements.

Next to the **Type** field, click  to display the Select Type dialog box. You can change

the type for the Organization, depending on your requirements.


Next to the **Location** field, click  to display the Select Type dialog box. You can specify a Location for the Organization, depending on your requirements.

6. If you have assigned a Type to the Organization, the **Attribute Values** tab is displayed. Any attributes that are defined for this type of Organization are displayed here. Click the **Value** field next to each defined attribute to display a list of available values for that attribute.

 **Note:** The **Attribute Values** tab is available only if you have previously applied a Type.

What to do next

When you have created an organization, you can model it to contain a structure using Organization Units and Dynamic Organization Units which can be connected using Hierarchy links.

 **Note:** Association links are not supported in the runtime model. If you create an association a warning is displayed.

Hierarchies and Associations

In the Organization Editor, the relationships between Organization Units are denoted by two similar types of Organization Unit Relationship, Hierarchy and Association.

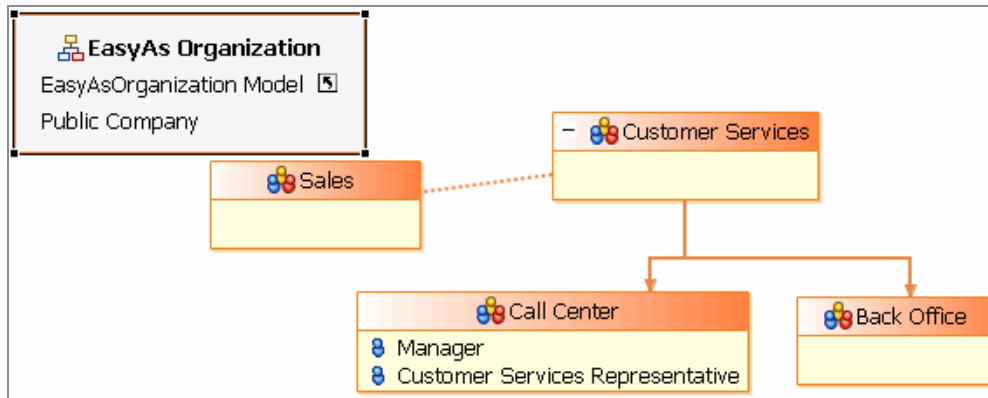
Hierarchy indicates a hierarchical relationship, for example between a department and its sub-departments.

Association can be used to indicate any sort of non-hierarchical relationship, depending on your requirements. For example, relationships between Organization Units might be based on factors such as resource or work allocation.

In the diagram editor, you can see whether a relationship is hierarchical or not because a Hierarchy is represented by a solid line with an arrowhead, an Association by a dotted line. This difference is controlled by one property of the Organization Unit Relationship, the **IsHierarchical** property. You can edit the value of this property to change a relationship

from Hierarchy to Association, or vice versa, simply by checking or unchecking a box on the **General** tab in the **Properties** view.

The following illustration shows a hierarchical relationship between the Customer Services Organization Unit and its sub-divisions Call Center and Back Office, and an Association between Customer Services and Sales.



Organization Units

An Organization Unit represents resources that are associated together because they fulfil a business need within the organization.

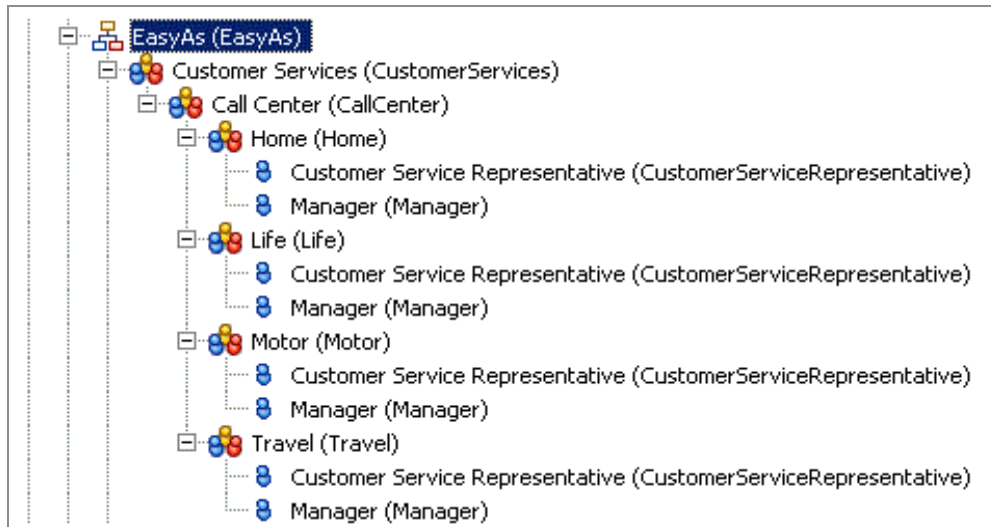
For example, an Organization Unit can be a department, project or location. Different Organization Unit Types are provided in the default Schema, and if it suits your organizational structure you can use them for different levels of organization or other different sorts of unit that might be present.

An Organization Unit is made up of Positions. A Position represents a set of responsibilities for a job of work to be performed in an Organization Unit. An Organization Unit can have many Positions.

Organization Units can be linked by relationships. These relationships can link Organization Units within the same Organization, or in different Organizations in the same Organization Model. (They do not link one Organization to another.)

Relationships can be hierarchical or otherwise. In the Organization Modeler diagram editor they are represented by Hierarchy or Association connections - see [Hierarchies and Associations](#).

An example of a Customer Services Organization Unit that contains Sub Organization Units and Positions is shown below.



You can specify privileges and system actions for an Organization Unit. An example of a privilege might be that the Accounts Department can authorize expenses up to \$500.

For each Organization Unit, there are various properties you can assign. For example, you can specify the location of the Organization Unit and how long it should exist for.

Creating an Organization Unit


You can create an Organization Unit within your Organization.


Procedure

1. In the Organization Model editor, double-click the organization in which you want to place an Organization Unit. The Organization Editor for that organization is opened.
2. Select the Organization Unit tool in the Organization Modeler palette, and click the empty part of the Organization diagram. This places an Organization Unit in the Organization.
3. At this point the **Label** field of the Organization Unit is automatically selected. Enter the label you require. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces.
4. If the **Properties View** is not already displayed, right-click the organization Unit you just created and select **Show Properties View**. The **Properties View** is displayed.
5. In the **Purpose** field on the **General** tab, and on the **Description** tab, you can enter some text to describe the purpose of the Organization Unit and/or any supporting


explanations, depending on your requirements.

6. If you are using the default schema, or if you have created your own schema and defined any Organization Unit Types, you can assign a Type to this Organization Unit by selecting it in the **Element** field.

Next to the **Location** field, click  to display the Select Type dialog box. You can specify a Location Type for the Organization.

To specify start and end dates for the Organization Unit, click  next to the date fields to display the calendars.

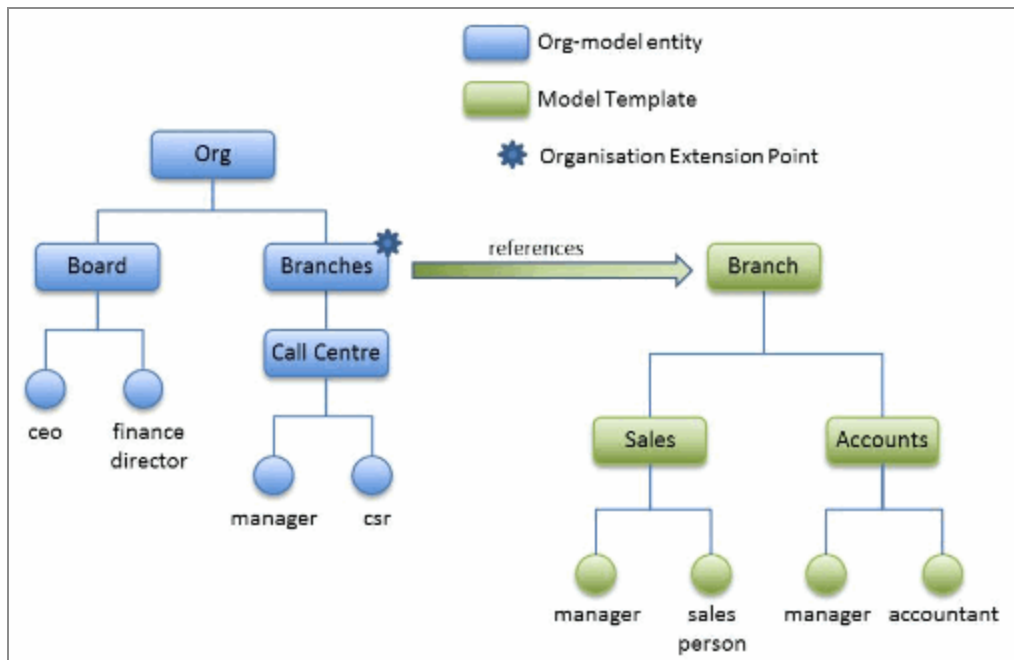
7. If you have applied a Type to the Organization Unit, the **Attribute Values** tab is displayed. Any attributes that are defined for this type of Organization Unit are displayed here. Click the **Value** field next to each defined attribute to display a list of available values for that attribute.

Click the **Privileges** tab. Click  to display the Select Type dialog box. You can specify the Privileges for the position.

Large Organizations with Branch Networks

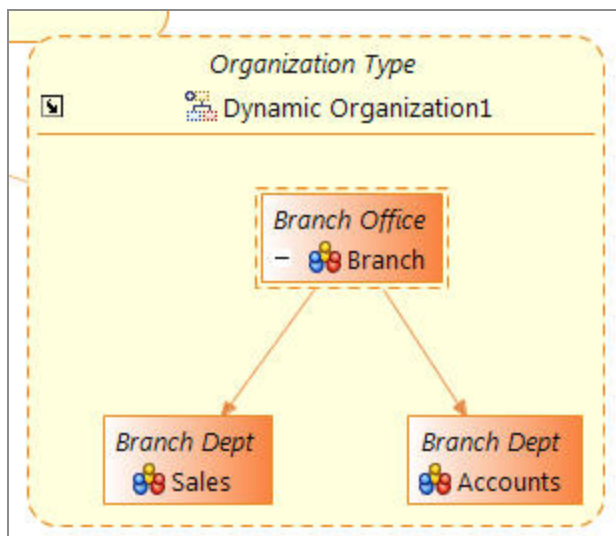
Large organizations with branch networks have particular requirements from their Organization Model structure as they want to replicate similar branch structures in multiple places. To do this, they can use a combination of static and dynamic organization models, meaning they do not need to model each branch individually when creating their organization model.

To model a large organization with a branch structure, you can create an organization containing extension points that point to Dynamic Organization Model templates that model a branch.



Dynamic Organizations

A dynamic organization is a common organizational pattern that can be referenced from a number of different organization models. It is used to represent repeating organization patterns, to avoid the need to model these individually everywhere they are required. For example, a bank might have a number of branches each containing a similar group of roles, so a dynamic organization could represent a branch and be used in different organization models.



The example above shows a dynamic organization representing a Branch. Dynamic organizations are referenced from dynamic organization units within an organization. See [Dynamic Organization Units](#).

Dynamic Organizations differ from static organizations in a number of ways:

- Only one root organization unit is allowed in a dynamic organization.
- Participants within a process (known as Dynamic Organization Participants) can be created as external references with a reference to an organization entity within a Dynamic Organization. See [Dynamic Organization Participants](#).
- When a Dynamic Organization Participant is assigned to a task you need to identify the correct instance of the Dynamic Organization to use to resolve this participant at runtime (as there might be a number of instances generated dynamically). This is done using [Dynamic Organization Identifier Mapping](#).

Creating a Dynamic Organization

You can create a Dynamic Organization within your Organization Model.

A Dynamic Organization is an organizational pattern that can be referenced from a number of different Organizations.

Procedure

1. Activate the Organization Model Editor for your Organization Model.
2. Select the Dynamic Organization tool in the Organization Model Editor palette, and click the empty part of the Organization Model editor. This places a Dynamic Organization in the Model.



Note: You can alternatively expand the Organization Model in the Project Explorer. Right-click **Organizations** and select **Add Child > Dynamic Organization**. A new Dynamic Organization is created.

3. At this point the **Label** field of the Dynamic Organization is automatically selected. Enter the label you require. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces. For example, if you put Head Office in the **Label** field, the **Name** is HeadOffice.
4. In the **Purpose** field on the **General** tab, and on the **Description** tab, you can enter

some text to describe the purpose of the Organization and/or any supporting explanations, depending on your requirements.

5. If you have assigned a Type to the Organization, the **Attribute Values** tab is displayed. Any attributes that are defined for this type of Organization are displayed here. Click the **Value** field next to each defined attribute to display a list of available values for that attribute.



Note: The **Attribute Values** tab is available only if you have previously applied a Type.

What to do next

Once you have created a Dynamic Organization, you can open it and model it by adding organization units and positions. See [Creating a Dynamic Organization Unit](#).

The Properties section of the Dynamic Organization is the same as that of an Organization but has an additional tab for the **Dynamic Organization Identifiers**. Dynamic Organization Identifiers are arbitrary strings that you can define. They have values assigned at runtime to identify an instance of the Dynamic Organization. The Dynamic Organization participant in a process that references an organization entity within this Dynamic Organization requires a mapping between process data (which provides the values for the identifiers at runtime) and these identifiers. See [Dynamic Organization Identifier Mapping](#).

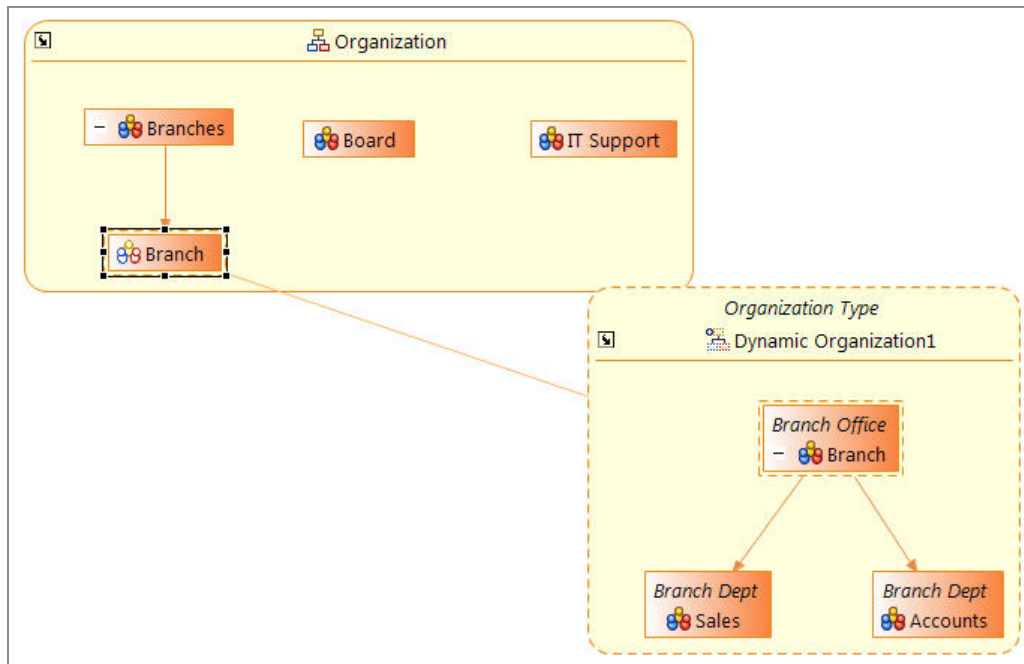
Dynamic Organization Units

Dynamic Organization Units are created within Organizations to reference an organizational pattern (Dynamic Organization). For example, they could be used where you want to refer to a number of branches with the same structure, without creating each individually within the Organization using the Organization Modeler.

See [Dynamic Organizations](#).

In the example below, the dynamic organization unit **Branch** has a reference to the Dynamic Organization called **Dynamic Organization1**. **Dynamic Organization1** consists of a hierarchy of organization units which make up a branch structure.

i Note: The entity **Branch** never gets deployed to BPM, and no entity of that name is generated in the organization model. This is only a visual marker that is used to represent the reference to the Dynamic Organization.



Creating a Dynamic Organization Unit

You can create a Dynamic Organization Unit within your Organization. This can be used to reference a Dynamic Organization. This in turn allows you to model a repeating organizational pattern such as a Branch in an organization.

Before you begin

See [Creating an Organization](#).

See [Creating a Dynamic Organization](#).

Procedure

1. In the Organization Model editor, double-click the organization in which you want to place a Dynamic Organization Unit. The Organization Editor for that organization is opened.

2. Select the Dynamic Organization Unit tool in the Organization Modeler palette, and click on the empty part of the Organization diagram. This places a Dynamic Organization Unit in the Organization.
3. At this point the **Label** field of the Dynamic Organization Unit is automatically selected. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces. The Label and Name are derived from the root organization unit of the referenced Dynamic Organization. This is not editable.
4. If the **Properties View** is not already displayed, right-click the Dynamic Organization Unit you just created and select **Show Properties View**. The **Properties View** is displayed.

What to do next

Dynamic Organization Identifiers are arbitrary strings that are defined on a Dynamic Organization by the user. These strings have values assigned at runtime to identify an instance of the Dynamic Organization. The Dynamic Organization participant in a process that references an organization entity within this Dynamic Organization has a mapping between process data (which provides the values for the identifiers at runtime) and these identifiers.

Position

A Position represents a set of responsibilities for a job of work to be performed in an Organization Unit.

A Position can only be assigned to one Organization Unit. It represents the responsibilities of the position within the context of the Organization Unit. The position of Administrator within the Human Resources Organization Unit might be different to the same position in the Finance Organization Unit, for example. An Organization Unit can contain many Positions.

You can specify an ideal number of people to have in a Position by specifying the **Number** field. This does not mean that you must have this number of resources in that position or that only resources with these requirements can fulfill this position. You can specify what the ideal requirements of the position are.

You can specify privileges and system actions associated with a Position. As an example of privileges the Accounts Organization Unit might be able to sign off expenses up to \$500 for example, but the Accounts Manager might be authorized to sign off expenses up to \$1000.

Whether positions inherit the privileges specified for an Organization Unit is determined by the run-time environment to which the Organization Model is exported. It is not defined in the Organization Modeler itself.

For each Position, there are various properties you can assign. For example, you can specify the location of the Position and how long it should exist for, and you can also specify Capabilities and Resources for a Position.


Creating a Position

You can create a Position within your Organization Unit.


Procedure

1. Select the Position tool in the Organization Modeler palette, and click the Organization Unit where you want to create the Position.
2. At this point the **Label** field of the Position is automatically selected. Enter the label you require. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces.
3. If the **Properties View** is not already displayed, right-click the Position you just created and select **Show Properties View**. The **Properties View** is displayed.
4. In the **Purpose** field on the **General** tab, and on the **Description** tab, you can enter some text to describe the purpose of the Organization and/or any supporting explanations, depending on your requirements.
5. If you are using the default schema, or if you have created your own schema and defined any Position Types, you can assign a Type to this Position by selecting it in the **Element** field.

Next to the **Location** field, click  to display the Select Type dialog box. You can specify a Location for the Position.

To specify start and end dates for the Organization, click  next to the date fields to display the calendars.

6. If you have applied a Type to the Position, the **Attribute Values** tab is displayed. Any attributes that are defined for this type of Position are displayed here. Click the **Value** field next to each defined attribute to display a list of available values for that attribute.

Click the **Capabilities** tab. Click  to display the Select Type dialog box. You can specify the Capabilities for the Position.

7. Do the same for the **Privileges** tab.

Groups

A Group represents a job type within your organization. It allows resources to be grouped by their job characteristics. These might be general job characteristics or characteristics that apply in a particular context.

For example, an Insurance Company might have groups for Loss Adjuster or Claims Handler. At the same time, you might have employees who have specializations in the Home Insurance sector. You might want to group these employees together based on these specializations, as shown below.



You do not assign Positions within the organization model, nor named users, to Groups when you are working with Organization Modeler. Users are assigned to groups at runtime using the Organization Browser.

Groups can be hierarchical, in other words a Sub Group can be created from a parent Group, or exist alongside each other. All members of a sub-group are members of the parent Group.

You can specify system actions, capabilities and privileges for Groups. A Group can have as many Capabilities as you want. This is useful for example, if you require resources with particular capabilities. You can group resources together based on their Capabilities.

For each Group, there are various properties you can assign. For example, you can specify a description and purpose for the Group.

Creating a Group

You can create a group within your Organization Model.


Procedure

1. In the **Project Explorer**, expand the folder for your Organization Model.
2. Right-click the **Groups** folder and select **Add Child > Group**. Right-click the Group and select **Rename**. Type the label of the Group you wish to create. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces.



Note: To add a sub-group, right-click the group where you want to create your sub-group and select **Add Child > Group**.

3. If the **Properties View** is not already displayed, right-click the Group you just created and select **Show Properties View**. The **Properties View** is displayed.
4. In the **Purpose** field on the **General** tab, and on the **Description** tab, you can enter some text to describe the purpose of the Organization and/or any supporting explanations, depending on your requirements.

Click the **Capabilities** tab. Click  to display the Select Type dialog box. You can specify the Capabilities for the position.

5. Do the same for the **Privileges** tab.

Participants

When a process designer creates a user task, they can define the participant(s) who performs the task at runtime. Participants are used to identify who or what performs an activity. For example, in a hiring process, a person (human participant) interviews the candidate and an email system (system participant) sends out an automatic follow-up reminder.

Participants are defined in the following ways:

- **statically**, by specifying one or more organizational entities - groups, positions, organization units or organizations.

- **dynamically**, by using runtime data to identify the required organizational entities.
- **using expressions**, by building a query that interrogates the organization model to identify the required organizational entities.

This flexibility allows a process designer to handle both simple and complex distribution scenarios without impact on the overall process design. For example:

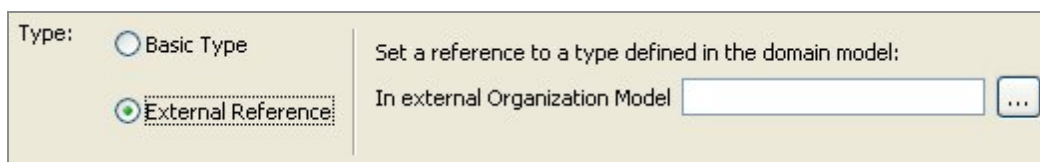
- offer a user task to all Customer Service Representatives.
- allocate a user task to an accountant if the value to be signed off is less than \$5000, but allocate it to an Accounts Manager if the value is \$5000 or more.
- allocate a user task to a single loss adjuster who holds at least level 2 motor insurance certification and is based in the Chicago office.

There are two types of participant:

- *user task participants* represent the users who perform the work defined in user tasks. These participants must be defined as *external references* in an organization model used by the process, not as *basic types*.
- *system participants* are used to identify a task that is performed by the system.

Participants from the Organization Model

You can create participants by creating an external reference to types defined in a model of your own organization or in a Dynamic Organization.



The screenshot shows a configuration window for a participant. On the left, under the 'Type:' label, there are two radio buttons: 'Basic Type' (unselected) and 'External Reference' (selected). To the right of these buttons, there is a section titled 'Set a reference to a type defined in the domain model:'. Below this title, there is a text input field labeled 'In external Organization Model' followed by a small square button with three dots (a dropdown menu).

You can create an external reference to the following parts of the organization model (static or dynamic):

- Positions
- Groups
- Organization Units
- Types from the default meta model

You can set a participant by using dynamic performer fields so that the work items generated appear in managed work lists.

Note: When a Dynamic Organization Participant is assigned to a task you need to identify the correct instance of the Dynamic Organization to use to resolve this participant at runtime. This is done using Dynamic Organization Identifiers which are mapped to process data.

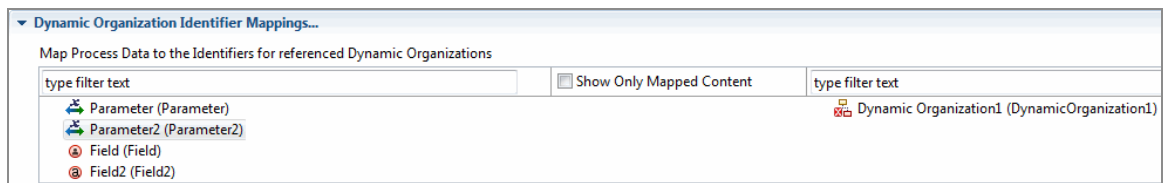
Dynamic Organization Identifier Mapping

When a Dynamic Organization Participant is assigned to a task you need to identify the correct instance of the Dynamic Organization to use to resolve this participant at runtime. This is done using Dynamic Organization Identifiers which are mapped to process data.

Note: The mappings are between the process data and the Dynamic Organization Identifiers of the referenced Dynamic Organization (**not** the Dynamic Organization).

Procedure

1. In the business process, select the **Work Resource** tab, and expand **Dynamic Organization Identifier Mappings....**



2. Map your process data (data fields and parameters) to a Dynamic Organization Identifier (which you set up when you created the Dynamic Organization Model). See [Dynamic Organizations](#) and [Creating a Dynamic Organization](#).

What to do next

A Dynamic Organization declares its Identifier Fields. These are arbitrary fields that are used to uniquely identify a generated instance of the Dynamic Organization at runtime.

When a Dynamic Organization is assigned to an Extension Point, those Identifier Fields must be mapped/assigned to named LDAP Attributes. This is done after deployment.

A generated instance of a Dynamic Organization takes its Identifier values from those named Attributes; of the LDAP Entry from which it originates.

A Dynamic Organization Participant carries values for the Dynamic Organization Identifiers. These values are derived from process data (data fields and parameters) mapped to those Dynamic Organization Identifiers. With this information the User Task can identify the instance of the Dynamic Organization in which the Participant can be found.

Dynamic Organization Participants

Dynamic Organization Participants can be created as external references to an organization entity within a Dynamic Organization. These participants can be assigned to a task like any other participant.

A Process Task can reference multiple participants. However, if any of those participants are Dynamic Organization Participants, then all Dynamic Participants of that task must reference entities within the same Dynamic Organization.

When a Dynamic Organization Participant is assigned to a task you need to identify the correct instance of the Dynamic Organization to use. You do this using **Dynamic Organization Identifiers**. The process data fields and parameters that provide these values have to be mapped to these identifiers.

Capabilities and Privileges

Capabilities represent the skills that are available within an organization, for example language skills or possession of a professional qualification. Privileges represent the authority that an Organization Unit, Position or Group can have within an organization, such as the authority to approve expenditure up to a defined amount.

Once a Capability has been created it can be assigned to Groups and Positions. Capabilities assigned to a Group or Position represent "entry criteria" for that group or position: it is necessary to have that capability in order to be a valid member of the group or position.

You can assign Privileges to Groups, Positions and Organization Units. Privileges might be based on budgets and spending, work/product approval, resource allocation, or other

factors. By specifying Privileges, you can see the chains of authority throughout your organization.

You can also associate a **system action** with a privilege. System actions are tasks that might need to be authorized in some way. Like Privileges, they can be assigned to Organization Units, Positions or Groups. At run time, only users who hold the associated privilege are then allowed to execute that system action. See [System Actions](#).

Whether positions inherit the privileges specified for an Organization Unit is determined by the run-time environment to which the organization model is exported. It is not defined in the Organization Modeler itself.

You can create categories for Capabilities and Privileges. Categories are a way of grouping your Capabilities and Privileges into meaningful units. You could create a Category for Language and then create Capabilities within that category for each language spoken in your organization; for example French, Polish, Chinese, and so on.

You can add extra information about a Capability or Privilege by assigning a Qualifier. For example, you might have created a Capability that represents an exam qualification but for a particular Position you might want to qualify that Capability by specifying a grade as well. To add qualifiers, check the **Has qualifier** box on the **General** tab in the Properties view. The **Qualifier** tab then becomes available. On that tab you can name and describe the qualifying information, and specify a data type that determines its allowable values, and if required specify a default value. For a description of the data types that qualifying information can have, see [Data Types](#). Then, when you assign the Capability or Privilege to a Group or Position, you can specify the value of the Qualifier by typing it into the **Value** field on the **Capabilities** or **Privileges** tab.

You can assign other properties to Capabilities and Privileges. For example, you can specify a purpose and/or description for the Capability or Privilege.

Creating Capabilities

You can add Capabilities to your Organization Model.

Procedure

1. In the **Project Explorer**, expand the folder for your Organization Model.
2. You can either:
 - Right-click the **Capabilities** folder and select **Add Child > Capability**. A new

Capability is created.

- Right-click a Category and select **Add Child > New Capability**. A new **Capability** is created.
3. Right-click the Capability you just created and select **Rename**.
 4. Type the label of the Capability you wish to create. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces.
 5. If the **Properties View** is not already displayed, right-click the Capability you just created and select **Show Properties View**. The **Properties View** is displayed.
 6. Click the **General** tab.
 7. You can specify a type for your capability by doing the following:
 - a. Select the **Has qualifier** check box. The **Show qualifier ...** option is displayed.
 - b. Click the **Show qualifier ...** option. The **Qualifier** tab is displayed.
 - c. Click the **Qualifier** tab. In the **Type** field, select an available type from the drop-down list.
 - d. If required, specify a default value for the qualifier.
See [Capabilities and Privileges](#) for more information on the types you can specify.
 8. In the **Purpose** field on the **General** tab, and on the **Description** tab, you can enter some text to describe the purpose of the Privilege and/or any supporting explanations, depending on your requirements.

Creating Capability Categories

You can add Capability Categories to your Capabilities in your Organization Model.

Procedure

1. In the **Project Explorer**, expand the folder for your Organization Model.
2. Right-click the **Capabilities** folder and select **Add Child > Capability Category**. A new **Category** is created.
3. Right-click the Category you just created and select **Rename**.
4. Type the label of the Category you wish to create. The **Name** field is automatically

filled with the same text as the **Label**, but without any internal spaces.

Creating Privileges

You can add Privileges to your Organization Model.

Procedure

1. In the **Project Explorer**, expand the folder for your Organization Model.
2. You can either:
 - Right-click the **Privileges** folder and select **Add Child > New Privilege**. A new **Privilege** is created.
 - Right-click a Category and select **Add Child > New Privilege**. A new **Privilege** is created.
3. Right-click the Privilege you just created and select **Rename**.
4. Type the label of the Privilege you wish to create. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces.
5. If the **Properties View** is not already displayed, right-click the Privilege you just created and select **Show Properties View**. The **Properties View** is displayed.
6. Click the **General** tab.
7. You can specify a type for your privilege by doing the following:
 - a. Select the **Has qualifier** check box. The **Show qualifier ...** option is displayed.
 - b. Click the **Show qualifier ...** option. The **Qualifier** tab is displayed.
 - c. Click the **Qualifier** tab. In the **Type** field, select an available type from the drop-down list.
 - d. If required, specify a default value for the qualifier. See [Capabilities and Privileges](#) for more information on the types you can specify.

i Note: Organization model deployments to the same major version are always additive.

This means that changes in the qualifier value of privileges reflects every previous value you set. So, if you set the qualifier for Position 'Tester' with Privilege 'All' to 10 and deploy the organization model, and then change the qualifier to 11 and redeploy it (as a different minor version), Position 'Tester' now has two assignments of privilege 'All'; one with a qualifier of 10, and with a qualifier of 11.

8. In the **Purpose** field on the **General** tab, and on the **Description** tab, you can enter some text to describe the purpose of the Privilege and/or any supporting explanations, depending on your requirements.

Creating Privilege Categories

You can add Privilege Categories to your Privileges in your Organization Model.

Procedure

1. In the **Project Explorer**, expand the folder for your Organization Model.
2. Right-click the **Privileges** folder and select **Add Child > Privilege Category**. A new **Category** is created.
3. Right-click the Category you just created and select **Rename**.
4. Type the label of the Category you wish to create. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces.

Locations

Locations represent the locations that your organization uses. A location might be a place, an office building, or even a room in an office, depending on your requirements.

You can create locations in the Organization Model and then assign them to Organizations, Organization Units and Positions. You can see how your organization is distributed across its various locations.


You can also specify other properties for Locations.


Creating a Location

You can create a Location for an Organization Model.

Procedure

1. In the **Project Explorer**, expand the folder for your Organization Model.
2. Right-click the **Locations** folder and select **Add Child > Location**.
3. Right-click the Location and select **Rename**.
4. Type the label of the Location you wish to create. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces.
5. If the **Properties View** is not already displayed, right-click the Location you just created and select **Show Properties View**. The **Properties View** is displayed.
6. In the **Purpose** field on the **General** tab, and on the **Description** tab, you can enter some text to describe the purpose of the Location and/or any supporting explanations, depending on your requirements.

Next to the **Type** field, click  to display the Select Type dialog box. You can change the type for the Location, depending on your requirements.

To specify start and end dates for the Location, click  next to the date fields to display the calendars.

7. If you have assigned a Type to the Location, the **Attribute Values** tab is displayed. Any attributes that are defined for the Type are displayed here. Click the **Value** field next to each defined attribute to display a list of available values for that attribute.

Resources

Resources are used to specify items such as people, equipment or buildings. You can use Resources within your organization model to identify such items and define how they relate to Organization Units and Positions.

A resource can be assigned a Resource Type if Resource Types are defined in your schema, and has any Attributes defined for that type. For example, you might want to define attributes of **FirstName** and **Surname** for a Resource that represents an employee, or **TelephoneNumber** for both an employee and for another type of Resource that represents a meeting room. See [Schemas](#) for more information about Schemas and Resource Types.

The following Resource Types are provided in the default Schema:

- Durable Resource Type. You can use this for items like buildings which have a long lifespan.
- Consumable Resource Type.
- Human Resource Type. A Human Resource Type must always be present in the Schema, and there can be only one Human Resource Type present in the schema.

See [Creating a Schema](#) for details of creating a Schema with or without using the standard Types.

Resources and TIBCO Business Studio - BPM Edition

Any Resources that you have defined are not exported to TIBCO Business Studio - BPM Edition, and are not used at runtime.

i Note: The only exception to this is that the Human Resource type, and any attributes defined for it, are exported to TIBCO Business Studio - BPM Edition.

Therefore, a Human Resource type must always be present in the Schema. Even if you choose not to include the standard Types in your Schema, a Human Resource type is always present, and it cannot be deleted from the Schema.

Resources are mapped to entities in the organization model using the *Organization Browser*. You can:

- Map BPM Resources, which represent actual user IDs obtained from an LDAP-compliant directory, to Positions and Groups in the organization model,
- Map attributes assigned to the organization model's Human Resource Type to LDAP attributes in the LDAP sources you have used.

i Note: This means that any sort of information that you have reflected in defining Resources within Organization Modeler, including for example if you have used Resources to represent anything other than users, is not represented in ActiveMatrix BPM after deployment.

Creating a Resource


You can create a Resource for your Organization Model.

Procedure

1. In the **Project Explorer**, expand the folder for your Organization Model.
2. Right-click the **Resources** folder and select **Add Child > Resource**. A new **Resource** is created.
3. Right-click the Resource and select **Rename**.
4. Type the label of the Resource you wish to create. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces.
5. If the **Properties View** is not already displayed, right-click the Resource you just created and select **Show Properties View**. The **Properties View** is displayed.

Note: For a complete description of all the properties you can configure for a Resource.

6. In the **Purpose** field on the **General** tab, and on the **Description** tab, you can enter some text to describe the purpose of the Resource and/or any supporting explanations, depending on your requirements.

Next to the **Type** field, click  to display the Select Type dialog box. You can change the type for the Resource depending on your requirements.

7. If you have assigned a Type to the Resource, the **Attribute Values** tab is displayed. Any attributes that are defined for the Type are displayed here. Click the **Value** field next to each defined attribute to display a list of available values for that attribute.

See [Attributes](#).

Queries

Queries are a way of identifying an entity within the organization model to use as a suitable participant for a task in a business process.

Note: The use of queries is not currently supported in the Organization Model.

You can specify queries either:

- As strings of text. These are not checked or validated within Organization Modeler. A business analyst might enter the query as free text, which a solution designer can implement when the analysis is implemented as a process.
- As Resource Query Language (RQL). This is a scripting language which you can use to write queries specifying which entity within the organization model should be selected for work allocation. See [Resource Query Language](#).

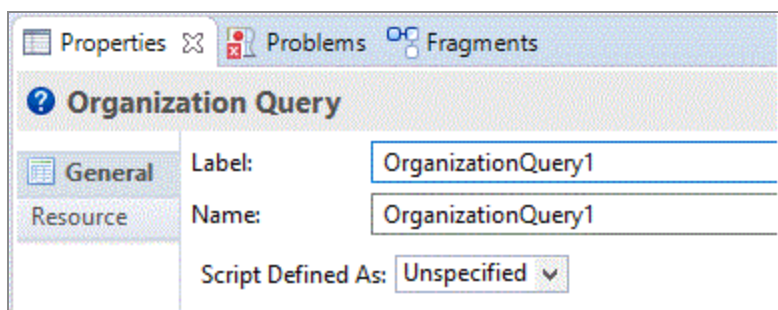
Note: You should consider using Dynamic Organization Participants as the alternative to RQL as it is more efficient in most situations. See [Dynamic Organization Participants](#)

Creating Queries

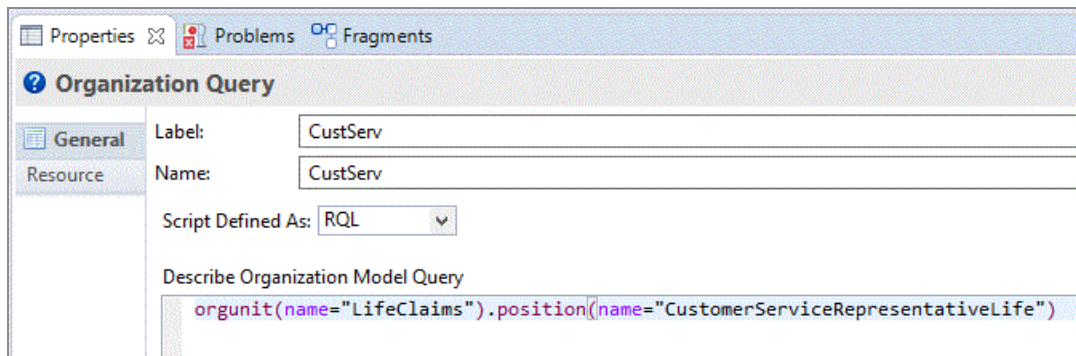
You can create a Query for an Organization Model.

Procedure

1. In the **Project Explorer**, expand the folder for your Organization Model.
2. Right-click the **Queries** folder and select **Add Child > Organization Query**. A new **Organization Query** is created.



3. Right-click the Query and select **Rename**.
4. Type the label of the Query you wish to create. The **Name** field is automatically filled with the same text as the **Label**, but without any internal spaces.
5. On the **General** tab of the Properties View, select either **Free Text** or **RQL** from the dropdown list in the **Script Defined As** field. An input field is displayed.



6. Type the Query into the **Describe Organization Model Query** field. You can either enter free text that describes the solution designer what you intend the query to accomplish, or use the structure of RQL to enter a query.

System Actions

System actions are actions that a user might wish to perform at runtime but that need to be authorized, or need to be restricted to users with a certain level of authority.

These actions might include, for example re-allocating work-items, skipping work-items, viewing another user's work list, or administering resources.

This authorization is implemented by associating system actions with **privileges** within Organization Modeler. See [Capabilities and Privileges](#) for more details about privileges.

In Organization Modeler:

- For the Organization Model, the **System Actions** tab of the **Properties** view lists all the system actions that are available, and any privileges with which each is associated.
- For Organization Units, Positions and Groups, the **System Actions** tab of the **Properties** view lists the subset of system actions that are available for that class of entity, and any privileges with which each is associated.

In all these cases you can associate a system action with one or more privileges. As described in [Capabilities and Privileges](#), privileges can have qualifiers which determine the level of the privilege. At run time, only users who hold the associated privilege with any required level of qualifier (or if more than one privilege is associated with a particular system action, users who hold **all** the associated privileges) are then allowed to carry out that system action.

At runtime, a list of system actions and privileges is maintained, as defined in the organization model, and thus determines whether a user is authorized to carry out a particular action.

System Actions Reference

System actions are shown at two levels: the organization model level and at the "scoped" level (for groups, organization units, and positions).

The tables below show the system actions that can be associated with privileges at those two levels. The tables are further divided into categories.

Organization Model-Level System Actions

Administration	
System Action	Provides Access To
System Administration	<p>Users with this system action have access to some of the administrative actions like deployment/undeployment, shared resource management and configuration management. User's with this System Action can access Administrative UI.</p> <p>For any other action, please refer to the other system actions.</p> <p>.</p>
Application Development Platform	
System Action	Provides Access To
Application Development	<p>Various functions:</p> <ul style="list-style-type: none"> • Access and launch App dev capability • Create or upload a user interface(UI) application • Manage the available UI applications, that is, publish and edit the applications
Case Document	Delete case documents

Application Development Platform	
System Action	Provides Access To
Administration	
Case Document User	Upload and view case documents

Audit Trail	
System Action	Provides Access To
Query Audit	Various functions: <ul style="list-style-type: none"> • Store and retrieve comments for a process instance • Generate an audit trail for cases, process instances, and work items • Retrieve audit data in tabular format for charting tools
Write Audit	Protects against users calling a REST API to create custom audit entries

Business Services	
System Action	Provides Access To
List Business Services	Various functions: <ul style="list-style-type: none"> • List business services • List business service categories • List page flows • List case actions • Query business services • Query business service categories

Case Data	
System Action	Provides Access To
Create / UpdateCase Data	Create cases in the case data store Update, link, and unlink cases
Delete Case Data	Delete cases from the case data store
Read Case Data	Find, read, and navigate cases
Organization Model	
System Action	Provides Access To
Browse Model	Browse the organization model
Create Resource Admin	Create resources and edit resource information
Delete LDAP Admin	Delete LDAP containers
Delete Resource Admin	Delete resources
Export LDAP Admin	Export LDAP container and resource mapping information
Import LDAP Admin	Import LDAP container and resource mapping information
LDAP Admin	Various functions: <ul style="list-style-type: none"> List LDAP containers

Organization Model	
System Action	Provides Access To
	<ul style="list-style-type: none"> • Save LDAP container • Execute LDAP queries • List LDAP connections • List candidate resources • Set extension points and candidate queries
Organization Admin	<p>Set extension points (both the "LDAP Admin" system action and this system action give access to set extension points)</p> <p>This system action also allows the user to see all organizations, even if organization relationships are set up.</p>
Read Parameters	Controls viewing and editing resource attributes in the Organization Browser
Read Push Destinations	Controls viewing and editing push destinations in the Organization Browser
Resolve Resource	Find and list resources
Resource Admin	<p>Various functions:</p> <ul style="list-style-type: none"> • Create, update, and delete resource • List candidate resources • Set candidate queries • Purge deleted resources • Delete LDAP container
User Settings	List, retrieve, save, or delete user settings that are persisted on the server
Write	Controls editing resource attributes in the Organization Browser

Organization Model	
System Action	Provides Access To
Parameters	
Write Push Destinations	Edit push destinations for an organization or resource
Process Management	
System Action	Provides Access To
Bulk Cancel / Purge Process Instance	Cancel process instances Purge completed process instances automatically from the system
Bulk Resume / Suspend Process Instance	Suspend process instances Resume suspended process instances
Cancel / Purge Process Instance	Cancel a process instance Purge completed process instances automatically from the system
Handle Process Migration	List, set, unset, or clear migration rules
Query Process	Query and list process instances and templates
Resume / Suspend Process Instance	Suspend process instances Resume suspended process instances Fix halted process instances
Start Process	Start an instance of a process

Work Management	
System Action	Provides Access To
Auto Open Next Work Item	Open the next work item automatically after the previous one is closed
Change Allocated Work Item Priority	Change priority of an allocated work item
Change Any Work Item Priority	Change priority of any work item
Close Other Resources Items	Close work items that are currently allocated to other users
Open Other Resources Items	Open work items that are currently allocated to other users
Pend Work Item	View "pended" work items, that is, work items that have been hidden until a specified date/time, or period of time has expired
Reallocate To Offer Set	Reallocate the work item to the offer set
Reallocate Work Item To World	Reallocate the work item to all users
Set Resource Order Filter Criteria	Retrieve or set filter and sort criteria for work item lists for a resource
Skip Work Item	Skip a work item in a work list
View Global Work List	Retrieve a work item list for ALL resources, or retrieve a work list containing work items associated with a specific global reference
View Work List	Various functions: <ul style="list-style-type: none"> Retrieve a work list for an organizational entity

Work Management	
System Action	Provides Access To
	<ul style="list-style-type: none"> • Retrieve list of all public work views • Add or remove resources from a public view • Delete a work list view • Lock or unlock a work list view
Work Item Allocation	Allocate or unallocate a work item

Scoped System Actions

The following system actions are considered "scoped", that is, they are set on specific groups, organization units, and positions using the Organization Modeler. The scoped system actions are specifically used to control access to, and functions from, supervised work views.

Scoped System Actions

Work Management	
System Action	Description
Close Other Resources Items	Close work items that are currently allocated to other users
Open Other Resources Items	Open work items that are currently allocated to other users
Reallocate To Offer Set	Reallocate the work item to the offer set
Reallocate	Reallocate the work item to all users

Work Management	
System Action	Description
Work Item To World	
Set Resource Order Filter Criteria	Retrieve or set filter and sort criteria for work item lists for a resource
View Work List	Controls access to supervised work views: <ul style="list-style-type: none"> • To create a supervised work view for an organizational entity, you must possess this system action on the group, organization unit, or position. • To create a supervised work view for an individual resource, you must possess this system action on a specific position to which the resource has been mapped.
Work Item Allocation	Allocate or unallocate a work item

Schemas

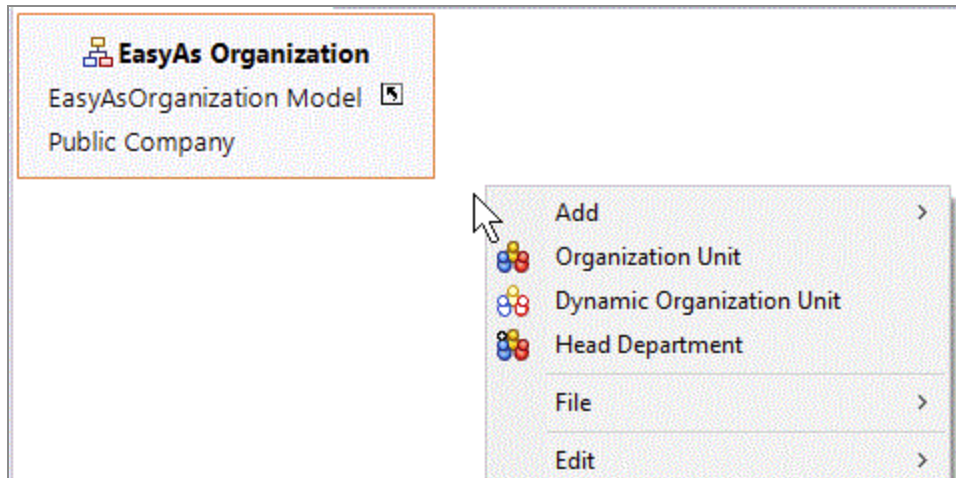
You can create an embedded Schema for use in the definition of your organization model, by using the Organization Modeler.

By using a Schema, you can specify Types that you intend to use for certain elements within your organization. By setting up these types, you can create a generic model or template for your organization model. By defining different Types of Organization, Organization Unit, Position, Resource and so on, and choosing and defining Attributes for those Types, you can extend the model provided by the default schema until it is a close representation of the components that make up the organization that you wish to model.

The containment relationships between components can also be defined in the schema's Types. For example, from an Organization element that you have assigned an Organization

Type, Organization Modeler can create a child Organization Unit of a defined type, because the relationship between the two Types is described in the schema.

An example of this is shown below. One type of Organization - the Public Company organization type - is defined in the standard Schema. One type of Organization Unit element - Head Department - is defined as a Member of that Organization Type. If your Organization is defined as a Public Company, and you right-click in the Organization Editor to add an Organization Unit, you can select either an un-Typed Organization Unit, or a Head Department unit.



You can create specific organizational structures in a Schema, so that when you need to create an instance of that structure in your organization model, you can use the structures defined in the Schema. For example, you might want to create an organization structure for temporary projects that consists of specific Organization Units and Positions. You can create this structure in a Schema then create an instance of a temporary project based on the Schema.

The benefits of Schemas are:

- You can use Schemas as templates for organization models that need to be exchanged between systems. Some extended enterprises refer to organizational concepts differently, for example, "region" vs. "district". By specifying types for particular components, you create a generic schema for your Organization Model that enables it to be exchanged with other systems.
- You can create a language or vocabulary to express the organization concepts that are applicable to your organization.
- You can create additional semantic data for your existing organization concepts. For example, for a Position whose type is Manager, you could create an attribute called

bonus to specify what type of bonus should be given.

When you use a Schema within the Organization Modeler, you can either:

- Use the default Schema, which comes delivered with standard Types for the schema elements,
- Create an empty Schema which does not contain any standard Types (except for the Human Resource Type, which must always be present).

You can also decide to use a mixture of the two approaches by adding your own types to the default schema.


Organization Modeler Default Schema

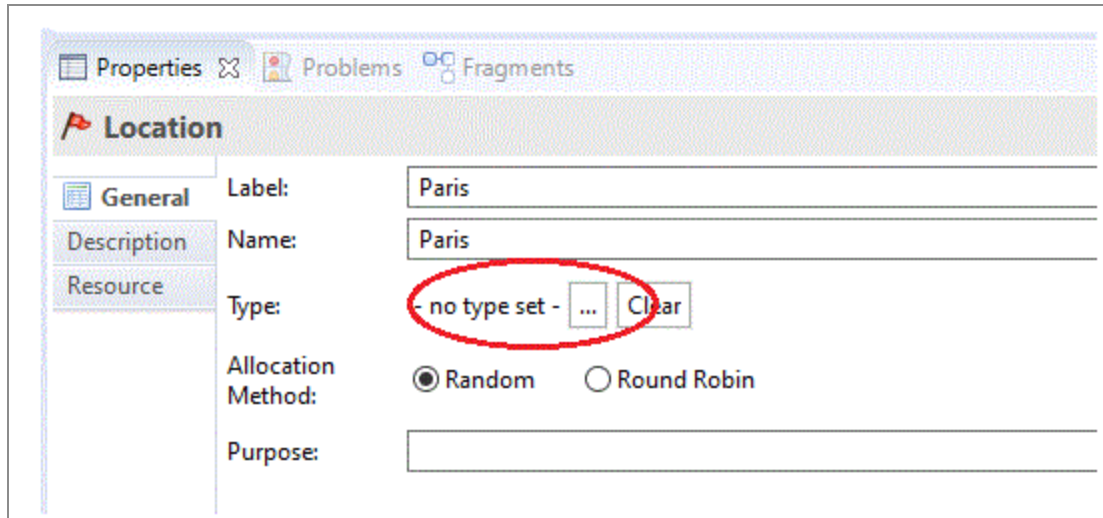
When you create certain components within your Organization Model, the Organization Modeler allocates standard types to these components. Not all the components within the Organization Modeler have a standard type.

The following table describes the components that have types provided in the default schema, what they are, and what members each of those types contains:

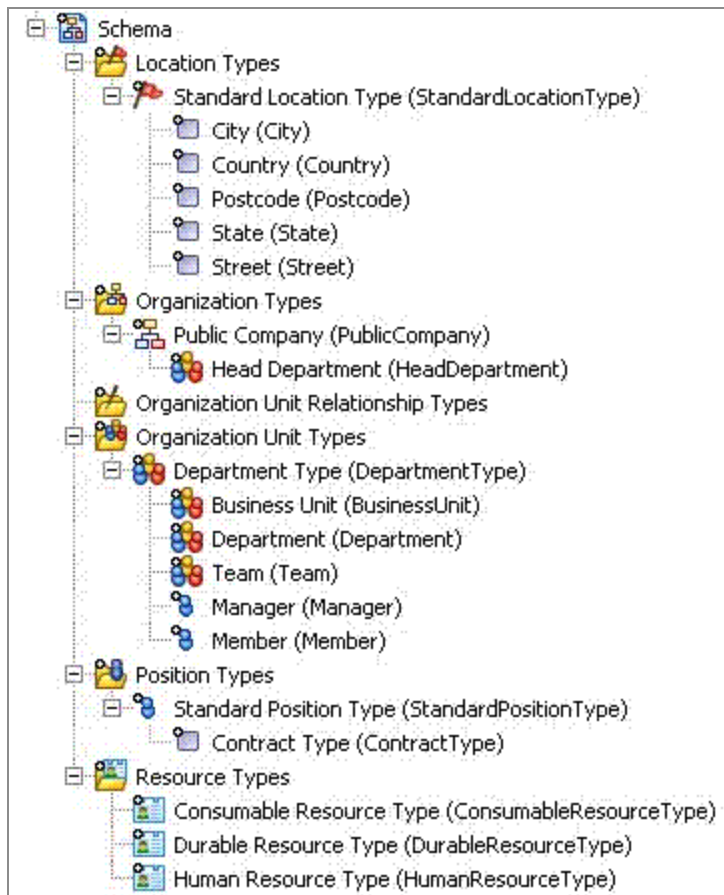
Component	Standard Type	Unit Members	Position Members
Organization	Public Company	Head Department	
Organization Unit	Department Type	Department Business Unit Team	Manager Member
Position	Standard Position Type		
Location	Standard Location Type		
Resource	Human Resource Type Consumable Resource Type Durable Resource Type		

When you create an Organization Model, you can decide whether to use the Standard Types provided by the Organization Modeler, create new Types, or create your own Schema, depending on your requirements.

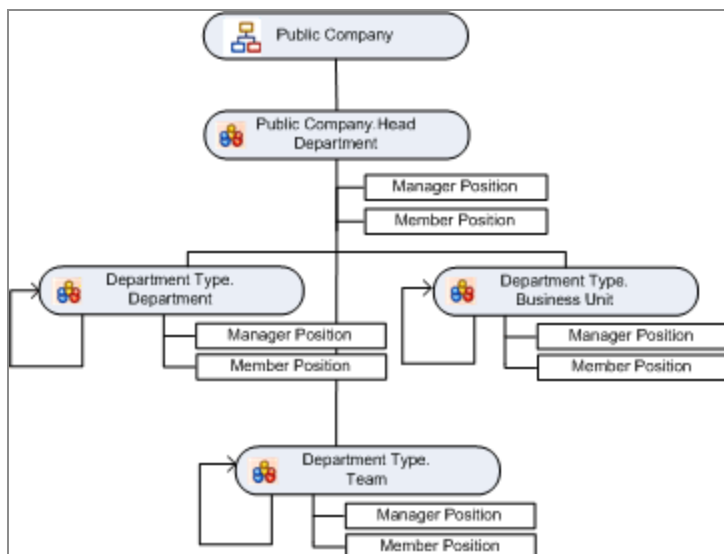
When a component that has a standard type defined for it is created, you can assign the type in the **Properties View** for the component. By default no type is assigned, as in the following illustration. You can assign a type by clicking  and selecting either the standard type or any other type that has been defined.



The following diagram shows the default schema as it appears in the Project Explorer, if you chose to apply it when creating the project:



The following diagram shows how the default schema reflects an organizational structure.



In this diagram:

- **Public Company** is the delivered Organization type.
- **Head Department** is an Organization Unit type that is a member of **Public Company**. This type has a multiplicity defined as **0..1**, so there can only be one **Head Department**.
- **Department**, **Business Unit** and **Team** are the other Organization Unit types delivered. All of these need to be in a hierarchical relationship with another typed organization unit as their parent. The hierarchy does not need to be that shown in the diagram, however; any one of the delivered organization units can be the parent of any other.

Attributes

For each of the Types in the schema you can create Attributes. For each attribute, you can add some extra semantic information to each individual type.

For example, for a Position whose type is Manager, you could create an Attribute called bonus to specify what type of bonus should be given to this Position. The Standard Position Type in the default schema has the attribute **Contract Type** assigned to it. The Standard Location Type delivered in the default schema has the following attributes already defined:

- Country
- State
- City
- Street
- Postcode

When you create an attribute, you must specify a data type that determines its allowable values. For a description of the data types an attribute can have, see [Data Types](#).

i Note: Do not change the data type of an Attribute assigned to any Type in the Schema, once the Organization Model including it has been deployed to the Directory Engine. Since multiple versions of a Model can exist, changing the data type of an Attribute between versions can create inconsistencies.

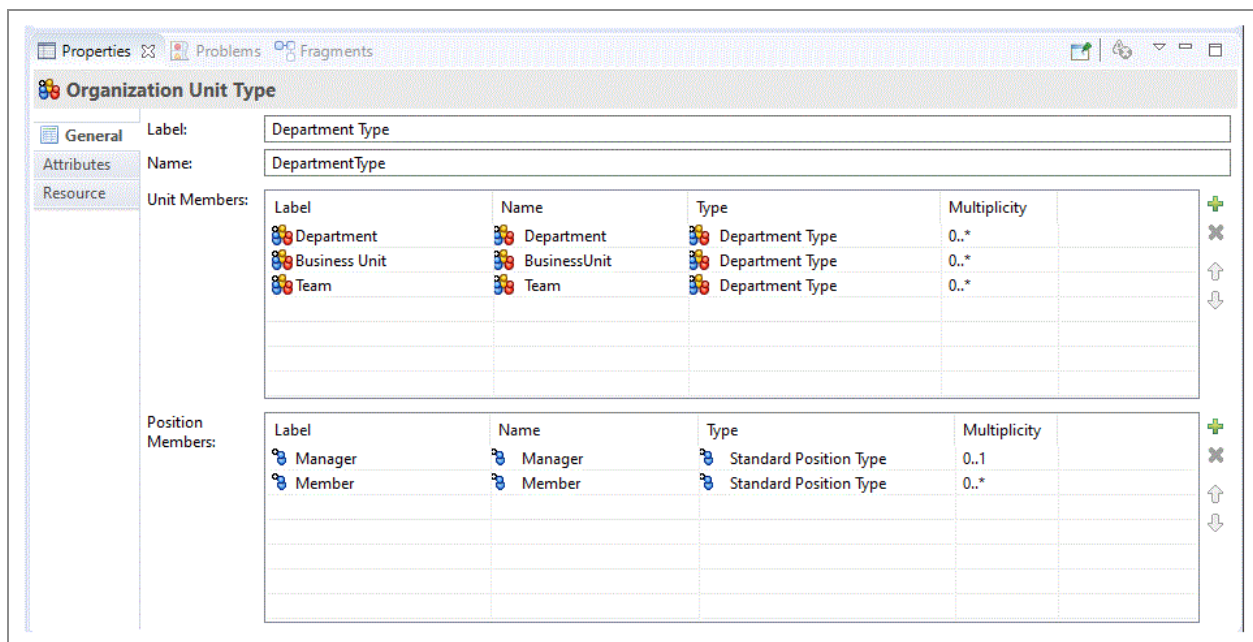
Members

For the Organization Type and Organization Unit Type you can create Members.

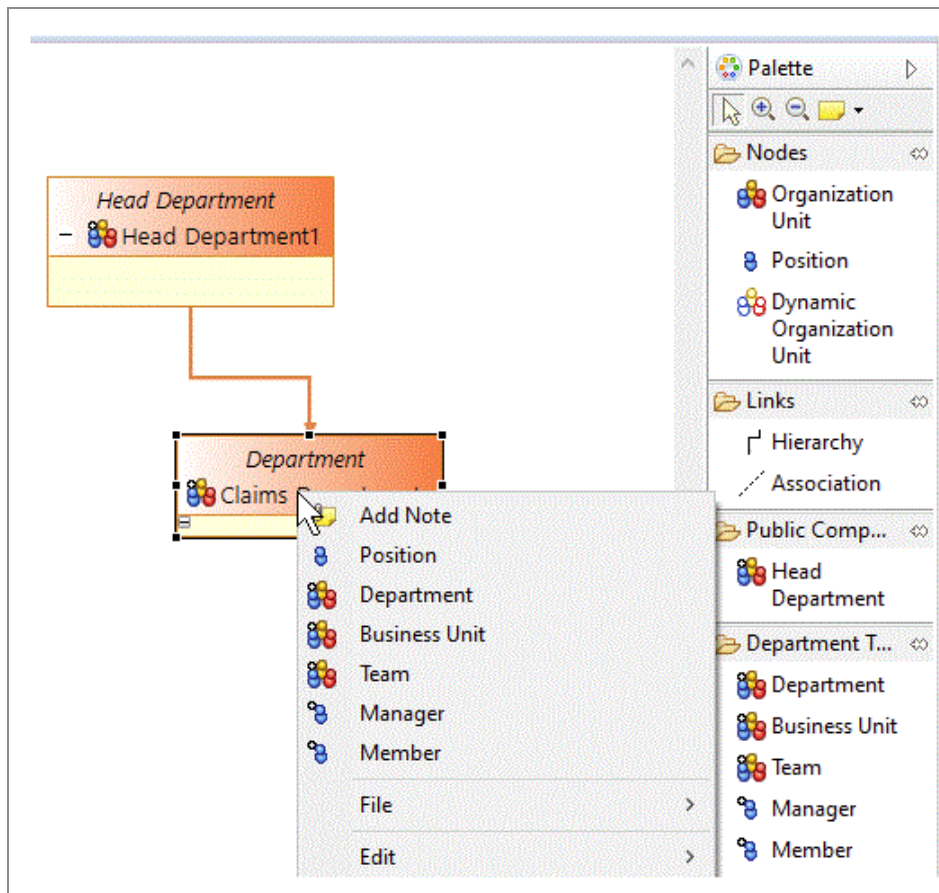
The following table describes what Members you can create for each type:

Type	Members
Organization Type	Unit Members
Organization Unit Type	Unit Members, Position Members

Using Members, you can group together a set of Position Types and Organization Unit Types into meaningful units. Once you have created your Organization Unit Types and Position Types, you can create a template structure by grouping them together within an Organization Type or Organization Unit Type as Members of that Type. An illustration of the Department type delivered in the default schema is shown below.



The Department, Business Unit and Team Organization Unit Types have been grouped together in the Department organization unit type as unit members of the type. This means that when an Organization unit element is created in an Organization Model using the default schema, these elements are available from the **Add Child** menu, from the context menu or from the palette. This is illustrated below.



When you subsequently add an Organization Unit to your Organization diagram, you can select one of these Elements from the Element field on the **General** tab, and assign it to the new Organization Unit.

Multiplicity

You can specify whether you need to allow for multiple copies of an element.

The following table describes the multiplicity values you can specify:

Indicator	Meaning
0..1	No instances or one instance (optional).
0..n	Multiple instances from zero to <i>n</i> where <i>n</i> is greater than zero.

Indicator	Meaning
0..*	Any number of instances; * denotes that there is no limit.
$n..m$	Multiple instances where n and m are zero or more. For example: 1..2 1..* The latter would mean that there must be at least one instance, but there is no upper limit.

Managing Work Using Organization Models

An organization is a collection of people, grouped and related to each other in different ways according to the needs of the enterprise. An organization model formalizes and defines the different elements of the enterprise organization (the organization's entities, their attributes and the relationships between them) that are available for use by a process.

These elements comprise:

- structural elements - organizations, organization units and positions.
- groups, which define the specification for a job of work to be performed, providing a *functional* view of the organization.
- descriptive elements - capabilities, privileges and locations, which provide additional information about other organizational elements, or about the resources (users) that belong to them.
- resources, which can represent items such as people, equipment or buildings.

i Note: Resources in the Organization Model are organization model entities that represent real users. To execute a process successfully, you must map real users to organization model entities using the *Organization Browser*. Once resources have been mapped, when a process is executed, a user task participant is translated into the real user or users who should receive the corresponding work item, see [Mapping Resources to the Organization Model](#).

Using Organization Models in a Process

You can use organization model entities in a process: either as user task participants (at runtime, these entities are converted into one or more real users who receive the work items resulting from user tasks) or to make decisions about how to route work.

Procedure

1. Create or obtain an organization model. See [Creating or Importing an Organization Model](#).
2. Create process participants and map them to organization model entities. See [Using Organization Model Entities as Process Participants](#).
3. Deploy the organization model.
4. Map resources (users) to the organization model. (This is not, however, a design-time activity.) See [Mapping Resources to the Organization Model](#).
5. Deploy the process.
6. Run the process.

Creating or Importing an Organization Model

Organization models are created using the Organization Modeler.

An organization model can exist in the same project as a process that uses it, or it can reside in a different project. In the latter case, the project containing the organization model should be selected as a **Project Reference** (see [Referencing Other Projects](#)).



Note: An organization model can only be placed in a special folder marked as an **Organization Models Folder**. Organization Models folders are marked in Project Explorer with this icon .

Once the organization model has been created, it can be used to define participants in the process. See [Assigning Participants to a User Task](#).

Importing an Existing Organization Model

You can either:

- import a complete project that contains an organization model, or
- import an organization model (.om) file into an **Organization Models** special folder in a project.



Note: This folder must be a special folder defined as an **Organization Models** folder. If the folder is not already properly configured, right-click it and select **Special Folders > Use as Organization Models Folder**.

Organization Model Entities as Process Participants

Every user task in a process that is to be executed must have at least one participant. The participant can be defined in various ways.

The participant can be defined as the following:

- as a particular organizational entity in an organization model used by the project. See [Assigning Participants to a User Task](#) for more details.
- as an **organization model query**, a statement which uses Resource Query Language to create an expression that locates the required participant within the organization. See [Using a Participant Expression to Define a Participant](#) for details of how to do this.

To be available for use by a process, an organization model must be defined either in the same project as the process, or in a referenced project.

Using Capabilities and Privileges in Allocating Work to Process Participants

Using capabilities and privileges allows you to allocate work at runtime.

For example,

- allocate a user task to an accountant if the value to be signed off is less than \$5000, but allocate it to an Accounts Manager if the value is \$5000 or more.
- allocate a user task to a single loss adjuster who holds at least level 2 motor insurance certification and is based in the Chicago office.

To allocate work to process participants using capabilities and privileges, you need to use an organization query. This allows you to enter a query using a script or expression. This is evaluated when a referenced task is executed at runtime, so the actual participant is resolved and the activity dispatched and offered to the participant. A query could resolve to:

- a participant in the package/process. For example, you could define a capability called French speaker and then create a query that resolves a user task to any user who possesses the French speaker capability.
- an entity in the organizational model. For example, you could define a query that resolves a user task to Accounts Managers who can sign off values of \$5000 or more.

See [Assigning Participants to a User Task](#) for more information.

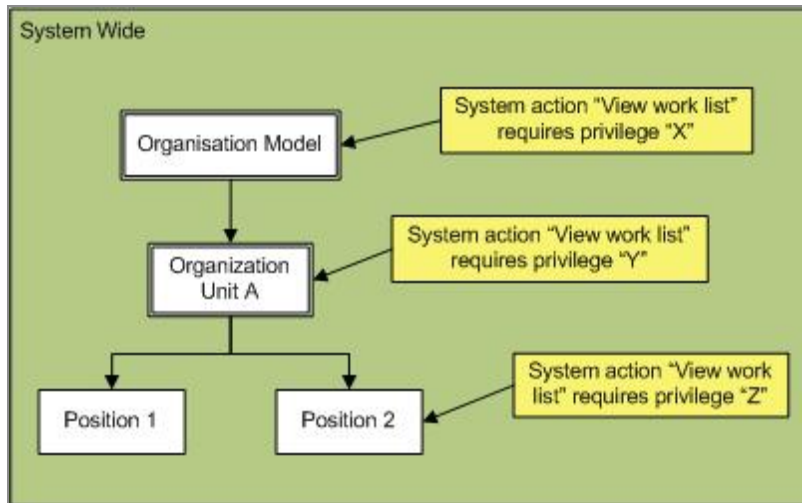
Using System Actions for Processing Work

System actions are actions that a user might wish to perform at runtime but that need to be authorized, or need to be restricted to users with a certain level of authority. These actions might include, for example, re-allocating work-items, skipping work-items, viewing another user's work list, or administering resources.

This authorization is implemented by associating system actions with privileges within Organization Modeler.

- Privileges can be assigned to any system action at the organization model level.
- Privileges can also be assigned to some system actions at the level of the organization unit, position or group.

For example, in the following diagram the "View work list" system action has been associated with three different privileges, "X", "Y" and "Z", at three different levels - the organization model, organization unit A, and position 2.



This means that a user must hold privilege "X", "Y" or "Z" to view the work list of a user who holds Position 2.

If a user wants to view the work list of a user who holds Position 1, they must hold privilege "X" or "Y". This is because no privilege has been associated with Position 1, so any privileges associated with the parent entity are used instead. If privilege "Y" had not been associated with Organization Unit A, the user would instead need privilege "X", defined in the parent Organization Model.

As well as assigning different privileges at different levels, as shown above, qualifiers on the same privilege can be used to refine how access to a particular system action is controlled. (When comparing a required privilege to a held privilege, if either side is not qualified the comparison is positive. If both sides are qualified, the qualifications must match for the comparison to be positive.)

Controlling access to system actions by the application of (user-defined) privileges within the organization model provides an organization with a powerful and completely flexible way to customize and tailor users' access to system functions.

For more information, see [System Actions](#).

Mapping Resources to the Organization Model

Once you have deployed an organization model, you can then map resources - real users - to the model's groups and positions. You must do this before attempting to run any process that uses this organization model. Once resources have been mapped, when a process is executed, a user task participant is translated into the real user or users who should receive the corresponding work item.

You map resources to the organization model using the *Organization Browser*.

About Participants

Participants are the entities assigned to carry out the tasks in a process.

There are two types of participants:

- *user task participants* represent the users who perform the work defined in user tasks. These participants must be defined as *external references* in an organization model used by the process, not as *basic types*.
- *system participants* are used to identify a task that is performed by the system.

When using participants as part of a process, you can:

- create a participant,
- define who receives the work, by assigning one or more participants to a user task, see [Assigning Participants to a User Task](#).
- define how to distribute work to users, see [Defining How Work is Assigned to Users](#).
- delete a participant.

Assigning Participants to a User Task

Participants define who might receive work items generated from a user task.

A user task must have at least one participant, and can have as many as are required by the process. (A validation error is flagged on a user task that does not have at least one participant defined.)

Participants can only be assigned to a user task if they are defined in the process package (at process or package level) and are valid entities in an organizational model used by the process - see [Participants](#).

You can use a combination of the following methods to define a participant for a user task:

- selecting a named participant.
- using a performer data field or parameter. See [Using a Performer Data Field or Parameter to Dynamically Define a Participant](#)

- using a participant expression. See [Using a Participant Expression to Define a Participant](#).
- using organization entities in performer data fields/parameters to allow you to deliver work dynamically to an organizational entity

Using a Performer Data Field or Parameter to Dynamically Define a Participant

A performer data field/parameter is a special type of data field/parameter that you can select as a participant for a user task. By assigning a value to the performer data field/parameter earlier in the process, you can dynamically define a participant for a user task. You can populate a performer data field/parameter with one or more organization entity GUIDs, or a single valid non-array RQL expression.

i Note: A parameter can only be defined at the process level.

Using a performer field, you can deliver work dynamically to an organizational entity, so that the work items appear in managed work lists. For example:

- Dynamically deliver to a group by name (e.g. using an organization entity GUID:
`bpm.process.getOrgModel().groupByName('MyGroup')`)
- Dynamically deliver to a position within an organization unit by name (for example:
`orgunit(name='KEYTeam').position(name='AdditionalStaff') union orgunit(name='Agency').position(name='Contractor')`)

**Note:**

You can also use non-RQL scripting to identify dynamic performers. For example:

```
var groups = bpm.process.getOrgModel().groupByName('MyGroup');
if (groups != null && groups.length == 1) {
  // If you find the group that you want, then use its GUID as
  // the performer field value.
  data.performer = groups[0].getGuid();
} else {
  // If no groups match the given name OR there is more than one
  // group with the same name,
  // then take appropriate application-specific remedial action
  // here.
}
```



Note: If you use the presentation channel settings (push destinations) to deliver notification of work items via email, on the Work Resource tab for the user task, you must set the Distribution Strategy to **Allocate to One** rather than **Offer to All**. For example, if you have a performer field set to: `resource(name='tibco-admin')`, tibco-admin receives an email notification of a work item **only** if the Distribution Strategy is **Allocate to One**.

Procedure

1. Add the performer data field/parameter as a participant to the user task, using the Properties View.
2. Make sure that the process logic assigns a suitable value to the parameter before the user task is executed - for example, by using a script.

You can assign a script.

You can populate a performer field with one of the following:

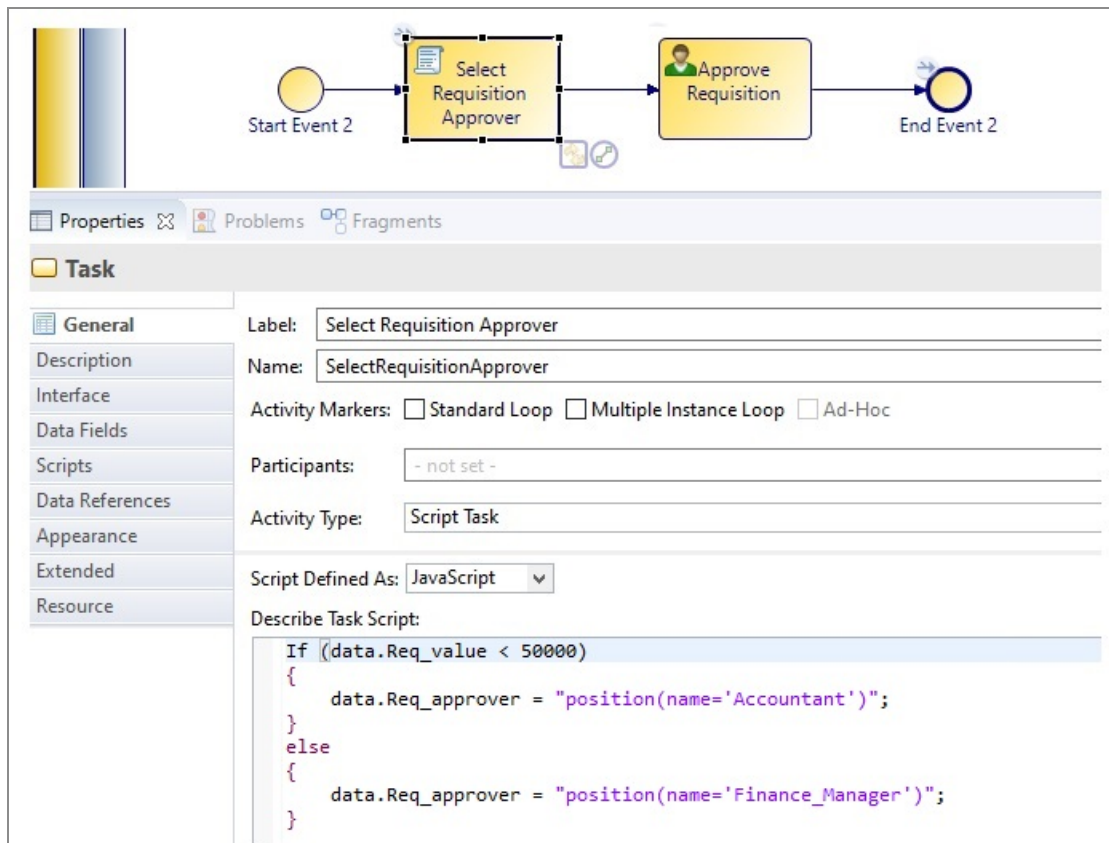
- an organization entity GUID
- multiple organization entity GUIDs (using a performer array field that contains multiple GUIDs - each of the performer fields in the array contains an organization entity GUID)
- a single valid non-array RQL expression (that resolves to one or more organization entities).

- a resource GUID. For example:

```
var users= bpm.process.getOrgModel().resourceByName
('JohnSmith');
if (users!= null && users.length == 1) {
    // If you find the user that you want, then use its GUID as
    the performer field value.
    data.performer = users[0].getGuid();
} else {
    // If no users match the given name OR there is more than
    one user with the same name,
    // then take appropriate application-specific remedial
    action here.
}
```

i Note: Using an organization entity GUID or multiple organization entity GUIDs means that dynamic performers allow references to specific organizational entities, so that they appear in the relevant managed work lists. You should only need to use RQL if you want to offer work based on capabilities, privileges or intersections of organization entities. In most cases, writing a script to identify the organization entity GUIDs you need and populating the performer fields with these are much more efficient than using RQL.

For example, the annotated process extract below shows part of a purchasing process that contains a user task to approve a requisition. The process requires that if the requisition value is less than \$50000, this task can be performed by an Accountant. If it is more than or equal to \$50000, it must be performed by the Finance Manager.



The Select Requisition Approver script task assigns a value to the Req_approver performer data field, based on the value of the requisition which is held in Req_value (and which we assume to have been defined earlier in the process).

The Approve Requisition task assigns the work item to the Req_approver performer data field - the value of which is either "Accountant" or "Finance_Manager" (both of which are also defined as participants for the process).

What to do next

Using a performer field, you can deliver work dynamically to an organizational entity, so that the work items appear in managed work lists.

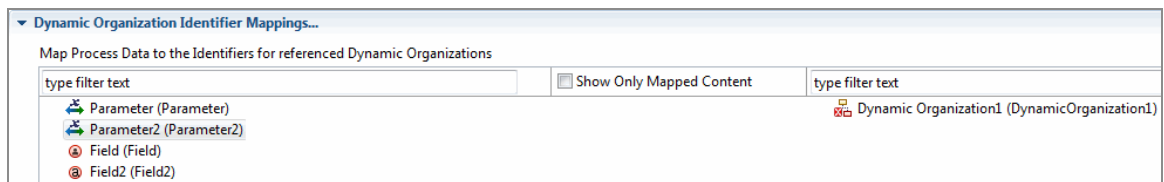
Dynamic Organization Identifier Mapping

When a Dynamic Organization Participant is assigned to a task you need to identify the correct instance of the Dynamic Organization to use to resolve this participant at runtime. This is done using Dynamic Organization Identifiers which are mapped to process data.

Note: The mappings are between the process data and the Dynamic Organization Identifiers of the referenced Dynamic Organization (**not** the Dynamic Organization).

Procedure

1. In the business process, select the **Work Resource** tab, and expand **Dynamic Organization Identifier Mappings....**



2. Map your process data (data fields and parameters) to a Dynamic Organization Identifier (which you set up when you created the Dynamic Organization Model). See [Dynamic Organizations](#).

What to do next

A Dynamic Organization declares its Identifier Fields. These are arbitrary fields that are used to uniquely identify a generated instance of the Dynamic Organization at runtime.

When a Dynamic Organization is assigned to an Extension Point, those Identifier Fields must be mapped/assigned to named LDAP Attributes. This is done after deployment.

A generated instance of a Dynamic Organization takes its Identifier values from those named Attributes; of the LDAP Entry from which it originates.

A Dynamic Organization Participant carries values for the Dynamic Organization Identifiers. These values are derived from process data (data fields and parameters) mapped to those Dynamic Organization Identifiers. With this information the User Task can identify the instance of the Dynamic Organization in which the Participant can be found.

Using a Participant Expression to Define a Participant

You can use the **Organization Model Query** option to create an *expression* to define a participant. The expression sets out a definition that the participant in question must meet. This defines the participant in terms of organization model entities.

For example, this enables you to:

- allocate a work item to the manager of the person who carried out a particular named task.
- allocate a work item to one named position if the value of the data field *x* is >2000, and allocate it to a different named position otherwise.

Participant expressions enable dynamic definitions of participants, since the participant who fits the criteria on one occasion might not be the same on another occasion.

Expressions can be used to describe concepts such as the following:

- The manager of a given organization unit
- A member of a given organization unit
- A given group
- The manager approving a given task

i Note: A push destination assigned to an organization entity (group, position, organization unit, and so on.) only works when the organization entity is explicitly identified as the participant, and not when it is defined as part of a participant reference.

These expressions can be made up of references to organization model entities:

- Lists of all members of a given organization unit
- Lists of all members in a named position in a given organization unit
- Lists of all members of a given named group
- List of all resources in a named position
- A specific named resource

For instance, you can select a group named "HealthSafety" this way:

```
group(name="HealthSafety")
```

Using Organization Entities in Performer Data Field or Parameter

You can use organization entities in performer data fields or parameters to allow you to deliver work dynamically to an organizational entity, so that the work items appear in managed worklists.

You can do this by populating a performer field with one of the following:

- an organization entity GUID
- multiple organization entity GUIDs (using a performer array field that contains multiple GUIDs)

i Note: A Process Task can reference multiple Participants. However, if any of those Participants are Dynamic Organization Participants, then all references to Dynamic Organization Participants by that Process Task must refer to entities within the same Organization Model Template.

- a valid DRQL expression that resolves to one or more organization entities. See [Best Practices when using Dynamic Resource Query Language \(DRQL\)](#)

Procedure

1. Add the performer data field/parameter as a participant to the user task.

Defining How Work is Assigned to Users

A number of design-time mechanisms are available to refine and control how work items generated from user tasks are assigned to users.

Offering and Allocating Work

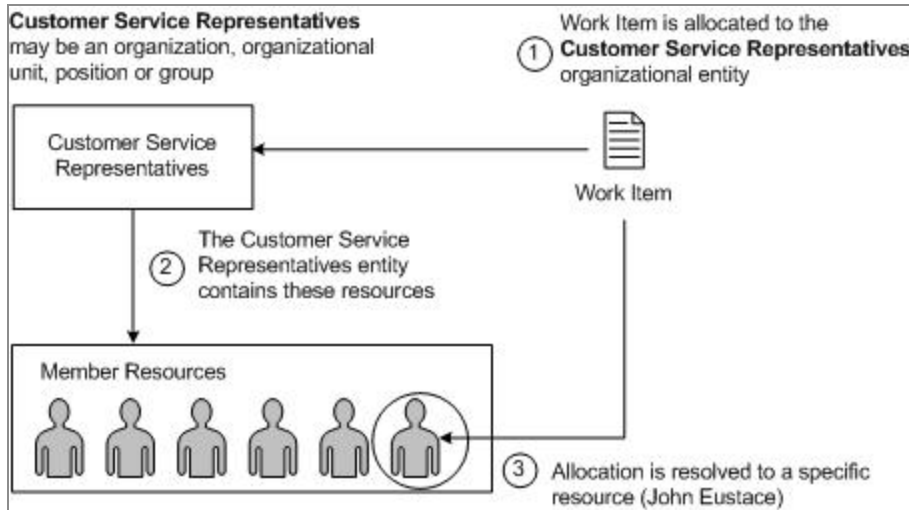
The participants assigned to a user task define the pool of users who are eligible to receive work items generated by that user task. You must also define whether a work item generated from a user task is *allocated* to or *offered* to the pool of eligible users:

- **Allocated work** is only sent to a single user, selected from the pool.

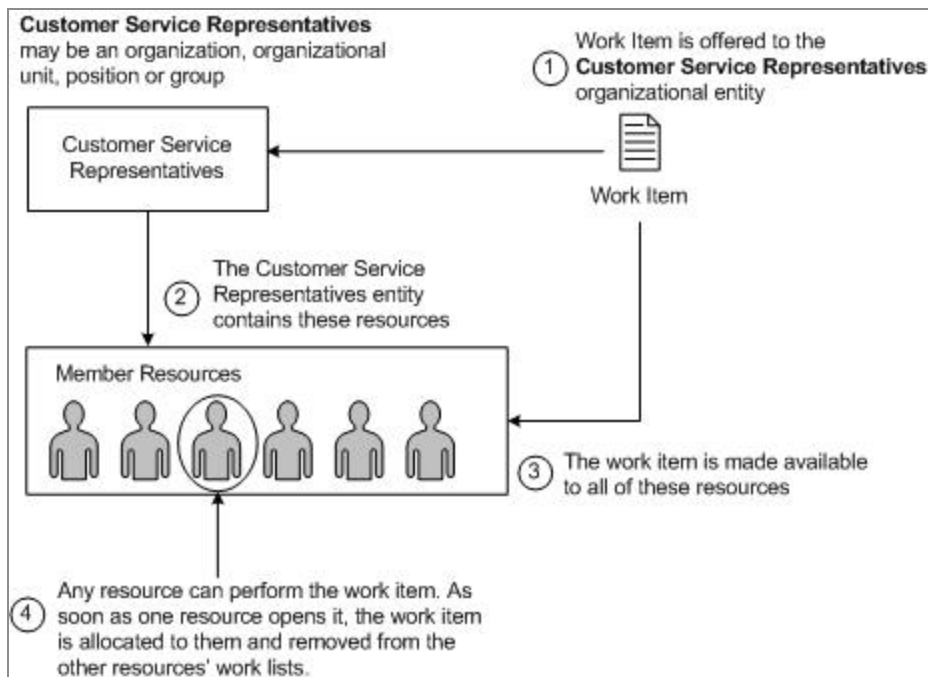
- **Offered work** is sent to every user in the pool.

The following diagrams show the difference between these methods.

Allocating a work item to a single user from the pool



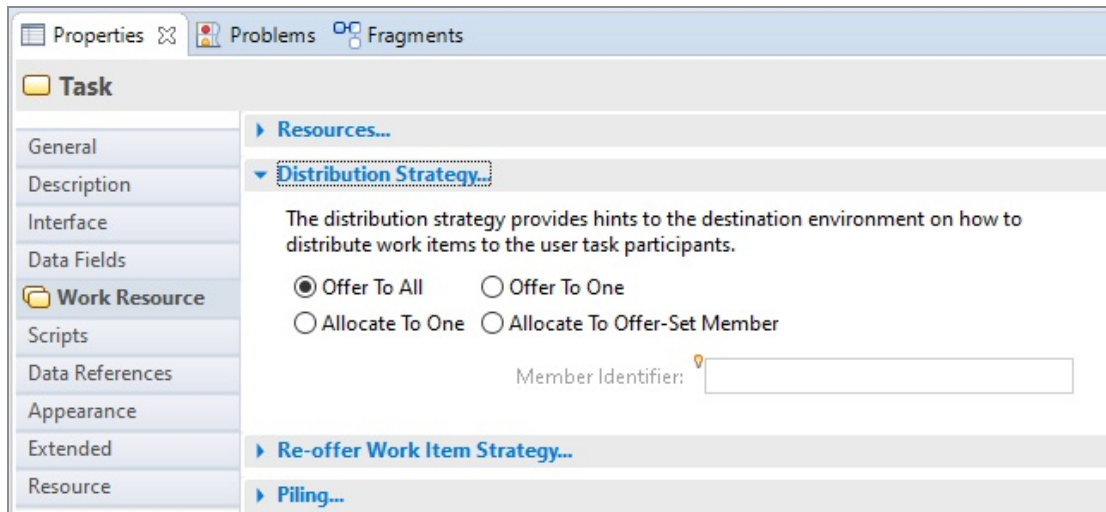
Offering a work item to every user in the pool



Procedure

1. Select the user task.

2. Select the **Work Resource** tab of the **Properties** view.



3. In the **Distribution Strategy** area, click:

- **Offer To All**, to offer a work item generated by this user task to all users in the pool of designated participants. This is the default option.
- **Offer To One**, to offer a work item generated by this user task to a single user in the pool of designated participants. **This option is not currently supported. Do not use it.** A validation error is displayed if this option is selected.
- **Allocate To One**, to allocate a work item generated by this user task to a single user in the pool of designated participants.
- **Allocate To Offer-Set Member**, to specify both an offer set for a work item, and a specific user to whom that work item should be allocated. When the work item is scheduled, if that user is not a valid member of the offer set, the work item is instead offered to the remaining members of the offer set, as if the Offer to all distribution strategy had been used instead. For more information, see [Allocating a Work Item to a Member of an Offer Set](#).

Distributing Work Within the Target Pool

Once you have identified the participants who are eligible to receive work items generated by a user task, the users within the target pool to whom the work items should be sent must be identified.

If the user task's specified distribution method is:

- **Offer**. The work item is offered to all users who are members of the specified

organizational entity (or entities).

- **Allocate.** The work item is allocated to the pool of users that the participant definition resolves to. It determines which user to use by selecting an allocation method. These are:
 - **Round-robin.** Work items are allocated to members in strict rotational order.
 - **Random.** Work items are allocated to members in random order.

Assign the allocation method from the **General** tab of the **Properties** view of the requisite organizational entity. If that entity does not have an allocation method, the allocation method defaults to Random.

i Note: Work items can be assigned to an organizational entity (such as a position) using an RQL statement. RQL is dynamic, which means that if the items referred to by the RQL changes in some way (for example if the resources mapped to an organizational position are changed), then this is reflected in the set of resources associated with the work item.

The work items assigned using an RQL statement do not appear in the Supervised Work List for the organizational entity (because the work items are offered to individual resources, not to the group being listed in the Supervised Work List). For example, if you define a Customer Services Representative position using resource query language, the work item is offered to each resource in the Customer Service Representative position but does **not** appear in the supervised work list.

If work items are offered to organization entities that contain no resources, the work item is scheduled, but does not appear in any worklists until a resource is mapped/added to the organization entity that the work item was offered to.

If work items are offered to organization entities that do not exist, the work item is scheduled to the undelivered group.

If you undeploy an organizational entity that is referenced by RQL and if work items are scheduled to that entity (or if work items are offered to organization entities which are deleted due to an organization model upgrade) then the work items transition to the pending state. To correct this, the organizational entity must be redeployed. In this scenario, an error is logged and audited.

Allocating a Work Item to a Member of an Offer Set

Use the **Allocate to offer-set member** distribution strategy to specify both an offer set for a work item, and a specific user to whom that work item should be allocated. When the work item is scheduled, if that user is *not* a valid member of the offer set, the work item is instead offered to the remaining members of the offer set, *as if* the **Offer to all** distribution strategy had been used instead.

The **Allocate to offer-set member** strategy allows you to support, for example, a case handler/account manager pattern, so that although the work item is originally allocated to a member of a team, the team manager can still:

- see all items that were originally offered to the team.
- re-allocate the work item to another member if required - for example, if the user who started the case is off work due to sickness.
- report on work from a team perspective.



Note: The **Allocate to offer-set member** distribution strategy cannot be used with the Chained Execution, Separation of Duties or Retain Familiar workflow patterns.

Procedure

1. Define a Performer field that is used to identify the user to whom the work item should be allocated.

The Performer field must not be an array field and must not contain an initial value defined as an RQL expression. In either case, an error marker is displayed against the field.
2. On the **Properties** tab of the User task, click **Work Resource > Distribution Strategy > Allocate To Offer-set Member**.
3. In the **Member Identifier** field, enter the name of the Performer field you defined earlier.
4. Modify the process so that it populates the Performer field with a suitable value before the user task is executed.

The performer field must, at runtime, contain the GUID of a single resource who is a valid member of the work item's offer set, as defined by the user task's participant.

Note: You can use scripting methods and attributes to obtain the resource's GUID. See "Process Manager and Work Manager Scripting" in the *TIBCO Cloud BPM Business Data Services Guide* for more information.

The performer field must not be populated by an RQL expression. (This results in a runtime error.)

Result

At runtime, the work item is allocated to the user identified by the GUID value in the performer field. If the performer field returns an invalid value:

- The work item is instead offered to all members of the offer set.
- One or more of the following error messages are returned (in the BPM log file and in relevant event or audit views).

Message ID of Error	Possible Causes
BRM_RESOURCE_INVALID_DESCRIBE_ENTITY	<p>The provided GUID belongs to an organizational entity that is not a resource.</p> <p>The performer field has been populated using an RQL expression (for example, <code>resource(name="Leon")</code>), so does not contain a GUID.</p>
BRM_WORKITEM_SCHEDULE_MEMBER_INVALID	The provided GUID belongs to an invalid resource.
BRM_WORKITEM_SCHEDULE_NOT_A_MEMBER	The provided GUID belongs to a resource who is not a member of the offer set.

Using Resource Patterns to Control How Work is Assigned

You can use the following resource patterns in your process to further control how work is assigned to users. Resource patterns are formal representations of the various ways in which resources are represented and utilized in workflows, as identified by the **Workflow Patterns initiative**.

- [Chained Execution](#)
- [Separation of Duties](#)
- [Retain Familiar](#)
- [Piling](#)

Chained Execution

Chained Execution allows you to specify that a user automatically starts the next work item in the same process instance as soon as the previous one has been completed.

To set up a chained execution pattern, you refactor the required user tasks as an embedded sub-process, and mark the sub-process to use chained execution.

User tasks that are to be chained together must meet the following requirements:

- They can be assigned to different Participants, but they still are offered to the same user *if* that user qualifies under the participant definitions for each task. For example, if one task is assigned to a position and the second to a group, a user who both holds that position and belongs to that group can be given both tasks. If the user is not a member of the group, the second task is offered to each of the group members.
- They must have **Offer to All** as their **Distribution Strategy** (see [Offering and Allocating Work](#)).

Separation of Duties

Separation of Duties allows you to specify that particular user tasks must be performed by different *users*, even though the tasks are assigned to the same *participant*.

Retain Familiar

Retain Familiar stipulates that you want a specific task to be executed by the same resource that executed a previous task in the same process instance.

For example, the resource that handles the initial customer contact is the same one that handles the follow-up call.

Piling

Piling allows you to specify that a particular participant completes work items that relate to a specific task, sequentially.

The work items might be in different processes. Once the work item is completed, if another work item that corresponds to the same task is present in the user's work list, it is immediately started. The benefit of this is that the same work is given to a specific resource who then gains experience in processing this particular task.

Resource Query Language

The Resource Query Language (RQL) is used to identify resources that meet a defined set of criteria. An RQL query returns a set of resources that match the criteria expressed in the query. Work can then either be allocated to one of those resources, or offered to multiple individual resources.

RQL is dynamic, and is evaluated when the work item is created and whenever it changes. This means that if the items referred to by the RQL change in some way (for example if the resources mapped to an organizational position are changed) this is reflected in the set of resources associated with the work item.

See [Assigning Participants to a User Task](#) for details of how RQL is used.

You might want to consider whether to use Resource Query Language or Dynamic Organization Participants. See [Dynamic Organization Participants](#).

Best Practices When Using Resource Query Language (RQL)

When using Resource Query Language (RQL) there are a number of issues you should bear in mind to ensure that your processes operate most efficiently when deployed. Planning when designing your process and RQL can save problems later on.

- All RQL is dynamic, so there are design implications because of that. You should be aware that changes to the deployed organization model are reflected in the RQL results set after a period of time.

- Model your system as relevant to the application - this is almost certainly not the same as the durable organization you might see in an organization chart. If you do, you might not need to use RQL in most cases.
- RQL is more complex than using basic participants. Where possible use basic participants - and only use RQL for special cases.
- In both new and existing projects, consider whether you need to use RQL at all, or whether it can be replaced more efficiently with Dynamic Organization Participants. Dynamic Organization Participants allow you to do many actions without the overhead of RQL and are quicker. Dynamic Participants only allow valid actions, and at runtime they perform better than RQL because you are explicitly specifying an internal identifier for an organization entity (GUID). So there is instant lookup, with no parsing. Using Dynamic Organization Participants is the preferred way of dynamically assigning work. See [Dynamic Organization Participants](#).
- Work items allocated using RQL do not appear in any managed work lists as they are not directly associated with specific organizational entities. You should use dynamic performer fields to get the association of work items to organizational entities functionality. See [Using a Performer Data Field or Parameter to Dynamically Define a Participant](#).
- Use RQL when you want to reference LDAP attributes or reference capabilities.
- When you deploy an organization model it is cached in memory. Actions using this in memory cache are quick (for example, RQL using references to an orgunit in the model). However, RQL that references resources, makes calls to the database and takes time.
- If you are looking at all LDAP attributes on a resource, looking them up in LDAP is time-consuming. If combined with operators such as OR, then it is even slower.
- You need to use RQL for more complicated queries - those involving union, intersect, and qualifying LDAP attributes for example.
- You could produce valid RQL which does not identify any actual resources. You do not receive any warning that this might happen, so must be careful to understand the results of the RQL you are providing.
- RQL cannot refer to Dynamic Organization Units (as these are replaced with dynamic instances at runtime).

Examples of Usage of RQL

When deciding what RQL queries to make, you should plan to minimize the number of calls they make to the database to make them efficient. For information on what options are available, see [Organization Entities](#).

```
orgunit(name='KEYTeam').position(name='AdditionalStaff') union orgunit
(name='Agency').position(name='Contractor')
```

This example queries the organization unit called KEYTeam and for positions called "AdditionalStaff". It then joins the result of this to a query of the organization unit called Agency, and the results who have the position Contractor.

```
orgunit(name='KEYTeam').position(name='Director' or
name='Management').capability (name='Language' qualifier='French' )
```

This example queries the organization unit called KEYTeam, and for positions named either Director or Management where the resource in the position has a capability of French language.

```
orgunit(name='KEYTeam').position(name='AdditionalStaff') intersect resource
(attribute.Area='DACH')
```

This example finds all the resources which have an LDAP attribute of AREA that is equal to DACH. This takes some time because of the querying of LDAP for all resources defined in BPM.

```
(orgunit(name='KEYTeam').all() intersect not orgunit
(name='Management').position(name='*')) union orgunit(name='Vice
Presidents*').position(name='Permanent')
```

This is an example of a complex RQL statement that takes excessive time, as it makes multiple calls on the database to resolve resource queries.

RQL Structure

RQL expressions are made up of the following components: Keywords, Operators, Organization Entities, and Combinators:

- [Keywords](#)
- [Operators](#)
- [Organization Entities](#)
- Combinators. See [Combining Expressions](#).

Keywords

The keywords described in this topic are permitted.

- **name.** The name of an entity. For example:
`name="javaDeveloper"`
- **type.** The type of the entity within the organization model schema, if it has a type. For example:
`type="JavaProgrammer"`
- **attribute.** An attribute of a resource that has been extracted from the LDAP source. This has the form **attribute.attributeName**. For example:
`attribute.phone = "+44(0)1793441300"`

The values associated with keywords can be qualified by combining them with other keywords, for example:

`(name="javaDeveloper" or type="JavaEngineer")`

- **qualifier.** The qualifier for a [Capability](#) or a [Privilege](#). For example:

`qualifier=" > 1000"`

The **qualifier** keyword is used by combining it with the **name** of a [Capability](#) or a [Privilege](#), for example:

`capability(name="Language" qualifier=Arabic)`

The '*' and '?' wildcard characters can be used:

`name="uk-*`

Operators

The operators listed in this topic are permitted

The operators are permitted in the following order of precedence:

- (,)
- =, <>, <, <=, >, >=
- not
- and

- or

Organization Entities

The organization entities listed can be referred to. One of these must be used as the starting point of every query.

- Organization
- Orgunit (organization unit)
- Position
- Location
- Capability
- Privilege
- Group
- Resource

i Note: A Push Destination assigned to an organization entity (Group, Position, Organization Unit, and so on.) only works when the organization entity is explicitly identified as the participant, and not when it is defined by the Resource Query Language.

The significance of specifying each of these organizational elements is set out in the following sections.

Organization

For example:

```
organization(name="EasyAs")
```

This expression returns all the resources allocated to any Position which is allocated to any Organization Unit within the Organization named **EasyAs**.

Orgunit (organization unit)

For example:

```
orgunit(name="Support-SWI")
```

This expression returns all the resources allocated to all the Positions within the organization unit named **Support-SWI**.

By default Organization Units are not treated recursively. That is, an expression specifying an organization unit does not return positions in a sub-unit of that organization unit. See [Navigating the Organization Model with RQL Queries](#) for more details.

Position

For example:

```
position(name="Manager")
```

This expression would return all the resources in a Position named **Manager**.

You can also locate resources based on their type in the organization model schema, for example:

```
position(type="UnitManager" or name="Manager")
```

This expression would return all the resources in a position which is an instance of the position type named **UnitManager**, or is a Position called **Manager**.

Location

For example:

```
location(name="NewYork")
```

This expression returns all the resources that:

- Have a location of **NewYork**, or
- Are allocated to a Position that:
 - is located in **NewYork**, or
 - is allocated to an Organization Unit that:
 - is located in **NewYork**, or
 - is allocated to an Organization that is located in **NewYork**.

Capability

For example:

```
capability(name="JavaProgrammer")
```

This expression includes all the resources that:

- Have the capability named **JavaProgrammer**



Note: RQL only makes use of capabilities that are assigned to Resources.

Capabilities might be assigned to Positions and Groups in the organization model. These represent "entry criteria" - that is, only resources with that capability should be assigned to that group or position. However, this is not enforced. Capabilities assigned to Resources indicate that the person represented by that Resource does actually have the capability in question.

Capabilities might be further refined by using a qualifier, as described for [Privilege](#).

Privilege

For example:

```
privilege(name="signoff" qualifier>10000)
```

All the resources allocated to a Position with:

- the privilege **signoff** with a qualifier value greater than 10000, or
- belonging to an Organization Unit with the privilege **signoff** with qualifier value greater than 10000.

An expression can also specify the privilege without specifying a qualifier. For example:

```
privilege(name="signoff")
```

This would return all the resources allocated to a Position or an Organization Unit with the privilege **signoff**, regardless of any qualifier.

Group

For example:

```
group(name="Health&Safety")
```

This expression returns all the resources in a group named **Health&Safety**.

Groups can operate recursively - that is, groups can contain groups. So this expression would also include all resources belonging to all sub-groups. This can be overridden - see [Navigating the Organization Model with RQL Queries](#).

Resource

For example:

```
resource(name="Clint Hill")
```

This expression returns all the resources with the name **Clint Hill**.

Unlike the other entities listed, Resources refer directly to actual users as defined in LDAP. When selecting resources you can start with a particular resource by referencing its name, for example:

```
resource(name="Clint Hill").position(name="abc").orgunit(type="efg").privilege  
(name="signoff" qualifier > 10000)
```

This expression returns all Resources that belong to an OrgUnit of type "efg" in which Clint Hill holds the position named "abc", and in which he holds the "signoff" privilege, qualified for a value of greater than 10000.

You can also query attributes from the LDAP database that have been passed to BPM, for example:

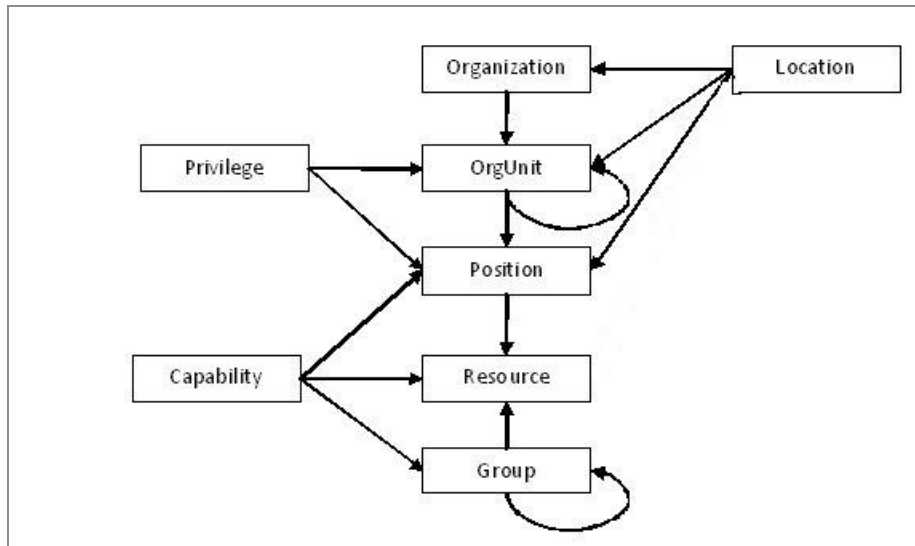
```
resource(attribute.phone="+44(0)1793*" and attribute.language="*Spanish*")
```

This returns the resources with a Swindon dialing code (+44 (0)1793) on their **phone** attribute and whose language attribute include the text **Spanish**.

Navigating the Organization Model with RQL Queries

RQL expressions, starting with one of the elements listed in [Organization Entities](#), locate resources by following the paths indicated by arrows in the diagram below. For example, an expression that specifies a capability finds all the positions, resources, and groups with that capability; and then these cascade down to find the resources in the positions and groups.

The following diagram shows how resources can be located from each of these starting points.



Using the . Operator

You can restrict the results returned by an expression by using the dot operator ("."). This can be used in two ways.

Using the dot operator to qualify organization entities

You can use the dot operator to specify a qualification of an organizational entity. For example:

```
orgunit(type="Sales").privilege(name='signoff' qualifier>10000)
```

This expression returns resources in those Organization Units of type **Sales** that have the privilege qualified by a value of greater than 10000.

i Note: This is not a simple "and" operator; the expression does not return resources that are in organization units of type **Sales** and that have the required privilege; it returns resources where the organization unit itself has that privilege.

The dot operator is not commutative: in other words, changing the order of the expression changes its meaning.

So, for example:

```
position(name='abc').orgunit(type='Sales')
```

Expressed in natural language, this example means:

All Resources, in all Positions of Org-Unit of type "Sales", in which the Position named "abc" .

Another example is:

```
orgunit(type="Sales").position(name="abc")
```

Expressed in natural language, this example means:

All Resources in the Position named "abc" of Org-Unit of type "Sales".

Using the dot operator in hierarchical relationships

The dot operator works differently when used to express hierarchical relationships between two organizational entities. For example:

```
orgunit(type="area office").position(type="manager")
```

returns Positions of type **manager** that are in organization units of type **area office**.

i Note: You can only use the dot operator to link organizational entities that are **directly** connected by arrows in the diagram in [Navigating the Organization Model with RQL Queries](#). For example:

```
organization(type="public company").position(type="manager")
```

would not be valid, because **organization** and **position** are not directly linked.

Using only children and all

In organization models, both Organization Units and Groups can be linked to other elements of the same type in a hierarchy.

However RQL expressions do not by default navigate this hierarchy.

For example the expressions:

```
orgunit(name="AreaNorth")
```

```
group(name="HomeInsurance")
```

would return only resources assigned directly to the named organization unit or group, not to other organization units or groups in a hierarchical relationship below it.

To override this default behavior, you can use one of the following modifiers:

- **only()**. For example

```
group(name="HomeInsurance").only()
```

This returns only resources assigned to the named element. (This is the default behavior both for Organization Units and for groups.)

- **children()**. For example

```
group(name="HomeInsurance").children()
```

This returns resources assigned to the named element and to any first level child elements, but not to elements any further below in a hierarchy.

- **all()**. For example

```
orgunit(name="AreaNorth").all()
```

This returns resources assigned to the named element and to any subordinate elements.

Combining Expressions

All expressions in RQL return the set of resources that correspond to that expression. You can combine expressions to produce more sophisticated queries using the following set operators:

- **union**. Returns resources that are in either expression. For example:

```
orgunit(name="Drivers").position(name="Developer") union orgunit  
(name="OsakaDev").position(name="JavaDeveloper")
```

returns any resource who is either a Developer in the Drivers organization unit or a Java Developer in the OsakaDev organization unit.

- **intersect**. Returns resources that are in both expressions. For example:

```
orgunit(name="Drivers").position(name="Developer") intersect group  
(name="JavaDevelopers")
```

returns any resource who is both a Developer in the Drivers organization unit and a member of the Java Developers Group.

Use an intersect to specify multiple qualifier values for a capability or a privilege. (If

the data type of the qualifier is EnumSet it is possible for the capability or privilege to have multiple qualifier values.) For example:

capability(name='Language' qualifier='French') intersect capability (name='Language' qualifier='Spanish')

- **subtract.** Returns resources that are in the first expression but not the second. For example

```
orgunit(name="Drivers").position(name="Developer") subtract group
(name="Managers")
```

returns any resource who is a Developer in the Drivers organization unit but is not a member of the Managers Group.

- **not.** Returns resources that are in the first expression but not the second. For example

```
orgunit(name="Drivers").position(name="Developer") intersect not group
(name="Managers")
```

returns any resource who is a Developer in the Drivers organization unit but is not a member of the Managers Group.

i Note: While similar in the resources they return, **subtract** returns the *relative complement* and must be used when performance is a consideration.

Another example might be:

```
(
orgunit(type="Support").position(type="SupportEngineer")
intersect
not group(name="Backdesk")
)
union
orgunit(name="Support").position(name="Manager")
```

This selects support engineers in the Support department who are not part of the back desk group, together with the support department's managers.

Combining expressions in this way can sometimes mean that a resource is included in the results for more than one reason. If this happens, the resource is only included in the result set once; that is, there is no duplication of results.

RQL Cleanup Configuration

When resource queries are no longer referenced, they should be deleted from the system.

The following resource query cleanup properties are available to configure when and how often resource queries are evaluated to determine which can be deleted:

- `ResourceQueryCleanerEnable` - Enables, or disables, the evaluation of resource queries to identify those that are no longer referenced. If this is disabled (false), automatic deletion of un-referenced resource queries does not take place.
- `ResourceQueryCleanerStart` - The time of day at which resource queries, that are no longer being used, are scheduled for deletion.

Cleanup of unused resource queries is invoked at this time each day that cleanup is scheduled according to the value of `ResourceQueryCleanerInterval`.

- `ResourceQueryCleanerInterval` - The interval between evaluations of resource queries to identify those that are no longer referenced, and can be deleted. This defaults to once per day (at the time specified in `ResourceQueryCleanerStart`).
- `ResourceQueryCleanerEnd` - The time of day at which the last resource query evaluation of the queries that can be deleted is accepted.
- `ResourceQueryCleanerLimit` - The number of resource queries checked for deletion in a single database transaction.
- `ResourceQueryCleanerPause` - The number of seconds between deleting batches of resource queries.

For information about configuring these properties, see "Directory Engine Configuration" in the *TIBCO Cloud BPM Administration* guide.

Work List Facade

The Work List Facade (WLF) project is the single place where you can define display values for work list attributes that are used throughout an organization.

You can only have one work list facade per organization and the Work List Facade project contains a single file.

Work Item Attributes

Work Item Attributes are values that can be assigned to work items from process data. They can then be displayed for the work list entries for the work items, and can be sorted on.

When Work Item Attributes are listed in the Mapper, you can use the Work List Facade to define a set of display labels, and assign them to the predefined work item attribute set.

In the Work List Facade editor, the attributes are sorted on the basis of the numeric values in their name: for example, attribute0, attribute1, attribute2. See [Creating a Work List Facade](#).

You can define display values for any of the 40 work item attributes. Work item attributes that have display values defined appear at the top of the list of attributes, above those with no display values assigned. See [Setting the Display Label for Work Item Attributes](#).

Physical Work Item Attribute Name	Type (Length)	Display Label
① attribute1	Fixed Point Number (0 decimals)	
② attribute2	Text (64)	
③ attribute3	Text (64)	
④ attribute4	Text (64)	
⑤ attribute5	Floating Point Number	
⑥ attribute6	Date Time and Timezone	
⑦ attribute7	Date Time and Timezone	
⑧ attribute8	Text (20)	
⑨ attribute9	Text (20)	
⑩ attribute10	Text (20)	
⑪ attribute11	Text (20)	
⑫ attribute12	Text (20)	
⑬ attribute13	Text (20)	
⑭ attribute14	Text (20)	
⑮ attribute15	Fixed Point Number (0 decimals)	
⑯ attribute16	Floating Point Number	
⑰ attribute17	Floating Point Number	
⑱ attribute18	Floating Point Number	
⑲ attribute19	Date Time and Timezone	
⑳ attribute20	Date Time and Timezone	
㉑ attribute21	Text (20)	
㉒ attribute22	Text (20)	
㉓ attribute23	Text (20)	
㉔ attribute24	Text (20)	
㉕ attribute25	Text (20)	
㉖ attribute26	Text (20)	
㉗ attribute27	Text (64)	
㉘ attribute28	Text (64)	
㉙ attribute29	Text (64)	
㉚ attribute30	Text (64)	
㉛ attribute31	Text (64)	
㉜ attribute32	Text (64)	
㉝ attribute33	Text (64)	
㉞ attribute34	Text (64)	
㉟ attribute35	Text (64)	
㊱ attribute36	Text (64)	
㊲ attribute37	Text (64)	
㊳ attribute38	Text (64)	
㊴ attribute39	Text (255)	
㊵ attribute40	Text (255)	

Note: As work lists might be used to display work items from many different TIBCO BPM Enterprise applications simultaneously, the use of work item attributes must be coordinated between those applications.


Creating a Work List Facade

The Work List Facade allows you to optionally declare the design-time definition of a set of display labels for the predefined work item attribute set. The display label set is contained in the Work List Facade.

See [Work Item Attributes](#).

Procedure

1. Select **File > New > Work List Facade Project**.
2. Enter the project name.
3. Click **Finish**.

 **Note:** If you have deleted the Work List Facade file for any reason, you can add a new Work List Facade file. From the Work List Facade Special Folder select **New > Work List Facade**. However, multiple Work List Facade files are not allowed.

What to do next

Now go to the work list facade editor to define the display names of your work list attributes. See [Setting the Display Label for Work Item Attributes](#).

Setting the Display Label for Work Item Attributes

The work list facade editor is a table editor that allows you to optionally set the display label for any of the predefined set of 40 work item attributes.

Display labels are used throughout the organization.

Procedure

1. When you create a Work List Facade project, the **Work Item Attributes** table opens. If it is not already open, select **WorkListFacade.wlf** from your Work List Facade project.
2. Add values to the Display Label column.

Work Item Attributes		
Physical Work Item Attribute Name	Type (Length)	Display Label
① attribute1	Fixed Point Number (0 decimals)	Phone Number
Ⓜ attribute2	Text (64)	Name
Ⓜ attribute3	Text (64)	Address Line 1
Ⓜ attribute4	Text (64)	Address Line 2
Ⓜ attribute5	Floating Point Number	
Ⓜ attribute6	Date Time and Timezone	
Ⓜ attribute7	Date Time and Timezone	
Ⓜ attribute8	Text (20)	
Ⓜ attribute9	Text (20)	
Ⓜ attribute10	Text (20)	
Ⓜ attribute11	Text (20)	
Ⓜ attribute12	Text (20)	
Ⓜ attribute13	Text (20)	
Ⓜ attribute14	Text (20)	
① attribute15	Fixed Point Number (0 decimals)	
Ⓜ attribute16	Floating Point Number	
Ⓜ attribute17	Floating Point Number	

You can add as many or as few display values as you want. When you use the Work List Facade Editor as described in [Work Item Attributes](#), to add values, the attributes with display values appear at the top of the list of attributes wherever they appear in the user interface.

The attribute name and type column are **read-only**. Only the Display Label can be edited. The Display Labels have length restriction of 64 characters.

What to do next

When you have completed your display labels, deploy the Work List Facade project.

If you need to change your display labels at any time, edit them as described and redeploy the Work List Facade project.

Mapping Process Data to Work Item Attributes

You can map process data to work item attributes to allow automatic work item attribute assignment on all user tasks which implicitly or explicitly reference the mapped process data. This can be done to avoid the use of scripts to achieve the same result.

Procedure

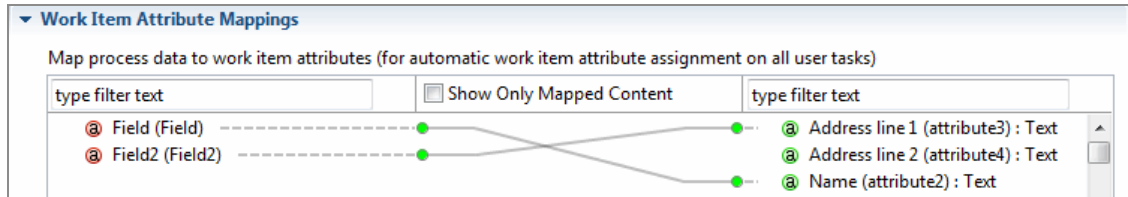
1. On the Work Resource tab of the Business Process, expand **Work Item Attribute Mappings**.

The left column shows all data fields defined for the process. The right column lists

the 40 work item attributes available, with those with defined display labels at the top of the list. When there is no Work List Facade file in the Workspace scope, the Work Item Attributes are sorted by the numeric values in their name. See [Setting the Display Label for Work Item Attributes](#)

2. Map a data field to an attribute.

For example, **Field** could be mapped to the **Name** attribute. Do this for all data fields you would like to map to attributes.



Each user task in the process, with the mapped process data associated, contains the script equivalent to the mappings defined here.

i Note: If you use the Work Manager script from the **Scripts** tab for a user task, you can only use the physical attribute names (attribute1, attribute 2, and so on) which appear in content assist and not the display names.

Importing Pre-Version 5.x Projects into TIBCO Business Studio - BPM Edition Version 5.x

Projects that were created in pre-version 5.x TIBCO Business Studio - BPM Edition can be imported into TIBCO Business Studio - BPM Edition version 5.x and newer.

Some features that were available in pre-version 5.x projects are no longer available in version 5.x (and newer). See [What's Changed?](#) Any features contained in your pre-version 5.x projects that are no longer supported can cause problem markers to appear in TIBCO Business Studio - BPM Edition. After you have imported your pre-version 5.x projects, you must perform some post migration tasks to fix these problem markers. See [Post Migration Tasks](#).

Also note that when you are importing pre-version 5.x projects, the Import wizard determines which projects are dependencies of other projects, and assists you with importing those projects first. After importing the lower-level projects, any problem markers should be resolved before continuing to import the higher-level projects, as the higher-level projects depend on the lower-level projects operating properly.

What's Changed?

Some features that were available in pre-version 5.x ActiveMatrix BPM are no longer available in ActiveMatrix BPM version 5.x and newer.

Feature	Description
BPM Projects	<p>There are some changes to BPM projects.</p> <ul style="list-style-type: none">Previously, you could create projects for destinations other than for ActiveMatrix BPM, as well as for other add-ons like simulation. This is no longer the case. There are no other destination environments. Therefore, configuration for destination environments is no longer

Feature	Description
	<p>shown.</p> <ul style="list-style-type: none"> • The existing BPM Developer Project now changes to BPM Process Project. • You must have separate projects for business object models and organization models. <p>Note: TIBCO Business Studio - BPM Edition provides a quick fix to move business object models or organization models into dedicated projects.</p>
Web Services	<p>ActiveMatrix BPM version 5.x+ does not support:</p> <ul style="list-style-type: none"> • Using web services with WSDL documents. You must convert web-service invocation tasks to REST service tasks that invoke equivalent REST services. • Exposing service operations that external clients or other processes can invoke using WSDL documents. Any of your processes that were previously exposed as web services can now be called using the run-time public REST API. <p>Content and configuration relating to WSDL documents is removed on importing pre-version 5.x ActiveMatrix BPM projects to version 5.x+. See WSDL Activities for more information.</p>
Data	<p>Run-time business data internal representation has changed from XML to JSON which means that XML is not supported as a mechanism for creating business object models.</p> <p>Process data fields are now wrapped in a data object. See Scripting.</p>
Data mapping	<p>As data internal representation is changing from XML to JSON, data-mappings are being standardized for the data mapper grammar. All JavaScript/XPath mappings for all WSDL related activities; service task, receive task, send tasks, message events, catch WSDL fault events and all JavaScript mapped to non-WSDL related event / task types (sub-process, signals, catch error) are automatically converted to data mapper grammar,</p>

Feature	Description
Forms	<p>Note: You can use pre-version 5.x TIBCO Business Studio - BPM Edition side-by-side with version 5.x TIBCO Business Studio - BPM Edition. This means you can look at your data mappings in your original project to help you re-map your data in a new project.</p>
	<p>See Data Mapping for more information.</p>
Scripts	<p>Some API methods are different between pre-version 5.x and version 5.x. When importing pre-version 5.x projects to TIBCO Business Studio - BPM Edition, the API methods are converted. See the <i>TIBCO Business Studio™ - BPM Edition Forms User's Guide</i>.</p> <p>As data internal representation is changing from XML to JSON there are changes to the JavaScript capabilities. For example, multiple instance data is now represented as arrays rather than specialized list objects. Many JavaScript capabilities have also been standardized between process scripts and form scripts.</p> <p>Factory methods have also changed. See Scripting.</p>
Business Object Models	<p>The following are changes to Business Object Models:</p> <ul style="list-style-type: none"> Existing Business Object Models that were generated from WSDL (previously in the Generated Business Objects folder) are automatically moved to a Business Objects folder. This is because they might still be used in processes, even if they are not used to invoke WSDL services. A new concept of a terminal state is introduced. All case classes must have a case state attribute and that attribute must nominate the case state enumerations that indicate when a case is complete. You must set a case to one of its terminal states when there is no more work to be done on a case. The Case Identifier attribute has changed. You can choose to generate case identifiers automatically, or enter them manually. If you choose Auto, the Case Identifier can include a prefix and/or a suffix. For example, UK-1234-A. See Case Identifiers.

Feature	Description
	<ul style="list-style-type: none">Some BOM primitive types are no longer supported: Attachment, DateTime, Duration, ID, Integer, Object.
Deployment	You no longer export your project as a DAA (Distributed Application Archive). Instead, you generate deployment artifacts (RASC (Runtime Application Shared Concept) files). These files are deployed using the ActiveMatrix BPM Administrator. Generating deployment artifacts is now done using the Deployment menu when right-clicking a project in TIBCO Business Studio - BPM Edition.

Importing Pre-Version 5.x Projects

TIBCO Business Studio - BPM Edition contains an Import wizard that you can use to import pre-version 5.x projects into TIBCO Business Studio - BPM Edition version 5.x and newer.

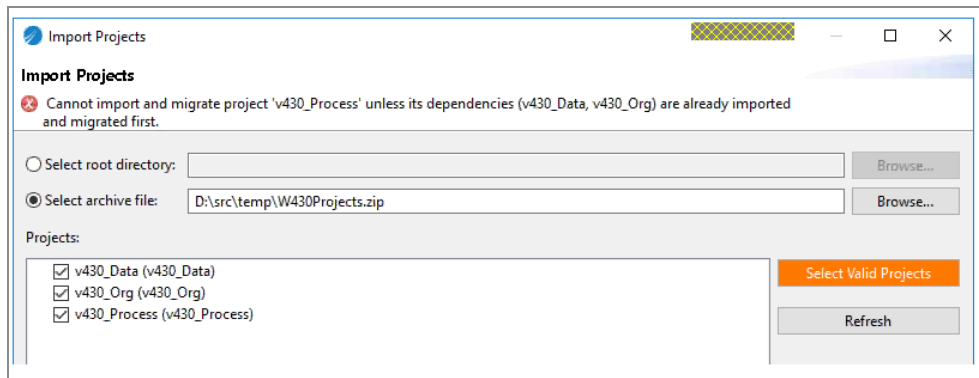
When projects are imported into TIBCO Business Studio - BPM Edition, it's important that lower-level projects - which other projects are dependent upon - are imported first. For instance, if you have a process project that is dependent on an organization model project, the organization model project must be imported first, then the process project can be imported.

The Import wizard in TIBCO Business Studio - BPM Edition assists in importing projects with dependencies by indicating that there are projects that require migration whose dependencies have not already been imported, and allowing you to filter these out of the selection. After the lower-level projects are imported, you use the Import wizard again to import the projects whose dependent projects have already been imported. This process is then repeated for each tier of projects until all have been imported.

Procedure

1. Start TIBCO Business Studio - BPM Edition with a blank workspace.
2. Select **File > Import > Existing Studio Projects into Workspace**.
3. On the Import Projects dialog box, select either **Select root directory** or **Select archive file**, then browse to and select the project folder or archive file for the pre-version 5.x project.

If the project folder or archive contains multiple projects with dependencies, a message is displayed stating that it cannot import a project unless its dependencies are imported first. For example:



4. Click **Select Valid Projects**.

The Import wizard deselects the projects requiring migration that have dependencies.

5. Click **Finish**.

The Import wizard imports the projects that were selected on the Import Projects dialog box.

6. Resolve any issues that are flagged by problem markers.

This can make the task of fixing problems more manageable by avoiding problems raised for higher-level projects that are actually caused by problems in lower-level projects. See [Post Migration Tasks](#).

7. Repeat steps 2 - 6 until all of the projects in the archive are imported.

Post migration tasks

After you have imported your pre-version 5.x TIBCO BPM Enterprise projects, any features contained in your projects that are no longer supported might have problem markers.

You must perform some post migration tasks to fix these problem markers. See the following subsections.

Business Object Model Data Types

Pre-version 5.x ActiveMatrix BPM uses JavaScript representation for business object model data types.

You can use the problem markers to help you fix any issues with your data type fields.

- Some BOM primitive types are no longer supported: Attachment, DateTime, Duration, ID, Integer, Object. Any references to these types cause problem markers.
- For simple type fields, the generic Object type is no longer supported for attributes. In JSON, use a text attribute and place your escaped JSON in the text attribute instead. In other words, this replaces XSD:any.
- Integers are now shown as a fixed point number with zero decimal places. Any integer types are automatically converted on import.
- There are two types of number:
 - Number.
 - Fixed point number.

Both number types are implemented as JavaScript Numbers with extension functions to help with various interactions. Fixed point numbers allow you to specify a fixed number of decimal places and values assigned to these are rounded appropriately.

- Sub-packages are no longer supported. If your business object model has sub-packages, you must move all your classes up to the top-level package.
- For other restrictions, use the problem markers to help you reach a solution.

WSDL Activities

ActiveMatrix BPM version 5.x+ does not invoke or expose service operations that external clients or other processes can invoke using WSDL documents. Content and configuration relating to WSDL documents is removed on importing pre-version 5.x ActiveMatrix BPM projects to version 5.x TIBCO Business Studio.

- Version 5.x+ TIBCO Business Studio processes do not support activities with a service type of **Web Service**.
 - To invoke business processes from business services or pageflow processes,

replace the web service tasks with a call sub-process activity and configure the sub-process activity to be invoked asynchronously.

- To invoke external clients, you must use REST services. To do this, on the **General** tab in the **Properties View** for your activity, set **Service Type** for your activity to **REST Service**.
- Incoming message activities (in other words, all those activities involved in exposing processes as web services) are no longer supported. Instead, you can use new incoming request activities (formerly receive task and intermediate untriggered or no-type events) in your process and trigger these using the generic runtime API services.
- Incoming request-with-reply patterns are not currently supported. As an alternative you might consider using an asynchronous send-two-ways pattern. The request is made from an external system via an Incoming request event (using the generic runtime API services), the 'reply' is implemented as a REST service send task that invokes a service in the external system to send the required reply data.
- Correlation is now supported using either process-instance-id (same as pre-version 5.x) or business data. It means that you can now correlate certain incoming request event or task triggered by the Process REST API to an individual process instance by matching the correlation-data values in the input payload of the Process API with the correlation-data values in the interface properties associated with the incoming request event or task.
See [Controlling Flow from an External Application](#) for more details.

Other Service Tasks

Some service types are no longer available for activities with a type of **Service Task**.

Service Type	Description
Database	You must use REST services. To do this, on the General tab in the Properties View for your activity, set Service Type for your activity to REST Service .
Java	You must use REST services. To do this, on the General tab in the Properties View for your activity, set Service Type for your activity to REST Service .

Service Type	Description
Document operation	You can control documents using the ActiveMatrix BPM Client or the generic runtime API.
Decision	You can use JavaScript, or in complex scenarios, create a REST service to perform the same calculations.

Data Mapping

Mapping data now only supports the Data Mapper grammar. When you import your pre-version 5.x projects, your data mappings are automatically converted from JavaScript mapping to Data Mapper mapping grammar. After migration, problem markers are generated if you have any scripted mappings that do not have explicit return statements.

Participant Configurations

As incoming message activities and WSDL invocation are not supported, you should remove your web service provider participants. Secondly, for service tasks that invoke REST services, the participant configuration has changed. On import, the old participant configuration is removed and you must add the new configuration for these participants.

Multi-Instance Sub-Processes

There is a change in the processing of multi-instance sub-processes.

In multi-instance sub-process call tasks instances, the individual sub-process can complete in any order. Output mappings can be made from a non-array sub-process parameter into an array field in the calling process. The target array field is populated from the values returned by the sub-process instances.

In pre-version 5.x ActiveMatrix BPM, regardless of the order in which the sub-process instances completed, the output value was assigned into the target array element *in the order in which the sub-process instances were started*. In other words, the return value from

the first sub-process instance was always assigned to the first target array field element, the second sub-process instance output value was always assigned to the second target array field element. If the sub-process instances completed in a different order than they were started then the target field had null values assigned to their corresponding target array field elements.

In version 5.x+ this is no longer the case. The target array field elements are now assigned in the order in which the sub-process instances complete. In other words, if the third sub-process instance completes first then its output value is the first element in the target array field.

Therefore, for single to multi-instance output mappings, the target list must be configured as **Append to Target List**. To enforce this, a problem marker is raised on any multi-instance sub-process single to multi-instance output data mapping, unless you configure it as **Append to Target List**.

If your version 5.x BPM application is designed to rely on the list element index corresponding to the specific order in which the sub-process instance was invoked, this no longer works in version 5.x+.

Scripting

There are some changes to scripting in version 5.x, for example, process data fields are now wrapped in a data object and factory methods have changed. Some differences are automatically fixed for you when you import your projects for the first time. Depending on the complexity and nature of your scripts, you might need to fix some scripts manually by addressing the problems markers raised on those scripts.

Retrieving and Setting Process Relevant Data

At run-time, instances of BOM classes are created to represent particular instances of the generic BOM class. These instances are referred to as Business Objects. You can retrieve and set business object attributes, using business data scripting. For example, in pre-version 5.x, you could have the following:

```
var custName = customerInstance.name;
```

and you could set the value of the Name business object property as follows:

```
customerInstance.name = "Clint Hill";
```

In version 5.x, process relevant data are now wrapped in a data object. To set the same value of the Name process parameter or data field property, you must now do the following:

```
data.customerInstance.name = "Clint Hill";
```

Business Object Creation by Factory

i Note: This applies to server-side scripting only. Factory usage in client-side scripting in pre-version 5.x is the same as version 5.x.

Business Objects are created by factories. There is a factory method for each class within the BOM (for example, `createCustomer`, `createOrder`, and so on). For example, in pre-version 5.x, you could have the following:

```
com_example_ordermodel_Factory.createOrder()
```

In version 5.x all BOM factory methods are accessed via a single factory object. To use the same factory method in version 5.x, you must now do the following:

```
factory.com_example_ordermodel.createOrder()
```

i Note: You can assign the values of two BOM attributes to the same BOM class instance object. In this situation, changes are made by reference, and the changes are reflected in both places. However, once the script is complete, the data are saved and that single value is stored and, thereafter, behaves as two independent objects. This differs from the behavior in pre-version 5.x, where a business object instance can only be contained by a single container at any given instant.

Working with Enumerated Types (ENUM)

i Note: This applies to server-side scripting only. In client-side scripting, enumerated types are handled in version 5.x in the same way they were handled in pre-version 5.x.

If you want to categorize objects as different types, instead of using a number or a free format string, you can use an Enumerated Type (ENUM). In pre-version 5.x, you could use unqualified names to define ENUMs in scripts. In version 5.x, this is no longer supported. You must use a fully qualified name (qualified by the package name) for the enumerations in the script. The qualified name of enumerations to be used in scripts is similar to the

factory names, with the qualified name formatted to replace dot '.' by '_' an underscore character.

Secondly, the fully qualified name must now be wrapped in a `pkg` object. For example:

```
com_example_shared_ColorEnum.RED
```

becomes

```
pkg.com_example_shared_ColorEnum.RED
```

Case Data Access Methods

Prior to ActiveMatrix BPM version 5.x, specific JavaScript classes were dynamically generated to handle specific BOM case classes. ActiveMatrix BPM version 5.x has been changed to provide a statically defined (that is, it is always there) JavaScript class that is used for all case classes, where the case class is identified in the method parameters using its fully qualified name (that is, "<bom_package_id>.<case_class_name>").

Upon migration from pre-version 5.0, scripts are automatically changed to use the equivalent new method, assuming there is an equivalent method, and that the referenced Business Data projects have been imported.

ScriptUtility

`ScriptUtil` provided methods to create various types of object, to modify `Duration` objects, and to serialize business objects into or deserialize them from their XML representation in pre-version 5.x. These are no longer provided. `ScriptUtility` should now only be used for copying objects. See [ScriptUtil](#) for more information.

Some Script Utilities No Longer Supported

The following script utilities are no longer supported:

- `DateTimeUtil`. These methods were used to create date, time and date-time objects of the types used in pre-version 5.x.
- `DataUtil`. `DataUtil` provided a single method that allowed you to create a `List` object for use in scripting in pre-version 5.x.

Regular Expression Patterns for Text Fields No Longer Supported

In pre-version 5.x, when defining a Primitive Type, text fields could be constrained to match certain patterns. Regular expression patterns are no longer supported.

List Objects No Longer Supported

In pre-version 5.x a process data field specified as an array could refer to multiple instances of a business object. Similarly, a BOM class attribute could refer to multiple instances of a business object. Multiple instances of business objects used to be handled using a `List` object. `List` objects are no longer supported. In version 5.x, you must use JavaScript Arrays instead.

WorkManagerFactory and Process Classes

`WorkManagerFactory` and `Process` classes have the following changes:

- `WorkManagerFactory` is renamed as `workManager`.
- `workManager` and `process` objects are now wrapped in a `bpm` object.

Example 1

```
var theOfferSet = WorkManagerFactory.getWorkItem().getWorkItemOffers();
```

becomes

```
var theOfferSet = bpm.workManager.getWorkItem().getWorkItemOffers();
```

Example 2

```
var myActivityLoopIndex = Process.getActivityLoopIndex();
```

becomes

```
var myActivityLoopIndex = bpm.process.getActivityLoopIndex();
```

Forms

There are some changes to forms in version 5.x. ActiveMatrix BPM, for example, there are some changes to get and set methods. Unless stated otherwise, when importing a pre-

version 5.x project into version 5.x TIBCO Business Studio - BPM Edition that use scripts that contain any of the following features, problem markers are generated that you must fix.

There are also changes to business object model data types and scripting that impact forms, for example changes to business object model data types. See [Business Object Model Data Types](#) and [Scripting](#).

Scripting API changes

- Script actions, computation actions and validation scripts are all migrated automatically. External JavaScript files must be migrated manually.
- There are some changes to get and set methods. Previously, properties used to be read or written using a get or set method respectively. Now, such get/set method pairs have been replaced by direct properties, which can be used as lvalues (to write them) or rvalues (to read them). Read-only properties never had a set method, and such properties cannot be used as lvalues. For a full list of the new properties and remaining methods, see "API for Scripting" in the *TIBCO Business Studio™ - BPM Edition Forms User's Guide*. `<obj-expr>.getXxx()` and `<obj-expr>.setXxx(<value-expr>)` calls are automatically migrated to use property access syntax `<obj-expr>.xxx` and `<obj-expr>.xxx = <value-expr>` respectively.
- Multi-valued structured-type properties are now JavaScript Array objects, not List objects. Most List method calls are automatically migrated to use the corresponding Array API. See `List.iterator()` below for exceptions to this.
- Multi-valued simple-type values can no longer be assigned.
- In pre-version 5.x, field values could be accessed using `f.controlName`. In version 5.x, `f.xxx` and `f[<expr>]` are replaced by `control.xxx.value` and `control[<expr>].value`, respectively.
- In pre-version 5.x parameter values could be accessed using `p.paramName`. In version 5.x, `p.xxx` and `p[<expr>]` are replaced by `data.[expr]` and `data[xxx]`, respectively.

CSS Best Practice

Version 5.x ActiveMatrix BPM now enforces CSS best practice, by removing all the deprecated configuration properties that used to control rendering of form components by non-CSS means. Following import of a pre-version 5.x project, any such properties (for example, font size, weight, color, and so on,) must be re-applied using pure CSS

techniques. As part of this, Font and Layout property tabs and their associated property sections have been removed.

Resource/Model Name Length

You can no longer use long names for your resource/models. If you have used long names, when importing to version 5.x, you might receive problem markers that you must fix.

Duration Forms Component Type No Longer Supported

In pre-version 5.x ActiveMatrix BPM, the duration control allowed the users to specify a duration using a configurable set of temporal units. The duration control is no longer supported in version 5.x.

Custom Controls API changes

The API for custom controls has changed to use direct property access instead of `get/set` method calls. Custom controls supply a JavaScript wrapper script which (in common with all external JavaScript resources) is not migrated. You must migrate the JavaScript wrapper script manually. See "Custom Controls" in the *TIBCO Business Studio™ - BPM Edition Forms User's Guide*.

List.iterator()

For `List.iterator()` and all calls on the resulting `Iterator` object such as `hasNext()`, `next()`, `add()`, `remove()` and so on, are not automatically migrated. You must manually refactor any code that uses `List.iterator()`.

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

The [TIBCO Product Documentation](#) website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The following documentation for this product is available on the [TIBCO Business Studio™ - BPM Edition Product Documentation](#) page:

- *TIBCO Business Studio™ - BPM Edition Release Notes*
- *TIBCO Business Studio™ - BPM Edition Installation*
- *TIBCO Business Studio™ - BPM Edition Application Designer's Guide*
- *TIBCO Business Studio™ - BPM Edition Modeling Guide*
- *TIBCO Business Studio™ - BPM Edition Forms User Guide*

How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the [TIBCO Support](#) website.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to [TIBCO Support](#) website. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to [TIBCO Community](#).

Legal and Third-Party Notices

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, Business Studio, and TIBCO Business Studio are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SOFTWARE GROUP, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of Cloud Software Group, Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2015-2023. Cloud Software Group, Inc. All Rights Reserved.