TIBCO ActiveMatrix BusinessWorks™ SmartMapper

User's Guide

Software Release 6.0
November 2013

Two-Second Advantage®

Important Information

Contents

Figures

Tables

Preface

This user's guide explains how to use TIBCO ActiveMatrix BusinessWorks SmartMapper with TIBCO BusinessWorks™.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix BusinessWorks SmartMapper Documentation

The following documents form the TIBCO ActiveMatrix BusinessWorks SmartMapper documentation set:

- *TIBCO ActiveMatrix BusinessWorks SmartMapper Installation Guide* Read this manual for instructions on site preparation and installation.
- *TIBCO ActiveMatrix BusinessWorks SmartMapper User's Guide* Read this manual for instruction on using the product.
- *TIBCO ActiveMatrix BusinessWorks SmartMapper BUI User's Guide* Read this manual for instructions on managing relationship data in a web-based user interface.
- *TIBCO ActiveMatrix BusinessWorks SmartMapper Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Product Documentation

You may find it useful to read the documentation for the following TIBCO products:

- TIBCO Designer™
- TIBCO Administrator™
- TIBCO ActiveMatrix BusinessWorks™
- TIBCO Rendezvous®
- TIBCO Enterprise Message Service™
- TIBCO Hawk®
- TIBCO Runtime Agent™

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
ENV_NAME *TIBCO_HOME*	TIBCO products are installed into an installation environment. A product installed into an installation environment does not access components in other installation environments. Incompatible products and multiple instances of the same product must be installed into different installation environments. An installation environment consists of the following properties: • **Name** Identifies the installation environment. The name is appended to the name of Windows services created by the installer and is a component of the path to the product in the Windows Start > All Programs menu. This directory is referenced in documentation as *ENV_NAME*. • **Path** The folder into which the product is installed. This folder is referenced in documentation as *TIBCO_HOME*. The default value of *TIBCO_HOME* depends on the operating system. For example, on Windows systems, the default value is C:\tibco.
SmartMapper_HOME	TIBCO ActiveMatrix BusinessWorks SmartMapper is installed into a directory within *TIBCO_HOME*. This directory is referenced in documentation as *SmartMapper_HOME*. The default value of *SmartMapper_HOME* depends on the operating system. For example, on Windows systems, the default value is C:\tibco\smartmapper\6.0.
code font	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use MyCommand to start the foo process.
bold code font	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type the username **admin.** • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled: MyCommand [**enable** | disable]

Table 1 General Typographical Conventions

Convention	Use
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO BusinessWorks Concepts*. • To introduce new terms. For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: MyCommand *pathname*
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
[ ]	An optional item in a command or code syntax. For example: MyCommand [optional_parameter] required_parameter
|	A logical OR that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: MyCommand para1 | param2 | param3

Table 2 Syntax Typographical Conventions

Convention	Use
{ }	A logical group of items in a command. Other syntax notations may appear within each logical group. For example, the following command requires two parameters, which can be either param1 and param2 or param3 and param4: MyCommand {param1 param2} | {param3 param4} In the next example, the command requires two parameters. The first parameter can be either param1 or param2 and the second can be either param3 or param4: MyCommand {param1 | param2} {param3 | param4} In the next example, the command can accept either two or three parameters. The first parameter must be param1. You can optionally include param2 as the second parameter. And the last parameter is either param3 or param4. MyCommand param1 [param2] {param3 | param4}

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts, a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access All TIBCO Documentation

After you join TIBCOmmunity, you can access the documentation for all supported product versions here:

http://docs.tibco.com/TibcoDoc

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support as follows:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 Introduction

This chapter gives an overview of TIBCO ActiveMatrix BusinessWorks SmartMapper.

Topics

Product Overview

TIBCO ActiveMatrix BusinessWorks SmartMapper is focused on addressing three problems common to solutions that are part of enterprise application integration (EAI):

- Lookups
- Cross-Referencing
- Configurable Business Rules

Lookups

Any two applications are likely to have different code sets for things like currency, countries, status codes, and so forth. These code sets generally represent pick lists in applications. As the number of applications increase or more packaged applications are thrown in the mix, it becomes prohibitively difficult to make these code sets all the same. As the list expands, it also becomes unwieldy to hardcode these lists into maps; therefore, the majority of these lookups are done in databases.

Database lookups introduce a number of problems:

- Lookups are difficult to write, as they require advanced programming skills.
- To save time, the data models become oversimplified, which makes their growth difficult over time.
- Database access can be very slow.

Cross-Referencing

The most common integration need in EAI is data synchronization. Data synchronization typically requires the development of three things for each object:

- A mapping, to transform data from one schema to another. Three mapping models are needed if there is a common object model.

  See Using a Common Data Model: Three Mapping Models on page 7 for more information on this case.

- A process, to define the business rules that control how data is processed.

- Cross-referencing, which allows users to
  - Match an entity in one application with its semantic equivalent in another application. For example, an account ID in Siebel might be labeled acme-123 and the same account in SAP might be labeled 00001890.
  - Follow a previously created link to perform updates or to preserve references.
  - Match static data such as codes or a pick-list of data among applications. For example, a pick list in one system may use USD and another system pick list may use Dollars.

Configurable Business Rules

Through the Data Models and Cross-Referencing Functions, users can do the following:

- Set thresholds in a process, and make these thresholds configurable from the administrative GUI rather than by redoing the entire process.

  For example, a business may choose that all purchase orders over $10,000 must be manually approved. If the business now chooses to change that threshold to $5,000, they only need to change the value in the database through the SmartMapper GUI.
- Set closed loop integration. TIBCO ActiveMatrix BusinessWorks processes can set the values required by SmartMapper. For example, you can set stocking levels in BusinessWorks, and have SmartMapper incorporate these levels into its processes.

How TIBCO ActiveMatrix BusinessWorks SmartMapper Addresses Business Problems

TIBCO ActiveMatrix BusinessWorks SmartMapper offers a unique set of features that both simplify and enhance the process of setting up, deploying, using, and managing cross-referencing services for data synchronization. These features can dramatically reduce the time and cost of integration projects.

The following is an overview of what TIBCO ActiveMatrix BusinessWorks SmartMapper offers:

- **Speed** High-performance cache, parallel execution, and distributed server options ensure that lookups are not the bottleneck.
- **Simplicity** A graphical user interface models relationships, applies them to maps, and even generates the tables without the need for coding. This dramatically reduces both the development time and the total cost of ownership.
- **Power** Modeling the right relationships the first time rather than forcing them into a one-size-fits-all model saves endless extra steps and workarounds.
- **Flexibility** Neither the model nor the processes is tied to the deployment data store options, which allows the system to grow as a company's needs grow.

This additional set of features is built on the power of BusinessWorks and offers the same ease-of-use, standards, and lifecycle management capabilities that are available in BusinessWorks.

Product Features

TIBCO ActiveMatrix BusinessWorks SmartMapper includes the following major features:

- Cross-reference activities, which are used within a BusinessWorks process. Cross-reference activities allow you to look up, create, update, and delete mappings, and to generate keys.
- SmartMapper Wizard, which allows data translation in the context of a hierarchical object.
- Batched approach, which allows lookups to be processed efficiently and writes to be all-or-nothing.
- Non-key attributes. Support for a non-key based Dynamic Lookup activity.
- The association relationship type where there are only two participants. The relationship type is set on the Relationship resource.
- Composite key lookups: multiple attributes can be used as search criteria at the same time to perform lookups, which might produce multiple results.
- Use of XML standards such as XSD for definitions and XPath for attributes
- Server, server-less, or cache only
- Simplified correlation of data from multiple sources
- Leverages and features all of the deployment, administration, security and management capabilities of BusinessWorks.

Bulk Load and Bulk Extract Activities

- Bulk extract of data. The Bulk Extract activity enables the bulk extraction of cross-referencing data for reporting purposes. You can query entity or relationship data by providing an entity name or relationship name. You can also filter the results of the Bulk Extract activity.
- Dynamic loading and dynamic query. The Bulk Load activity allows you to load SmartMapper data and the Dynamic Lookup activity allows you to query SmartMapper data.
- Creation and modification timestamps for records can be included in the Dynamic Lookup and Bulk Extract activities. A participant level timestamp for the relationship-based Bulk Extract activity is now available.

Available through TIBCO Designer GUI

- Zero-coding approach with SmartMapper Wizard-driven graphical entity relationship modeling and mapping. This approach enables lookups, new key generation, and maintenance of the mapping model.
- Entity relationship modeling. This allows you to model objects and keys, and to link modeled objects through relationships.

Available through TIBCO Administrator

- Use of TIBCO Administrator to maintain data related to rules and cross-reference services

Performance and Scalability

- High-performance caching mechanisms for optimal throughput and scalability.
- Queries are performed in parallel.
- Cross-referencing engine shared among multiple instances of BusinessWorks (ActiveMatrix BusinessWorks SmartMapper Enterprise Server only). Shared cross-referencing takes place in the Maximize Throughput scenario. For further information, see Maximize Throughput Scenario (Many Small Objects) on page 10.
- Shared multiple adapter instances among one instance of BusinessWorks. This is the Minimize Latency scenario. See Minimize Latency Scenario (One Large Batch) on page 11.

Complexity of Relationships

- Multiple keys
- The same object may participate in many relationships
- Ability to handle complex object relationships: one-to-many, many-to-one, many-to-many, non-key data

Compatibility

For information on which platforms this product is compatible with, see the readme file.

Using a Common Data Model: Three Mapping Models

A common data model is used when data is mapped into a canonical data format such as OAG.

See http://www.openapplications.org.

When using a publish and subscribe /event-driven architecture, a source does not necessarily know what its destinations are. Further, if multiple applications are connected to the network as peers, it is helpful if each application does not have to know about the internal data formats of all of the others. In this case, you can set up one standard common data model and have all the sources map to and from that model.

Generally, when performing cross-referencing between two applications with a common data model, the following three mapping models are necessary:

- **Publish mapping** This mapping goes from a source application to a common data model. It can be used to generate primary keys if the objects have not yet been created in the cross-reference mapping tables.

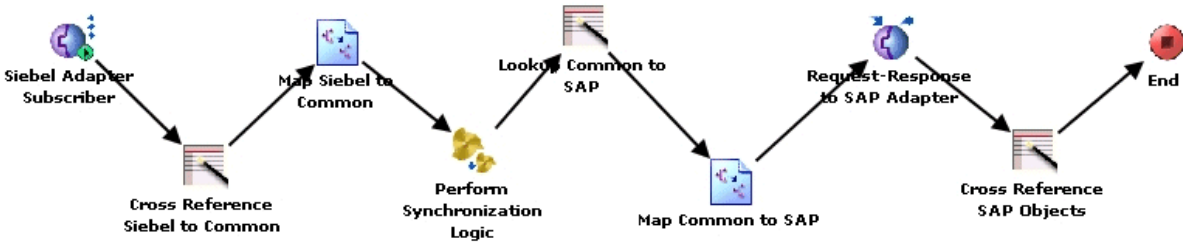
  The sample projects address the publish mapping only, but the software can be used to build any one of these mappings.

- **Request mapping** This mapping maps data from a common data model to a target application. In request mapping, if a primary key lookup fails, it usually does not matter because the adapter creates a key in the application. The target adapter generates these entities. For example, when you create a customer in SAP, request mapping will generate a customer ID, which is returned in the response from the SAP BAPI.

- **Response mapping** After a key is successfully created into the target application, the newly-generated keys must be matched to the common keys.

Figure 1 shows all three mapping models.

Figure 1 Three Mapping Models in Action



Publish mapping maps an entity (say Customer_Number) from the Siebel adapter to the common model in TIBCO ActiveMatrix BusinessWorks SmartMapper. A common ID can be generated in the common model. From there, the mapper takes the entity as defined in the common model and performs the synchronization logic.

Request mapping uses the synchronization logic results, and generates a request to the SAP adapter. If SAP uses external numbering, then, if necessary, during Request Mapping, a key can be generated that will correspond to a SAP identifier.

Response mapping establishes the link between the common ID and the one generated by the SAP application.

Architecture

There are as many ways to set up TIBCO ActiveMatrix BusinessWorks SmartMapper as there are ways to set up BusinessWorks. However, the following are some of the more likely scenarios:

- **Development** In this case, all the TIBCO ActiveMatrix BusinessWorks SmartMapper components run on one machine. The project is undeployed. The data store is a local database or a file system.
- **Small project** In this case, TIBCO ActiveMatrix BusinessWorks SmartMapper is installed on the remote BusinessWorks machine. The database is installed on the remote BusinessWorks machine.
- **Large project or enterprise** In this case, there are multiple BusinessWorks engines that contain TIBCO ActiveMatrix BusinessWorks SmartMapper and a TIBCO Administrator server. The file system is used for static data on BusinessWorks engines. The SmartMapper Enterprise Server runs on the database machine.

Scaling TIBCO ActiveMatrix BusinessWorks SmartMapper

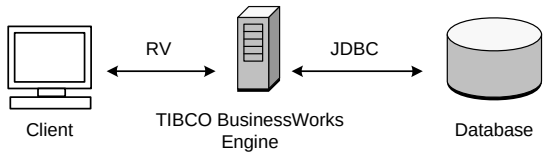
You can scale TIBCO ActiveMatrix BusinessWorks SmartMapper depending on the project requirements.

Figure 2, Figure 3, and Figure 4 show how to scale TIBCO ActiveMatrix BusinessWorks SmartMapper.

Using TIBCO ActiveMatrix BusinessWorks SmartMapper for a Small Project

In the simplest case, you might need only one BusinessWorks engine, as shown in Figure 2. This is a typical small project where a data source is sending messages to BusinessWorks. Mapping is done in process and TIBCO ActiveMatrix BusinessWorks SmartMapper accesses a JDBC data source as needed.

Figure 2 Small Project with One BusinessWorks Engine



Using TIBCO ActiveMatrix BusinessWorks SmartMapper for More Complex Projects

In more complex situations, you need to consider the loads and scale TIBCO ActiveMatrix BusinessWorks SmartMapper depending on whether you are getting many messages or a smaller number of large messages.

For example, imagine 10,000 orders, each with many lines and at least one lookup per line.

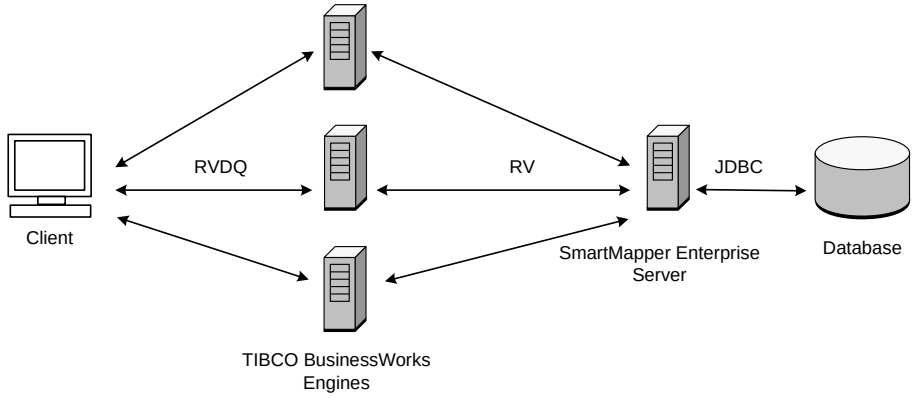
- In one case, each order would be a separate inbound message. See Maximize Throughput Scenario (Many Small Objects) on page 10.
- In the second case, the whole batch of orders is a single message. See Minimize Latency Scenario (One Large Batch) on page 11.

Maximize Throughput Scenario (Many Small Objects)

If you are receiving many messages, tune your system for high message throughput by using multiple BusinessWorks engines. As throughput increases, more BusinessWorks engines are used to handle the load.

The SmartMapper Enterprise Server brokers the transactions to the data store. If you have many messages, you need to scale the number of BusinessWorks engines that you use, but the SmartMapper Enterprise Server can be shared among all the BusinessWorks engines.

Figure 3 Maximized Throughput Scenario with Multiple BusinessWorks Engines



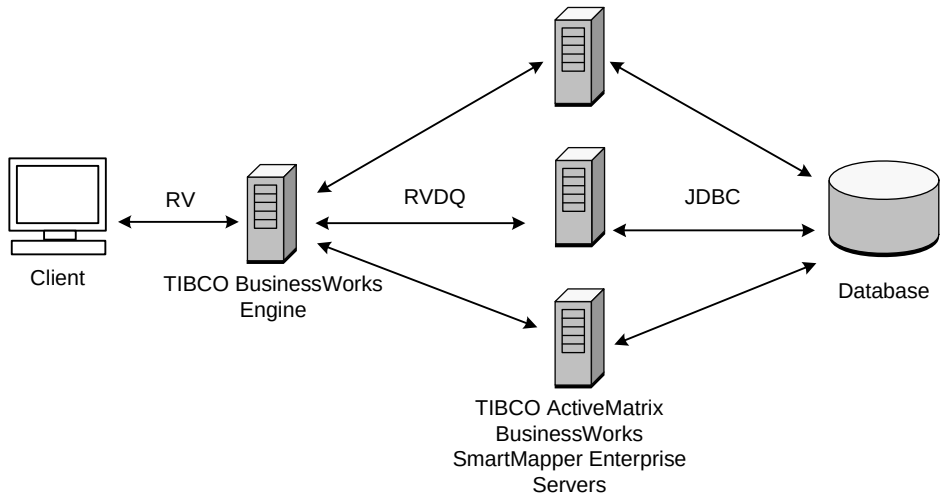
Minimize Latency Scenario (One Large Batch)

If you are receiving large messages, tune to minimize latency by using multiple SmartMapper Enterprise Servers (instead of multiple BusinessWorks engines).

In some cases like legacy, payroll, or supply chain planning systems, huge messages are generated. For example, one message might contain a month's worth of orders. If there is only one message, adding BusinessWorks engines does not help, but the SmartMapper Wizard can still perform lookups in parallel.

Also, by using distributed queues, multiple SmartMapper Enterprise Servers can work against the load. You change the batch size on the SmartMapper Enterprise Server to a reasonable chunk size (100) and set the subscriber to use RVDQ. The RVDQ (Rendezvous Distributed Queue service) facilitates load balancing.

Figure 4 Minimized Latency Scenario with Multiple SmartMapper Enterprise Servers



Chapter 2 Getting Started

This chapter describes the basic steps required to configure and run TIBCO ActiveMatrix BusinessWorks SmartMapper in TIBCO Designer.

Topics

Overview

TIBCO ActiveMatrix BusinessWorks is a scalable, extensible, and easy to use integration platform that allows you to develop and test integration projects. TIBCO ActiveMatrix BusinessWorks includes a graphical user interface, TIBCO Designer, for defining business processes, and an engine that executes the processes.

For detailed information about how to configure processes, see TIBCO Designer documentation, which can be accessed by selecting **Help > Designer Help** from the menu in the TIBCO Designer window.

A typical configuration and deployment procedure includes the following steps:

1. Starting TIBCO Designer
2. Creating a Project
3. Creating an ER Model
4. Creating a TIBCO ActiveMatrix BusinessWorks Process
5. Adding Activities to the Process
6. Testing the Process
7. Deploying a Project

Starting TIBCO Designer

TIBCO Designer is used to configure TIBCO ActiveMatrix BusinessWorks SmartMapper instances. This section introduces how to start TIBCO Designer.

To start TIBCO Designer, execute one of the following platform-specific commands:

- On Microsoft Windows

  From the Start menu, select **All Programs > TIBCO > TIBCO Designer *version_number* > TIBCO Designer**

  or

  From the command line, run *TIBCO_HOME*\designer*version_number*\bin\designer.exe.

- On UNIX

  Run *TIBCO_HOME*/designer/*version_number*/bin/designer.

After performing the previous command, the TIBCO Designer Startup panel is displayed.

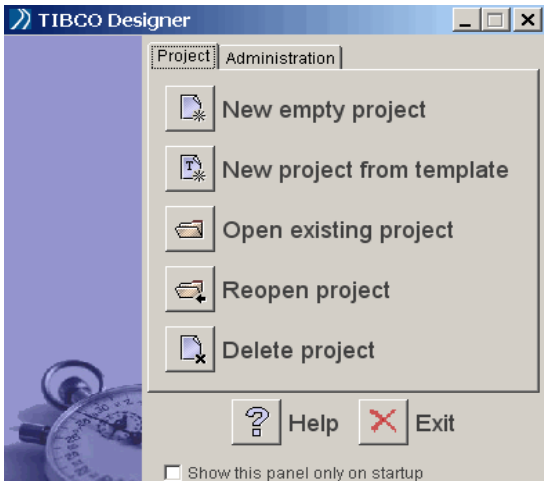
Creating a Project

After starting TIBCO Designer, you can create a project or open an existing project in the displayed Startup panel. A project contains the configuration files to define options used by a runtime plug-in. This section describes how to create a project in TIBCO Designer.

To create a project, complete the following steps:

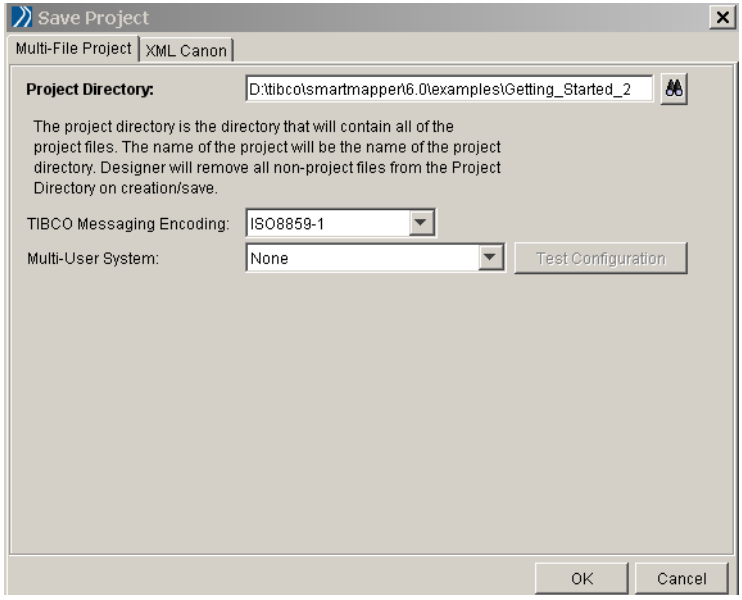
1. Click the **New Empty Project** button in the TIBCO Designer Startup panel, as shown in Figure 5. The Save Project dialog is displayed.

Figure 5 Create a New Project



2. Save the project in the specified location. Click the button next to the Project Directory field in the Multi-File Project tab. Navigate to the location where you intend to save the project and specify a name for the project, as shown in Figure 6. Click the **OK** button.

Figure 6 Save a New Project



TIBCO Designer window is displayed with the newly created project.

Creating an ER Model

After creating a project, you can create an ER Model. The SmartMapper ER Model is a reusable logical data model that contains a set of applications, entities, relationships, and storage service. An ER Model must be created prior to creating a SmartMapper process or using the SmartMapper adapter service.

To create an ER Model, complete the following tasks:

- Task A, Create Applications and Entities, page 18
- Task B, Define a Relationship, page 19
- Task C, Create a Storage Service, page 20

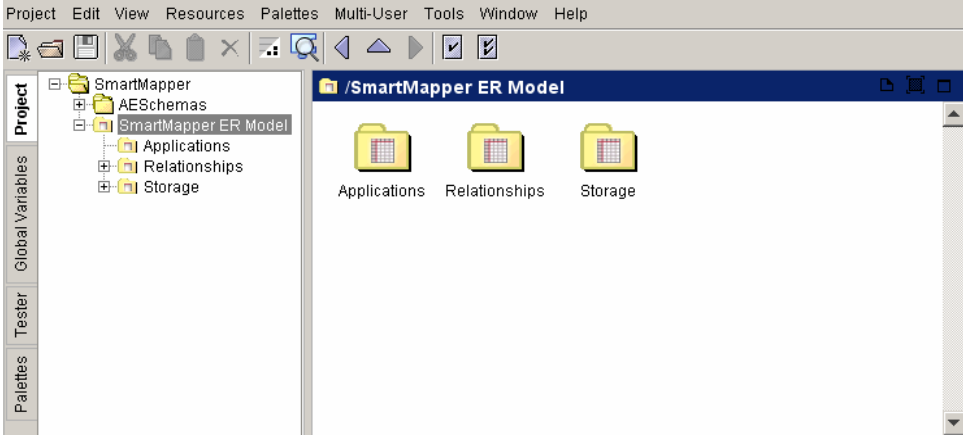
Task A Create Applications and Entities

An entity is an object that is being referenced. See Entity on page 31 for more details.

To create applications and entities, do the following:

1. Open the project you created in the Creating a Project procedure.
2. Click the **Palettes** tab in the Project panel and expand the **SmartMapper Resources** palette.
3. Drag the **SmartMapper ER Model** resource to the Design panel and configure the ER Model in the Configuration panel. See SmartMapper ER Model on page 82 for details about configuring the ER Model.
4. Expand ***Project Name*** > **SmartMapper ER Model** in the Project panel. All the components of an ER Model are displayed, as shown in Figure 7.

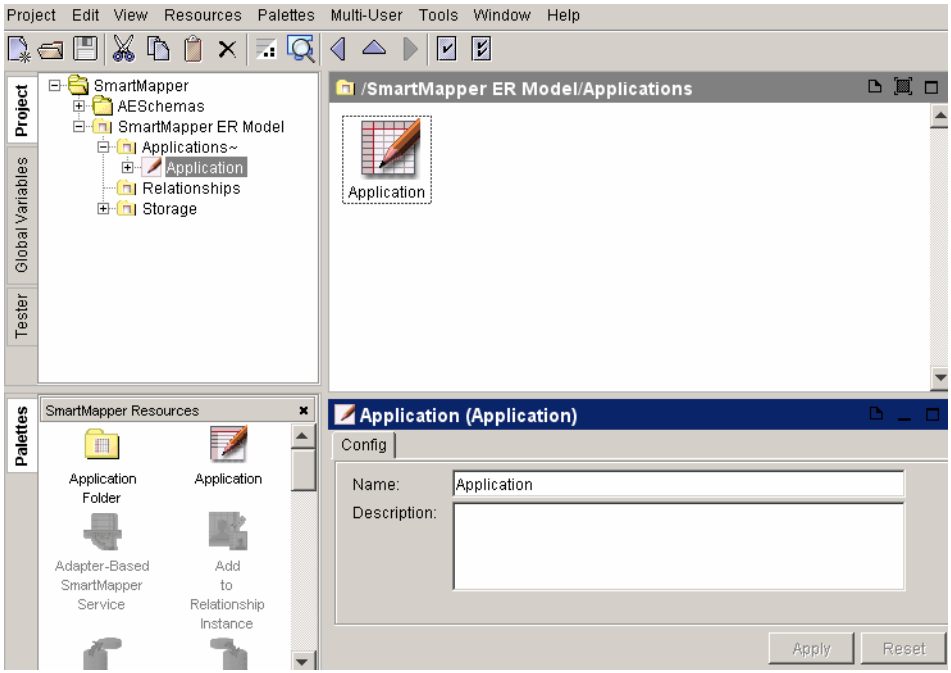
Figure 7 The SmartMapper ER Model



5. Double-click the **Applications** folder in the Design panel, and then click the **Palettes** tab in the Project panel. The Application resource is available in the SmartMapper Resources Palette panel.
6. Drag the **Application** resource to the Design panel and configure the application in the Configuration panel.

   Repeat this step to add more applications.

Figure 8 Adding an Application



7. Double-click the application that you created under the Applications folder, and then click the **Palettes** tab in the Project panel. The Entity resource is available in the SmartMapper Resources Palette panel.
8. Drag the **Entity** resource to the Design panel and configure the entity in the Configuration panel. See Entity on page 84 for more details about configuring the entity.

   Repeat this step to add more entities to the selected application.

Task B Define a Relationship

A relationship is a collection of entities that users want to bind together. See Relationships on page 33 for more details.

To define a relationship, do the following:

1. Click the **Relationships** folder in the project tree, and then click the **Palettes** tab in Project panel. The Relationship resource is available in the SmartMapper Resources Palettes panel.
2. Drag the **Relationship** resource to the Design panel and configure the relationship in the Configuration panel. See Relationship on page 86 for details about configuring the relationship.

   Repeat this step to add more relationships.
3. Double-click the **Relationship** resource in the project tree, and then click the **Palettes** tab in the Project panel. The Participant resource is available in the SmartMapper Resources Palettes panel.
4. Drag the **Participant** resource to the Design panel and configure the participant in the Configuration panel. See Participant on page 87 for details about configuring the participant.

   Repeat this step to add more participants.

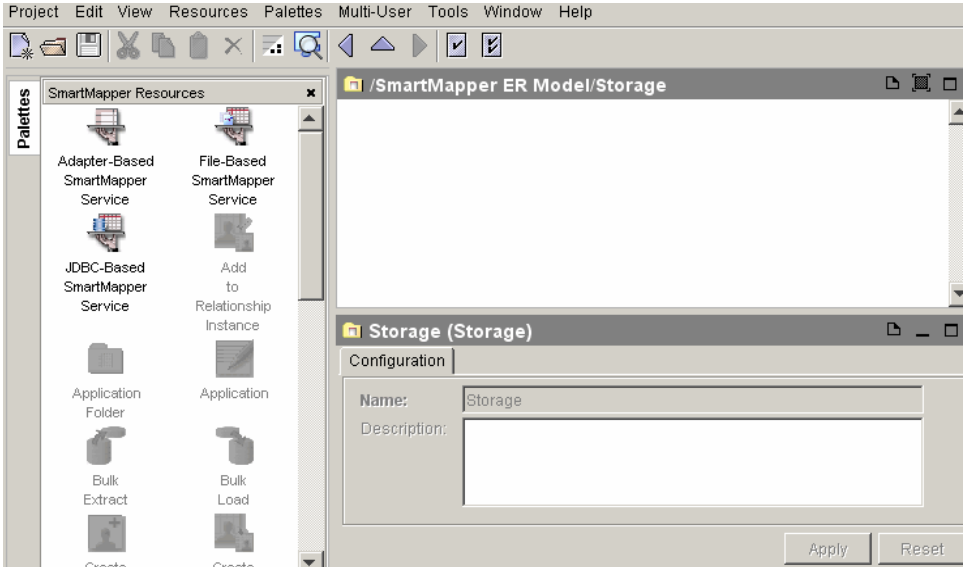
Task C Create a Storage Service

TIBCO ActiveMatrix BusinessWorks SmartMapper provides three options to store the relationship data. See Storage on page 35 for more details. Take the JDBC-Based SmartMapper Service as an example.

To define a storage service, do the following:

1. Click the **Storage** folder in the project tree, and then click the **Palettes** tab in Project panel. All the available storage services are displayed in the SmartMapper Resources Palettes panel, as shown in Figure 9.

Figure 9 Available Storage Services



2. Drag the **JDBC-Based SmartMapper Service** resource to the Design panel and configure the storage service in the Configuration panel. See JDBC-Based SmartMapper Service on page 88 for details about configuring the storage service.

   Note: The Active check box must be selected in the Configuration tab.

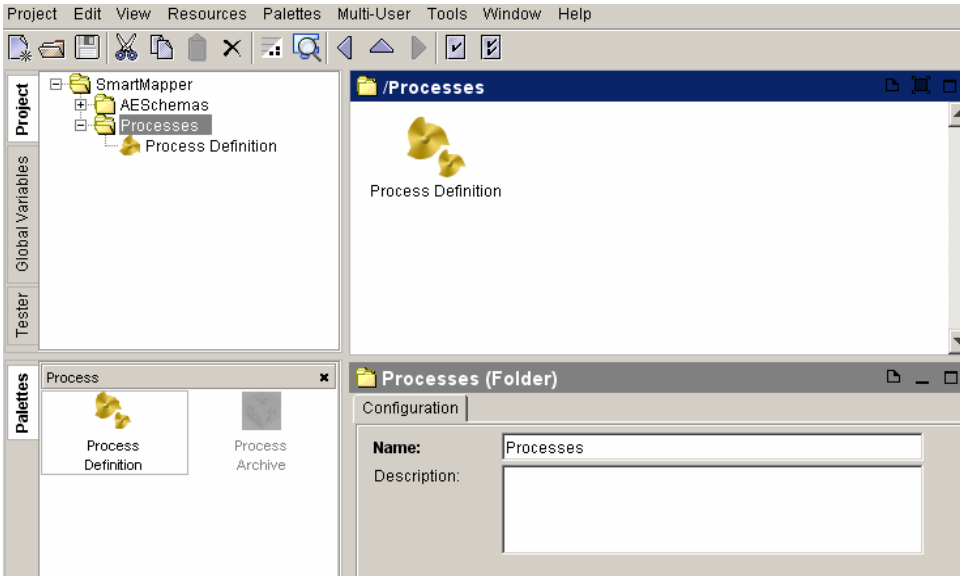
Creating a TIBCO ActiveMatrix BusinessWorks Process

After creating the ER model, you can create a TIBCO ActiveMatrix BusinessWorks process to deal with certain workflows. This section introduces how to create a process.

Complete the following steps to create a process:

1. Right-click the project folder you created in the Creating a Project procedure, and then select the **New Folder** item from the menu that is displayed. Rename the folder to Processes.
2. Click the **Processes** folder. Expand the **Process** palette in the Palettes panel, and drag the **Process Definition** icon to the Design panel, as shown in Figure 10.

Figure 10 Process Definition



3. Configure the process based on your requirements.

   See *TIBCO Designer User's Guide* for more information about how to configure processes.
4. Save the project.

Adding Activities to the Process

After creating a TIBCO ActiveMatrix BusinessWorks process, you can add activities to the created process.

To add activities to the process, do the following:

1. Double-click the created process. The Start and End activities are displayed in the Design panel.
2. Select SmartMapper resources to define the process. Expand the **SmartMapper Resources** palette in the Palettes panel, and drag the activities to the Design panel. You can also add activities from other palettes to the process. For example, File activities.

If the **SmartMapper Resources** palette is not shown in the Palettes panel, select **Palettes > Adapters > SmartMapper Resources** from the TIBCO Designer menu to make the palette visible.

3. Click the  button in the TIBCO Designer toolbar to draw transitions between activities.
4. Configure each activity in the process.

   For more information about SmartMapper activity configurations, see SmartMapper Activities on page 103.

Once the process definition is complete, you can use the test mode tool to test the process. See Testing the Process on page 24 for more details.

Testing the Process

After creating and configuring a process, you can test the process by using the test mode tool. Testing the process helps you check whether the process works properly before deploying it. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information.

Deploying a Project

Before deploying a project, you must create an Enterprise Archive file (EAR file) that contains the configuration for the process definition. You can upload the archive to TIBCO Administrator to deploy the associated application.

Complete the following steps to deploy a configured project:

1. Save the project in the TIBCO Designer window. Make sure the processes saved in the project can be triggered.
2. Create an Enterprise Archive. Select **Tools > Create Project EAR** from the TIBCO Designer menu.
3. Expand the Enterprise Archive you created in Step 2, and then select the **Process Archive** resource. Click the  button in the Processes tab. The Select A Resource dialog is displayed.
4. Expand the folder to select the process definition you want to include and then click the **Apply** button.
5. Build the archive. Click the Enterprise Archive item you created in Step 2 and then click the **Build Archive** button in the Configuration tab. An EAR file is generated and saved in the location specified in the File Location field in the Configuration tab.
6. Deploy the project in TIBCO Administrator. Start TIBCO Administrator and upload the EAR file for the project. Deploy the application and start the process.

   See *TIBCO ActiveMatrix BusinessWorks Administration* for more information about how to deploy a project in TIBCO Administrator.

Chapter 3 Life Cycle

This chapter describes the lifecycle you can expect when working with TIBCO ActiveMatrix BusinessWorks SmartMapper. Read it to get an introduction to the tasks you will face when configuring an Entity Relationship (ER) Model, including activities in TIBCO ActiveMatrix BusinessWorks processes that reference the model.

Topics

Overview

TIBCO ActiveMatrix BusinessWorks SmartMapper allows you to maintain cross-referencing relationships across applications. It provides an easy-to-use graphical user interface for modeling, assembling, deploying, and managing these relationships. Before proceeding, read through the descriptions of the terms in Table 3. Doing so will help you understand the concepts presented in this chapter.

Table 3 Cross-Referencing Relationships Across Applications

Creating Keys	You should know the relevant fields in your application so you can choose your keys well. The choice of keys is an important decision and not one to be taken lightly. The key defines an entity's identity with respect to the entities in other systems. This is further complicated when an entity participates in many relationships or is referenced as a foreign key on other schema. For example, when choosing keys for customers and addresses, consider how they relate to each other, how they relate to their peers in other systems, and how they are referenced on sales orders. You create a key for an entity by adding an attribute to the entity. You must define a key to set up the entity.
Attribute	A field in an entity. Non-key attributes are supported. Certain non-key attributes are also nullable. If an entity has multiple attributes marked as keys, they form a composite key. Non-key attributes, if they exist, are not part of the key.
Composite Key	A composite key is used when an entity needs more than one attribute for unique identification, such as: If a social security number attribute and an email address attribute are keys, and they are placed in one entity, then both are necessary to identify the entity. If order lines are numbered 10, 20, 30, and so on in each non-unique order, then the OrderNumber plus the Line Number attribute form the unique composite key.
Data Synchronization	The processes of replicating data while performing the necessary semantic and syntactic transformations. **Example:** The customer Cisco in Siebel might also be represented in SAP. Both systems generate a unique key according to their own algorithm. You could choose to match these directly, or you could choose their own common format and have all the sources map to and from that model. Cross-referencing is used in data synchronization. For example, you can create a customer and then update that customer. You can create an object (such as a sales order) that depends on the customer. In this case, you use cross-referencing to associate the new object with the customer.

Table 3 Cross-Referencing Relationships Across Applications

Key	A value that represents some object. A value that is a link to data in another location. A key is what an adapter uses to retrieve and uniquely identify a set of information such as a customer or an order. A key can be a single value or a set of values (composite). None of the values can be null or blank. In this software, an entire entity is used as a key to access data. For example, the Siebel_Account entity contains an attribute called AccountId. AccountId is the key which the Siebel adapter uses to retrieve an account. In another example, in an order, the order lines are numbered 10, 20, 30, and so on. This is done in every order, so the line numbers are not unique. However, the OrderNumber attribute plus the Line Number attribute, as part of an entity like Siebel_SalesOrder, form a key that is unique. This is a composite key because it is comprised of more than one value.
Primary Key	The unique key to an object. The most concise way to uniquely identify an object. The primary key to an object must be one (or more) of the attributes of the entity which represents the object.
Foreign Key	A foreign key represents an object other than the object that contains the key. For example, an order entity might contain an order key and a customer key. The order key is the primary key for the order. The order key refers to the order itself, which is a record in the table of orders. The customer key (say customerid) is a foreign key in the order. It refers to the table of customer records, not the table of orders. The customerid foreign key in the order links the order to the customer record. The customerid is the primary key of the customer that placed the order.
Static Data	Data that does not change, or data that is changed manually instead of through a process. Examples of static data are: • State codes • Currency codes • Units of measure TIBCO ActiveMatrix BusinessWorks SmartMapper does not differentiate between static and dynamic data. Therefore, the same TIBCO ActiveMatrix BusinessWorks SmartMapper functionality applies to both.

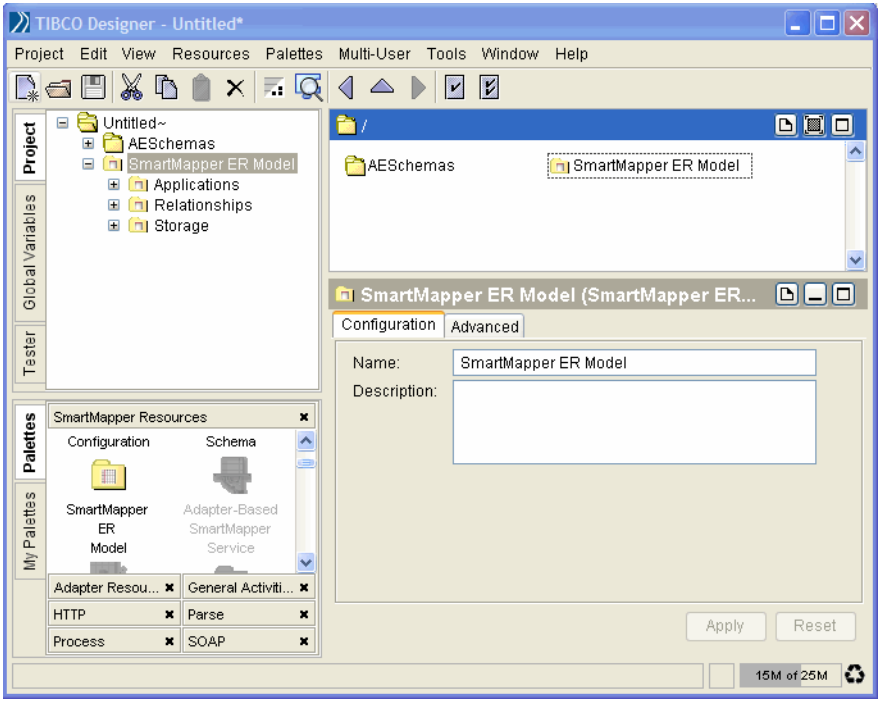
Model Phase

The first step is to set up the ER Model, which is a reusable logical data model that contains a set of applications, entities, and relationships that you configure in TIBCO Designer. See Creating an ER Model on page 18 for details about how to create an ER Model.

In this phase, you model entity relationships and create mappings between applications.

Each ER Model can be assigned to one TIBCO ActiveMatrix BusinessWorks SmartMapper service and to one persistent storage. If multiple ER Models are created, they must be separated. That is, a relationship can only refer to entities in its own ER Model.

Figure 11 ER Model as Presented in TIBCO Designer



The ER Model consists of applications, relationships and storage. These are described in the following sections:

- Applications, page 31
- Relationships, page 33
- Storage, page 35

Applications

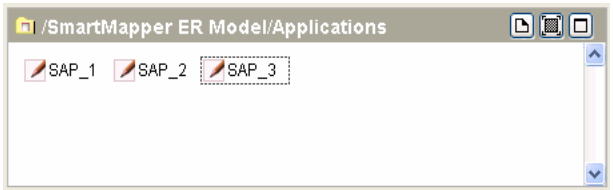
An application is a system with a unique data set and numbering scheme. In TIBCO Designer, application folders are logical containers for entities. For example:

- An instance of an ERP system such as Oracle, Siebel, or SAP
- A code set such as ISO

You can have multiple instances of one application, for example, three SAP applications that all use different numbering schemes. In this case, the three applications must be set up as separate applications in TIBCO Designer.

Note: Only digits, alphanumeric characters, and underscores can be contained in the application name. Besides, a digit cannot be used as the first character of the application name.

Figure 12 Applications Set Up in TIBCO Designer



Entity

In TIBCO ActiveMatrix BusinessWorks SmartMapper, an entity represents something that is being referenced, such as a customer record.

In most cases, an entity is a proxy for an object. It contains the information necessary to retrieve or identify the object.

Entities are composed of one or more attributes. Entities are created and grouped logically in Application folders like SAP, Oracle, and Siebel. Once created, entities are placed in relationships to form participants.

An entity represents a complete key to the object. This is the case whether the entity contains one or many attributes, and whether those attributes are keys or values.

Note: Only digits, alphanumeric characters, and underscores can be contained in the entity name. Besides, a digit cannot be used as the first character of the entity name.

For example, in a list of name-value pairs, name and value are both entities:

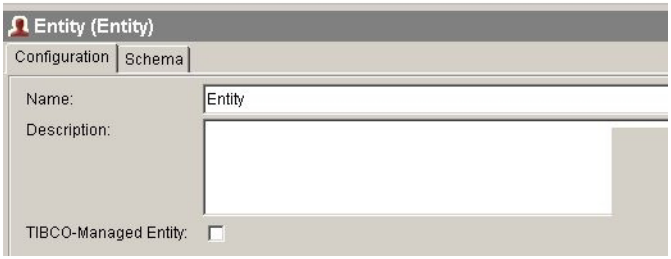
- SAP_customer. This entity might have just one attribute, but the entity is a concept that represents the real customer in the SAP system, which has hundreds of attributes.
- Siebel_account
- ISO country code

- Name and value

The following rules apply to entities:

- Keys have limited data types
- All keys are strings or integers
- Non-key attributes are supported
- Certain non-key attributes are also nullable

Figure 13 Entity Configured in the ER Model



TIBCO-Managed Entity

A TIBCO-Managed Entity is identified using a check box that is available in the Configuration tab of each entity (as shown in Figure 13).

An entity is marked as a TIBCO-Managed Entity when TIBCO ActiveMatrix BusinessWorks SmartMapper is responsible for generating new keys.

For a TIBCO-Managed entity, keys are generated internally rather than assigned, because the entity does not correspond to an external system. These keys are used for a common data model.

Relationships

A relationship is a collection of entities that a user wants to bind together in a particular way. It may also be a way to describe the binding.

Note: Only digits, alphanumeric characters, and underscores can be contained in the relationship name. Besides, a digit cannot be used as the first character of the relationship name.

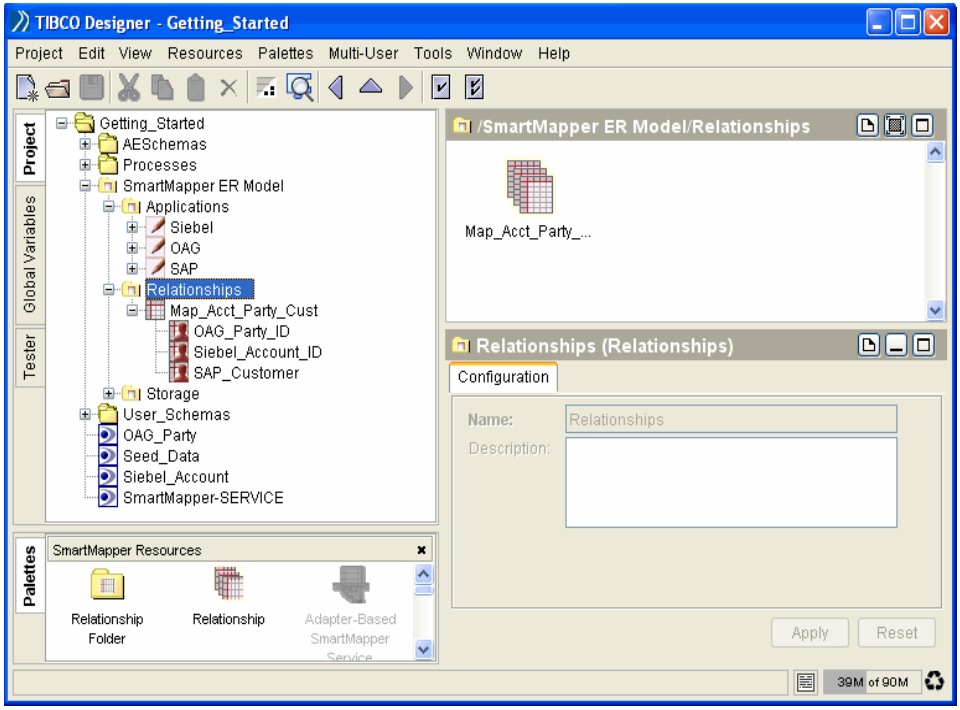
The relationship can be one of the following two types:

- **Identity relationship** In this relationship all participants are peers. See Identity Relationship on page 216 for more details.
- **Association relationship** In this relationship there are only two participants in each relationship. See Association Relationship on page 218 for more details.

An entity may participate in many relationships within one ER Model, but must not play two roles in the same relationship.

For example, one customer relationship contains the entities Siebel_Account_ID and SAP_Customer. These entities play roles in the customer relationship, and the same entities can participate in other relationships. However, the entities cannot each play more than one role in each relationship.

Figure 14 Relationship Defined in the ER Model

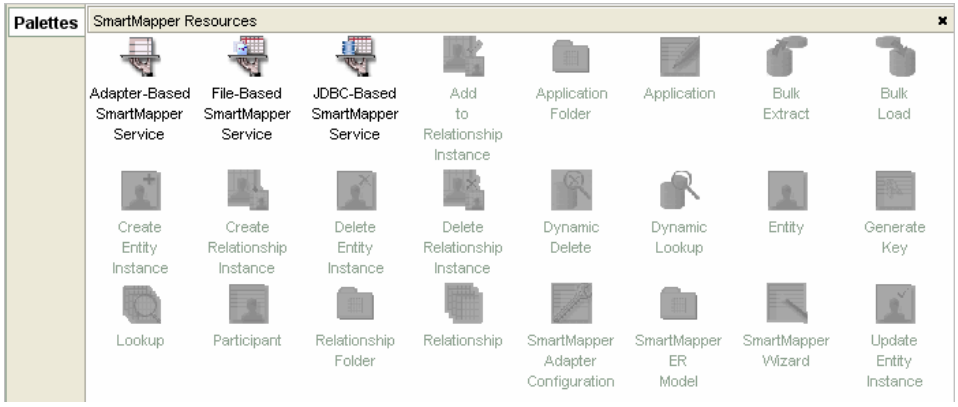


For more information, see Relationships in TIBCO ActiveMatrix BusinessWorks SmartMapper on page 215.

Storage

A storage service defines the back-end implementation and storage characteristics of an ER Model. It specifies whether the data is stored in a file or a database, or is accessed through the Enterprise Server. You can choose one of three storage services: JDBC, File or Adapter-based (Enterprise Server). See Create a Storage Service on page 20 for more details about creating a storage service.

Figure 15 Choosing One of the Three Storage Services



The file-based service is very useful for prototyping and for production where the data is static. Think of the file service as a read-only cache. While it's possible to write to the file, such as for seeding purposes, it is very slow. In addition, there's some risk of data corruption when doing concurrent writes.

A SmartMapper storage service is associated with an ER Model. Assigning an ER Model to a service designates that service as the only data manager for that ER Model.

An ER Model can only be assigned to one storage service at a time. If an ER Model assigned to one service is assigned to another service, it is automatically un-assigned from the first service and assigned to the new service.

A storage option may be chosen after a process is created, but must be chosen before the process is run. Multiple storage options and ER Models may be created. There is a one-to-one correlation between storage options and ER Models.

Assemble Phase

Once you have set up the ER Model, you can bind and configure the components into a TIBCO ActiveMatrix BusinessWorks process. There are two ways to do this, using individual ActiveMatrix BusinessWorks SmartMapper activities, or using the SmartMapper Wizard. See Adding Activities to the Process on page 23 for details about how to add SmartMapper activities to a process.

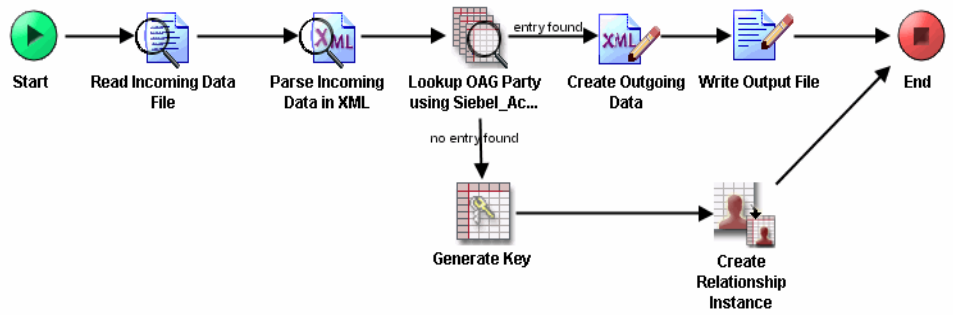
You may need to use the SmartMapper Wizard activity when you are doing a large number of lookups. The SmartMapper Wizard enables many lookups to be processed in parallel, allowing the process to run quickly.

You can bulk extract data from ActiveMatrix BusinessWorks SmartMapper for reporting purposes and filter the results. You can also dynamically load bulk data into ActiveMatrix BusinessWorks SmartMapper using the Bulk Load activity.

Using ActiveMatrix BusinessWorks SmartMapper Activities

Figure 16 shows a completed BusinessWorks process that uses SmartMapper activities. The process reads and parses incoming data, then determines if the key already exists. If it does not exist, a key is generated. The key is used the next time the process is run.

Figure 16 Completed BusinessWorks Process

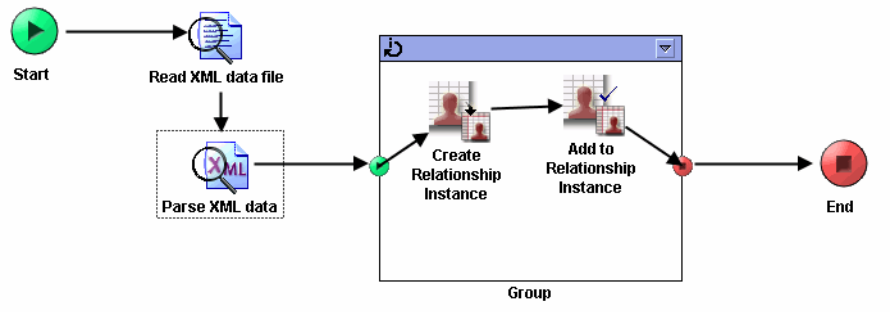


You must also seed the data for the keys in the standard tables. This means importing key data from the clients that will be synchronized using TIBCO ActiveMatrix BusinessWorks SmartMapper.

As shown in Figure 17, the SmartMapper activities can be used to read from the source and populate the data store. The data may also be entered manually in TIBCO Administrator.

See Simple Tutorial: Seed and Lookup on page 169 for details.

Figure 17 SmartMapper Used to Populate the Data Store

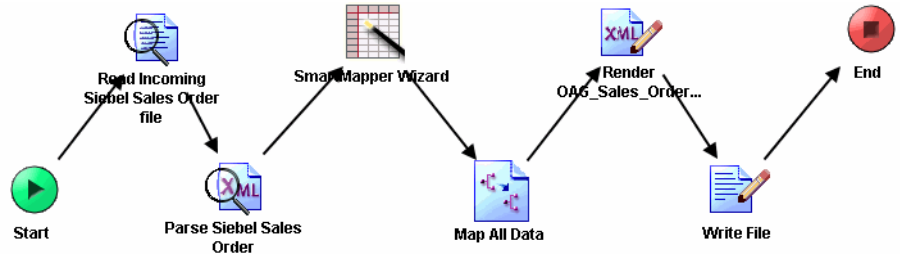


Using the SmartMapper Wizard

The SmartMapper Wizard allows you to select primary keys, foreign keys and other lookup data, and then assign relationships to them using a single activity.

Once this has been accomplished, the SmartMapper Server automatically performs lookups, generates new IDs, and maintains mappings as needed.

Figure 18 Completed BusinessWorks Process that Uses the SmartMapper Wizard



Using the Bulk Load and Bulk Extract Activities

Bulk data loading and extraction are supported using the Bulk Load and Bulk Extract activities.

Figure 19 shows a bulk load process. You must provide the input format as an entity, one participant relationship, or two participant relationship, the ER Model to use, the criteria to load, and the Name or Index. You can also enable verbose reporting.

Figure 19 Bulk Load Process

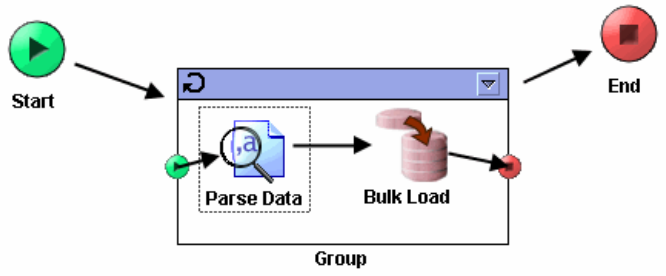
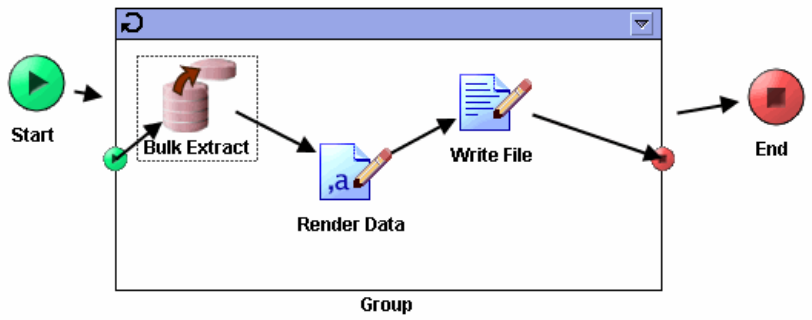


Figure 20 shows a bulk extract process. When defining a Bulk Extract activity, you must provide the output format as an entity, one participant relationship, or two participant relationship, and the ER Model to use.

It is also possible to filter the results of the Bulk Extract activity.

You can specify that the operation be processed in subsets, which allows you to process smaller batches of rows instead of retrieving one large result set. You can also include record timestamps, such as create and update timestamps, as well as participant-level timestamps for the relationship-based Bulk Extract activity.

Figure 20 Bulk Extract Process



Deployment Phase

When you deploy, you attach the ActiveMatrix BusinessWorks process to a physical architecture. You can deploy by using:

- an ActiveMatrix BusinessWorks engine that is configured with ActiveMatrix BusinessWorks SmartMapper activities.
- the SmartMapper Enterprise Server. This allows you to share the cross-referencing server across multiple integration projects.

  Generally the SmartMapper Enterprise Server would run on the same machine as the database server itself. Multiple SmartMapper Enterprise Servers could communicate to the same database if throughput demands are high enough. Multiple data stores may be used in an enterprise as long as the data is disjoint.

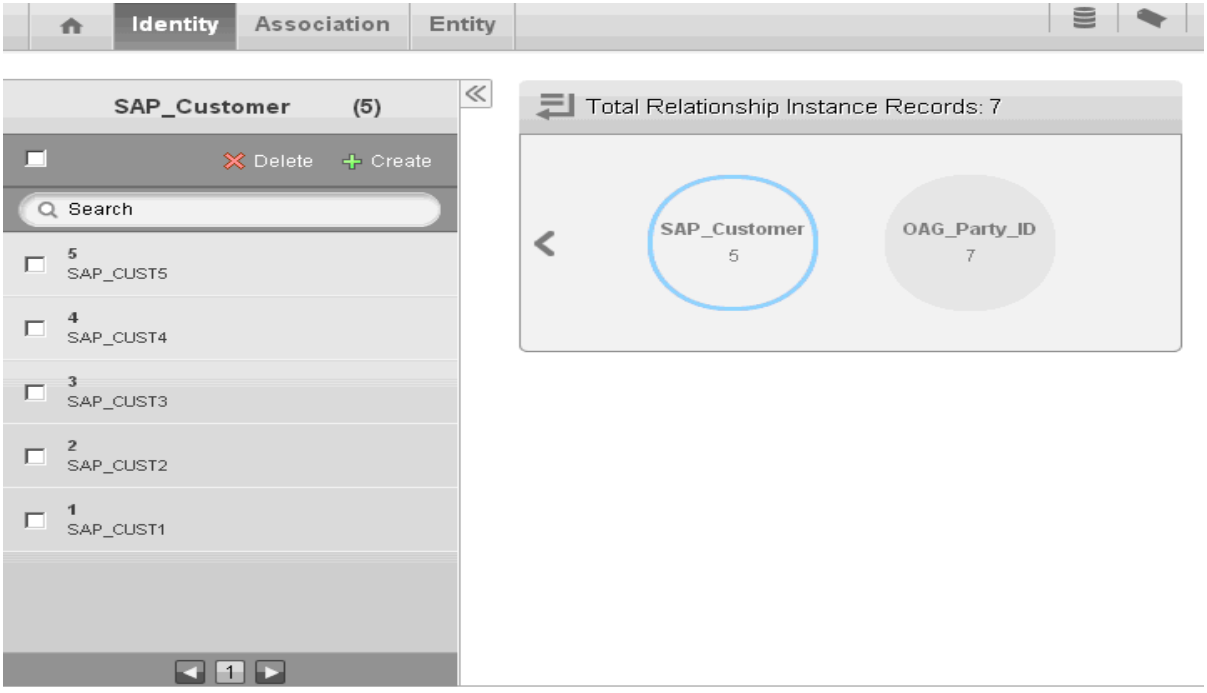
See Architecture on page 9 for more information about deployment.

Management Phase

Two ways are available to manage the relationship data:

- **Use TIBCO Administrator** After you deploy an ActiveMatrix BusinessWorks SmartMapper service, it is managed using TIBCO Administrator. For information about deploying an application in TIBCO Administrator, see the *TIBCO Administrator User's Guide*. You can also use the ActiveMatrix BusinessWorks SmartMapper plug-in into TIBCO Administrator to view participants and manage relationships. See Managing Relationship Data with TIBCO Administrator on page 67 for details.
- **Use SmartMapper Business UI** TIBCO ActiveMatrix BusinessWorks SmartMapper Business UI is a web-based user interface, which allows users to add, edit, and delete relationships, participants, and entities. See *TIBCO ActiveMatrix BusinessWorks SmartMapper Business UI User's Guide* for more details.

Figure 21 SmartMapper Business UI: Managing Identity Relationship Data



Chapter 4 Loading and Extracting Data

This chapter explains the ER Model; the three methods used to load data; and one method used to extract data.

Topics

Overview

This chapter explains how to parse and load data. A project that contains three processes is described. Each process uses a different ActiveMatrix BusinessWorks SmartMapper activity to load the same data. A short example is also included that shows how to extract data using the Bulk Extract activity.

Bulk Load Activity

The Bulk Load activity is designed for two use cases. 1) seeding a large amount of data into SmartMapper and 2) enabling dynamic writes where the relationship name is passed in with the data rather than modeled at design time.

This allows you to seed multiple, different relationships from one set of data, and to link all participants in one call where the SmartMapper Wizard activity (described next) allows only pairs.

The activity provides mechanisms to “fail” a row at a time in the case of collisions.

Besides, you can also use the Bulk Load tool to load relationship and entity data to the specified database. See Bulk Load Tool on page 160 for more details about how to use this tool.

SmartMapper Wizard Activity

The SmartMapper Wizard activity is designed for mapping a hierarchical object such as a sales order. It can do different operations such as seeding and lookups.

All writes in a SmartMapper Wizard operation succeed or none do, and running the operation multiple times gives the same result as running it once.

Create Relationship Instance Activity

The Create Relationship Instance activity provides a simple, functional mechanism for creating a logical “row” in ActiveMatrix BusinessWorks SmartMapper.

The activity seeds data into its respective entity table. It may not, however, have the speed or the exception handing capability when compared to the other activities, which are optimized for certain use cases.

Bulk Extract Activity

The Bulk Extract activity allows you to extract cross-referencing data for reporting purposes and filter the results. You can query either entity or relationship data by providing just name of the entity or relationship.

You can use participant-level timestamps for the relationship-based Bulk Extract activity.

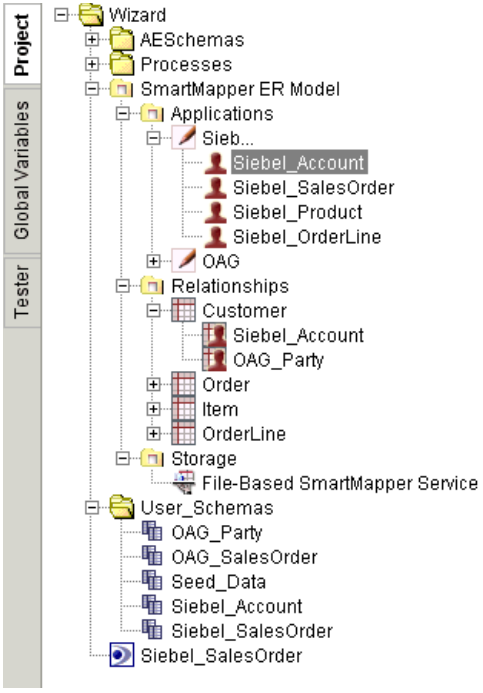
Besides, you can also use the Bulk Extract tool to extract cross-referencing data from the specified database. See Bulk Extract Tool on page 158 for more details about how to use this tool.

ER Model and BusinessWorks Activities

Each process described in this chapter uses the same SmartMapper ER Model and BusinessWorks Data Format activity. This section provides an overview of the ER Model and the activity.

The ER Model has an entity associated with each application and each entity is assigned to a relationship. An entity represents something that is being referenced, such as a customer record. For example, Figure 22 shows the ER Model defined for the processes.

Figure 22 SmartMapper ER Model



This example uses a file-based SmartMapper service. Instead, a database can be used for storage and the JDBC-based SmartMapper service activity can define the connection to the database instance.

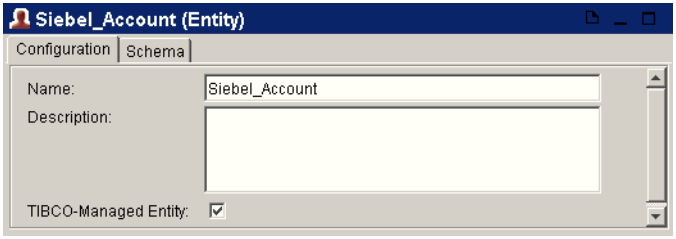
Applications and Entities

Two tabs are available for configuring an entity: Configuration and Schema.

Configuration Tab

The Configuration tab allows you to name the entity and enter its description.

Figure 23 Entity: Configuration Tab

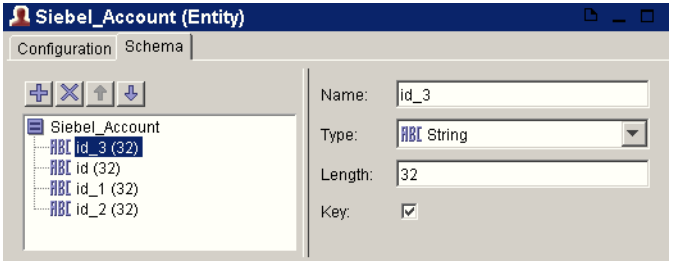


An entity can be defined with the TIBCO-Managed Entity field selected. When the TIBCO-Managed Entity check box is selected, it allows SmartMapper to generate a key for an entity if one is not returned in the lookup. The goal here is to create a relationship among three applications — represented by three columns in the text file — and to generate a global, unique identifier (GUID).

Schema Tab

By default, schema is not defined for an entity. You must define the schema.

Figure 24 Entity: Schema Tab



Clicking the + button allows you to add and configure a schema. The default attribute name is id.

The name can be any value less than 31 characters. Each entity has a key set assigned with either a string or integer attribute.

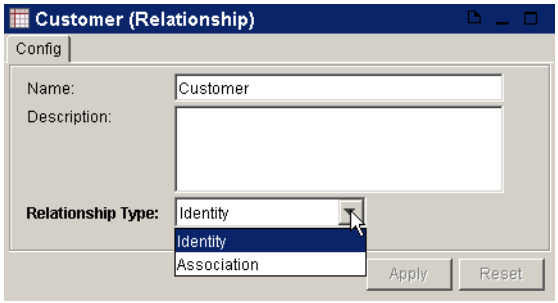
Relationships and Participants

You can configure relationships and participants that belong to these relationships. For more information, see Relationships on page 33.

Relationships

Select a relationship in the project panel.

Figure 25 Relationship



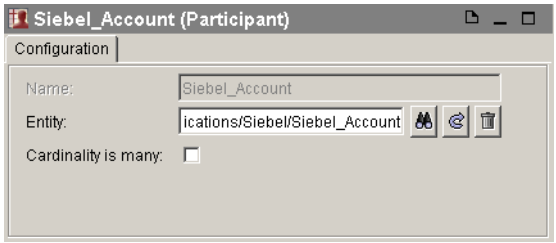
You can name a relationship, enter its description, and choose the relationship type. There are two types of relationships:

- Identity: used to keep corresponding keys for different applications
- Association: this relationship can have only two participants and is used to model relationships of two orthogonal entities

Participants

Select a participant in the project panel.

Figure 26 Participant



You can map this participant to a particular entity.

If the Cardinality Is Many check box is selected, the relationship can have a collection of entities as a peer with the other entities.

BusinessWorks Data Format Activity

The BusinessWorks Data Format activity is used in ActiveMatrix BusinessWorks SmartMapper processes.

This activity contains the specification for parsing the string using the Parse Data activity. See *TIBCO ActiveMatrix BusinessWorks Palette Reference* for details.

The source data is in a spreadsheet, so the configuration fields are defined as shown in Figure 27, Figure 28, and Figure 29.

Figure 27 Data Format Activity: Configuration Tab

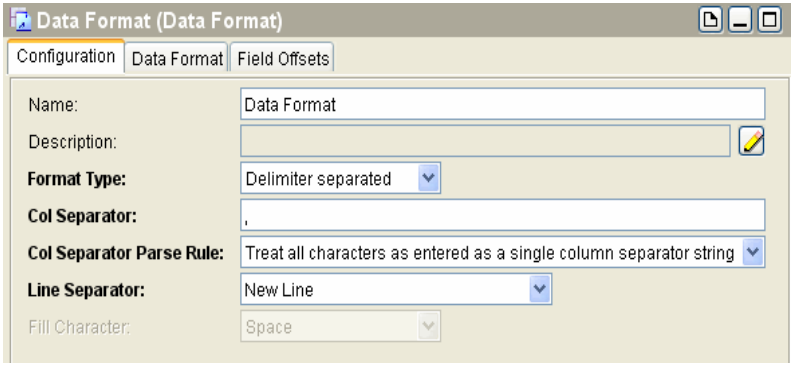
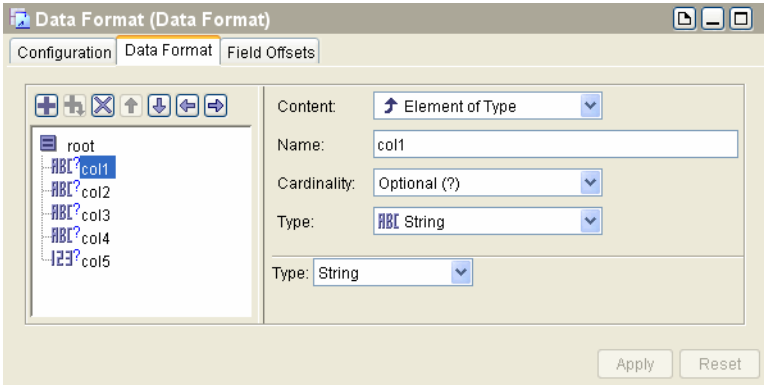


Figure 28 shows five columns in the spreadsheet for each schema. Each element is defined with the same settings for the Cardinality field and two Type fields.

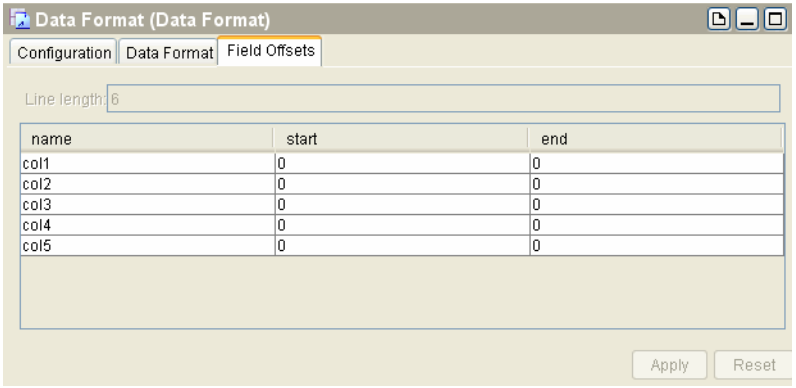
Figure 28 Data Format Activity: Data Format Tab



When processing fixed format text, you must specify the line length and column offsets. This allows the Parse activity to determine where columns and lines begin and end.

Figure 29 shows the Field Offsets tab that provides a way of specifying the format of fixed-width text.

Figure 29 Data Format Activity: Field Offsets Tab



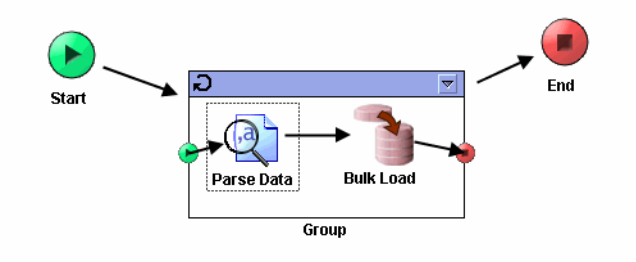
Using the Bulk Load Activity

The Bulk Load activity allows you to load large amounts of data. Unlike other activities, the Bulk Load activity does not stop if an error occurs. That is, if a record fails to load into the database, the activity will not stop.

The Parse Data and Bulk Load activities are used to manage the amount of data. The Parse activity handles data in chunks and hands each chunk over to the Bulk Load activity before processing the next chunk of data. Both activities are placed in a group so the action can be repeated until all the data is parsed and loaded.

A file is used as input in this example, but the mechanism is nearly identical when loading from a database. Figure 30 shows the process defined.

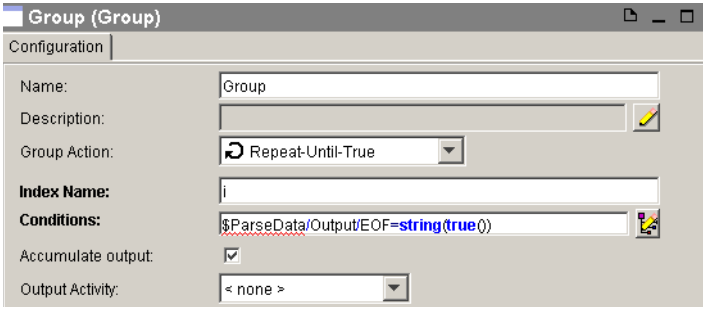
Figure 30 Bulk Load Activity Diagram



The iterator group is used to prevent BusinessWorks from parsing the entire file into memory at once. Ten thousands records could be handled fairly easily, but 100,000 would result in a noticeable slow down. A loop like this enables the process to scale for any size.

The Group activity configuration is shown in Figure 31.

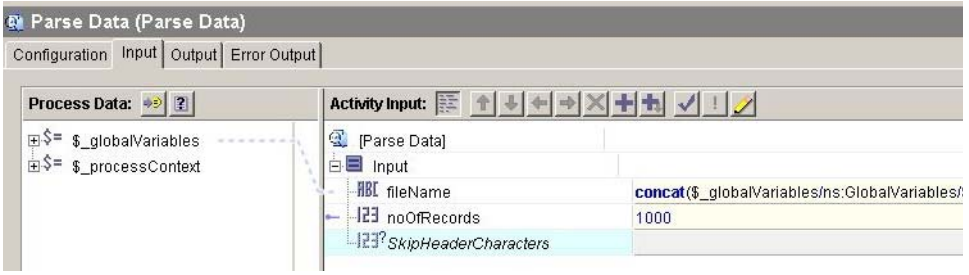
Figure 31 Group Activity Configuration Diagram



The group action repeats until the condition defined occurs.

The Parse Data activity takes input from a file and processes it, turning it into a schema tree based on the Data Format activity. See BusinessWorks Data Format Activity on page 47 for details.

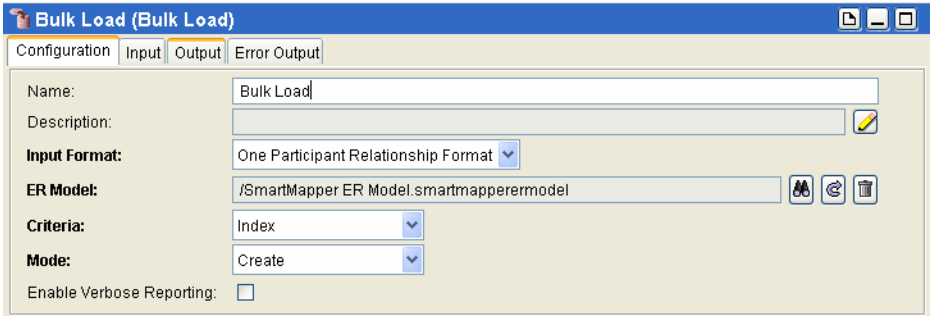
Figure 32 Parse Data Activity: Input Tab



Under the Input tab, the path name of the file that contains the records is given and the number of records to parse is defined. By setting the number of records, a limit is placed on the number of records that will be read or parsed at a time.

Figure 33 shows the Bulk Load Configuration panel. The Bulk Load activity allows you to load multiple records at once.

Figure 33 Bulk Load Activity: Configuration Tab



When the Criteria field is set to Index, it means the records will be loaded by index.

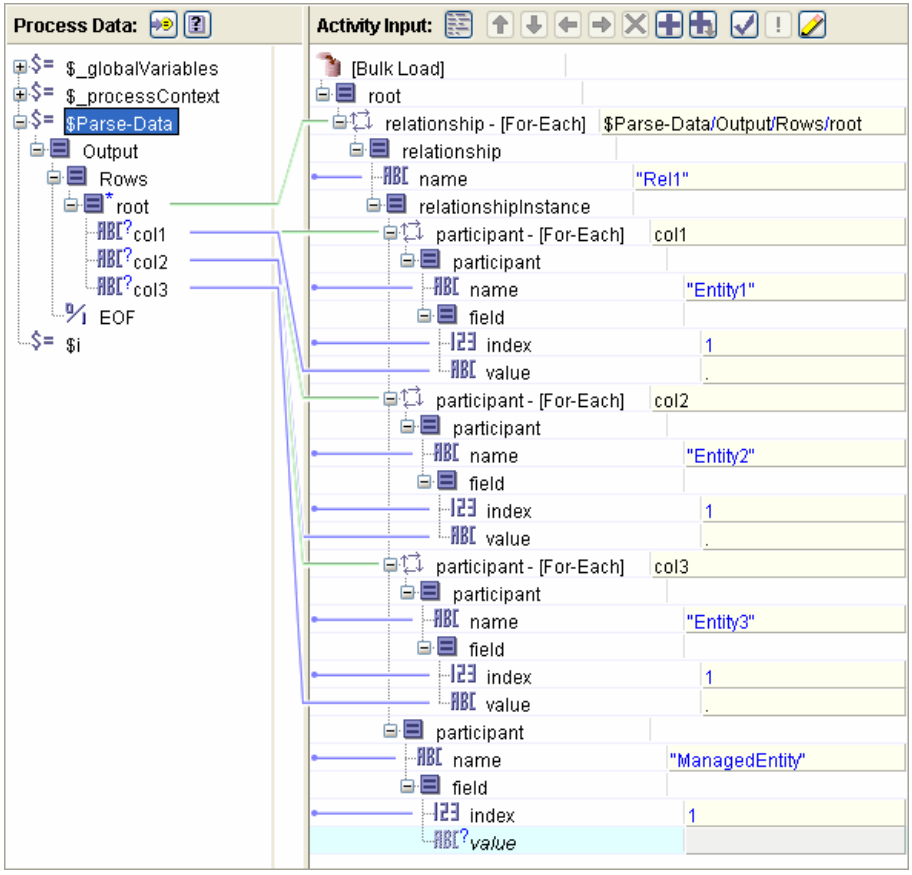
Any data load has a chance of collisions. If the Mode field is set to Create, the data is loaded with minimal collision detection. This is the fastest, most efficient way to seed an empty store. If the Mode field is set to Add, on the other hand, it checks for duplicate records first and ignores them, rather than throwing an error.

The Input structure is different depending on the Mode setting when the Input Format is set to One Participant Relationship Format.

See Create Mode Setting on page 114 for more information.

Figure 34 shows an example screen of the Bulk Load activity.

Figure 34 Bulk Load Activity: Activity Input



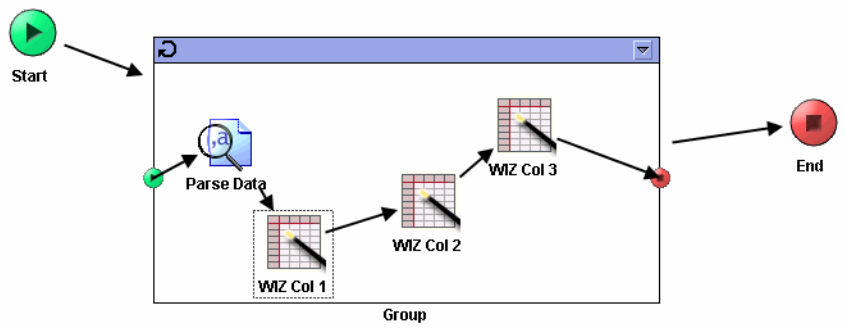
The Parse Data activity defined the elements for the input columns. The outcome of the Parse Data activity is used as the input of the Bulk Load activity.

In the Input tab, the Process Data and Activity Input panes are represented as schema trees. Under the Activity Input pane, click the icon to add an element to the schema tree.

Using the SmartMapper Wizard

The same data can be managed using the SmartMapper Wizard activity. The following diagram shows a process defined with three SmartMapper Wizard activities, one for each column in the source data file. The ER Model and Parse Data activity used in this process is the same as was described earlier in ER Model and BusinessWorks Activities on page 44.

Figure 35 Parse Data Activity: Diagram

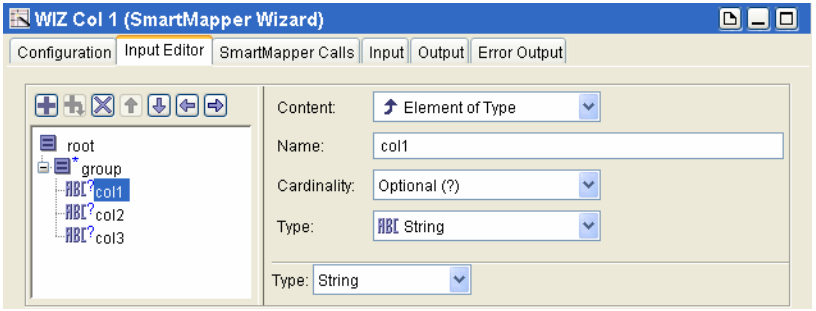


Note that while this is functionally equivalent to that described in the previous section, and will work when there are no errors, but it does not provide the same level of error handing.

Each column is associated with a SmartMapper Wizard activity. Essentially, each activity uses the same definition, but for a different column in the input data file (col1, col2, col3). We'll only describe how to define the first SmartMapper Wizard. The other Wizards are defined the same way, but with different column names.

Figure 36 shows the Input Editor.

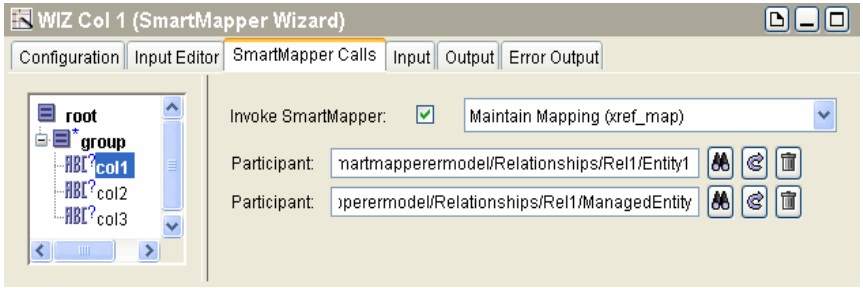
Figure 36 Input Editor



Three elements are defined: col1, col2 and col3.

The SmartMapper Calls tab allows you to annotate the input schema for logical locations for keys or references.

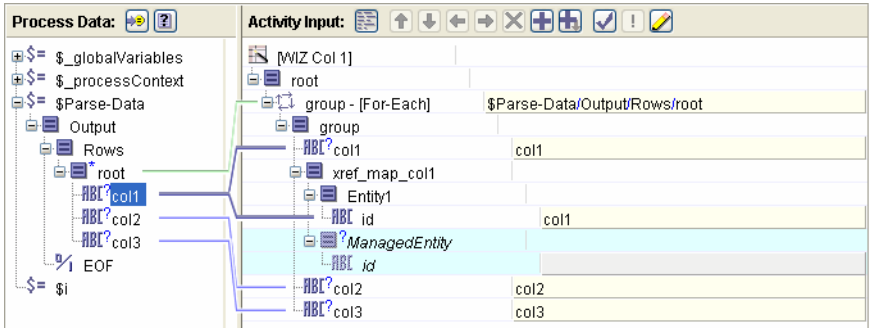
Figure 37 SmartMapper Calls Tab



Select the Invoke SmartMapper check box, and then select Maintain Mapping (xref_map) in the list. This maintains a one-to-one mapping between the given two participants.

The Input Editor defined the schema for the columns in the data file. Note that ActiveMatrix BusinessWorks SmartMapper generates a GUID for each row and maps it to the values in the first column.

Figure 38 Generating a GUID



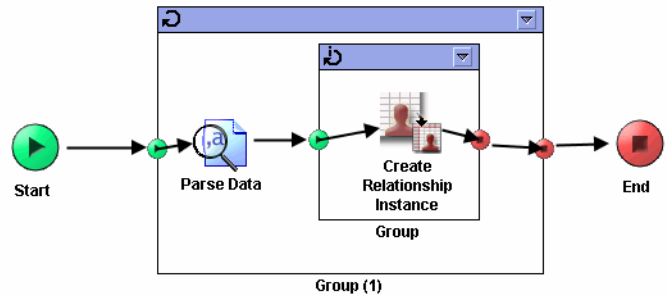
The second SmartMapper Wizard takes the output from the first and adds the data from the second column to the relationship.

Using the Create Relationship Instance

This primitive operation creates a set of entity instances and associates them in a relationship instance. Figure 39 shows a process defined with the Create Relationship Instance activity.

The SmartMapper ER Model and Parse Data activity used in this process are the same as those described earlier.

Figure 39 Create Relationship Instance Diagram

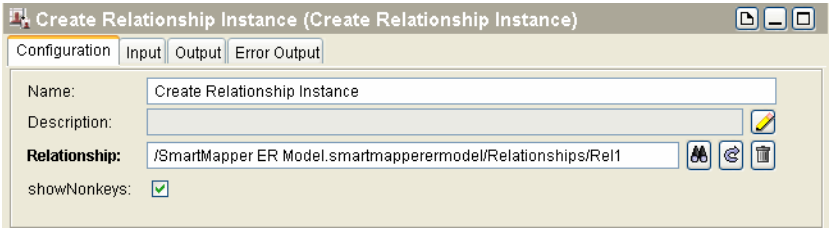


As shown in Figure 39, two group activities are used in the process as follows:

- The outer group controls how much of the file is read into memory.
- The inner group puts each record into ActiveMatrix BusinessWorks SmartMapper.

The configuration for the create relationship instance is shown in Figure 40. It is assigned the Rel1 relationship defined in the ER Model.

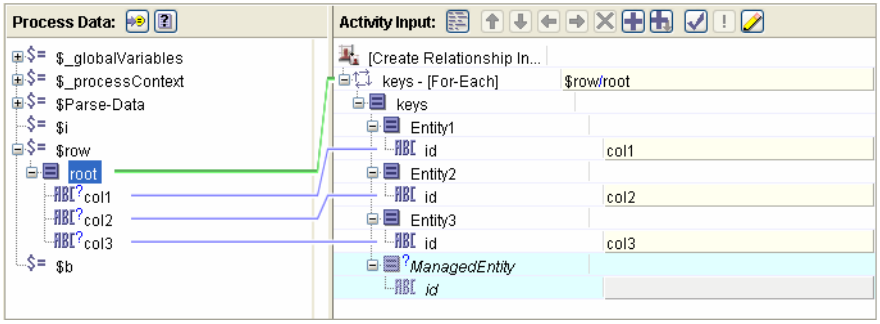
Figure 40 Create Relationship Instance: Configuration



The Parse Data activity defined the elements for the input columns.

In the Input tab, the process data and activity input are represented as schema trees. Under Activity Input, click the + icon to add an element to the schema tree.

Figure 41 Create Relationship Instance: Activity Input

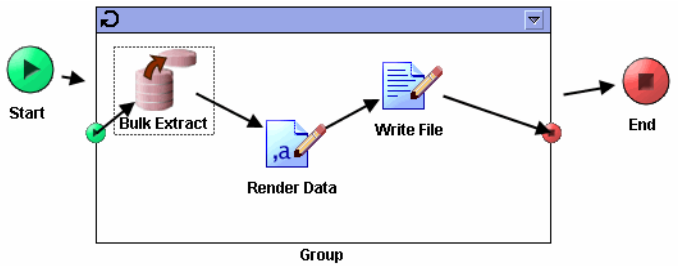


Extracting Data to the File

The Bulk Extract activity allows you to retrieve cross-referencing data. In this example, the activity retrieves data and writes it to a file using the original data format.

Figure 42 shows the project. Similar to the other projects described, the activities are placed in a group where each record is read using ActiveMatrix BusinessWorks SmartMapper. The number of records to read before writing is configured in the Bulk Extract activity.

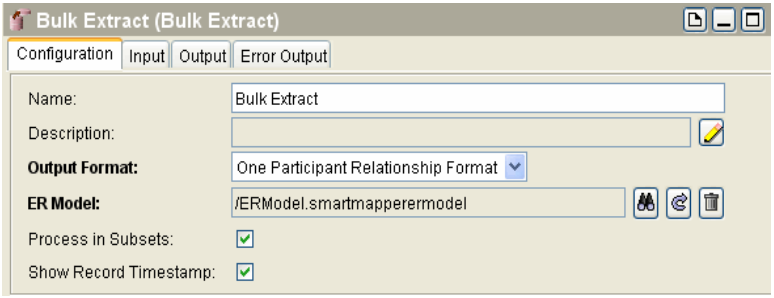
Figure 42 Extracting Data to the File



The Configuration tab for the Bulk Extract activity allows you to specify that the result set be processed in smaller batches rather than processing the entire result set at once. The Output Format field allows you to specify the relationship format, or entity format. The choice you make determines the structure that appears under the Output tab. See Output Tab on page 107 for more information.

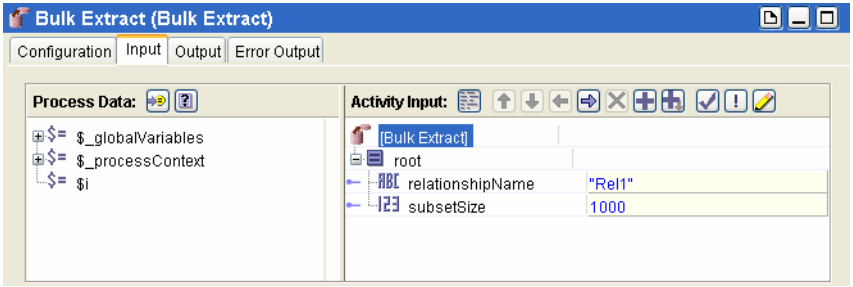
To include record timestamps, select the Show Record Timestamp field.

Figure 43 Show Record Timestamp



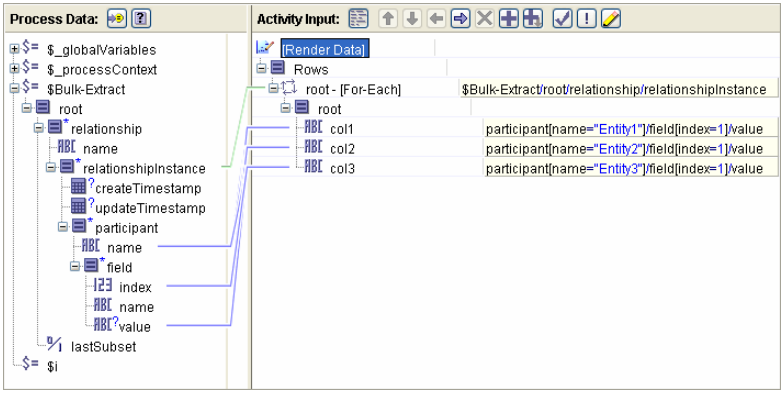
Under the Input tab, a root element was created where the relationship name and the subset size is set (elements are created by clicking the + icon). The subset size sets the size of each batch of records you wish to process.

Figure 44 Bulk Extract: Input Tab



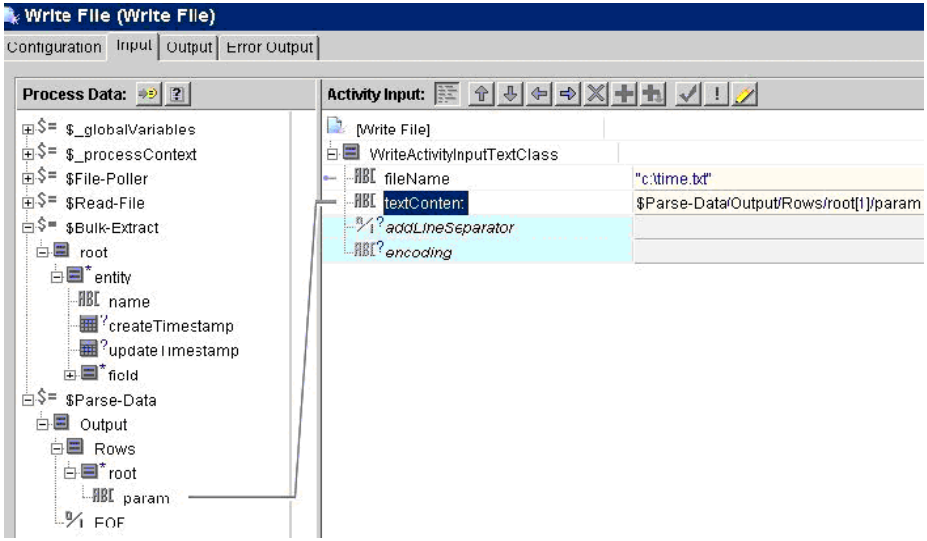
The Render Data activity allows you to retrieve a result set from a database. Figure 45 shows the Input tab in the activity where the output data from Bulk Extract activity is to be transformed to the columns used in the spreadsheet.

Figure 45 Bulk Extract: Activity Input



The Write File activity writes the text content to the specified file. The file name is given in the input field, which is shown in Figure 46 on page 58.

Figure 46 Write File: Input Tab



Chapter 5 Using the SmartMapper Adapter Service

This chapter describes how to use the SmartMapper adapter.

Ensure that TIBCO ActiveMatrix BusinessWorks SmartMapper Enterprise Server is installed. For TIBCO ActiveMatrix BusinessWorks SmartMapper Plug-in, the SmartMapper adapter service is unavailable.

Topics

Overview of the SmartMapper Adapter

The Adapter-Based SmartMapper Service accesses a SmartMapper adapter to manage the relationship data. Running as a Subscription Service, the SmartMapper adapter receives the requests triggered by the SmartMapper activities and executes the operations on the database server.

The SmartMapper Wizard activity sends requests to SmartMapper services in batches, the SmartMapper adapter can handle these requests in an efficient way when using the Adapter-Based SmartMapper Service. For more details, see Batching, page 141.

A typical configuration and deployment procedure for a SmartMapper adapter service includes the following steps:

1. Starting TIBCO Designer, page 15
2. Creating a Project, page 16
3. Creating an ER Model, page 18
4. Creating a SmartMapper Adapter Instance, page 61
5. Starting the SmartMapper Adapter, page 62
6. Managing the SmartMapper Adapter using TIBCO Administrator, page 64

This chapter starts with creating a SmartMapper adapter instance. See Getting Started on page 13 for details about starting TIBCO Designer, creating a project, and creating an ER Model.

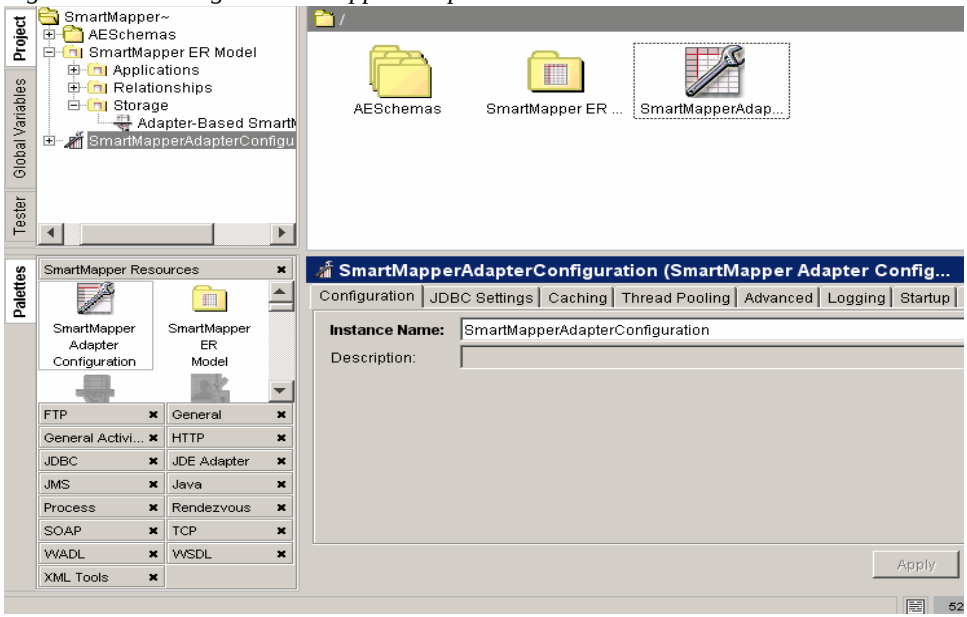
The ER Model must be configured with the Adapter-Based SmartMapper Service active.

Creating a SmartMapper Adapter Instance

After creating an ER Model with the Adapter-Based SmartMapper Service, follow these steps to create a SmartMapper adapter instance:

1. Click the project name and expand the SmartMapper Resources Palettes panel.
2. Drag the **SmartMapperAdapterConfiguration** resource to the Design panel, as shown in Figure 47, and then configure the SmartMapper adapter instance. See SmartMapper Adapter Configuration on page 98 for details about configuring the SmartMapper adapter.

Figure 47 Creating a SmartMapper Adapter Instance



3. Double-click the **SmartMapperAdapterConfiguration** resource in the Design panel, and then click the **Adapter Services** folder. The Subscription Service is displayed.
4. Configure the Subscription Service. See Adapter Service Configuration on page 195 for details about configuring the subscription service.

Starting the SmartMapper Adapter

After configuring the adapter with the Subscription Service, you can start the adapter in the following two ways:

- Starting the Adapter with the Adapter Tester, page 62
- Starting the Adapter from the Command line with a Repository File, page 62

Starting the Adapter with the Adapter Tester

To start the adapter with the Adapter Tester, the adapter and TIBCO Designer must be installed on the same machine.

1. Start TIBCO Designer.
2. Select **Tools** > **Show Adapter Tester**.
3. In the left pane, select the SmartMapper adapter instance that you want to start.
4. Click the **Run Settings** tab. In the Working Directory field, enter a directory to place running files.
5. In the Adapter Executable field, click the button to locate the adapter executable. By default, the executable is located in the *SmartMapper_HOME*\bin directory.
6. In the TRA Template field, click the button to locate the adsmartmapper.tra file. By default, the file is located in the *SmartMapper_HOME*\bin directory.
7. Click the **Apply** button.
8. Click the **Start** button. To view the output messages, click the **Console** tab.

Starting the Adapter from the Command line with a Repository File

To start the adapter from the command line with a repository file, the project must be run as a local repository and saved in DAT (repository) format. See the *TIBCO Designer* documentation for more information about repository files.

Task A Convert the Project to a Repository File

To export the project to a local repository:

1. Start TIBCO Designer.
2. Select **Project** > **Export Full Project...**. The Export Project dialog appears.
3. Click the **Local Repository** tab and enter the project name and the output directory in the Export Project dialog. Click the **OK** button.

4. In the Create Project dialog, select **File Type** and **TIBCO Messaging Encoding**. Click the **Yes** button.

Task B Start the Adapter

The adapter can be run by specifying the path of the DAT file in the repo URL, in the properties file or TIBCO Runtime Agent.

1. At the command prompt, go to the *SmartMapper_HOME*\bin directory, which hosts the adapter executable.
2. Enter the following command to start the adapter:

```
adsmartmapper.exe -system:repourl repository_file_url -system:configurl    configuration_url
```

Managing the SmartMapper Adapter using TIBCO Administrator

This section explains how to manage the SmartMapper adapter service with TIBCO administrator:

Creating an EAR File in TIBCO Designer

Generate an Enterprise Archive file (EAR) that contains information about the adapter services to deploy.

The EAR file contains information on what you wish to deploy. This could be one or more adapter services, one or more TIBCO BusinessWorks process engines, or both.

Building an archive creates the EAR file, which you can then deploy from TIBCO Administrator. If you make changes to the business processes or adapter services included in the archive, you need to rebuild the archive. Saving the project does not affect the archive.

In TIBCO Designer, follow these steps to create an EAR:

1. Configure the SmartMapper adapter service. See Creating a SmartMapper Adapter Instance on page 61 for details.
2. Select the adapter project in the Project panel.
3. Drag the **Enterprise Archive** resource from the General Palettes panel to the Design panel, and then specify a name and a location for the Enterprise Archive file in the Configuration panel.
4. Select the **Enterprise Archive** project you have created in the Project panel.
5. Drag the **Adapter Archive** resource from the Palettes panel to the Design panel, and then configure the Adapter Archive in the Configuration panel.
6. Select the **Enterprise Archive** project in the Project panel, and then click the **Build Archive** button in the Configuration panel.

See Also

See *TIBCO Designer User's Guide* for more information about this procedure. The guide is available from the Designer Help menu.

Deploying the Project

Before deploying a project, the machine on which the adapter is installed must be part of a TIBCO administration domain. After you have installed the TIBCO Administration Server, any machine on which you install TIBCO Runtime Agent (required by an adapter) can be added to the administration domain. The TIBCO software installed on the machine is then visible and accessible via the TIBCO Administrator GUI.

When you deploy a project, startup scripts and other information about the different components are sent to the machines to which the components were assigned. The project data store and TIBCO Administration Server are updated with the deployed components.

To deploy a project:

1. Import the EAR file into TIBCO Administrator.
2. Assign adapter archives in the EAR file to adapters installed in the administration domain and likewise assign process archives to process engines.
3. Specify startup options for each adapter service.

See Also

See *TIBCO Administrator User's Guide* for an introduction to the TIBCO administration domain and detailed information about the above steps.

See *TIBCO Administrator Server Configuration Guide* for fault tolerance information.

Starting or Stopping the Adapter

The TIBCO Administrator Application Management module allows you to start, and stop deployed applications.

To start an adapter service from the module:

1. In the Administrator GUI left pane, expand **Application Management** > *Application-Name* > **Service Instances**.
2. In the Service Instance panel, select the check box next to the adapter service.
3. Click the **Start Selected** button.

   The status changes from Stopped to Starting up to Started.

4. To stop the adapter service, click the **Stop Selected** button.

Do not use a TIBCO Rendezvous message to stop the run-time adapter. Instead use TIBCO Hawk.

See Also

See *TIBCO Administrator User's Guide* for more information.

Monitoring the Adapter

TIBCO Administrator offers a number of monitoring options.

- Specify alerts and TIBCO Hawk rulebases for each machine in the domain.
- Specify alerts and Hawk rulebases for each adapter service.
- View the log for each adapter service.

See Also

See *TIBCO Administrator User's Guide* for information about configuring the above monitoring options.

Chapter 6 Managing Relationship Data with TIBCO Administrator

This chapter describes how to deploy a SmartMappper project and how to manage relationship data with TIBCO Administrator.

Topics

Overview

TIBCO ActiveMatrix BusinessWorks SmartMapper plug-in for TIBCO Administrator allows you to view and manage runtime data in the ER Model tables. Using TIBCO Administrator, you can maintain relationship data whether or not the project is deployed. This allows you to seed the data for testing.

Building an Enterprise Archive (EAR)

In TIBCO Designer, a project can have one or more **Enterprise Archive (EAR)** resources that correspond to an .ear file. To create an application definition in TIBCO Administrator, you must load an .ear file.

An Enterprise Archive consists of the SmartMapper Archive resource, which corresponds to a deployable service in TIBCO Administrator.

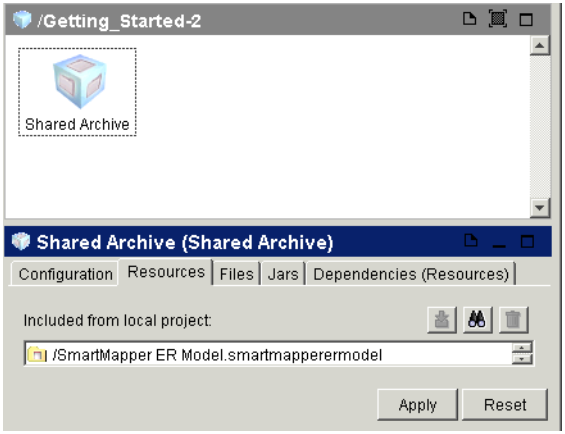
To learn more about enterprise archives and how they are built, see *TIBCO Designer User's Guide.*

To Create a SmartMapper Enterprise Archive (EAR)

1. In TIBCO Designer, create a new SmartMapper project with at least one process and one service.
2. With the top-level folder selected, open the General palette.
3. Locate the Enterprise Archive resource and drag the resource to the configuration panel.
4. Double-click the **Enterprise Archive** resource.

   The Shared Archive resource appears in the design panel.
5. Click on the Resources tab.

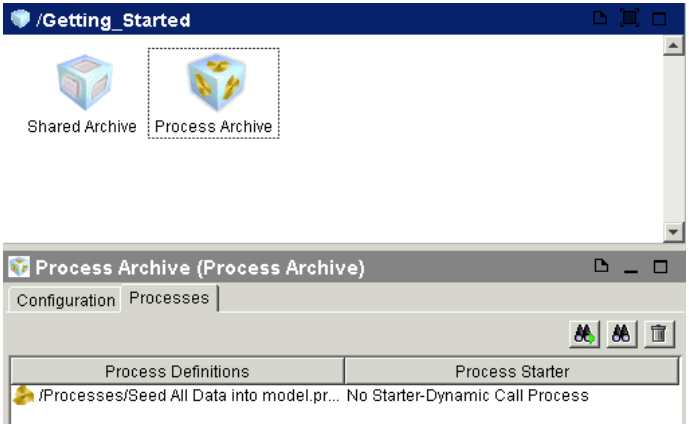
Figure 48 Adding a Project to the Archive



6. Browse to the SmartMapper ER Model you have created.
7. Select the project and click **OK**.

8. Click **Apply**.
9. Drag the **Process Archive** resource from the Palettes panel to the Design panel.
10. Select the **Processes** tab.

Figure 49 Adding a Process to the Archive



11. Browse with the button “Add a non -process starter process to this archive” (binoculars without the green arrow) and select the process you wish to add to the archive, such as “Seed All Data into model”.
12. Click **Apply**.
13. Repeat steps 10 and 11 to add other processes if required.
14. Click the **Enterprise Archive** resource in the project panel, then click the **Build Archive** button and save the project.

Adding a SmartMapper Application

To learn more about adding and administering TIBCO applications, see *TIBCO Administrator User's Guide*.

To add a SmartMapper application, complete the following steps:

1. Select **Start > All Programs > TIBCO > TIBCO Administrator *version_number* > TIBCO Administrator**.
2. Select a domain and log in.
3. Click **Application Management**.
4. Click the **New Application** button on the Application Management panel.

   The Upload EAR File dialog appears.

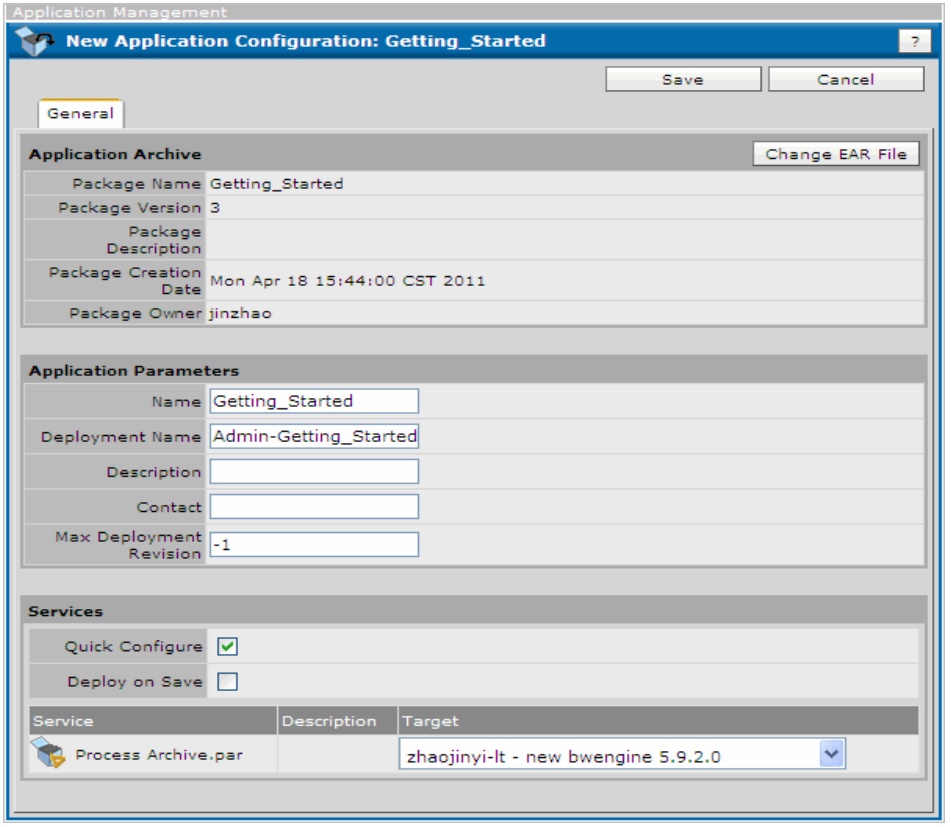
Figure 50 Upload EAR File

5. Browse to the archive file you wish to upload.

   To learn how to build an enterprise archive file (EAR), see Building an Enterprise Archive (EAR) on page 69.
6. Click the **OK** button.

   The new application appears.

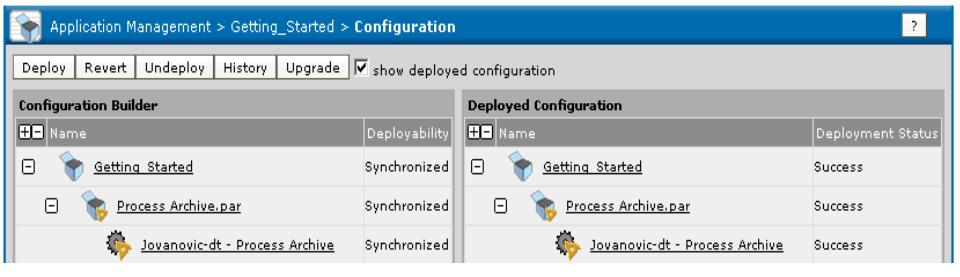
Figure 51 New SmartMapper Application



If you want to deploy the application immediately, select the **Deploy On Save** check box.

7. Click the **Save** button, as shown in Figure 51.
8. Click the **Deploy** button.
9. Click the **OK** button. The application is deployed.

Figure 52 Deployed Application



Opening a SmartMapper Project

To learn more about adding and administering TIBCO applications and services, see *TIBCO Administrator User's Guide*.

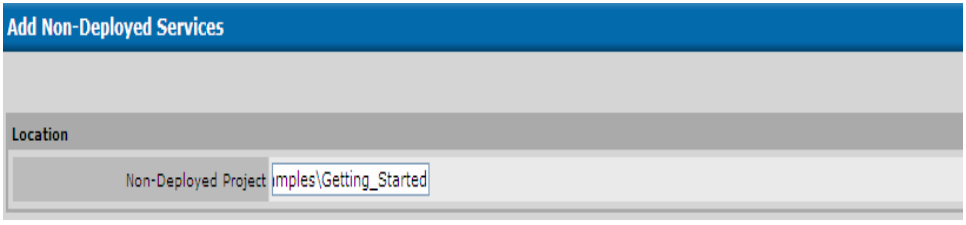
Opening a Non-Deployed SmartMapper Project

To open a non-deployed project, complete the following steps:

1. Select **Start > All Programs > TIBCO > TIBCO Administrator *version_number* > TIBCO Administrator.**
2. Select a domain and log in.
3. Click the **SmartMapper 6.0.0** node under the Resource Management folder.
4. Click **Add Non-Deployed** button on the right panel.

   Type the file pathname to that project in the Non-Deployed Project text box, such as D:\tibco\smartmapper*version*\examples\Getting_Started**.**

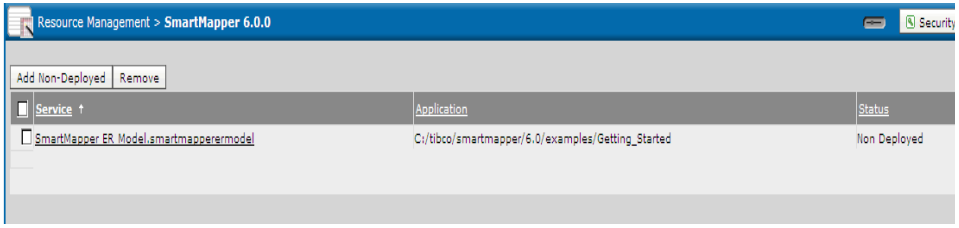
Figure 53 Adding Non-Deployed Service



5. Click the **OK** button.

   The project appears as a service.

Figure 54 Deployed SmartMapper Project



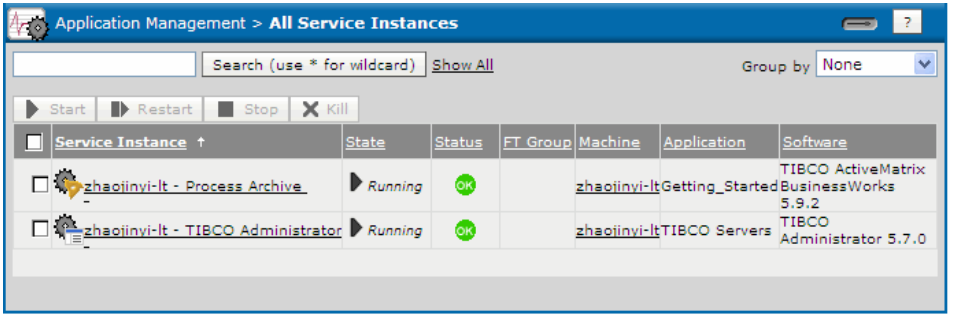
6. Click the service name to open the service. See Managing Relationships on page 75 to learn about managing relationships for the selected service.

Opening a Deployed SmartMapper Project

To open a deployed project, complete the following steps:

1. Select **Start > All Programs > TIBCO > TIBCO Administrator *version_number* > TIBCO Administrator.**
2. Select a domain and log in.
3. Click the **All Service Instance** node under Application Management folder.

Figure 55 SmartMapper Project Deployed



4. Click the service name to open the service.

   See Managing Relationships on page 75 to learn about managing relationships for the selected service.

Managing Relationships

This section shows how to create a relationship instance, search for a relationship, and remove or edit keys in a relationship.

It assumes that you have opened a non-deployed project or a deployed service as explained in Opening a SmartMapper Project on page 73.

The Edit SmartMapper Detail screen allows you to do the following:

- Create a Relationship Map, page 76
- Search for a Relationship, page 77
- Remove or Edit Keys in a Relationship, page 79

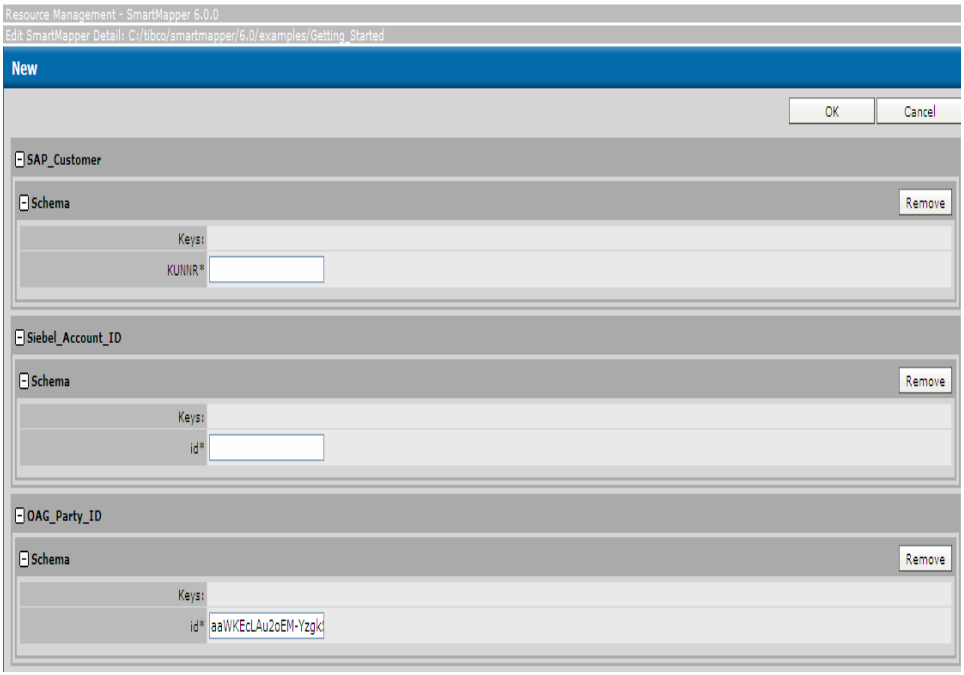
Create a Relationship Map

To create a relationship, do the following:

1. Click the **New Map** button under Relationship Instances.

   The New dialog appears.

Figure 56 Creating a Relationship Map



2. Enter a value for each of the keys contained in that relationship.
3. Click the **OK** button.

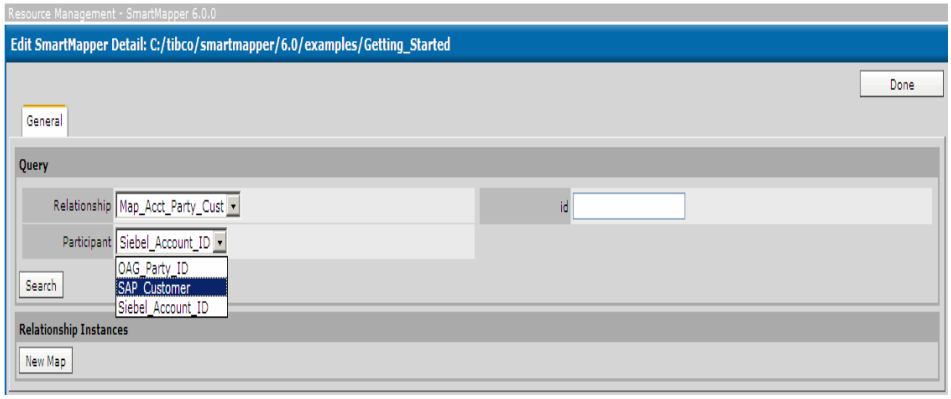
   This creates a relationship between the OAG_Party, which is a TIBCO-managed entity, and the listed accounts.

Note that a key was automatically generated for the TIBCO-managed entity (in this case, OAG_Orderline.

Search for a Relationship

1. In the Edit SmartMapper Detail panel, first select a relationship from the Relationship dropdown list.
2. Select a participant from the Participant dropdown list.

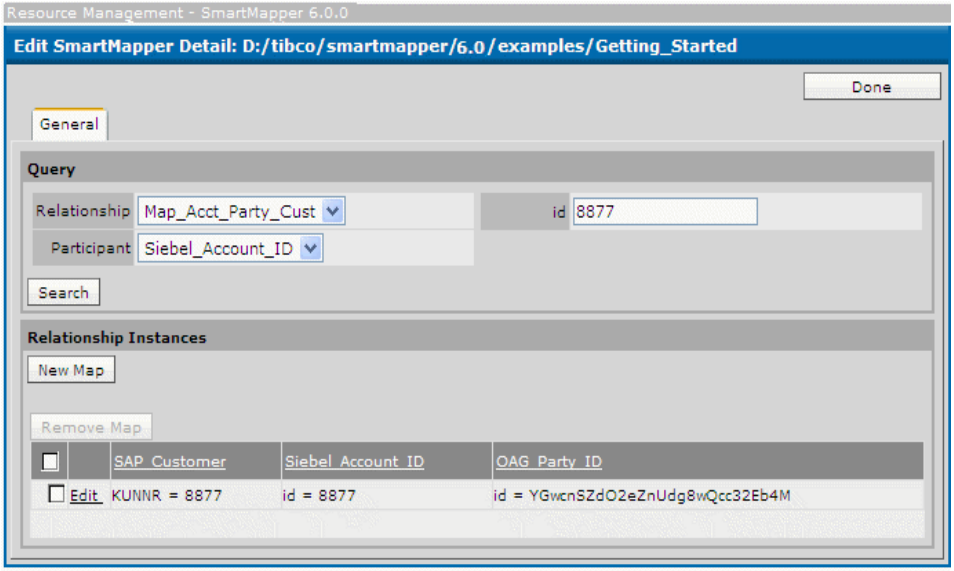
Figure 57 Selecting a Participant



   - If you select an entity that is an account ID (such as Siebel_Account_ID), the ID window will appear on the right.
   - If you select an entity that represents a concept with multiple attributes (such as SAP_Customer), the KUNNR window will open on the right.
3. If the selected entity is an account ID, type the value in the ID field.

   If the selected entity that represents a concept with multiple attributes, type the value in the KUNNR field.
4. Click the **Search** button. The query results appear.

Figure 58 Query Results for a Relationship Search



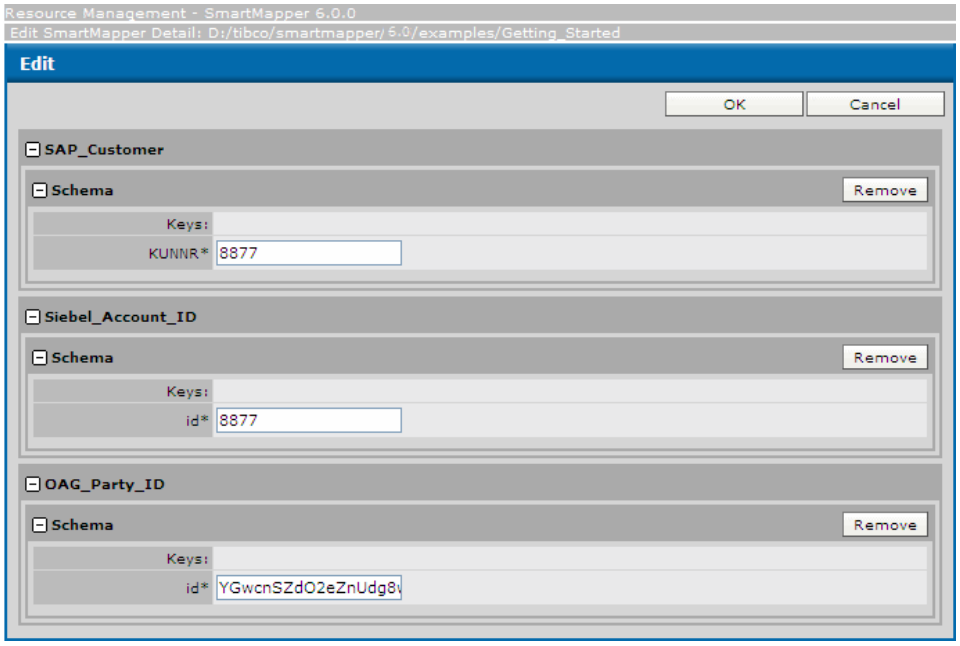
The relationship instance for the entity Siebel_Account_ID is displayed. It contains the account ID for the entity.

Remove or Edit Keys in a Relationship

To remove or edit a key from a relationship instance, complete the following steps:

1. Click **Edit** beside the selected relationship instance, as shown in Figure 58. The Edit dialog appears.

Figure 59 Editing Keys



2. On this screen you can:
   - Remove an instance by clicking **Remove**.
   - Edit an instance by changing the values.
   - If a participant has more than one instance, you can add multiple instances of the participant to the relationship instance.
3. Click the **OK** button when done.

Chapter 7 SmartMapper ER Model Resources

This chapter describes the ER Model resources. Other resources are described in SmartMapper Activities on page 103.

Topics

SmartMapper ER Model

Resource

The SmartMapper ER Model resource represents a logical data model that contains a set of applications, relationships and storage. It defines a set of relationships and their participating entities.

The tabs that are displayed depend on the status of the project: the tabs shown in this section are those displayed when the ER Model icon has been moved to the design panel.

Applications Folder

The Applications folder contains one or multiple applications. An application must contain at least one entity. An application groups the entities based on logical systems.

One or more Application folders can be added into the Applications folder.

See also Applications on page 31.

Relationships Folder

A relationship defines the set of participants that can be used for cross-referencing. Any entity instance that belongs to a relationship instance can be used to find any of the other entity instances in the same relationship instance. In other words, entity instances that participate in a relationship instance cross-reference each other.

One or more Relationship folders can be added into the Relationships folder.

See also Relationships on page 33.

Storage Folder

Assigning an ER Model to a service designates that service as the only data manager for that ER Model. An ER Model can only be assigned to one storage service at a time.

A storage service folder is available in the ER Model. When you open this folder, you can choose one of three storage services: JDBC, File and Adapter-based.

See also Storage on page 35.

Configuration Tab

Table 4 SmartMapper ER Model: Configuration Tab

Field	Description
Name	The name to appear as the label for the resource.
Description	A short description of the resource.

Advanced Tab

Table 5 SmartMapper ER Model: Advanced Tab

Field	Description

ID Generator Class	A fully-qualified Java class name of the class used to generate keys for TIBCO-Managed entities.
	If this value is supplied, the specified class is used to generate keys for all of the ER Model entities that are marked as TIBCO-Managed.
	Note that this class must implement the interface com.tibco.solution.xref.autoid.AutoIdGenerator.
	This is defined in xref.jar:
	`package com.tibco.solution.xref.autoid;`
	`import com.tibco.solution.xref.XRefException;`
	`public interface AutoIdGenerator {`
	`String generateId( String entityName ) throws XRefException;`
	`}`
	For more information, see Generate Key on page 139.
	In some cases, the user may have id generators from a sequence in a database that supports sequences. In other cases, users can use custom JDBC classes to generate IDs.
	For further information on using custom classes to generate IDs, see Appendix A, Adding Your Own GUID Generation Class, on page 190.

Entity

Resource

An entity resource is an object that is being referenced, for example, a customer.

To place an entity in the TIBCO Designer design panel, you must have an application folder open in the panel.

For more information on entities, see Entity on page 31.

Configuration Tab

Table 6 Entity: Configuration Tab

Field	Description
Name	The name to appear as the label for the resource in the process definition. **Note:** Only digits, alphanumeric characters, and underscores can be contained in the entity name. Besides, a digit cannot be used as the first character of the entity name.
Description	Short description of the resource.
TIBCO-Managed Entity	If this is selected, during TIBCO ActiveMatrix BusinessWorks processing, TIBCO ActiveMatrix BusinessWorks SmartMapper will generate a key for this entity if one is not returned in the lookup. Selecting this check box also means that the schema is automatically completed for the entity. An entity may be defined as TIBCO-Managed in support of a common data model. The Generate Key activity generates a key for a TIBCO-Managed entity. See Generate Key on page 139. Note that when you select TIBCO-Managed Entity, the key schema has only one attribute ID, which must be a string of length less than 100 characters, as in id(100).

Schema Tab

The Schema tab defines the key set of the entity. Use the + button to add more attributes. Click **Apply** after adding attributes. All the attributes you add to the entity are keys.

Table 7 Entity: Schema Tab

Field	Description
Name	The name of the key. Some database implementations will have reserved words, which cannot be column names (and thus attributes). For example, if you are using Microsoft SQL Server, do not name an attribute *key*. If an attribute is named *key*, an error will be returned when synchronizing the schema. **Note:** Only digits, alphanumeric characters, and underscores can be contained in the name. Besides, a digit cannot be used as the first character of the name.
Type	Type can be String or Integer. The string length becomes the actual length in the database. Therefore, if you are using double-byte data, you may want longer strings.
Length	The maximum length. This applies only to the String type.
Key	When selected, the attribute is part of the participant's key.

Relationship

Resource

A relationship resource defines a relationship and identifies whether it is an identity or association relationship.

A relationship folder is available under the Relationships folder.

Configuration Tab

Table 8 Relationship: Configuration Tab

Field	Description
Name	The name to appear as the label for the resource. **Note:** Only digits, alphanumeric characters, and underscores can be contained in the relationship name. Besides, a digit cannot be used as the first character of the relationship name.
Description	Short description of the resource.
Relationship Type	Select Identity or Association: • **Identity** There are one or multiple participants per relationship, where one of the participants acts as a primary object. A primary object can exist independently of other objects. For example, Customer, Order, and Employee are considered primary objects, but address, and account information are not because they are contained within a primary object. • **Association** There are only two participants in each relationship.

Participant

Resource

A participant is a member of a relationship. The participant associates an entity to a relationship and specifies the entity's cardinality within that relationship.

A participant is the role that an entity plays in a relationship. You can add a participant to the design panel only when you have a Relationship folder open.

When a participant is added to a relationship in TIBCO Designer, an entity must be selected to represent the participant.

When the Cardinality is Many button is selected, you can link many instances to a single instance of other participants in the relationship. For example, you can link three different Siebel accounts, all referring to one customer, to a single SAP customer account.

Note that an association relationship is always many-to-many.

Configuration Tab

Table 9 Participant: Configuration Tab

Field	Description
Name	The name to appear as the label for the resource
Entity	The entity selected to represent the participant The entity must be from the same ER Model and cannot appear more than once in a relationship. However, it may participate in other relationships in the same ER Model.
Cardinality is many	If selected, this entity is on the many side of a one-to-many relationship. For an association relationship, the cardinality on both sides as assumed to be many. If cleared, this entity has a one-to-one relationship. **Note:** This option is available only when the relationship type is Identity.

JDBC-Based SmartMapper Service

Resource

The JDBC-Based Service accesses a JDBC database directly to manage SmartMapper data. This service runs inside a TIBCO ActiveMatrix BusinessWorks engine and uses a JDBC driver to read and write data to a database.

To use this service, a database must be installed, a user must have appropriate permissions, and a script must be run to create the initial metadata tables. This is described in the *TIBCO ActiveMatrix BusinessWorks SmartMapper Installation Guide*, Setting Up Tables for Databases.

Under the JDBC Settings Tab, click the **Synch ER Model to DB** button to run the script.

If you assign two (or more) ER Models to the same database, you must configure each to use a different user account to prevent the tables from colliding.

Configuration Tab

Table 10 JDBC-Based SmartMapper Service: Configuration Tab

Field	Description
Name	The name to appear as the label for the resource.
Active	This sets this storage service as active. You can set many storage services, but only one can be active at a time. If you have multiple services, checking the Active box on one automatically sets the Active boxes in the other services to inactive (unchecked).

JDBC Settings Tab

This tab defines the storage mechanism. SmartMapper ER Model synchronization is a process of creating metadata table schemas in the database. Before running TIBCO ActiveMatrix BusinessWorks SmartMapper runtime activities, users synchronize the ER Model by clicking the **Sync ER Model w/ DB** button.

SmartMapper uses its own connection pool separate from BusinessWorks and hence none of the smartmapper activities supports the JDBC transaction feature that comes with BusinessWorks.

Table 11 JDBC-Based SmartMapper Service: JDBC Settings Tab

Field	Description
Database	The database type. Use the arrow to select your database type.
JDBC Driver Class	The fully-qualified java class name of the JDBC driver. This is automatically given when you select your database type.
JDBC URL String	The appropriate URL string for the driver.
User	The database user name.
Password	The database password.
Connection Pool Size	TIBCO ActiveMatrix BusinessWorks SmartMapper database access always uses connection pooling. This value determines the maximum number of connections available in the pool.
Login Timeout (sec.)	The amount of time in seconds before the login attempt times out.
Test Connection	Tests the JDBC properties by using them to connect to the database.
Synch ER Model w/DB	The synchronization process creates new database tables. The database user specified in the User field needs to have 'create table' privileges. SmartMapper activity runtime does not create any new database tables and hence the user running the activities does not need to have those privileges. Currently, TIBCO ActiveMatrix BusinessWorks SmartMapper requires that the user running activities be the same user who has done the synchronization and hence created the tables. In a multi-user environment, it is desirable that one database user can do the synchronization, and that other users who run the activities have privileges just to run the activities and not to be able to synchronize. This is not possible with current implementation of TIBCO ActiveMatrix BusinessWorks SmartMapper. To avoid accidental database synchronization in such cases, follow the explanation given in Caching Tab on page 92.

ER Model and Database Synchronization

If the back-end SmartMapper service is JDBC-Based or Adapter-Based, the ER Model must be applied to the underlying JDBC database. This is done through the ER Model/database synchronization process.

Synchronization involves creating the database tables and indexes that are required for storing a ER Model in the database.

In the JDBC-Based Service the process is initiated using JDBC Settings Tab > Synch ER Model w/DB.

In the SmartMapper Adapter Configuration, the synchronization process is initiated using JDBC Settings Tab > Synch ER Model w/DB.

Generally, it is not good practice to modify an existing table. The data in the table may become inconsistent with the new schema.

If you delete and recreate an entity, a new table will be created. You can then manually convert the data using whatever tools or algorithm seems appropriate.

When you click **Synch ER Model w/DB**, the Synchronize ER Model dialog is displayed.

Table 12 Synchronize ER Model Dialog

Field	Description
Added Objects	The ER Model objects that are created in the database when synchronization is invoked.
Audit	When selected, auditing is enabled and all participant tables and relationship tables will include the creation_date and modified_date attributes. When cleared, the attributes are not included.
Removed Objects	The ER Model objects that are removed. The database tables for removed objects are not actually dropped during synchronization. Instead, they are marked as deleted and are not used by TIBCO ActiveMatrix BusinessWorks SmartMapper. In other words, the data is left intact for removed objects.
Perform Sync	Creates or removes the TIBCO ActiveMatrix BusinessWorks SmartMapper objects in the database. This function will generate an SQL script, same as the one generated by the function “Generate SQL”, but in this case such script will not be displayed in a window for editing.
Generate Complete Script	Generates a complete SQL script for the current ER Model. Click the **Copy to Clipboard** button to copy the SQL script to a local file. Hence, users can copy the same ER Model to a new database by running the generated SQL script in an empty database.
Generate SQL	Brings up a window that contains the SQL that is executed if you click **Synch ER Model w/DB**. This is useful for database administrators that may want to edit the SQL and apply it manually.

Reload ER Model from a Database

In multi-user environments, one user is usually responsible for designing and modifying the ER Model while other users use it for their projects in read only mode. The Reload feature allows the users to import an ER Model, previously synchronized by another user, from a SmartMapper database instance.

In the case of the JDBC-Based Service, you will proceed as follows:

- Create an instance of the ER Model by dragging an ER Model resource.
- Add an instance of the JDBC storage by dragging the JDBC-Based SmartMapper Service storage resource.
- Initiate the Reload process by using the tab JDBC Settings > Synch ER Model w/DB > Reload From DB.

In the SmartMapper Adapter Configuration, the synchronization process is initiated using the JDBC Settings > Synch ER Model w/DB > Reload From DB.

When you click the Reload From DB tab, the reload dialog will appear.

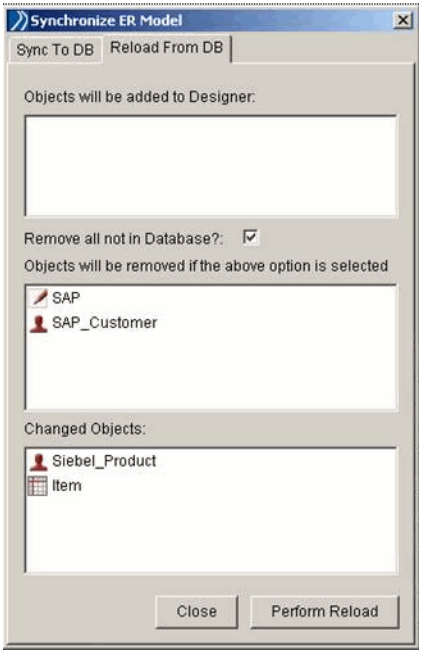
Table 13 Reload from Database Dialog

Field	Description
Added Objects	The ER Model objects which will be added to the designer project from the database.
Remove all not in database	All the objects from the existing ER Model in the designer project will be removed
Perform reload	Imports ER Model objects from the database to the designer project.

Use caution when selecting the Remove all not in database check box. It is better to leave it clear unless you want to explicitly remove the ER Model objects from your project.

Figure 60 shows an example of using the feature Reload From Database.

Figure 60 Synchronize ER Model



Edited ER Model Objects and Synchronization

The synchronization process handles added and removed ER Model objects. ER Model objects that have been edited since the last synchronization, that is, entities that had keys added or had the key name or type changed, are also synchronized with a new table created in the database. However, the original object still exists in the database, it cannot be used by BusinessWorks SmartMapper process anymore.

It is not feasible to automate the editing of database table definitions. Actions like changing a database column data type may involve dropping the table and re-creating it. Also, the data in a table would have to be exported and then re-imported into the new table. Applying an edited ER Model to a database must be done manually.

Caching Tab

There is a cache associated with the storage. The cache is controlled by the number of entries. The Least Recently Used algorithm swaps entries out of the cache as new ones are added.

The cache sync interval is important when multiple projects are accessing the same tables and doing updates or deletes. The cache sync interval controls the frequency at which the cache ensures its consistency with the database.

Table 14 JDBC-Based SmartMapper Service: Caching Tab

Field	Description
Enable Caching	Select to enable caching. Clear to disable caching.
Cache Type	Selects a caching type from the list. Two types are available: • **Local** Stores the caching data to a local memory. • **ActiveSpaces** Stores the caching data using TIBCO ActiveSpaces. **Note:** Ensure that TIBCO ActiveSpaces is installed.
The following fields are available when Local is selected in the Cache Type field.	
Maximum Number of Entries	Determines the maximum number of entity instances to be stored in the cache. Once this number is exceeded, the excess cache entries are culled from the cache using a least-recently-used algorithm.
Cache Synchronization Interval (Seconds)	The time interval, in seconds, that cache housekeeping is performed. This involves the following: Culling the appropriate cache entries when the cache has grown too big Synchronizing the cache to the database for entries that have been changed by other SmartMapper processes
The following fields are available when ActiveSpaces is selected in the Cache Type field.	
Metaspace Name	Specifies the name of the metaspace to connect to.
Discovery URL	Specifies the discovery URL to use to discover each metaspace.
Listen URL	Specifies the listen URL to use to listen for the incoming connections from new members to the metaspace.
Relationship Capacity	Specifies the maximum number of relationship instances per seeder for the space. If the number of relationship instances exceeds the specified number, the relationship instances that are not frequently used will be deleted.
Replication All	Specifies whether to replicate all the relationship data or not. Selecting this check box means that all the relationship data will be replicated on all the seeders.
Distribution Role	Specifies a distribution role for a newly joined member. Two options are available: • **Seeder** plays an active role in maintaining the space by providing CPU and RAM. • **Leeche** plays a passive role which has access to space data but provides no resources.

File-Based SmartMapper Service

Resource

While the JDBC-based service accesses a database, the file-based services do not. Instead, the file-based services are 100% cached. The file seeds the cache. Writes may be done for seeding and maintaining the file, but this is not a substitute for a database where there are large volumes of writes.

The file-based service runs inside the TIBCO ActiveMatrix BusinessWorks process engine and reads and writes the SmartMapper data in an xml file.

The file-based service is used for small sets of data that are not often edited. Data stored in a file-based service are always read in their entirety into memory. The file-based service does not perform well in write-intensive applications.

File-based service is generally used:

- for development and demos

  or

- when the data is more or less static. Often, in this case, read time is much more important than write time.

The integrity of a file-based data store is lower than a database. File-based storage should not be used for data that cannot be regenerated.

The Bulk Load, Bulk Extract, Dynamic Lookup, and Dynamic Delete activities are not supported for use with the File-Based SmartMapper Service.

Configuration Tab

Table 15 File-Based SmartMapper Service: Configuration Tab

Field	Description
Name	The name to appear as the label for the resource.
Active	This sets this storage service as active. You can set many storage services, but only one can be active at a time. If you have multiple services, checking the Active box on one automatically sets the Active boxes in the other services to inactive (cleared).
Data File	Specify the file that stores the SmartMapper data. Click **Browse** to use an existing data file, or click **New** to specify a new file.

Adapter-Based SmartMapper Service

Resource

The Adapter-Based SmartMapper Service accesses a SmartMapper Adapter (Enterprise Server) to manage the data.

To use this service, you must have the TIBCO ActiveMatrix BusinessWorks SmartMapper Enterprise Server package installed.

See Using the SmartMapper Adapter Service on page 59 for more details about creating, starting, and deploying a SmartMapper adapter service.

If the SmartMapper Adapter is modified, it must be re-linked to this service.

Configuration Tab

Table 16 Adapter-Based SmartMapper Service: Configuration Tab

Field	Description
Name	The name to appear as the label for the resource.
Active	Select to set this storage service as active. You can set many storage services, but only one can be active at a time. If you have multiple services, selecting the Active box on one automatically sets the Active boxes in the other services to inactive (cleared).
Transport Type	• **Rendezvous Reliable** Ensures that each multicast or broadcast message is received as long as the physical network and packet recipients are working, and that the loss of a message is detected. This choice can compensate for brief network failures because it can retransmit a message on request if the first attempt failed. • **Rendezvous Distributed Queue** Indicates load balancing should be enabled. An RVCMQ Session allows applications to use distributed queues for certified delivery, to any number of listeners using queuing member sessions that act together to process inbound task messages. • **JMS Queue** A message sent to a queue is consumed by one and only one receiver. Each message has only one receiver though multiple receivers may connect to the queue. The first receiver to access the queue gets the message. The other receivers do not.
Load Configuration from Adapter	Displays a popup window from which you select a SmartMapper Adapter Configuration resource you created earlier. By loading the configuration from the adapter, you ensure that the subject name and quality of service are in synch between the client and the server.

Transport Tab

Table 17 provides descriptions of the fields in the Transport Tab under Rendezvous Reliable Transport. See *TIBCO Rendezvous Concepts* and *TIBCO Enterprise Message Service User's Guide* for more details about the following transport types: Rendezvous Distributed Queue and JMS Queue.

Table 17 Adapter-Based SmartMapper Service Transport Tab: Rendezvous Reliable Transport

Field	Description
Subject	Specify the subject name to be used when subscribing. By default a service uses a message subject that is generated using the Domain and Deployment global variables, the adapter acronym, the adapter instance name and the service name. If you use this default subject, make sure the values for Domain and Deployment are not empty. You can type a different TIBCO Rendezvous subject name from the default in this field. See *TIBCO Rendezvous Concepts* for information about specifying subject names.
Service	(default value = null) The Rendezvous daemon divides the network into logical partitions. Each TibrvRvdTransport communicates on a single service. A transport can communicate only with other transports on the same service. To communicate on more than one service, a program must create more than one transport. It must create one transport for each service.
Network	(default value = null) Every network transport communicates with other transports over a single network interface. On computers with more than one network interface, the network parameter instructs the Rendezvous daemon to use a particular network for all outbound messages from this transport.
Daemon	The daemon parameter instructs the transport object how and where to find the Rendezvous daemon and establish communication.

Advanced Tab

The Advanced tab contains the fields as shown in Table 18.

Table 18 Adapter-Based SmartMapper Service Advanced Tab

Field	Description
Batch Size	The SmartMapper Wizard activity sends requests to services in batches. The batch size parameter determines the number of requests included in a single message sent to the adapter. A value of 0 sends all requests generated by the Wizard in a single message. Setting the batch size to 0 means that the entire object's worth of lookup requests is sent in one batch. This ensures that writes are done all or nothing. See Batching on page 141 for more information.

Field	Description
Timeout in seconds	The number of seconds to wait before a SmartMapper operation is considered to be timed out. If the timeout elapses before the SmartMapper Adapter replies, an exception is thrown.

SmartMapper Adapter Configuration

Resource

The SmartMapper Adapter Configuration resource runs the service as an adapter.

To use this service, you must have the TIBCO ActiveMatrix BusinessWorks SmartMapper Enterprise Server package installed. See Using the SmartMapper Adapter Service on page 59 for more details about creating, starting, and deploying a SmartMapper adapter service.

The service usually runs on the database server, and usually runs when there are high throughput requirements and/or multiple projects wanting to access the same relationships. This service is otherwise identical to the JDBC service.

Most users do not need to configure this adapter, since the default configuration for this resource provides functionality for TIBCO ActiveMatrix BusinessWorks SmartMapper.

If you choose to use the Adapter Configuration resource, you must be at the project level to drag the Configuration resource from the palette to the design panel.

The default configuration for the services mentioned above is defined through global variables. For more information about services and global variables, see Appendix B, Adapter Service Configuration, on page 195

Configuration Tab

Table 19 SmartMapper Adapter: Configuration Tab

Field	Description
Instance Name	The name to appear as the label for the resource.
Description	A short description of the resource.

JDBC Settings Tab

The fields in this tab are the same as the fields in the JDBC Settings Tab on page 89 with the following exception:

- **SmartMapper ER Model** Click the browse button to associate an existing entity relationship model with the adapter.

Caching Tab

This tab has the same fields as the Caching Tab on page 92.

Thread Pooling Tab

The SmartMapper Adapter employs a thread pool to execute all SmartMapper operations. This allows the adapter to be very efficient executing operations with a high degree of concurrency. The pool is initialized with the minimum number of threads in the pool and is allowed to grow to the maximum thread count as concurrent requests come in and existing threads are busy.

The number of threads in the pool determines the number of operations that can simultaneously be executed. The larger the thread-count, the more computer resources have to be used to maintain and switch contexts between threads.

It is generally not useful to have more threads than database connections.

Table 20 SmartMapper Adapter: Thread Pooling Tab:

Field	Description
Minimum Thread Count	The minimum number of threads to maintain in the Adapter thread pool.
Maximum Thread Count	The maximum number of threads to maintain in the Adapter thread pool.

Advanced Tab

The advanced tab sets the ID Generator Class.

For information on the ID Generator Class, see Advanced Tab on page 83 and Adding Your Own GUID Generation Class on page 190.

Logging Tab

Table 21 SmartMapper Adapter: Logging Tab

Field	Description
Use Advanced Logging	When Use Advanced Logging is not selected (the default), you can set two standard output destinations (sinks) for trace messages and set the tracing level for the roles selected. When Use Advanced Logging is selected, you have complete control to select the destinations and associating desired roles with each of the destinations. To create and configure the sinks, select the log sinks folder under the Advanced folder in the project panel. To create sinks, drag and drop the Generic Log Sink icon from the palette panel into the design panel. From the configuration panel, select the sink type. The following are the sink types available: File, Hawk, Network, STDIO. When File and STDIO sinks are created from the Generic Log Sink they offer further configuration options. For the File sink, the file limit, file count, and the option to append or overwrite can be specified. When created by default, this is set to 30000 bytes, 3 and append mode respectively. For the STDIO sink, the option to write to stdout or stderr can be selected. When created by default, stdout is selected. The Hawk sink uses the hawk session, created and used by the adapter for monitoring purposes, to send tracing messages to the TIBCO Hawk monitor or Display. The configuration for the Hawk sink involves specifying the MicroAgent Name that must be specified in the configuration panel. The Network sink is used to publish tracing message on TIBCO Rendezvous. The configuration for the network sink involves specifying the session, and the subject on which the trace messages needs to be published. For all the sinks, the name and description for the sinks can be provided optionally.
Log to Standard I/O	(STDIO Sink) When selected, trace messages are displayed in the command prompt window where the adapter is started. When not selected, trace messages are not displayed in the window.
Log File	Specify the name of the log file (log sink) to which trace messages are written. Global variables can be used to specify the location of the log file. The roles available are Info, Debug, Warning, and Error messages. The trace message generated depends on the roles selected. Turning on the roles can affect the performance of the adapter. Therefore, it is recommended that you turn on the required roles only.
Log Info, Debug, Warning, or Error Messages	Trace messages of the selected level(s) will be collected in the named log sink. You can configure what levels of trace messages you want logged, and where trace messages are sent. There are three types of logs (log sinks) that you can configure to hold trace messages, corresponding to three levels (roles) of trace messages, Information, Warning and Error. A fourth level of trace messages, Debug, is reserved and should not be enabled unless requested by the TIBCO Product Support Group. This option writes a lot of information to the log file and significantly reduces the speed of the adapter.

Startup Tab

The Startup tab need not be configured for a standard adapter configuration. By default, the startup behavior of the adapter consists only of displaying a banner when it starts. No services are activated, and no schema is loaded.

Table 22 SmartMapper Adapter: Startup Tab

Field	Description
Show Startup Banner	When selected (the default), the startup banner displays the run-time adapter version, the infrastructure version on which the adapter is built, and copyright information in the console window when the adapter is started.
Metadata Search URL	The field specifies the location where the adapter searches for base schemas. The adapter searches for any schema that has been defined and saved at this location, and that should be loaded at startup. The default value is /[repo name]/Schemas/Classes/ae/baseDocument

Monitoring Tab

Many of the following fields make use of global variables. For more about global variables, see Adapter Service Configuration on page 195.

Table 23 SmartMapper Adapter: Monitoring Tab

Field	Description
Enable Standard Microagent	Allows you to turn on or off the standard TIBCO Hawk Microagent. The way to turn it on or off is also configurable. By clicking the Globe icon, a standard check box or text value (true or false) can be used to turn the standard microagent on or off.
Standard Microagent Name	This is the name for the standard microagent that will be registered with the TIBCO Hawk system. In most cases the default value is used. The InstanceId variable need not be set because it is automatically set at run time by the run-time adapter.
Standard Microagent Timeout	Specifies the amount of time the Hawk Agent should wait for HMA method invocations to complete before timing them out. The default is 10000 milliseconds. Normally there is no need to change this value, however, on machines under extreme stress where method invocations are timing out, this option allows the timeout value to be increased.
Enable Class Microagent	Allows you to turn on or off the instance or class specific standard TIBCO Hawk Microagent. The way to turn it on or off is also configurable. By clicking the Globe icon, a standard check box or text value (true or false) can be used to turn the class microagent on or off.
Class Microagent Name	This is the name for the class microagent that will be registered with the TIBCO Hawk system. In most cases the default value is used.

Table 23 SmartMapper Adapter: Monitoring Tab (Cont'd)

Field	Description
Class Microagent Timeout	Specifies the amount of time the Hawk Agent should wait for HMA method invocations to complete before timing them out. The default is 10000 milliseconds. Normally there is no need to change this value, however, on machines under extreme stress where method invocations are timing out, this option allows the timeout value to be increased.
Default Microagent Session	Specify the name of the TIBCO Rendezvous session that will be used by the standard, class, and custom microagents.

The session name and the corresponding session are automatically generated by TIBCO Designer. Do not change the session name or the session. However, you can modify the session parameters if required. Navigate to the Sessions folder under the Advanced folder to modify the session parameters.

Make sure you have set the correct parameter value for the global variables that correspond to the TIBCO Hawk configuration. If the session parameters are not set properly, the microagents will not display in the TIBCO Hawk Display.

Chapter 8 SmartMapper Activities

This chapter explains the ActiveMatrix BusinessWorks SmartMapper activities that can be part of a BusinessWorks process.

See SmartMapper ER Model Resources on page 81 for more details about the ER Model related resources.

Topics

Bulk Extract

Activity

This activity enables the bulk extraction of cross-referencing data for reporting purposes. You can query entity or relationship data by providing an entity name or relationship name in the Input tab.

Because bulk extraction could result in a large number of records, you can retrieve results in smaller subsets. The activity can then be used in an iteration group to iterate over the results set. See Fetching Subsets of the Result Set in the *TIBCO BusinessWorks Palette Reference* manual for more information.

The Bulk Extract activity is not supported for use with the File-Based SmartMapper Service.

Configuration Tab

Table 24 Bulk Extract: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	A short description of the activity.
Output Format	Choose the output format. The output structure (under the Output tab) changes depending on your selection. • **Entity Format** Select to return entity data only. • **One Participant Relationship Format** Select if the relationship is an identity relationship. That is, there is one or multiple participants per relationship. • **Two Participant Relationship Format** Select if the relationship is an association relationship. That is, there are only two participants in each relationship and both relationships are many-to-many.
ER Model	Click the browse button to select the ER Model to associate with this resource.

Table 24 Bulk Extract: Configuration Tab (Cont'd)

Field	Description
Process in Subsets	Selecting this field specifies that you would like to process the result set in smaller batches rather than processing the entire result set at once. The subsetSize input element appears in the Input tab to allow you to specify the size of each batch of records you wish to process. Also, the lastSubset output element appears in the Output tab and is set to true when the last batch of records is being processed. • When Process in Subsets is cleared, complete results are returned on one execution. If the result set is large, you may want to use a distributed queue to improve performance by balancing the load among services. • When Process in Subsets is selected, performance cannot be improved significantly by using a distributed queue.
Show Record Timestamp	When selected, timestamps for records are included in the report. This works only if auditing is turned on. When cleared, timestamps are not included in the report.

Input Tab

Table 25 Bulk Extract: Input Tab (Sheet 1 of 3)

Field	Datatype	Description
entityName	String (repeating)	The name of the entity to be extracted. Appears only when Entity format is selected in the Output Format field under the Configuration tab.
relationshipName	String (non-repeating)	The name of the relationship to be extracted. Appears only when Relationship format is selected in the Output Format field under the Configuration tab.

Table 25 Bulk Extract: Input Tab (Sheet 2 of 3)

Field	Datatype	Description
extractionFilter	repeating	Each instance of the extraction filter is mapped to a condition in the SQL WHERE clause (WHERE <column> <operator> <value>). participantFieldName maps to the column name in the WHERE clause and where/orWhere field maps to the operator and value part of the WHERE clause. When multiple instances of the extractionFilter are provided, each of them is AND'ed if operator and value are specified using the where field. Conditions are OR'ed together if operator and value are specified using the orWhere field. In a filter, only when the value for the where field is empty, the value of the orWhere field will take effect. This filter contains the following: • **participantFieldName** This name should include both the participant name and field name separated by a dot for relationship base lookup. Only the field name should be specified for entity base lookup. This field also supports the reserved words createTimestamp and updateTimestamp for timestamp based filter criteria. • **where** Value of the where field will map to the operator and value part of the SQL where clause. The operator could be an SQL operator such as '>', '<', '=' or 'like'. When multiple instances of the extractionFilter are specified, the filter with the where field value is AND'ed with other filters. • **orWhere** Value of the orWhere field will map to the operator and value part of the SQL where clause. The operator could be an SQL operator such as '>', '<', '=' or 'like'. When multiple instances of the extractionFilter are specified, the filter with the orWhere field value is OR'ed with other filters. **Note:** No matter whether the first filter uses the orWhere field or the where field, SmartMapper always treats it as the where field. Example of the extractionFilter usage: ExtractionFilter Instance 1 -> partFieldName = 'id', where = '> 345' ExtractionFilter Instance 2 -> partFieldName = 'id', orWhere = 'like %345%' will map to following SQL WHERE clause: WHERE id > '345' OR id like '%345%' **Note**: An operator is a required part of the where/orWhere field value. Not specifying an operator could result in runtime errors. For more information, see the example BulkExtractFilter, which is available in the following directory: *SmartMapper_HOME*\examples.See Advanced Tutorial: Bulk Extracting Data with Different Criteria on page 184 for more details.

Table 25 Bulk Extract: Input Tab (Sheet 3 of 3)

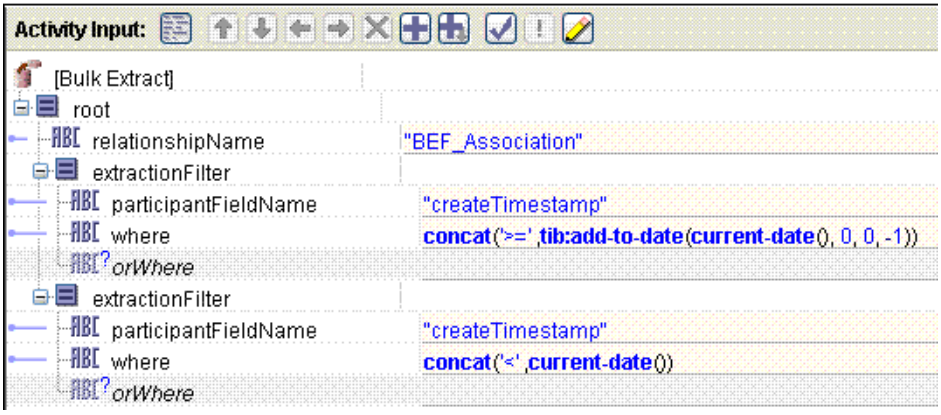
Field	Datatype	Description
subsetSize	integer	This element specifies the number of records to process in each batch. This allows you to process smaller batches of rows instead of retrieving one large result set.

Limitations to the extractionFilter are as follows:

- Filtering criteria cannot be parenthesized.
- For the Identity relationships, filtering criteria can be based only on one participant's fields.
- For the Association relationship, if filtering criteria is based on both participants, the AND operation is assumed across all participants.

Figure 61 shows an example for the extraction filter set to find a relationship that has been created yesterday.

Figure 61 Input Set for the Extraction Filter: Find a Relationship Created Yesterday



For more information, see Advanced Tutorial: Bulk Extracting Data with Different Criteria on page 184, which is available in the following directory:

SmartMapper_HOME\examples.

Output Tab

The structure under the Output tab display changes depending on the Output Format field setting as specified under the Configuration tab.

Entity Format Selected

The Output tab displays the following structure if Entity Format is selected in the Output Format field under the Configuration tab. **Entity** is a complex structure repeating datatype and provides the name of the entity as a string.

Table 26 Bulk Extract Output: Entity Format

Field	Datatype	Description
entity name	String (repeating)	The name of the entity selected in the Output Format field under the Configuration tab.
createTimestamp		The date when the timestamp for a record was created
updateTimestamp		The date when the timestamp for a record was updated
field		This is a complex structure repeating datatype that includes both key and non-key fields. The field order is dictated by the entity's schema definition. It contains the following elements: • index (integer) • name (string) • value (string)
lastSubset	Boolean	Appears only when Process in Subsets is selected under the Configuration tab. A value of true indicates that the current result set is the last subset.

One Participant Relationship Format Selected

The Output tab displays the following structure if One Participant Relationship Format is selected in the Output Format field under the Configuration tab.

Table 27 Bulk Extract Output: One Participant Relationship Format

Field	Datatype	Description
relationship name	String (repeating)	The name of the entity selected in the Output Format field under the Configuration tab.
relationshipInstance		Contains the following elements: • **createTimestamp** The date when the timestamp for a record was created • **updateTimestamp** The date when the timestamp for a record was updated • **participant** (see below for explanation)
participant		Contains the following elements: • **name** (string) • **createTimestamp** The date when the timestamp for a record was created • **updateTimestamp** The date when the timestamp for a record was updated • **field** (see below for explanation)
field		This is a complex structure repeating datatype that includes both key and non-key fields. The field order is dictated by the entity's schema definition. It contains the following elements: • index (integer) • name (string) • value (string)
lastSubset	Boolean	Appears only when Process in Subsets is selected under the Configuration tab. A value of true indicates that the current result set is the last subset.

Two Participant Relationship Format Selected

The Output tab displays the following structure if Two Participant Relationship Format is selected in the Output Format field under the Configuration tab.

Table 28 Bulk Extract Output: Two Participant Relationship Format

Field	Datatype	Description
relationship name	String (repeating)	The name of the entity selected in the Output Format field under the Configuration tab.
relationshipInstance		Contains the following elements: • **createTimestamp** The date when the timestamp for a record was created • **updateTimestamp** The date when the timestamp for a record was updated • **participant1, participant2** (see below for explanation)
participant1, participant2		Contains the following elements: • **name** (string) • **createTimestamp** The date when the timestamp for a record was created • **updateTimestamp** The date when the timestamp for a record was updated • **field** (see below for explanation)
field (for participant1 and participant2)		This is a complex structure repeating datatype that includes both key and non-key fields. The field order is dictated by the entity's schema definition. It contains the following elements: index (integer), name (string), and value (string).
lastSubset	Boolean	Appears only when Process in Subsets is selected under the Configuration tab. A value of true indicates that the current result set is the last subset.

Error Output Tab

The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 29 Bulk Extract: Error Output Tab

Field	Description
error Code	The error code for errors that are generated by the underlying SmartMapper engine.
msg	The text that describes the error.
input	A copy of the data that was input to the activity.

Bulk Load

Activity

The Bulk Load activity allows you to load large amounts of data. Unlike other activities, the Bulk Load activity does not stop if an error occurs. That is, if a record fails to load into the database, the activity will not stop.

The Bulk Load activity is not supported for use with the File-Based SmartMapper Service.

Configuration Tab

Table 30 Bulk Load: Configuration Tab.

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	A short description of the activity.
Input Format	Choose the input format: • **Entity Format** Select to load entity data only. • **One Participant Relationship Format** Select if the relationship is an identity relationship. That is, there are one or multiple participants per relationship. Note that you must have one participant in your ER Model where the Cardinality is many option is cleared if you select this option. That is, one participant must have a one-to-one relationship defined. See Create Mode Setting on page 114 for information about the Input structure that must be used when this option is selected. See Add Mode Setting on page 115 for information about the Input structure that must be used when this option is selected. See Update Mode Setting on page 116 for information about the Input structure that must be used when this option is selected. • **Two Participant Relationship Format** Select if the relationship is an association relationship. That is, there are only two participants in each relationship. The relationship is many-to-many.
ER Model	Click the browse button to select the ER Model to associate with this activity.

Field	Description
Criteria	• **Name** For each entity, load records by name. • **Index** Index is the order in which the fields are defined in the schema. Index starts from 1. If specified, loading is by index.
Mode	This field only appears when the Input Format field is set to One Participant Relationship Format. Select the mode: • **Create** Any data load has a chance of collisions. When running in Create mode, the data is loaded in with minimal collision detection. This is the fastest, most efficient way to seed an empty store. • **Add** This mode checks first for duplicate records and ignores them, rather than throwing an error. • **Update** Used to replace a relationship instance with a new one. For all three modes, see more details in the section One Participant Relationship Format on page 113.
Enable Verbose Reporting	When selected, a succeeded field and failed field is reported for each record. When cleared, the report will include details only about records that caused errors. A concise summary of successful loads and loads with warnings will be given. The report will appear as part of the Output tab.

One Participant Relationship Format

When using the Input Format **One Participant Relationship Format**, the input structure is different depending on the Mode setting.

- **Create** If the Mode setting is Create, a participant with the Cardinality is many option selected can be assigned multiple values in the same relationship instance. When Create is selected, all participants must be provided in a single relationship instance.
- **Add** If the Mode setting is Add, a participant with the Cardinality is many option selected *cannot* be assigned multiple values in the same relationship instance. Instead, each value must be assigned using a unique relationship instance.

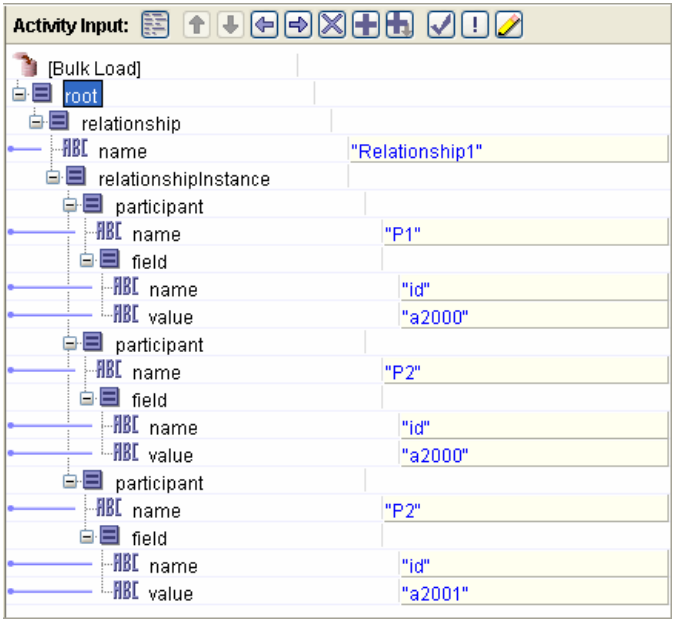
  When this mode is selected, only one participant can be primary. That is, participant entries can be split across multiple relationship instances, but each relationship instance must contain one participant (primary participant) that is already included in some other relationship instance.
- **Update** If the Mode setting is Update, Lookup is performed based on the primary participant, and if no instance is found, a new one will be created. If a lookup returns an existing relationship instance, it will be replaced with the new participant.

Create Mode Setting

Figure 62 shows a Bulk Load activity where the Input format is set to **One Participant Relationship Format** and the Mode is set to **Create**.

There are two participants, P1 and P2. P1 has the Cardinality is many option cleared (it has a one-to-one relationship) and P2 has the Cardinality is many option selected (it has a many-to-many relationship). P2 has two values assigned in the same relationship instance.

Figure 62 Input Set to One Participant Relationship; Mode Set to Create



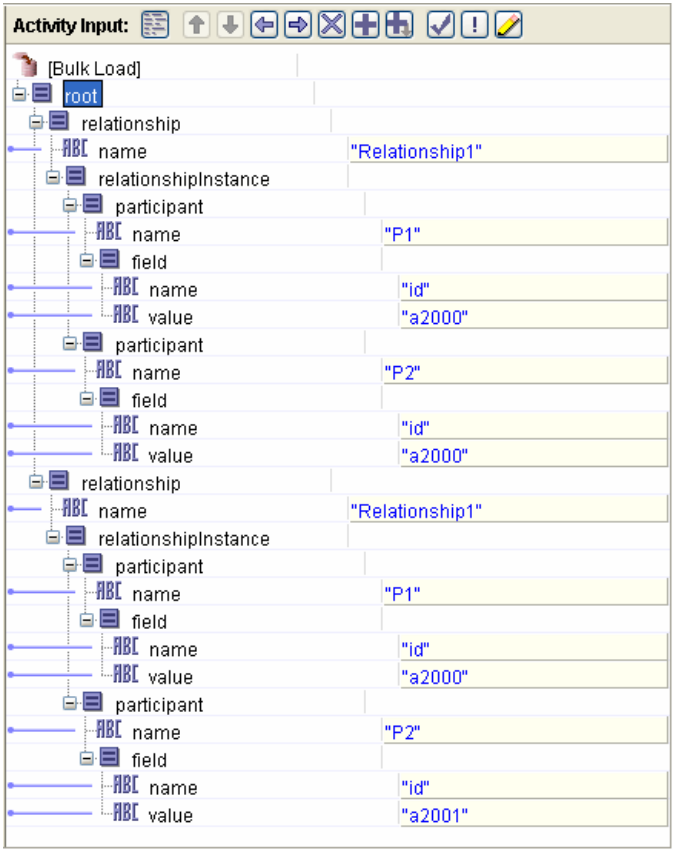
Add Mode Setting

Figure 63 shows a Bulk Load activity where the Input format is set to **One Participant Relationship Format** and the Mode is set to **Add**.

There are two participants, P1 and P2. P1 has the Cardinality is many option cleared (it has a one-to-one relationship) and P2 has the Cardinality is many option selected (it has a many-to-many relationship). P2 has two values assigned and each value is set in a different relationship instance.

Note also that if there were multiple participants with one-to-one relationships defined, only one participant could appear in subsequent relationship instances.

Figure 63 Input Set to One Participant Relationship; Mode Set to Add



Update Mode Setting

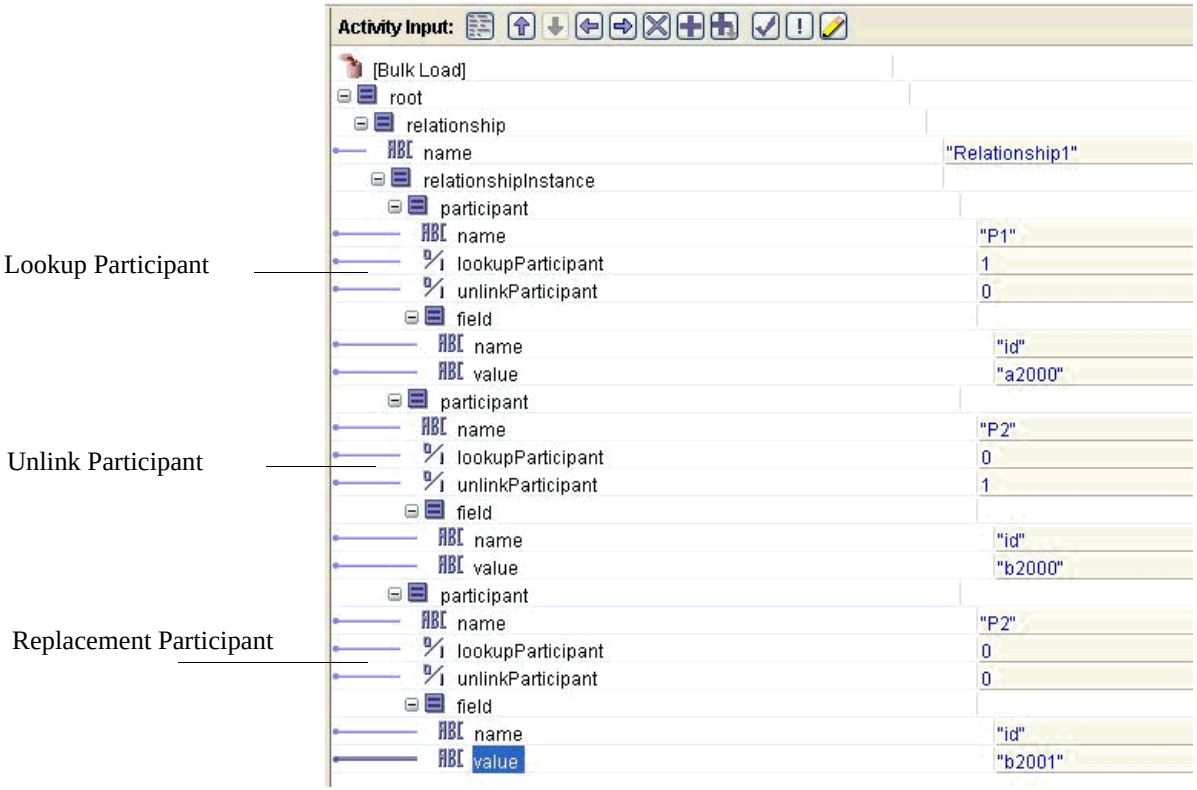
Figure 64 shows a Bulk Load activity where the Input format is set to **One Participant Relationship Format** and the Mode is set to **Update**.

There are two participants, P1 and P2. The lookupParticipant value of the P1 participant is 1, which means that the lookup is performed for a relationship instance based on this participant.

If there is a P2 Unlink Participant (value 1), then the value given in its field (b2000) will be replaced with the value given for the Replacement Participant (b2001).

If there is no P2 Unlink Participant (value 0), then the existing values in the database (such as b2003, b2004, b2005) will be replaced with the value given for the Replacement Participant (b2001). If there are no values to replace, a new value (b2001) will be added.

Figure 64 Input Set to One Participant Relationship; Mode Set to Update



Input Tab

The Input structure is different depending on the Mode setting.

See Create Mode Setting on page 114 for more information.

Table 31 Bulk Load: Input Tab

Field	Datatype	Description
entity	Complex Structure Repeating	Appears only when Entity Format is selected in the Output Format field under the Configuration tab. Provide the following as strings: • Name of the entity • Name of the field • **Field value** The value field could be empty when the entity or participant is TIBCO managed: in this case, the system will generate an ID value for it.
relationship	Complex Structure Repeating	Appears only when One Participant Relationship Format is selected in the Output Format field on the Configuration tab. Provide the following as strings: • Relationship name For the relationshipinstance, provide: • Participant name • **lookupParticipant** Looks for a relationship instance based on this participant. Values 1 or 0 indicate whether the lookup is performed. • **unlinkParticipant** Updates the existing participant with the new value. Values 1 or 0 indicate whether the update is performed. • Name of the field • **Field value** Key and non-key fields as specified by their schema definition. Each field includes the participant's field name and value.

Table 31 Bulk Load: Input Tab

Field	Datatype	Description
relationship	Complex Structure Repeating	Appears when Two Participant Relationship Format is selected in the Output Format field on the Configuration tab. Provide the following as strings: • Relationship name For the relationshipinstance, provide: • Participant1 name • Participant1 field name • **Participant1 field value** The key and non-key fields as specified by their schema definition. Each field includes the participant's field name and value. • Participant2 name • Participant2 field name • **Participant2 field value** The key and non-key fields as specified by their schema definition. Each field includes the participant's field name and value.

Use the Participant > Statement > Duplicate option for the participant as shown in Figure 65, and Field > Statement > Duplicate option for the participant's field as shown in Figure 66.

Figure 65 Duplicate Participant

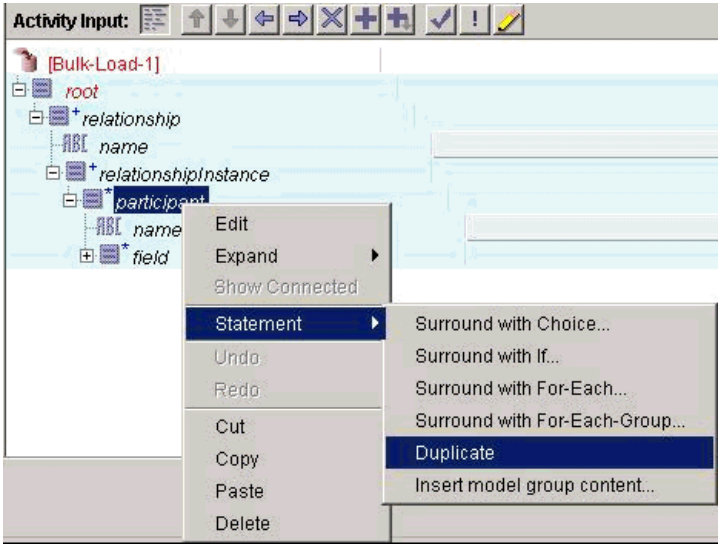
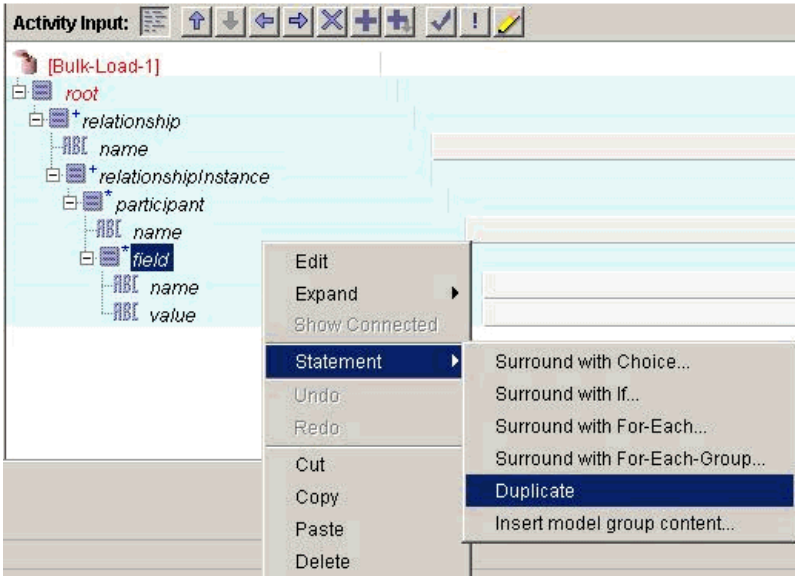


Figure 66 Duplicate Participant's Field



Output Tab

The structure under the Output tab display changes depending on the Output Format field setting as specified under the Configuration tab.

If Enable Verbose Reporting is selected under the Configuration tab, the structure under the Output tab will include a detailed report with an entry corresponding to each input record.

Error Output Tab

The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 32 Bulk Load: Error Output Tab

Field	Description
errorCode	The error code for errors that are generated by the underlying SmartMapper engine.
msg	The text that describes the error.
input	A copy of the data that was input to the activity.

Create Entity Instance

Activity

This primitive operation creates the data for an entity instance without relating the instance to any other object.

Configuration Tab

Table 33 Create Entity Instance: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	A short description of the activity.
Entity	Click the browse button to select the entity for which you are creating an instance.

Input Tab

Table 34 Create Entity Instance: Input Tab

Field	Description
Root	The schema for the entity to create. This is a repeating node allowing for the creation of multiple entity instances. See the *TIBCO BusinessWorks Process Design Guide* for an explanation of the activity input icons.

Output Tab

This tab contains a copy of the input.

Error Output Tab

The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 35 Create Entity Instance: Error Output Tab

Field	Description
msgCode	The plug-in code for the error. This is the code for errors that are generated by the SmartMapper plug-in.
msg	The text that describes the error.
errorCode	The error code for errors that are generated by the underlying SmartMapper engine.

Update Entity Instance

Activity

This primitive operation updates entity instance data. It sets the values or keys of an entity instance. This affects all relationships that the entity instance may participate in.

Configuration Tab

Table 36 Update Entity Instance: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	A short description of the activity.
Entity	The entity for which you are updating an instance.

Input Tab

Table 37 Update Entity Instance: Input Tab

Field	Description
oldObject	The schema for the entity. Map values here to identify the entity instance that is being updated.
newObject	Map values here to supply the new key values for this entity instance.

Output Tab

This tab contains a copy of the input.

Error Output Tab

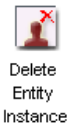
The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 38 Update Entity Instance: Error Output Tab

Field	Description
msgCode	The plug-in code for the error. This is the code for errors that are generated by the SmartMapper plug-in.
msg	The text that describes the error.
errorCode	The error code for errors that are generated by the underlying SmartMapper engine.

Delete Entity Instance

Activity



This primitive operation removes an orphaned entity from the system, that is, an entity which is not part of a relationship.

An exception is thrown if the entity is part of a relationship.

Configuration Tab

Table 39 Delete Entity Instance: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	A short description of the activity.
Entity	The entity for which you are deleting an instance.

Input Tab

Table 40 Delete Entity Instance: Input Tab

Field	Description
Root	The schema for the entity to delete. This is a repeating node allowing for the removal of multiple entity instances.

Output Tab

This tab contains a copy of the input.

Error Output Tab

The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about error messages.

Table 41 Delete Entity Instance: Error Output Tab

Field	Description
msgCode	The plug-in code for the error. This is the code for errors that are generated by the SmartMapper plug-in.
msg	The text that describes the error.
errorCode	The error code for errors that are generated by the underlying SmartMapper engine.

Create Relationship Instance

Activity

This primitive operation creates a set of entity instances and associates them in a relationship instance. The activity will also seed data into its respective entity table, if the data is not there.

For example, a relationship instance can link Customer abc-123 in Siebel to Customer 10000001 in SAP.

Configuration Tab

Table 42 Create Relationship Instance: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	A short description of the activity.
Relationship	The relationship for which this instance will be created.
showNonkeys	When selected, non-key data is shown. Use this when data is not seeded into the entity tables.

Input Tab

The input contains the keys of all the entities that participate in the chosen relationship. Map values into the keys to create the entity instances and associate them in a new relationship instance.

See the *TIBCO BusinessWorks Process Design Guide* for an explanation of the activity input icons.

Output Tab

The output contains the data for all the entity instances that were created. This will always be a copy of the input data.

Error Output Tab

The error output tab contains any error messages that may be generated. See Error Output Tab on page 127 for more details about error messages

When an error occurs during the execution of an activity, error data is generated. The Error Output tab of the activity displays the schema for the error data. The following information is displayed:

Table 43 Create Relationship Instance: Error Output for Tab

Field	Description
msgCode	The plug-in code for the error. This is the code for errors that are generated by the SmartMapper plug-in.
msg	The text that describes the error.
errCode	The error code for errors that are generated by the underlying SmartMapper engine.

Add to Relationship Instance

Activity

This primitive operation adds entity instances to an existing relationship instance.

Use this activity to associate more entity instances to an existing relationship instance, and to seed data into its respective entity table (if the data is not there).

Note that this doesn't apply to an association relationship: an "instance" is always a pair.

Configuration Tab

Table 44 Add to Relationship Instance: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	A short description of the activity.
Participant	The participant for an existing entity instance that is already a member of a relationship instance. This entity instance is used to locate the relationship instance to which the new entity instances will be added.
showNonkeys	When selected, non-key data is shown. Use when data is not seeded into the entity tables.

Input Tab

Table 45 Add to Relationship Instance: Input Tab

Field	Description
BaseEntry	The schema for the existing entity instance specified by the Participant field. Map values for an entity instance that already exists and is part of an existing relationship instance. See the *TIBCO BusinessWorks Process Design Guide* for explanation of activity input icons.
newEntries	Contains the schemas for the other participants in this relationship. Map values here to create new entity instances and add them to this relationship.

Output Tab

This tab contains a copy of the input values.

Error Output Tab

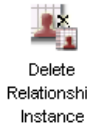
The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 46 Add to Relationship Instance: Error Output Tab

Field	Description
msgCode	The plug-in code for the error. This is the code for errors that are generated by the SmartMapper plug-in.
msg	The text that describes the error.
errCode	The error code for errors that are generated by the underlying SmartMapper engine.

Delete Relationship Instance

Activity



This primitive operation removes entity instances from a relationship instance. It does not delete entity instances themselves, but it removes the user specified memberships if they exist. Sometimes this activity will leave orphan data, such as side entries in an identity relationship instance that is left; in this case, you need to use this activity multiple times in order to clean the data.

To accomplish smart deletion, use Dynamic Delete, page 136.

To delete the entity instance as well, use Delete Entity Instance, page 124.

Configuration Tab

Table 47 Delete Relationship Instance: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	A short description of the activity.
Relationship	The relationship from which to remove entity instances.

Input Tab

Table 48 Delete Relationship Instance: Input Tab

Field	Description
keys	Contains the schemas for the participants in this relationship. Map values here for each of the entity instances that should be removed from this relationship.

Output Tab

This tab contains a copy of the input values.

Error Output Tab

The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 49 Delete Relationship Instance: Error Output Tab

Field	Description
msgCode	The plug-in code for the error. This is the code for errors that are generated by the SmartMapper plug-in.
msg	The text that describes the error.
errCode	The error code for errors that are generated by the underlying SmartMapper engine.

Lookup

Activity

This primitive operation accepts a participant as input, and then retrieves one or more participants by performing lookups on relationships. Null is returned whenever a lookup fails.

Configuration Tab

Table 50 Lookup: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	A short description of the activity.
Input Format	The type of data to be looked up. Two data types are supported: Entity and Relationship.
Entity	The entity that you want to look up. **Note:** This field is only available when Entity is selected in the Input Format field.
Input Participant	The participant which will be used to locate a relationship instance. **Note:** This field is only available when Relationship is selected in the Input Format field
Output Participants	The list of participants to look up. Note that the SmartMapper lookup activity supports lookup of only one output participant. **Note:** This field is only available when Relationship is selected in the Input Format field

Input Tab

Table 51 Lookup: Input Tab

Field	Description
Root	The schema for the input participant. These values will be used to find a relationship instance. See the *TIBCO BusinessWorks Process Design Guide* for an explanation of the activity input icons.

Output Tab

Table 52 Lookup: Output Tab

Field	Description
Root	The schemas for the participants that were found by the lookup. These will be the entity instances that were associated with the input participant in a relationship instance. The output data will contain nodes only for the output participants that were found. Output participants that are not found will return no nodes in the output data.

Error Output Tab

The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 53 Lookup: Error Output Tab

Field	Description
msgCode	The plug-in code for the error. This is the code for errors that are generated by the SmartMapper plug-in.
msg	The text that describes the error.
errCode	The error code for errors that are generated by the underlying SmartMapper engine.

Dynamic Lookup

Activity

This activity retrieves one or more participants by performing lookups on relationships.

In a dynamic lookup, relationship and participant information is derived from the data, rather than modeled at design time. This allows you to specify information during runtime (dynamically). You need not define data at design time.

If a lookup fails, the output is empty.

The activity also operates on a set of data, like the SmartMapper Wizard activity.

The Dynamic Lookup activity is not supported for use with the File-Based SmartMapper Service.

Configuration Tab

Table 54 Dynamic Lookup: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	Short description of the activity.
Input Format	Select the input format: Relationship or Entity. Dynamic Lookup can be key based or non-key based.
ER Model	Click the browse button to select the ER Model to associate with this activity.
Criteria	• **Name** For each entity, lookup records by name. • **Index** Index is the order in which the fields are defined in the schema. Index starts from 1. If specified, loading is by index.
Show Record Timestamp	When selected, timestamps for records are included in the report. When cleared, timestamps are not included in the report.

Input Tab

Table 55 Dynamic Lookup: Input Tab

Field	Datatype	Description
relationship	Complex Structure	Provide the following as strings: • Name of the relationship • inputParticipant name and field. This field is a complex structure (repeating). It includes only key fields. Field can be a name or index depending on the value selected for Criteria. Name is the field name in the entities schema, and value is the key value. • outputParticipant name. This name field could be empty, in which case all participants related to the given input participant will be outputted.
entity	Complex Structure	Provide the following as strings: • Entity name • Field name (field name in the entities schema) • Field value

Output Tab

This tab contains a copy of the input values with the output filled in.

Error Output Tab

The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 56 Dynamic Lookup: Error Output Tab

Field	Description
errCode	The error code for errors that are generated by the underlying SmartMapper engine.
msg	The text that describes the error.
input	A copy of the data that was input to the activity.

Dynamic Delete

Activity

This activity allows for deletion of a relationship instance using dynamic runtime inputs.

This activity is a smart deletion activity which means that it will not leave orphan data such as in the case of the activity Delete Relationship Instance on page 130.

The Dynamic Delete activity is not supported for use with the file-based SmartMapper service.

Configuration Tab

Table 57 Dynamic Delete: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	Short description of the activity.
Input Format	Select one of the two input formats: • **Relationship** A relationship is a collection of entities that a user wants to bind together in a particular way. For more details, see Relationships on page 33. • **Entity** Something that is being referenced, such as a customer record. For more details, see Entity on page 31.
ER Model	Click the browse button to select the ER Model to associate with this activity.
Criteria	• **Name** For each entity, lookup records by name. • **Index** Index is the order in which the fields are defined in the schema. Index starts from 1. If specified, loading is by index.

Input Tab

Table 58 Dynamic Delete: Input Tab

Field	Datatype	Description
relationship	Complex Structure	Provide the following as strings: • **Relationship name** Name of the relationship • **inputParticipant** name and field The inputParticipant field is a complex structure (repeating), which includes only key fields. It can be a name or an index, depending on the value selected for Criteria. The name entry is the field name in the entities schema, and the value is the key value. • **deletionParticipant** name and field The deletionParticipant field is a complex structure (repeating), which includes only key fields. It can be a name or an index, depending on the value selected for Criteria. The name entry is the field name in the entities schema, and value is the key value. This entry could be empty, which means that the whole relationship will be deleted. If only the deletionParticipant name is specified, it means that only the entries belonging to the given participant will be deleted. If all of the deletionParticipant entry is specified, it means that only the specified participant will be unlinked. If the entered participant's cardinality is one, the relationship that the participant participates will also be deleted.
entity	Complex Structure	Provide the following as strings: • **Entity name** Name of the entity • **Entity field** This structure contains entity field name and entity field value.

Output Tab

This tab contains a copy of the input values with the output filled in.

Table 59 Dynamic Delete: Output Tab

Field	Datatype	Description
relationship	Complex Structure	Contains the following elements: • StatusCode • Msg • name • instanceID • **inputParticipant** This structure contains the relationship name and field (with name and value strings) • **deletedParticipant** This structure contains the relationship name and field (with name and value strings)
entity	Complex Structure	Contains the following elements: • StatusCode • Msg • **Entity name** Name of the entity • **Entity field** This structure contains the index, entity field name and entity field value.

Error Output Tab

The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 60 Dynamic Delete: Error Output Tab

Field	Description
errCode	The error code for errors that are generated by the underlying SmartMapper engine.
msg	The text that describes the error.

Generate Key

Activity

This primitive operation generates a unique key for an entity. The operation does not create the entity. It takes an entity as a parameter to generate the key. The output has only one field: id.

If you want to add a special class to define GUIDs, see Adding Your Own GUID Generation Class on page 190.

Configuration Tab

Table 61 Generate Key: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	Short description of the activity.
Entity	The entity for which a key is being generated. See Entity on page 84.

Input Tab

This activity requires no input.

Output Tab

Table 62 Generate Key: Output Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
id	The generated key value.

Error Output Tab

The Error Output tab lists the possible exceptions that can be thrown by this activity. See Error Output Tab on page 127 for more details about the error messages.

Table 63 Generate Key: Error Output Tab

Field	Description
msgCode	The plug-in code for the error. This is the code for errors that are generated by the SmartMapper plug-in.
msg	The text that describes the error.
errCode	The error code for errors that are generated by the underlying SmartMapper engine.

SmartMapper Wizard

Activity

The SmartMapper Wizard allows you to do lookups, generate new keys, and maintain mappings within a single activity.

While the GUI does not enforce this, all operations within a SmartMapper Wizard activity must use the same ER Model.

Batching

The SmartMapper Wizard activity sends requests to SmartMapper services in batches. This is done for performance reasons and has different implications for the different service types.

- For the JDBC-Based SmartMapper Service, all operations invoked by the SmartMapper Wizard activity are done in one transaction.
- For the File-Based SmartMapper Service, all operations invoked by the SmartMapper Wizard activity are saved with one write to the file. In other words, if the SmartMapper Wizard activity involves creating three relationship instances, those three edits of the data result in one write operation to the file.
- For the Adapter-Based SmartMapper Service, the batch size parameter in the Advanced tab determines the size of the batches sent by the SmartMapper Wizard. In other words, if the SmartMapper Wizard activity invokes 100 operations and the batch size is set to 10, this will result in 10 messages being sent to the adapter each with 10 operations in them.

  Each of these batches is executed as soon as they are received and with a high degree of concurrency. In other words, the adapter does not wait for all the batches to arrive before starting to execute them. Further, the adapter executes the batches at the same time as much as possible. Data that is inter-data dependent should be grouped in the same batch.

  This makes the Adapter-Based SmartMapper Service very efficient for handling large maps with many SmartMapper operations. The batch size parameter can be used to tune this efficiency with small batch sizes yielding higher concurrency and large batch sizes minimizing network transmissions.

  If the transport (set up under the Adapter-Based SmartMapper Service, under Subscriber) is set up in DQ model, multiple adapter services may work on the data to maximize throughput.

Configuration Tab

Table 64 SmartMapper Wizard: Configuration Tab

Field	Description
Name	The name to appear as the label for the activity in the process definition.
Description	Short description of the activity.
Verbose	If selected, and if the Error on empty results field under the SmartMapper Calls tab is cleared, the Wizard will generate the following output: • groups — existing output • SmartMapper Exception • path. Path (XPath) pointing to the input member for which the Lookup failed. • errorCode. Error code. • Msg. Error message, detailing cause of error and possible resolution • inputRow. Row of input data (entity or relationship) that caused this error. This could be used as is for retries later on

Input Editor Tab

This tab is used to provide the logical source schema. The SmartMapper Wizard takes an XSD or you may build one, as with many activities.

SmartMapper Calls Tab

This annotates the input schema for logical locations for keys or references. The field selected does not actually have to be part of the key. The location is important because of the way the mapper establishes context.

The Input Participant and Output Participant fields are for participants available in the Relationships folder. Note that all operations within a SmartMapper Wizard activity must use the same ER Model.

In the maintain case, the second participant is enabled for input too.

To start using this tab, select an attribute.

Table 65 SmartMapper Wizard: Calls Tab

Field	Description
Input Schema	This is the schema tree in the left pane. It is used to browse the input schema and select the nodes where a SmartMapper operation should be invoked.

Field	Description
Invoke SmartMapper	Check this to invoke a SmartMapper operation at the selected schema node. Choose from the operations: • Relationship Lookup. See Invoke SmartMapper Listbox: Relationship Lookup Operation on page 143. • Entity Lookup. See Invoke SmartMapper Listbox: Entity Lookup Operation on page 144. • Maintain Mapping. See Invoke SmartMapper Listbox: Maintain Mapping Operation on page 144. This operation does not apply to the association relationship. • Maintain Mapping - One to Many. See Invoke SmartMapper Listbox: Maintain Mapping Operation: One-to-Many on page 145. This operation does not apply to the association relationship..

In the SmartMapper Wizard Call tab, the SmartMapper ListBox includes several choices, which are described below.

Invoke SmartMapper Listbox: Relationship Lookup Operation

This operation takes the participant input and returns an output participant. The two participants are the parameters. More than one instance will be returned if the output participant is in the many relationship. For further information, see Invoke SmartMapper Listbox: Maintain Mapping Operation: One-to-Many on page 145.

If the input evaluates to null, a lookup will not be performed. Such a lookup will return null unless you select the check box to throw an exception.

Figure 67 SmartMapper Wizard: Lookup Operation

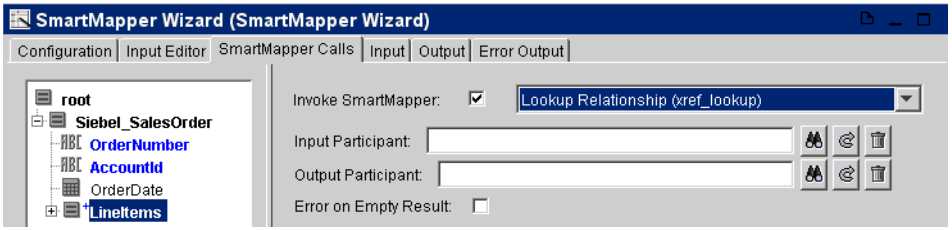


Table 66 SmartMapper Wizard: Lookup Operation Tab

Field	Description
Input Participant	The known participant. Note that the same ER Model must be used throughout the SmartMapper Wizard activity.
Output Participant	The participant that is being looked-up.

Field	Description
Error on Empty Result	When selected, the activity throws an exception if the lookup fails to find the output participant. If this box is cleared, see the Verbose field description under the Configuration tab.

Invoke SmartMapper Listbox: Entity Lookup Operation

This operation allows users to look up entities according to the input data.

Figure 68 SmartMapper Wizard: Entity Lookup Operation

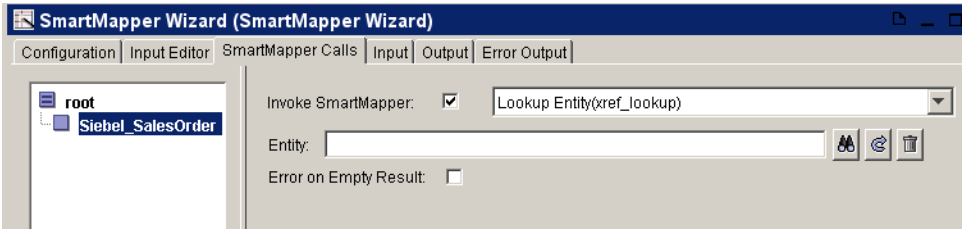


Table 67 SmartMapper Wizard: Entity Lookup Operation

Field	Description
Entity	Specifies the entity is being looked up. **Note:** the same ER Model must be used throughout the SmartMapper Wizard activity.
Error on Empty Result	When selected, the activity throws an exception if the lookup fails to find the entity. If this box is cleared, see the Verbose field description under the Configuration tab.

Invoke SmartMapper Listbox: Maintain Mapping Operation

This maintains a one-to-one mapping between two participants. This operation creates and associates two entity instances within a relationship instance. The creation/association only occurs if it does not already exist.

Figure 69 SmartMapper Wizard: Maintain Mapping Operation

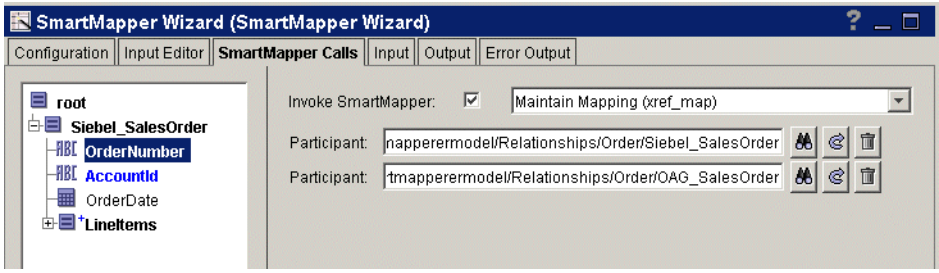


Table 68 SmartMapper Wizard: Maintain Mapping Operation

Field	Description
Participant	Select the two participants that are being maintained.

Invoke SmartMapper Listbox: Maintain Mapping Operation: One-to-Many

This maintains a one-to-many mapping. This operation employs the same semantics as the Maintain Mapping operation above, but this operation also creates and associates entity instances in a one-to-many mapping.

This operation also observes the TIBCO-Managed entity rules and generates keys for TIBCO-Managed entities only if no values are mapped to them in the Input tab.

Figure 70 SmartMapper Wizard: Maintain Mapping Operation One-to-Many

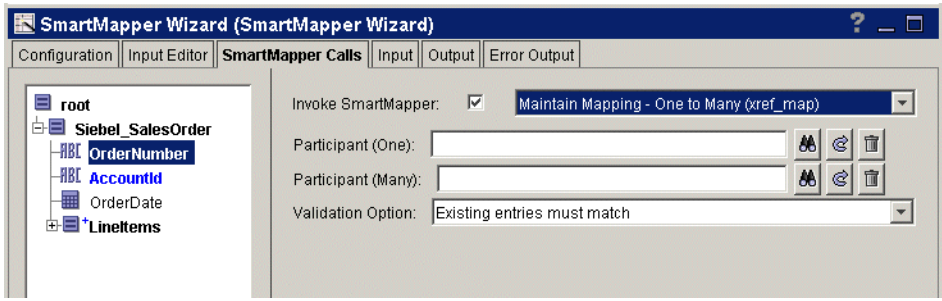


Table 69 SmartMapper Wizard: Maintain Mapping Operation One-to-Many

Field	Description
Participant (One)	The participant on the one side of the one-to-many association.
Participant (Many)	The participant on the many side of the one-to-many association. This participant must have selected the Cardinality Is Many check box in the ER Model definition.

Field	Description
Validation Option	Choose the validation behavior when the relationship instance already exists. Existing entries must match. In this case, the values passed into the Input tab must match the existing values with no additions or removal of entries on the many side of the relationship. Replace exiting entries. In this case, the values passed into the Input tab are meant to replace any existing values on the many side of the relationship. Add any new entries. In this case, the values passed into the Input tab are to be added if they do not already exist in the many side of the relationship.

Input Tab

In the Input tab for the SmartMapper Wizard Activity, the Activity Input schema is based on the schema provided in the Input Schema tab.

Also, the schema has been modified to include nodes for each operation defined in the SmartMapper Calls tab. These nodes are placed in the locations specified in the SmartMapper Calls tab. These SmartMapper nodes represent the input data to the SmartMapper operations defined in the SmartMapper Calls tab.

Dragging the Schema to the Design Window should bring up the autofill wizard to set all non-SmartMapper fields.

Only use copy on sub-schemas that do not contain cross-reference calls. This ensures that your final object will have both the lookups and the normal data in one object.

The SmartMapper Wizard allows calls to be batched to the database efficiently. Reads may be done in parallel. Writes may be done in one transaction. Exception semantics allow the capture of all lookup failures into one exception schema. Any real exception terminates the activity and rolls back any writes.

Configuration for all of this is set up in the service.

Your next activity should be a Mapper.

In SmartMapper Wizard, the input field values can be optional if both sides of the input entities participating in a relationship are null in the Maintain Mapping operation. Otherwise, you will get a message that the key value is not present if one of the entities is null.

Output Tab

The Output tab schema is also based on the schema provided in the Input Schema tab. SmartMapper nodes have also been placed in the output schema. These nodes represent the output returned by the SmartMapper operations.

You can view this in debug mode to see the new data

Error Output Tab

The error data contains information about which operation failed as well as the data that was passed into the operation.

Table 70 SmartMapper Wizard: Error Output Tab

Field	Description
msgCode	The plug-in code for the error. This is the code for errors that are generated by the SmartMapper plug-in.
msg	The text that describes the error.
errCode	The error code for errors that are generated by the underlying SmartMapper engine.
operationType	The type of the operation that failed: LOOKUP, MAINTAIN, or MAINTAIN_ONE_TO_MANY
path	The XPath path of the operation that failed.
input	A copy of the input data for the operation that failed.
participant1	The first input participant. This will contain the name/value pairs for the data of the first input participant.
participant2	The second input participant. This will contain the name/value pairs for the data of the second input participant. Note that for MAINTAIN_ONE_TO_MANY operations, this participant may have more than one instances.

For information on error codes, see Error Codes on page 201.

Chapter 9 Managing Local Cache Data with TIBCO Hawk

This chapter describes how to use TIBCO Hawk microagents to manage local cache data.

Topics

Overview of TIBCO Hawk

TIBCO Hawk is a sophisticated tool for enterprise-wide monitoring and managing of all distributed applications and systems. System administrators can use it to monitor adapters in a wide area network of any size. TIBCO Hawk can be configured to monitor system and adapter parameters and to take actions when predefined conditions occur. These actions include: sending alarms that are graphically displayed in the TIBCO Hawk display, sending email, paging, running executables, or modifying the behavior of a managed adapter.

Unlike other monitoring applications, TIBCO Hawk relies on a purely distributed intelligent agent architecture using publish or subscribe to distribute alerts. TIBCO Hawk uses TIBCO Rendezvous for all messaging and thus gains the benefits and scalability from the TIBCO Rendezvous features of publish/subscribe, subject name addressing, interest-based routing, and reliable multicast.

TIBCO Hawk is a purely event-based system that uses alerts. The agents are configured with rules that instruct them on everything from what and how to monitor to what actions to take when problems are discovered. Thus the workload is fully distributed throughout the enterprise. Every agent is autonomous in that it does not depend on other components to perform its functions.

The TIBCO Hawk Enterprise Monitor consists of these components:

- **Display**—GUI front end that displays alarms and provides editors to create rule bases, create tests, view messages, and invoke microagents to request information or initiate an action.
- **Agents**—Intelligent processes that perform monitoring and take actions as defined in rules.
- **Rulebases**—Rules that are loaded by agents to determine agent behavior.
- **Application Management Interface (AMI)**—Manages network applications via TIBCO Rendezvous and supports communication between a network application and monitoring TIBCO Hawk agents, including the ability to examine application variables, invoke methods, and monitor system performance.
- **Microagents**—Feed information back to TIBCO Hawk and expose action methods to rulebases.

  For more information, see the TIBCO Hawk documentation.

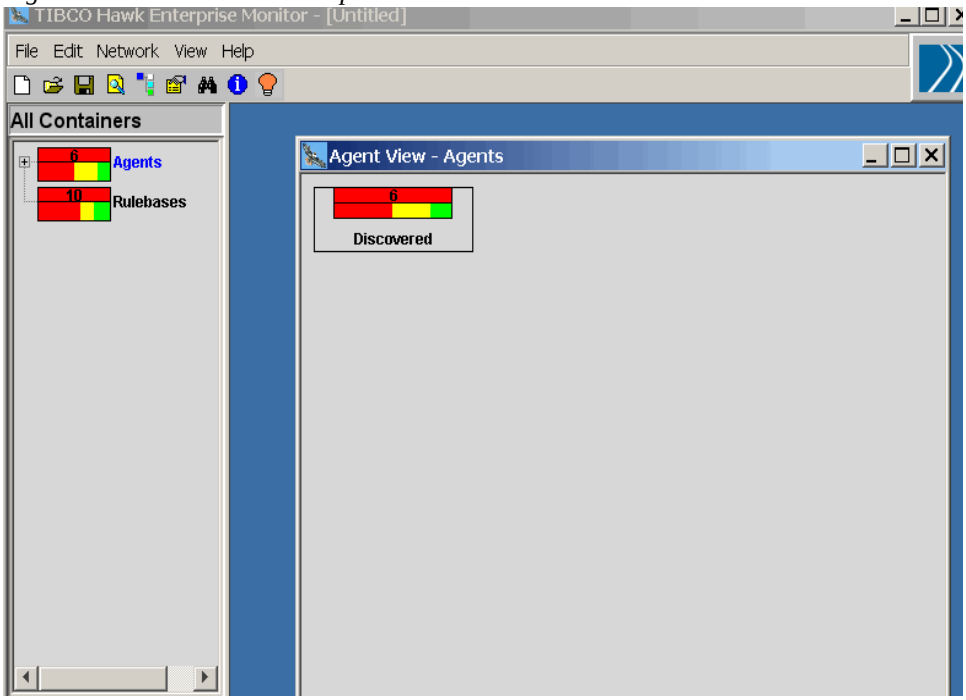
Invoking a SmartMapper Microagent Method

A set of default microagents is loaded when a TIBCO Hawk Agent is started.

To invoke a SmartMapper microagent method, do the following:

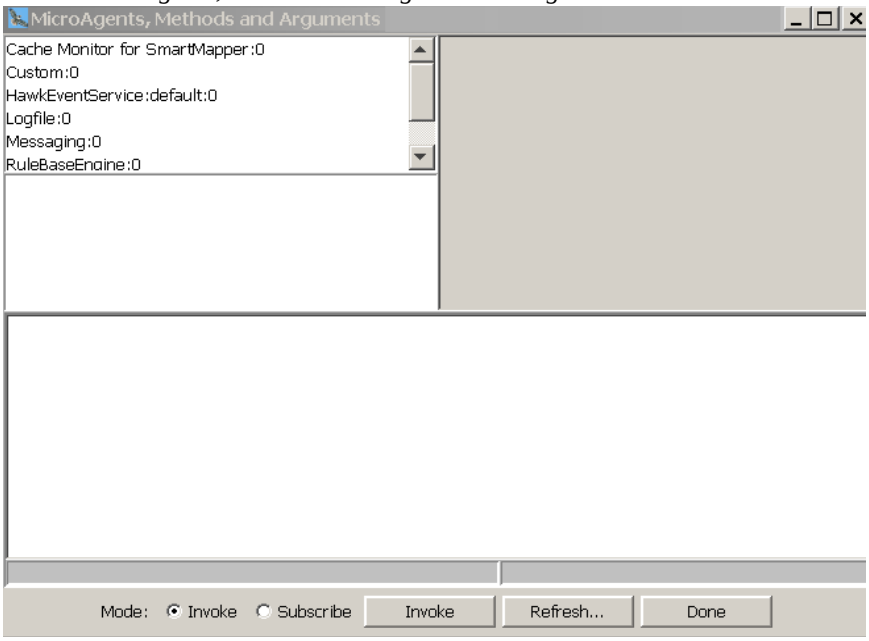
1. From the Start menu, select **All Programs > TIBCO > TIBCO Hawk *version_number* > Hawk Display**. The TIBCO Hawk Enterprise Monitor window is displayed, as shown in Figure 71.

Figure 71 TIBCO Hawk Enterprise Monitor



2. Right-click the **Agents** icon in the right panel. All the machines where all discovered agents are located are displayed.
3. Right-click the machine in which the agent resides and select **Get MicroAgents** from the list. The Microagent, Methods and Arguments dialog is displayed, as shown in Figure 72.

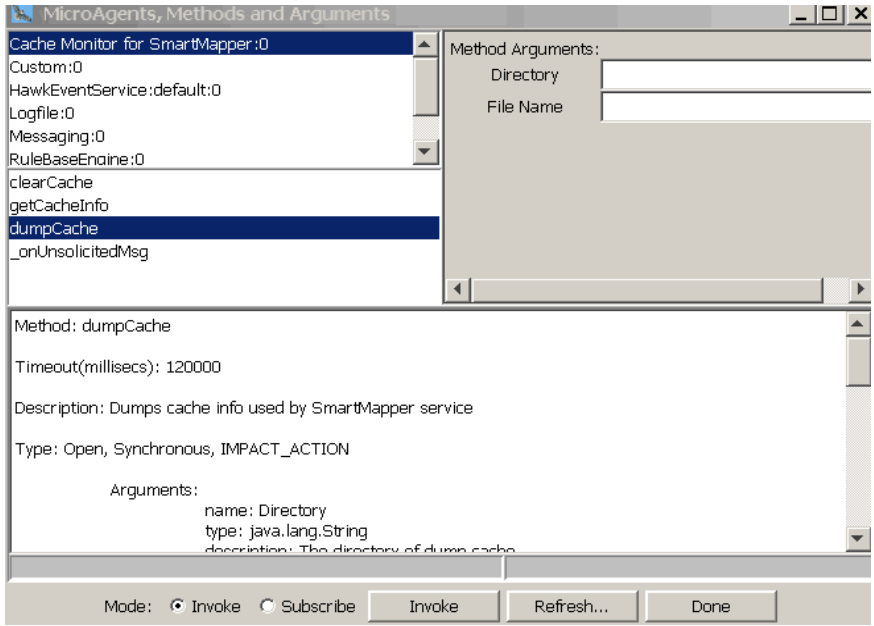
Figure 72 Microagents, Methods and Argument Dialog



4. Click the **Cache Monitor for SmartMapper** microagent on the upper-left list. A list of associated methods and text descriptions is displayed, as shown in Figure 73.

Ensure that the local caching function is enabled and a SmartMapper process is currently running.

Figure 73 Dump Cache Method



5. Click the name of the method to invoke, such as **Dump Cache**. Table 71 explains available SmartMapper methods.
6. Specify arguments for the method invocation. If the method accepts arguments, fields for each argument display in the upper-right panel. Detailed help text displays in the lower panel.
7. Click **Invoke** in the Mode field.
8. Click the **Invoke** button to invoke the selected method.

   The Invocation Results dialog displays the results returned by the method.
9. Click **Done** to close the dialog.

Table 71 Available Cache Methods for SmartMapper

Method	Description	Arguments
clearCache	Cleans up all the cache data.	No argument is required.
getCacheInfo	Returns cache information.	No argument is required.
dumpCache	Dumps cache data to a local file.	Two arguments must be specified: **Directory** The location where the cache file is saved. **File Name** The name of the cache file.

Chapter 10 SmartMapper Tools

This chapter introduces the command-line tools shipped with TIBCO ActiveMatrix BusinessWorks SmartMapper, which allow users to bulk load and extract cross-referencing data, compare two ER Model files, migrate relationship data, and upgrade SmartMapper databases.

Topics

Overview of SmartMapper Tools

A set of command-line tools is shipped with TIBCO ActiveMatrix BusinessWorks SmartMapper, which allows users to manage the cross-referencing data stored in the database and upgrade SmartMapper databases.

Ensure that the SmartMapper tools are selected when installing TIBCO ActiveMatrix BusinessWorks SmartMapper.

The following command-line tools are available:

- **Bulk Extract Tool** allows users to extract the relationship data or the entity data in batch.

  See Bulk Extract Tool on page 158 for more details.

- **Bulk Load Tool** allows users to load the relationship data or the entity data in batch.

  See Bulk Load Tool on page 160 for more details.

- **Differencing Tool** allows users to compare two ER Models.

  See Differencing Tool on page 162 for more details.

- **Migration Tool** allows users to migrate the relationship data residing in a database to another empty database.

  See Migration Tool on page 164 for more details.

- **Database Upgrade Tool** allows users to upgrade a database used for TIBCO ActiveMatrix BusinessWorks SmartMapper 5.x. After upgrading, the database can be used in TIBCO ActiveMatrix BusinessWorks SmartMapper 6.0.

  See Database Upgrade Tool on page 165 for more details.

A database must be accessed each time the command-line tools are run. Instead of specifying the database connection parameters one by one, you can prepare the databases to use prior to running the command-line tools. See Database Configuration on page 157 for more details about how to prepare a database for future use.

Database Configuration

You can set up the database configurations before running the SmartMapper command-line tools. If not, you have to type all the parameters used to connect to the database every time you run SmartMapper tools.

TIBCO ActiveMatrix BusinessWorks SmartMapper provides a db.properties file to set up database configurations. Hence, each time the SmartMapper tools are run, you just need to enter the database alias which points to the database that you want to use.

Complete the following steps to prepare databases for future use:

1. Locate the db.properties file in the *SmartMapper_HOME*\bin directory.
2. Open the db.properties file. The supported database types are MySQL, DB2, Oracle, and Microsoft SQL Server. For more details, see the readme file.
3. Set up database connection parameters according to the information listed in Table 72.

   Note: The asterisk (*) represents the alias of the database. It can be any character you want to use and cannot be omitted, which is used to locate the database. In the given example, the database alias is *mysql*.
4. Save your configurations.

Table 72 Database Configurations

Database Property	Description
*.type	The type of the database.
*.driver	The name of the driver class. **Note:** JDBC 4.0 is supported by JDK 6.0 by default. If JDBC 4.0 or above is being used, this parameter is ignored. SmartMapper will use the database driver defined in the database URL.
*.url	The URL to use to connect to the database.
*.username	The username to use when connecting to the database.
*.password	The password to use when connecting to the database.

Bulk Extract Tool

The Bulk Extract tool is used to export entity or relationship data to a local file.

To run this tool, do the following:

1. On the command line, enter the following command to direct to the location of SmartMapper tools. By default, all SmartMapper tools are located in the *SmartMapper_HOME*\bin directory.
   - **On Windows** cd *SmartMapper_HOME*\bin
   - **On Unix** cd *SmartMapper_HOME*/bin
2. Enter **bulkextract** on the command line and press **Enter** to get help information.
3. Enter the following command to export the desired entity or relationship.

   bulkextract -db <*dbconfig*> -f <*filename*> -r <relname> | -e <*entityname*> [-c <*criteria*>]

   Note: The entity and relationship data cannot be extracted simultaneously. That is, each time you run this tool, only one type of data can be extracted, either entity or relationship.

   Table 73 explains the parameters used in this command.

   A log file named *bulk_extract.log* is generated in the *SmartMapper_HOME*\bin\logs directory when running the tool.

Table 73 Bulk Extract Parameters

Parameter	Comment
-db	Specifies the database where the entity or the relationship that you want to export is stored. The value of this parameter is either the database alias configured in the *db.properties* file, or a group of parameters used to connect to the database. If you do not use the database alias, the following database related parameters must be specified: type, driver, URL, username, and password. For example: -db *mysql.jdbc.Driver jdbc:mysql:localhost:3306/test root admin*
-f	Specifies the name of the file where the extracted data is stored.
-e	Specifies the name of the entity to be extracted.
-r	Specifies the name of the relationship to be extracted. **Note:** The parameter -e and -r cannot be used together.

Table 73 Bulk Extract Parameters (Cont'd)

Parameter	Comment
-c	Specifies a criterion to filter the entity instance or the relationship instance that you want to export. This parameter is optional. When multiple filter criteria are provided, each of them is AND'ed if operator and value are specified using the where clause. Conditions are OR'ed together if operator and value are specified using the orWhere clause. A filter criterion consists of three parts, which are separated by comma (,). The first part is the filter condition, where or orWhere; the second part is the entity's field name, or the participant name and the corresponding field name separated by a dot; the third part is the value of the field. Use semicolon (;) to separate multiple filter criteria. For example: Filter enter data: -c "where,key1,like'%1%';where,key1,!='a21'" Filter relationship data: -c "where,A.key1,='a11';orWhere,C.key1,='c1'"

Bulk Load Tool

The Bulk Load tool is used to load a batch of entity or relationship data to a specified database.

Ensure that the data to be loaded conform to rules described in Data Format on page 161.

To run this tool, do the following:

1. On the command line, enter the following command to direct to the location of SmartMapper tools. By default, all SmartMapper tools are located in the *SmartMapper_HOME*\bin directory.
   - **On Windows** cd *SmartMapper_HOME*\bin
   - **On Unix** cd *SmartMapper_HOME*/bin
2. Enter **bulkload** on the command line and press **Enter** to get help information.
3. Enter the following command to load the relationship data to the specified database.

   bulkload -db <*dbconfig*> -f <*filename*> -r | -e

   Note: The entity and relationship data cannot be loaded simultaneously. That is, each time you run this tool, only one type of data can be loaded, either entity or relationship.

   Table 74 explains the parameters used in this command.

   A log file named *bulk_load.log* is generated in the *SmartMapper_HOME*\bin\logs directory after running the tool.

Table 74 Bulk Load Parameters

Parameter	Description
-db	Specifies the database where the entity or the relationship data is loaded. The value of this parameter is either the database alias configured in the *db.properties* file, or a group of parameters used to connect to the database. If you do not use the database alias, the following database related parameters must be specified: type, driver, URL, username, and password. For example: -db *mysql.jdbc.Driver jdbc:mysql:localhost:3306/test root admin*
-f	Specifies the name of the file that contains the entity data or the relationship data to be loaded. Ensure that the data in the specified file conform to the rules described in Data Format on page 161.
-e	Indicates that the data to be loaded is entity.
-r	Indicates that the data to be loaded is relationship.

Data Format

The data to be loaded must conform to the following rules:

- One file must contain one type of data, either entity or relationship. The file cannot contain both the entity data and the relationship data.
- The data must be in JSON format.
- One JSON object is one line.

Relationship Data Format

The following syntax shows the format of relationship data:

{"name":"*relationship_name*","participants":[{"name":"*participant_name*","fields":{"*key1*":"*value1*","*key2*":"*value2*","*key3*":"*value3*"}},{"name":"*participant_name*","fields":{"id":"*value*"}}]}

If the participant is a TIBCO-Managed entity, the participant should be {"id":"*value*"}. SmartMapper generates the entity with the specified value. If no value is specified, SmartMapper randomly assigns a value for the entity.

For example: {"name":"R1","participants":[{"name":"A","fields":{"key1":"a11","key2":"a2","key3":"a3"}},{"name":"B","fields":{"id":""}}]}

Entity Data Format

The following syntax shows the format of entity data:

{"name":"*entity_name*","fields":{"*key1*":"*value1*","*key2*":"*value2*","*key3*":"*value3*"}}

For example:

{"name":"A","fields":{"key1":"a1","key2":"a2","key3":"a3"}}

Differencing Tool

The Differencing tool is used to compare two SmartMapper ER Models. The ER Models to be compared can be located by an Enterprise Archive (EAR) file, a database, or directly providing the location of the ER Model where it is located.

To run this tool, do the following:

1. On the command line, enter the following command to direct to the location of SmartMapper tools. By default, all SmartMapper tools are located in the *SmartMapper_HOME*\bin directory.
   - **On Windows** cd *SmartMapper_HOME*/bin
   - **On Unix** cd *SmartMapper_HOME*/bin
2. Enter **diff** on the command line and press **Enter** to get help information.
3. Enter the following command to compare two ER Models.

   diff -src[ear <*EAR file directory!ER Model file*>][file <*ER model file path*>][db <*dbconfig*>]-dest[ear <*EAR file directory!ER Model file path*>][file <*ER model file path*> |[db <*dbconfig*>]

   Table 75 explains the parameters used in this command.

   A log file named *diff.log* is generated in the *SmartMapper_HOME*\bin\logs directory after running the tool.

Table 75 Differencing Tool Parameters

Parameters	Description
-src	Specifies the source ER Model to be compared. It is a composite parameter, following with a parameter used to define how to locate ER Models. The value of this parameter depends on which parameter is followed. Three parameters are available: ear, db, and file.
-dest	Specifies the destination ER Model that the source ER Model is compared with. It is a composite parameter, followed by a parameter used to define how to locate ER Models. The value of this parameter depends on which parameter is followed. Three parameters are available: ear, db, and file.
The following are parameters combined with -src and -dest parameters.	

Table 75 Differencing Tool Parameters (Cont'd)

Parameters	Description
ear	Specifies an Enterprise Archive (EAR) file in which the ER Model is to be compared. The value of this parameter consists of two parts separated by an exclamation mark (!). The first part is the path of the EAR file; the second part is the folder where the specified ER Model is located, which is detailed to the ER Model name. **Note:** If the file name or the file path contains spaces, enclose the value in the quotation marks (""). For example: diff -src file "*c:\temp\SmartMapper ERModel.ermodel*" -dest ear "*c:\sample.ear!Model\SmartMapper ERModel.smartmapperermodel*"
db	Specifies a database in which the ER Model is to be compared. The value of this parameter is either the database alias configured in the *db.properties* file, or a group of parameters used to connect to the database. If you do not use the database alias, the following database related parameters must be specified: type, driver, URL, username, and password. For example: diff -src *db mysql.jdbc.Driver jdbc:mysql:localhost:3306/test root admin* -dest db *mysql*
file	Specifies an ER Model file to be compared. The value of this parameter is the path of the ER Model file, which is detailed to the ER Model name. **Note:** If the file name or the file path contains spaces, enclose the value in the quotation marks (""). For example: diff -src file "*c:\temp\SmartMapper ERModel.ermodel*" -dest ear "*c:\sample.ear!Model\SmartMapper ERModel.smartmapperermodel*"

Migration Tool

The Migration tool is used to migrate cross-referencing data from one database to another empty database. TIBCO ActiveMatrix BusinessWorks SmartMapper supports migrating cross-referencing data between databases of different database types. The supported database types are MySQL, Oracle, DB2, and Microsoft SQL Server.

Note: Ensure that the database to which the data is migrated is empty.

To run this tool, do the following:

1. On the command line, run the following command to direct to the location of SmartMapper tools. By default, all SmartMapper tools are located in the *SmartMapper_HOME*\bin directory.
   - **On Windows** cd *SmartMapper_HOME*\bin
   - **On Unix** cd *SmartMapper_HOME*/bin
2. Enter **migrate** on the command line and press **Enter** to get help information.
3. Enter the following command to migrate the cross-referencing data from the source database to the destination database.

   migrate -src <*db configure*> -dest <*db configure*>

   Table 76 explains parameters used in this command.

A log file named *migrate.log* is generated under the *SmartMapper_HOME*\bin\logs directory after running the tool.

Table 76 Migrate Tool Parameters

Parameter	Description
-src	Specifies the source database from which the cross-referencing data is migrated. The value of this parameter is either the database alias configured in the *db.properties* file, or a group of parameters used to connect to the database. If you do not use the database alias, the following database related parameters must be specified: type, driver, URL, username, and password. For example: -db *mysql.jdbc.Driver jdbc:mysql:localhost:3306/test root admin*
-dest	Specifies the destination database to which the cross-referencing data is migrated. The value of this parameter is either the database alias configured in the *db.properties* file, or a group of parameters used to connect to the database. If you do not use the database alias, the following database related parameters must be specified: type, driver, URL, username, and password. For example: -db *mysql.jdbc.Driver jdbc:mysql:localhost:3306/test root admin*

Database Upgrade Tool

The database used for SmartMapper 5.x must be upgraded after installing SmartMapper 6.0. The Database Upgrade tool is used to upgrade the database used for SmartMapper 5.x to SmartMapper 6.0.

Note: Ensure that the 5.x database is synchronized and backed up.

To run this tool, do the following:

1. On the command line, run the following command to direct to the location of SmartMapper tools. By default, all SmartMapper tools are located in the *SmartMapper_HOME*\bin directory.
   - **On Windows** cd *SmartMapper_HOME*\bin
   - **On Unix** cd *SmartMapper_HOME*\bin
2. Enter **upgrade** on the command line and press **Enter** to get help information.
3. Enter the following command to migrate the cross-referencing data from the source database to the destination database.

   upgrade -db *<db config>* -ermodel *<ermodel file>*

   Table 77 explains parameters used in this command.

   A log file named *upgrade.log* will be generated in the *SmartMapper_HOME*\bin\logs directory after running the tool.

Table 77 Database Upgrade Tool Parameters

Parameter	Description
-db	Specifies the database to be upgraded. The value of this parameter is either the database alias configured in the *db.properties* file, or a group of parameters used to connect to the database. If you do not use the database alias, the following database related parameters must be specified: type, driver, URL, username, and password. For example: -db *mysql.jdbc.Driver jdbc:mysql:localhost:3306/test root admin*
-ermodel	Specifies the ER Model to be upgraded. The value of this parameter is the path of the EAR file, which is detailed to the ER Model name. For example: upgrade -db *mysql* -ermodel *c:\tmp\lookup-entity.ear!Model*

Chapter 11 Using Sample Projects

This chapter describes sample projects packaged with TIBCO ActiveMatrix BusinessWorks SmartMapper.

Topics

Overview of the Examples

TIBCO ActiveMatrix BusinessWorks SmartMapper is shipped with three examples. This chapter describes how to run the examples on a Microsoft Windows platform.

The following example folders are available in the *SmartMapper_HOME*\examples directory:

- **Getting_Started** shows how to seed data to the ER Model and look up relationship data.

  See Simple Tutorial: Seed and Lookup on page 169 for details.

- **Wizard** shows how to use the SmartMapper Wizard activity to perform a batch of SmartMapper operations.

  See Advanced Tutorial: SmartMapper Wizard on page 177 for details.

- **BulkExtractFilter** shows how to bulk extract relationship data using different criteria.

  See Advanced Tutorial: Bulk Extracting Data with Different Criteria on page 184 for details.

Simple Tutorial: Seed and Lookup

This tutorial addresses a common cross-referencing requirement in which an inbound message contains a reference to an object in one application that needs to be synchronized with objects in one or more other applications. The tutorial is included in the installation package in the directory *SmartMapper_HOME*\examples\Getting_Started .

Here is what the tutorial covers:

- First, the mapping relationships are created by defining the entities or keys involved in each relationship, and then defining the relationships themselves.
- Second, the service used for this mapping model is configured. The service defines where this mapping will be stored. Our options include file, database or adapter-based.
- Next, the data has to be seeded. A BusinessWorks process is included, which reads an XML file and uses the data in that file to populate the mapping.
- Finally, we will review and run the process to lookup a Siebel Account ID value in the cross-reference table.

Business Scenario

Here, the inbound message from a Siebel application references a Siebel customer through a Siebel_Account_ID field. Some of the data in this inbound message must be used to generate an outbound message to an OAG-based common data model. This OAG-based common data model uses an OAG_Party_ID field to refer to customers.

The inbound Siebel_Account_ID must be cross-referenced with the corresponding OAG_Party_ID on the OAG-based common data model at runtime. Also, if such a cross-reference does not exist, a new OAG_Party_ID must be created and cross-referenced with the inbound Siebel_Account_ID for possible future use.

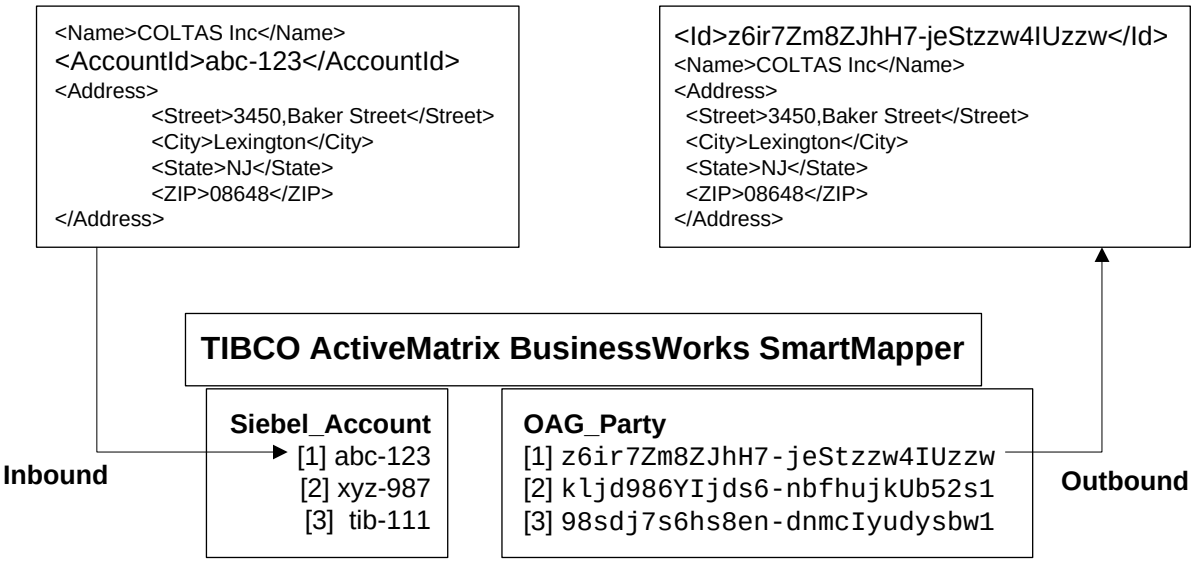
TIBCO ActiveMatrix BusinessWorks SmartMapper Solution

TIBCO ActiveMatrix BusinessWorks SmartMapper can perform the necessary cross-referencing required by the business scenario. The Siebel_Account_ID field is cross-referenced with the OAG_Party_ID field. When an inbound Siebel account is received by the TIBCO ActiveMatrix BusinessWorks SmartMapper cross-referencing engine at runtime, it is cross-referenced with the corresponding OAG_Party_ID, which is returned as output.

For entity values that do not exist in an underlying data store, TIBCO ActiveMatrix BusinessWorks SmartMapper dynamically generates an OAG_Party_ID, and then creates a cross-reference in the model at runtime.

Figure 74 shows how TIBCO ActiveMatrix BusinessWorks SmartMapper takes the inbound message from a Siebel application, cross-references the inbound Siebel_Account_ID with the corresponding OAG_Party_ID, and creates an outbound message for the OAG-based common data model that the other applications can understand.

Figure 74 Creating an Outbound Message for the OAG-based Common Data Model

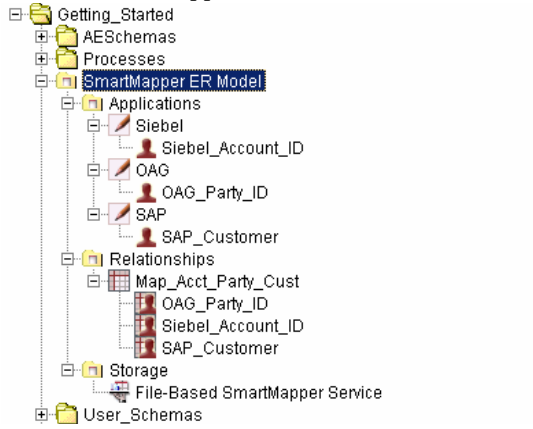


ER Model Overview

An ER Model must be created before creating a SmartMapper process. See Creating an ER Model on page 18 for details about creating an ER Model.

Three applications and one relationship are defined in this ER Model. As shown in Figure 75, the file-based SmartMapper service is used to store the relationship data. Each application has an entity associated with and each entity has a data schema defined. The entities become the participants of the Map_Acct_Party_Cust relationship.

Figure 75 Getting Started: SmartMapper ER Model



Process Definition

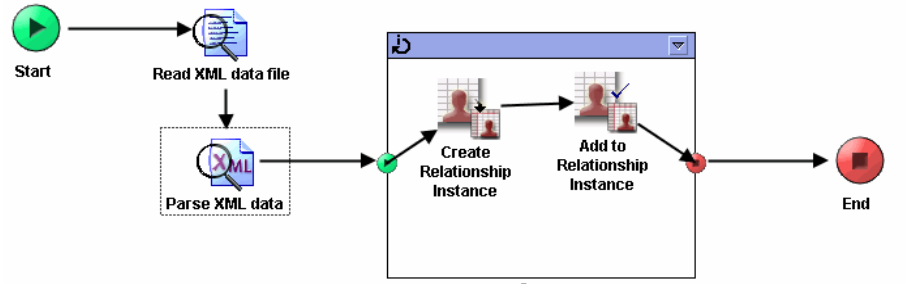
Two processes are included in this project:

- Seed All Data into Model, page 171
- Lookup Data, page 172

Seed All Data into Model

This process demonstrates how to seed relationship data to the created ER Model.

Figure 76 The Seed All Data into Model Process



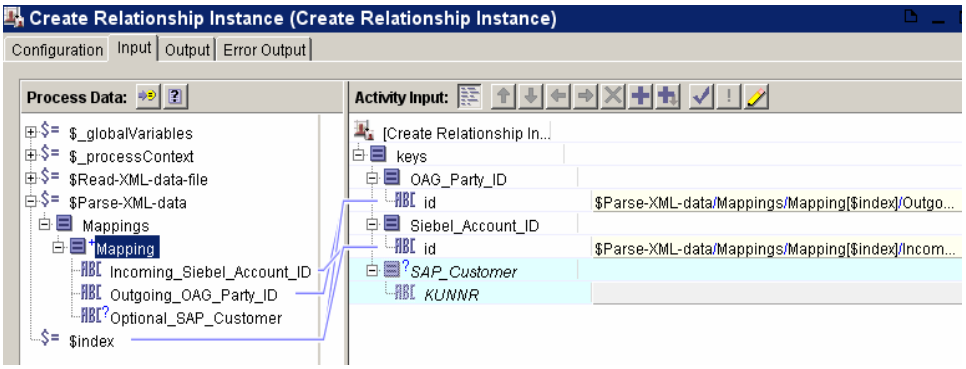
The process performs the following operations:

1. The Read XML data file activity reads the predefine file where contains the relationship data. The Seed_Data file is in the User_Schema folder.
2. The Parse XML data activity parses the schema data into XML data. The output of the Read XML data file activity is the input of this activity.

3. The Create Relationship Instance activity creates a relationship instance for the Map_Acct_Party_Cust relationship defined in the ER Model.

   The Siebel_Account_ID and the OAG_Party_ID are passed into this activity as input, as shown in Figure 77. These were parsed from the Seed_Data.xml file.

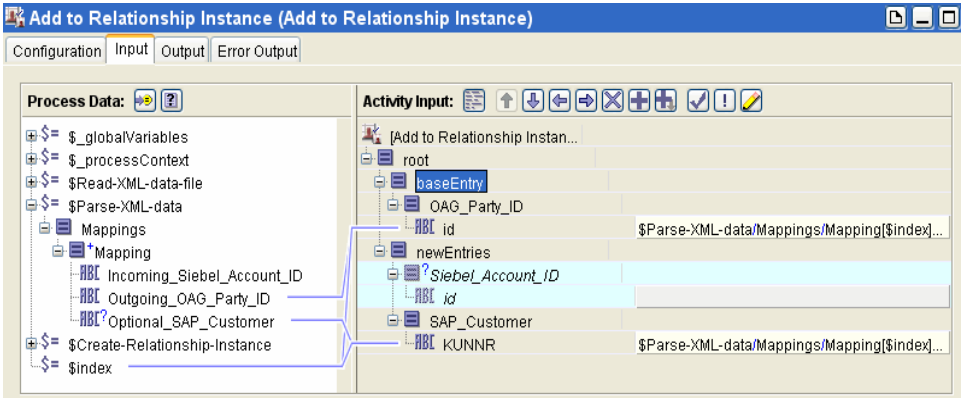
Figure 77 Create Relationship Instance: Input



4. The Add to Relationship Instance activity adds participants to the relationship instance added by using the Create Relationship Instance activity.

   The OAG_Party_ID under the baseEntry node is an entity instance that already exists in an existing relationship. Siebel_Account_ID and SAP Customer under the newEntries node are new entity instances that need to be created and added to the relationship.

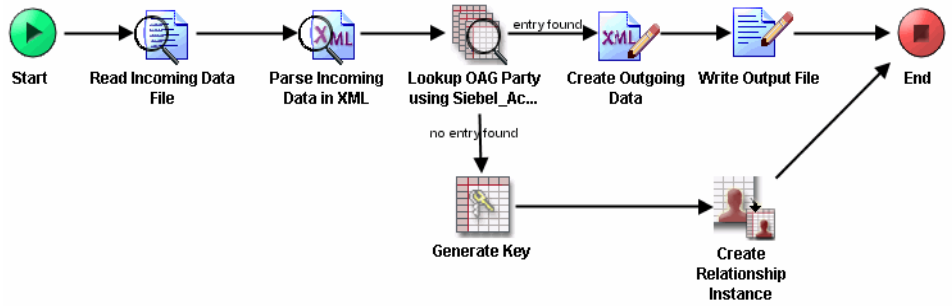
Figure 78 Add to Relationship Instance: Input



Lookup Data

This process shows how to look up Siebel_Account_ID in the cross-reference tables.

Figure 79 The Lookup Data Process



The following activities verify the mappings:

- Read Incoming Data File Activity
  - Input tab: the fileName in the Activity Input pane is configured to point to the Siebel_Account.xml file. The Siebel_Account.xml file is stored in the directory which is defined in the SmartMapper.demo.directory global variable.
  - Output tab: the output for this activity is a string contained in the textContent field. This will be used as an input for the Parse Incoming Data in XML activity.
- Parse Incoming Data in XML Activity
  - Input tab: the xmlString field in Activity Input pane contains the output from the Read Incoming Data File activity.
  - Output tab: content of the parsed Siebel_Account.xml file, based on the schema defined in the schema field in the Configuration tab.
  - Output Editor tab: the Siebel_Account.xsd schema in the Schema field is used to validate the contents of the Siebel_Account.xml file that the Read Incoming Data File activity produces.
- Look-up OAG Party using Siebel_Account Activity
  - Input tab: the Id field in the Activity Input pane is configured to point to the AccountId property of the Siebel_Account in the Process Data pane. The AccountId is the output of the Parse Incoming Data In XML activity.
  - Output tab: the output for this activity is the OAG_Party_ID key that is returned by the mapping service.

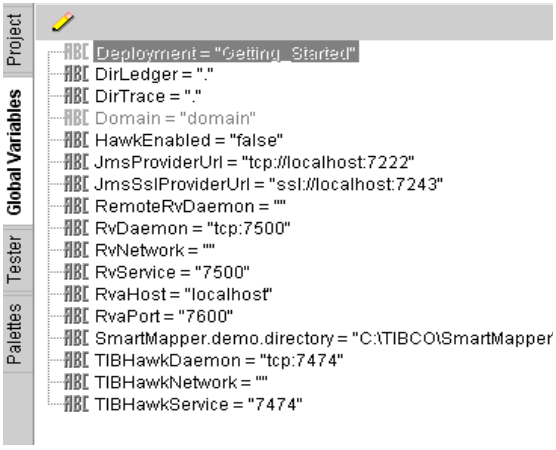
Setting Up the Project

Before running the project, follow these steps to configure the project:

1. Start TIBCO Designer.

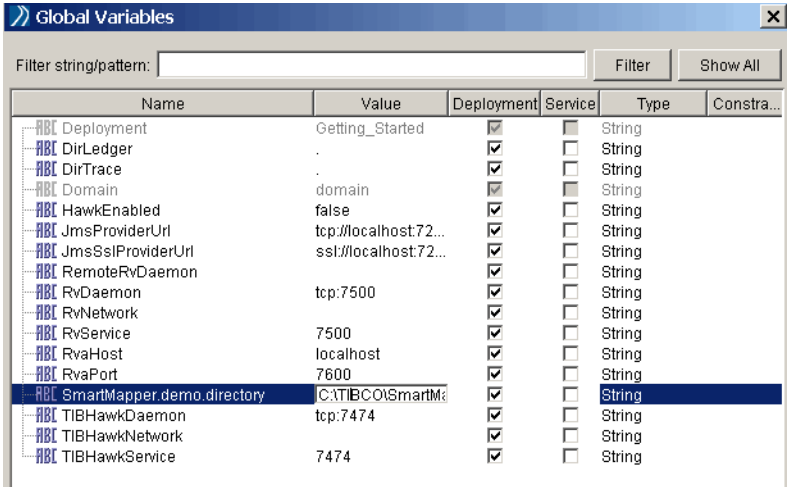
2. Click the **Open Existing Project** button in the TIBCO Designer startup panel. The Open Project dialog is displayed.
3. Click the **Browse** button next to the Project Directory field, and then select the Getting_Started folder, which is located in the *SmartMapper_HOME*\examples directory. Click the **OK** button. The project is displayed.
4. Click the **Global Variables** tab in the Project panel. The Global Variables panel is displayed as shown in Figure 80.

Figure 80 Seed and Lookup Example: Global Variables Panel



5. Click the button at the top. The Global Variables dialog is displayed.

Figure 81 Seed and Lookup Example: Global Variables Dialog



6. Click the **SmartMapper.demo.directory** variable and double-click its value in the Value column.
7. Provide the absolute path to the Getting_Started example. By default, the example is located in the *SmartMapper_HOME*\examples\Getting_Started directory.
8. Click the **OK** button.

Running the Project

After setting up the project, complete the following steps to trigger the Seed Data process or the Lookup Data process:

1. In the Project panel, click the **Tester** tab and then click the button at the top left corner in this panel. The Select Processes To Load dialog is displayed.
2. Expand the **Processes** folder and check the **Seed All Data into Model** process or the **Lookup Data** process.
3. Click the **Load Selected** button. The process is started.

   When the process executes, the elements of the process change colors depending upon what is occurring in the executing process instance. If all the transition lines change to green, it means the process runs successfully.
4. Click the button to return to the Design mode.

You can set breakpoints in the process definition at a specified point where you want to stop a running process and examine its state and process data, for example, you can set breakpoints before or after an activity.

See *TIBCO BusinessWorks Process Design Guide* for detailed information on using the test mode including setting breakpoints and the element colors in the test mode.

Expected Results

If the processes perform successfully, the green color is displayed between each activity.

Running the Lookup Data process will result in one of the following outcomes:

- If using the Siebel_Account_ID, the OAG_Party_ID will not be found in the mapping table, the process takes the No Entry Found transition.

  See Scenario 1: The Lookup Fails on page 176 for details.
- If using the Siebel_Account_ID, the OAG_Party_ID will not be found in the mapping table, the process takes the Entry Found transition.

  See Scenario 2: The Lookup Returns a Single Value on page 176 for details.

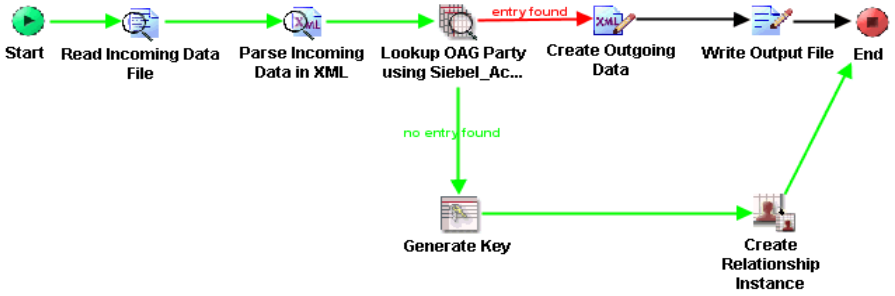
Scenario 1: The Lookup Fails

In this case, using the Siebel_Account_ID, a key for the OAG_Party_ID is not found in the cross-reference mapping tables.

When the lookup fails, the Generate Key activity comes into play.

Figure 82 shows the route taken by the process in the event that a lookup failure occurs.

Figure 82 No Key Found in Mapping Tables



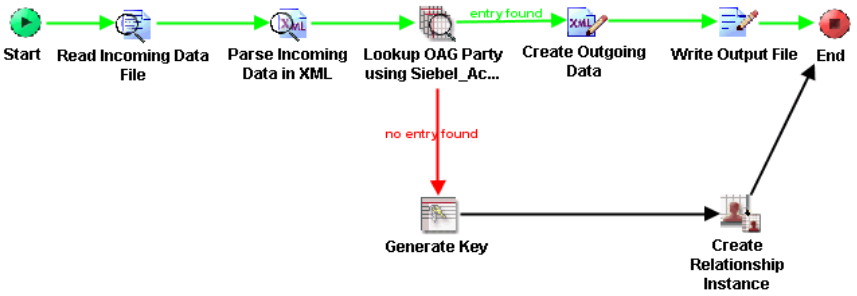
Scenario 2: The Lookup Returns a Single Value

In this case, using the Siebel_Account_ID, a key for the OAG_Party_ID is found in the cross-reference mapping tables.

When the user runs the process for a second time, the process will succeed because a key for OAG_Party_ID was generated (corresponding to the input Siebel_Account_ID) in the scenario in which the lookup failed.

Figure 83 shows the route taken by the process in the event that a lookup succeeds.

Figure 83 Key Found in Mapping Tables



Advanced Tutorial: SmartMapper Wizard

This tutorial shows how to use the SmartMapper Wizard in the context of a BusinessWorks process.

As is often the case, when a new sales order arrives, a corresponding order on a remote system must be created at runtime and linked to the inbound sales order.

Business Scenario

In this tutorial an order is mapped to a common data model so that it may be published out to the application. Some subscribers may be interested in doing billing, shipping, provisioning, or planning. All subscribers need to know that a new sales order was booked.

Because there are many things to cross-reference on a sales order, and because there are repeated elements like order lines, the SmartMapper Wizard is used.

TIBCO ActiveMatrix BusinessWorks SmartMapper Solution

The SmartMapper Wizard maps several fields to produce output that is used to generate an OAG_SalesOrder.

The following steps show an overview of how TIBCO ActiveMatrix BusinessWorks SmartMapper works:

1. The inbound message is taken from a Siebel application.
2. The inbound order number is cross referenced with the generated order number.
3. An outbound message is created that can be understood by other services on the integration backbone.

(1) Inbound Message from Enterprise Application

```
<OrderNumber>100</OrderNumber>
<AccountId>abc-123</AccountId>
<OrderDate>2003-01-31</OrderDate>
<LineItems>
  <LineNumber>10</LineNumber>
  <ProductId>101111</ProductId>
  <Quantity>3</Quantity>
</LineItems>
……
```

(2) TIBCO ActiveMatrix BusinessWorks SmartMapper Cross-Reference Tables

OrderNumber [1] **100**	ID [1]cross-referenced with remote ID: **1YrEz4qfZMI4v-Hjdxzzw4r-zzw**
AccountId [1] **abc-123** ……	PartyId [1] z6ir7Zm8ZJhH7-jeStzzw4IUzzw cross-referenced with remote PartyId: Wc5jtvOUZMI4xUHjdyzzw4r-zzw ……

(3) Outbound Message to Enterprise Application

```
<Id>1YrEz4qfZMI4v-Hjdxzzw4r-zzw</Id>
<PartyId>Wc5jtvOUZMI4xUHjdyzzw4r-zzw
 </PartyId>
 <DocumentDateTime>2003-01-31
  </DocumentDateTime>
<Line>
 <LineNumber>
  QvmSp5B3ZMI4xUHjdzzzw4r-zzw
  </LineNumber>
 <ItemId>
  cd6Nfd5IZMI4xUHje-zzw4r-zzw
  </ItemId>  <OrderQuantity>3</OrderQuantity>
</Line>
 ……
```

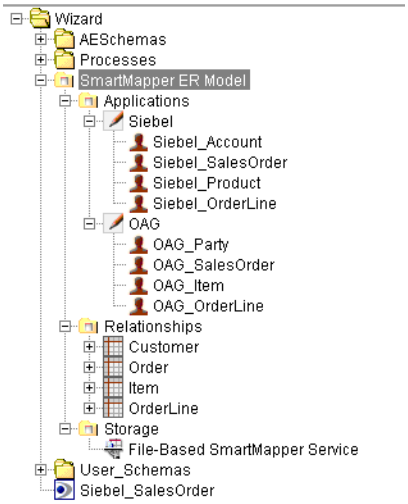
ER Model Overview

An ER Model must be created before creating a SmartMapper process. See Creating an ER Model on page 18 for details about creating an ER Model.

The ER Model used in this project contains two applications and four relationships. As shown in Figure 84, the file-based SmartMapper service is used to store the relationship data.

Four entities are added in each application and each entity has its own data schema associated with. Select one entity in the application node and click the **Schema** tab in the Configuration panel to view the defined schema.

Figure 84 The ER Model of the Wizard Project



Process Definition

The TIBCO ActiveMatrix BusinessWorks Create Sales Order process has been included in this example. Except for some general activities, such as the Read File and the Parse XML activities, the following activities are used:

- Reviewing the SmartMapper Wizard Activity, page 179
- Reviewing the Map All Data Activity, page 180

Reviewing the SmartMapper Wizard Activity

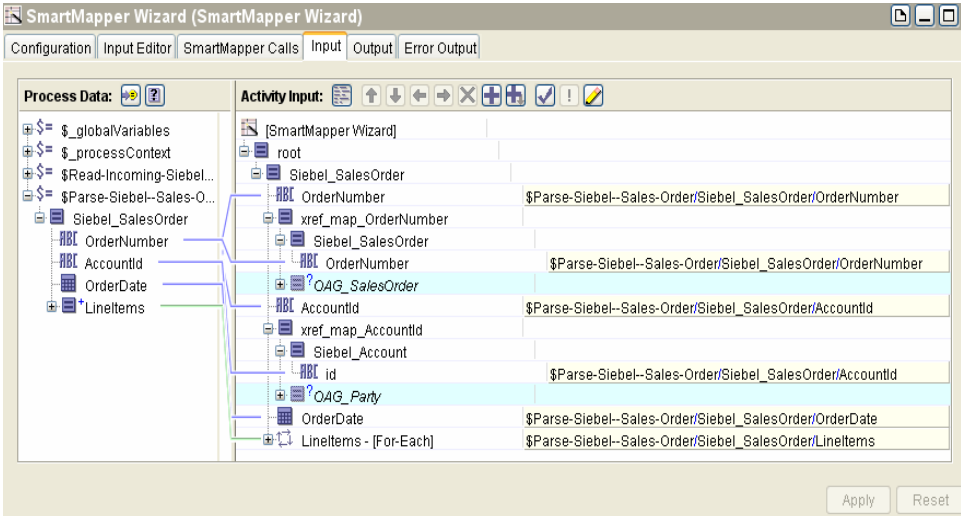
The SmartMapper Wizard activity allows you to perform several operations such as lookups, automatic key generation and mappings in the same activity. For more details about configuring the SmartMapper Wizard, see SmartMapper Wizard on page 141.

As shown in Figure 85, in this tab you can reference the actual fields in the inbound Siebel sales order. Because Maintain Mapping was chosen in the Invoke SmartMapper list in the SmartMapper Calls tab, you have the option to input the matched field, not just the input field.

If you do not want the SmartMapper Wizard to do the mapping, but rather give specific values for the input and output participants, complete both added elements, that is, all the participants under the xref node.

Note that the SmartMapper Wizard can only generate a key for an entity which is set as a TIBCO-Managed Entity. In this case, because the entities in the OAG application are all set as TIBCO-Managed Entity, their values can be empty, such as the OAG_SalesOrder and OAG_Party entities shown in the Activity Input pane.

Figure 85 Input Tab Configuration



Reviewing the Map All Data Activity

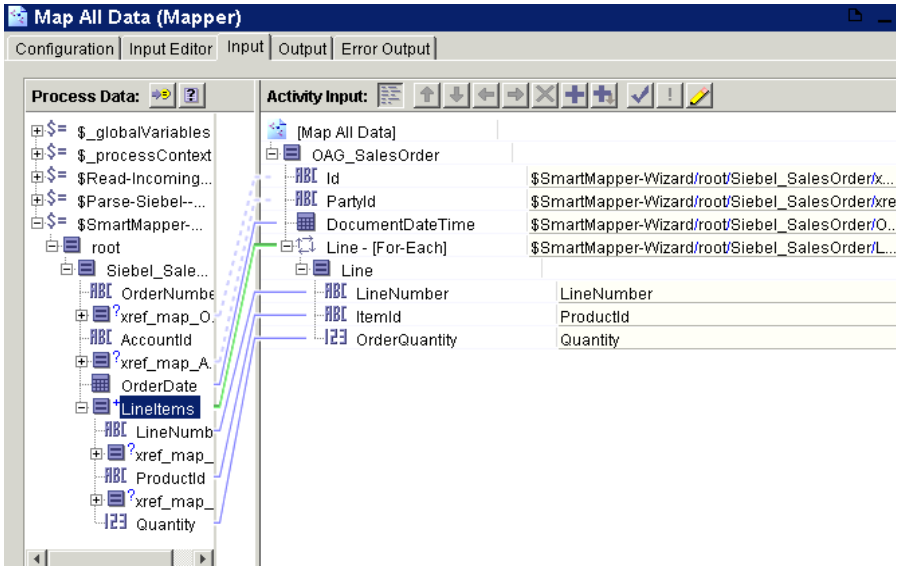
Now that the lookup or creation of fields corresponding to certain fields in the inbound Siebel sales order has been set up, the OAG sales order can be created. The Mapper activity can be used to create the OAG sales order.

As shown in Figure 86, the data passed as output from the SmartMapper Wizard activity is mapped into the fields of the OAG_SalesOrder schema in the Map All Data activity.

The green line from the SmartMapper Wizard LineItems field to the OAG sales order Line field indicates that the line sequence will contain the same number of elements as the inbound line item sequence.

The output of the Map All Data activity is an OAG sales order.

Figure 86 Map All Data: Input Tab

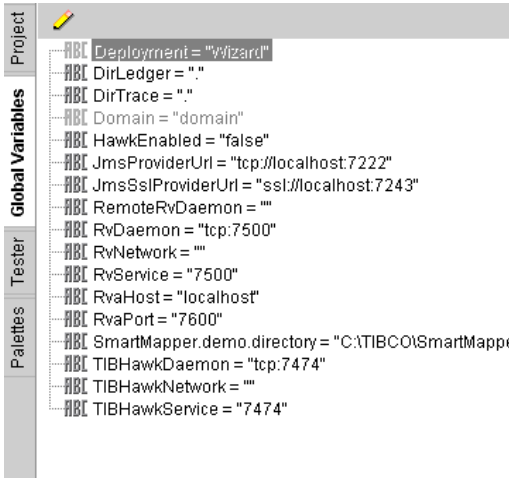


Setting Up the Project

Before running the project, follow these steps to configure the project:

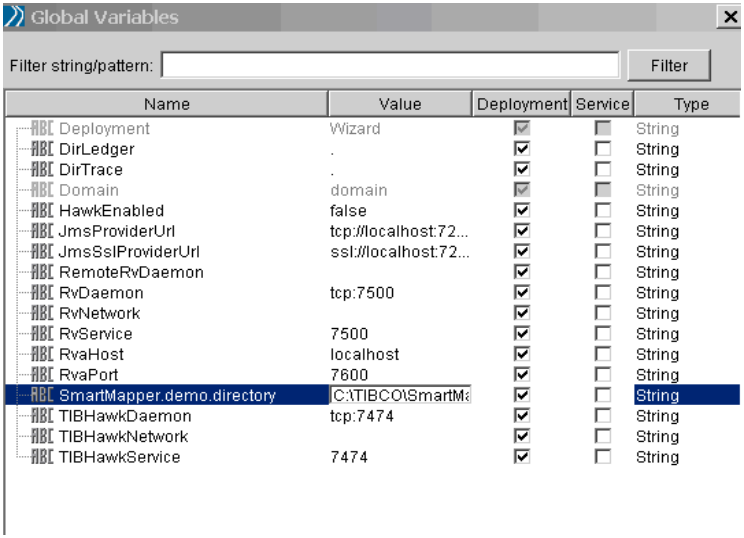
1. Start TIBCO Designer.
2. Click the **Open Existing Project** button in the TIBCO Designer startup panel. The Open Project dialog is displayed.
3. Click the **Browse** button next to the Project Directory field, and then select the wizard folder, which is located in the *SmartMapper_HOME*\examples directory. Click the **OK** button. The project is displayed.
4. Click the **Global Variables** tab in the Project panel. The Global Variables panel is displayed as shown in Figure 87.

Figure 87 Wizard Example: Global Variables Panel



5. Click the button at the top. The Global Variables dialog is displayed.

Figure 88 Wizard Example: Global Variables Dialog



6. Click the **SmartMapper.demo.directory** variable and double-click its value in the Value column.
7. Provide the absolute path to the Wizard example. By default, the example is located in the *SmartMapper_HOME*\examples\Wizard directory.
8. Click the **OK** button.

Running the Project

After setting up the project, complete the following steps to trigger the Create Sales Order process:

1. In the Project panel, click the **Tester** tab and then click the  button at the top left corner in this panel. The Select Processes To Load dialog is displayed.
2. Expand the **Processes** folder and check the **Create Sales Order** process.
3. Click the **Load Selected** button. The process is started.

   When the process executes, the elements of the process change colors depending upon what is occurring in the executing process instance. If all the transition lines change to green, it means the process runs successfully.
4. Click the  button to return to the Design mode.

You can set breakpoints in the process definition at a specified point where you want to stop a running process and examine its state and process data, for example, you can set breakpoints before or after an activity.

See *TIBCO BusinessWorks Process Design Guide* for detailed information on using the test mode including setting breakpoints and the element colors in the test mode.

Expected Results

If the processes perform successfully, the green color is displayed between each activity.

Advanced Tutorial: Bulk Extracting Data with Different Criteria

This tutorial shows how to use the Bulk Extract activity to look up relationship data based on different criteria. The example discussed in this section is located in the BulkExtractFilter folder.

The following search criteria are used in the BulkExtractFilter project:

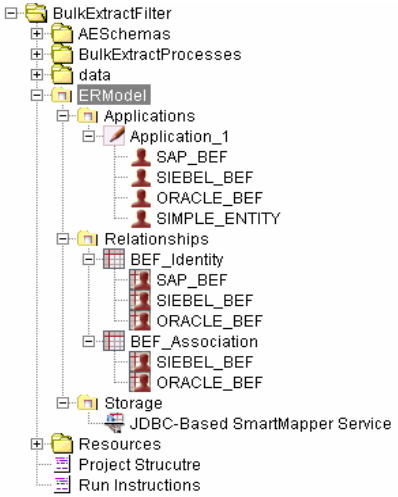
- Participant names
- Timestamp based criteria
- Like, And, and Or operations

ER Model Overview

An ER Model must be created before creating a SmartMapper process. See Creating an ER Model on page 18 for details about creating an ER Model.

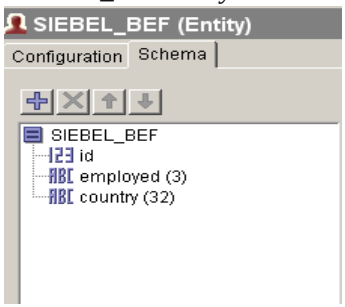
The ER Model used in this project contains one application and two relationships. As shown in Figure 89, the JDBC-Based SmartMapper service is used to store the relationship data.

Figure 89 BulkExtractFilter Project: ER Model



Four entities are added in the application and each entity has its own data schema associated with. Select one entity in the application node and click the **Schema** tab in the Configuration panel to view the defined schema.

Figure 90 Data Schema of the Siebel_BEF Entity



Two relationships are defined in this ER Model, one is an identity relationship and the other one is an association relationship. See Relationships in TIBCO ActiveMatrix BusinessWorks SmartMapper on page 215 for more details about the relationship types.

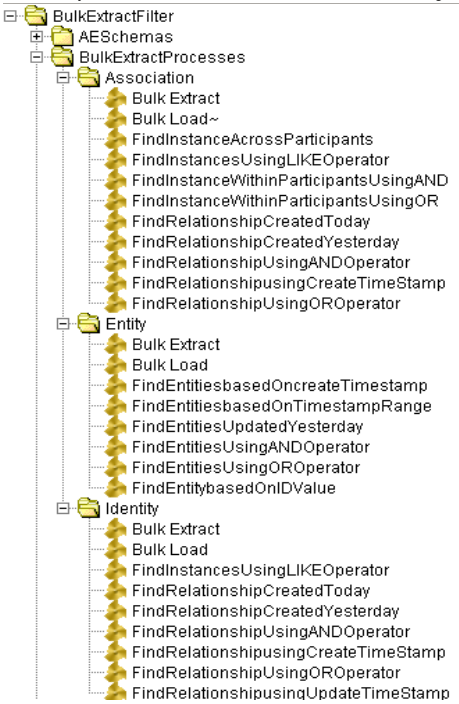
Process Definition

Three process folders are included in the BulkExtractFilter project: Association, Entity, and Identity. The processes in the Association and Identity folders are designed to look up association relationship data or identity relationship data according to different filter criteria; the processes in the Entity folder are designed to look up entities according to different filter criteria.

In general, the process name in each process folder indicates the filter criterion used. As shown in Figure 91, every process folder has the Bulk Extract and the Bulk Load processes defined:

- The Bulk Extract process extracts all the relationship data or the entity data.
- The Bulk Load process loads the specified sample data to the SmartMapper database.

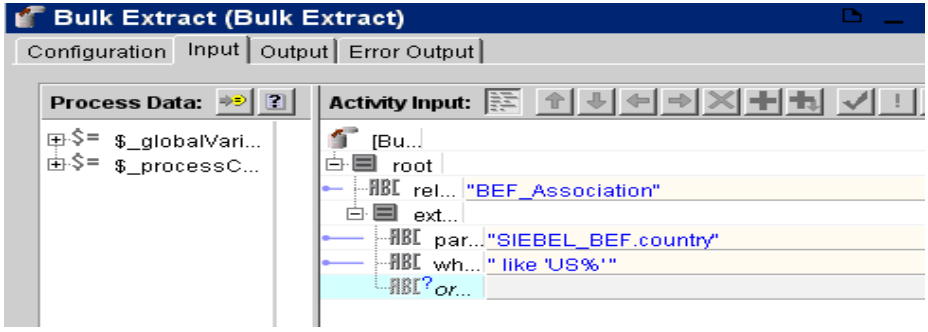
Figure 91 Processes Defined in the BulkExtractFilter Project



Take the FindInstancesUsingLIKEOperator process in the Association folder as an example. This process contains the following activities:

- The Bulk Extract activity is used to extract the relationship instances of the BEF_Association relationship. As shown in Figure 92, all the relationship instances whose participant names start with US will be extracted.

Figure 92 The Input Tab of the Bulk Extract Activity



- The Inspector activity is used to write the output of the Bulk Extract activity.

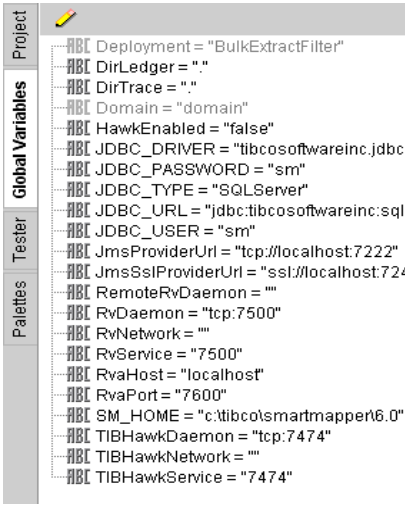
- The Log activity is used to write the output messages of the Inspector activity to a log file.

Setting Up the Project

Before running the project, follow these steps to configure the project:

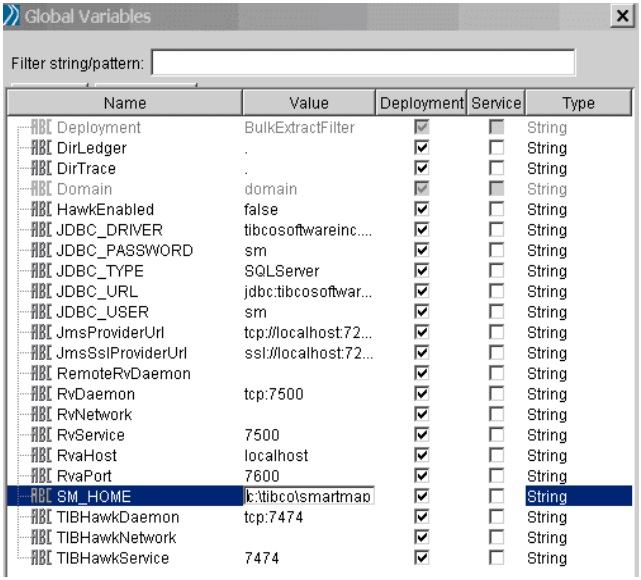
1. Start TIBCO Designer.
2. Click the **Open Existing Project** button in the TIBCO Designer startup panel. The Open Project dialog is displayed.
3. Click the **Browse** button next to the Project Directory field, and then select the BulkExtractFilter folder, which is located in the *SmartMapper_HOME*\examples directory. Click the **OK** button. The project is displayed.
4. Click the **Global Variables** tab in the Project panel and select the **SmartMapper.demo.directory** item.

Figure 93 BulkExtract Example: Global Variables Panel



5. Click the button at the top. The Global Variables dialog is displayed.

Figure 94 BulkExtract Example: Global Variables Dialog



6. Click the **SmartMapper.demo.directory** variable and double-click its value in the Value column.
7. Provide the absolute path to the BulkExtractFilter example. By default, the example locates in the *SmartMapper_HOME*\examples\BulkExtractFilter directory.
8. Click the **OK** button. The Project panel is displayed.
9. Expand **BulkExtractFilter > ERModel > Storage** and click the **JDBC-Based SmartMapper Service** in the Project panel.
10. Click the **JDBC Settings** tab in the Configuration panel and configure the database connection parameters. See JDBC-Based SmartMapper Service on page 88 for more details about the configurations.
11. Click the **Test Connection** button to validate the connection.
12. Click the **Synch ER Model w/DB** button. The metadata tables are synchronized to the specified database.

Running the Project

After setting up the project, complete the following steps to trigger the FindInstancesUsingLIKEOperator process:

1. In the Project panel, click the **Tester** tab and then click the button at the top left corner in this panel. The Select Processes To Load dialog is displayed.

2. Expand the **Association** folder, and then check the **Bulk Load** and the **FindInstancesUsingLIKEOperator** processes.

   The Bulk Load process seeds the relationship data to the specified database.

3. Click the **Load Selected** button. The process is started.

   When the process executes, the elements of the process change colors depending upon what is occurring in the executing process instance. If all the transition lines change to green, it means the process runs successfully.

4. Click the  button to return to the Design mode.

You can set breakpoints in the process definition at a specified point where you want to stop a running process and examine its state and process data, for example, you can set breakpoints before or after an activity.

See *TIBCO BusinessWorks Process Design Guide* for detailed information on using the test mode including setting breakpoints and the element colors in the test mode.

Expected Results

If the processes perform successfully, the green color is displayed between each activity. All the relationship instances of the BEF_Association relationship whose participant names start with US are extracted.

Appendix A Adding Your Own GUID Generation Class

This appendix describes an example of adding a GUID Generation class. This is an example of what you might choose to do, if it is appropriate to your system. This example is not meant to be implemented exactly as written.

Topics

GUID Class Overview

If appropriate to your system, you can add your own GUID generation class instead of using the class provided in the SDK. The class is per "ER Model" but it takes the entity name as a parameter. This appendix gives an example of such a class as a guideline.

For example you might build a SmartMapperSmartMapper ID Generator that generates ID's by incrementing an Oracle sequence.

For example, to install the ID generator, specify OracleSequenceBasedIdGenerator as the ID Generator Class for an ER Model (in Designer under the Advanced Tab of the ER Model configuration). Also, make sure this class will be found in the classpath.

A user cannot use the embedded Merant JDBC Drivers in TIBCO ActiveMatrix BusinessWorks for the use in their database-based ID Generation classes. This driver is locked for internal use only.

Users should include the classes of their JDBC drivers in their TIBCO ActiveMatrix BusinessWorks, TIBCO Runtime Agent and TIBCO Administrator classpath.

Class Overview

Class OracleSequenceBasedIdGenerator

public class OracleSequenceBasedIdGenerator

A simple implementation of a SmartMapper ID Generator that is backed by an Oracle sequence. (This is an example only).

Table 78 GUID Class Overview: Constructor and Method Overview

OracleSequenceBasedIdGenerator ()	Constructor
generateId(java.lang.String entityName)	Method

Table 79 GUID Class Overview: Constructor Detail

Name	Description
OracleSequenceBasedIdGenerator	public OracleSequenceBasedIdGenerator() Default constructor (required). Initializes a connection to the database

Table 80 GUID Class Overview: Method Detail

Name	Description
generateId	public java.lang.String generateId(java.lang.String entityName) throws com.tibco.solution.xref.XRefException Generates an ID by incrementing a sequence • **Parameters** entityName - The name of the SmartMapper Entity that is requesting a new ID • **Returns** The generated ID • **Throws** XRefException

Sample Code For GUID Generation

This code is example only.

```
import com.tibco.solution.xref.autoid.DefaultAutoIdGenerator;
import com.tibco.solution.xref.XRefException;
import java.sql.*;

/**
 * A simple implementation of a SmartMapper ID Generator that
 * is backed by an Oracle sequence.
 */
public class OracleSequenceBasedIdGenerator extends DefaultAutoIdGenerator {
  private Connection connection;

  /**
   * Default constructor (required). Initializes a connection to the database
   */
  public OracleSequenceBasedIdGenerator() {
    try {
      Class.forName("oracle.jdbc.OracleDriver");
      connection = DriverManager.getConnection(
        "jdbc:oracle:thin:@myhost:1521:mySID" );
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }

  /**
   * Generates an ID by incrementing a sequence
   *
   * @param entityName The name of the SmartMapper Entity that is requesting a
   *                   new ID
   * @return The generated ID
   * @exception XRefException
   */
  public String generateId(String entityName) throws XRefException {
    try {
      //select the next sequence value from the sequence "myseq"
      Statement statement = connection.createStatement();
      ResultSet rs = statement.executeQuery("select myseq.nextval from dual");
      rs.next();
      long nextVal = rs.getLong(1);
      return Long.toString(nextVal);
    }
    catch (Exception e) {
      throw new XRefException(e);
    }
  }
}
```

Appendix B Adapter Service Configuration

This appendix is for advanced users only.

Topics

Modifying the Subscriber Service

When you bring a SmartMapper Adapter Configuration Resource to the design panel, TIBCO Designer adds a Subscriber service to the project tree. This service is added under the Adapter Services folder. The SmartMapper Enterprise Server uses this service to listen to the requests from the client process that uses SmartMapper activities.

To display the Subscriber service:

1. Select the SmartMapper Adapter Configuration Resource in the tree.
2. Expand **Adapter Services**.
3. Select **Subscriber**.

Configuration Tab

Only the Description field is available for input.

Transport Tab

Table 81 Modifying the Subscriber Service: Transport Tab

Field	Description
Message Subject	Contains the subject on which the server listens. You can edit this field to change the subject.
Quality of Service	Reliable. This chooses the Reliable quality of service. This choice determines what is displayed in the Advanced folder below the SmartMapper Adapter Services folder. See Modifying the Reliable Quality of Service on page 197. Distributed Queue. This chooses the Distributed Queue quality of service. This choice changes what is displayed in the Advanced folder below the SmartMapper Adapter Services folder. See Modifying the Distributed Queue Quality of Service on page 198.
Session Reference	Every adapter can have one or more sessions configured for it. Sessions encapsulate stateful connections to TIBCO Rendezvous and other messaging sources. The session object shown in this field is initially supplied by the adapter, depending on the Quality of Service selected. You can change the session by browsing for it in the project panel.
Endpoint Reference	You can drag a different endpoint, browse for another endpoint resource, go to the referenced endpoint to edit its properties or delete the endpoint. Endpoint reference objects are explained in the *TIBCO Designer Adapter Resource Management Guide*.

Modifying the Reliable Quality of Service

Before proceeding, note that the Subscriber Quality of Service must be set to **Reliable**. Check the Quality of Service setting by selecting the Subscriber service and clicking the Transport tab.

1. Select the SmartMapper Adapter Configuration Resource in the tree.
2. Expand **Advanced**.
3. Expand **Sessions**.
4. Expand **DefaultRVSession**.
5. Select **DefaultRVSession**.

Table 82 Modifying Reliable QOS: DefaultRVSession

Field	Description
Service	(default value = null) The Rendezvous daemon divides the network into logical partitions. Each TibrvRvdTransport communicates on a single service. A transport can communicate only with other transports on the same service. To communicate on more than one service, a program must create more than one transport. It must create one transport for each service.
Network	(default value = null) Every network transport communicates with other transports over a single network interface. On computers with more than one network interface, the network parameter instructs the Rendezvous daemon to use a particular network for all outbound messages from this transport.
Daemon	The daemon parameter instructs the transport object about how and where to find the Rendezvous daemon and establish communication.
SSL	Check this box to enable SSL.

For further information on quality of service and SSL, see TIBCO Rendezvous documentation.

Modifying the Distributed Queue Quality of Service

Before proceeding, note that the Subscriber Quality of Service must be set to **Distributed Queue**. Check the Quality of Service setting by selecting the Subscriber service and clicking the Transport tab.

1. Select the SmartMapper Adapter Configuration Resource in the tree.
2. Expand **Advanced**.
3. Expand **Sessions**.
4. Expand **DefaultRVCMQSession.**
5. Select **DefaultRVCMQSession**.

Table 83 Modifying Distributed Queue QOS: DefaultRVCMQSession

Field	Description
Service	(default value = null) The Rendezvous daemon divides the network into logical partitions. Each TibrvRvdTransport communicates on a single service. A transport can communicate only with other transports on the same service. To communicate on more than one service, a program must create more than one transport. It must create one transport for each service.
Network	(default value = null) Every network transport communicates with other transports over a single network interface. On computers with more than one network interface, the network parameter instructs the Rendezvous daemon to use a particular network for all outbound messages from this transport.
Daemon	The daemon parameter instructs the transport object how and where to find the Rendezvous daemon and establish communication.
SSL	Check this box to use SSL
CMQ Name	The name of the distributed queue. This name is in the same format as TIBCO Rendezvous subject names. Worker Weight. The weight of the worker (this pertains to the worker processing queue requests, not to BusinessWorks process engines). Relative worker weights assist the scheduler in assigning tasks. When the scheduler receives a task, it assigns the task to the available worker with the greatest worker weight.

Field	Description
Complete time	The complete time parameter of the scheduler affects the reassignment of tasks: If the complete time is non-zero, the scheduler waits for a worker to complete an assigned task. If the complete time elapses before the scheduler receives completion from the worker, the scheduler reassigns the task to another worker. Zero is a special value, which specifies no limit on the completion time—that is, the scheduler does not set a timer and does not reassign tasks when task completion is lacking. All members implicitly begin with a default complete time, which is zero; programs can change this parameter. Complete time is in milliseconds.
Schedule Heartbeat	The scheduler sends heartbeat messages at this interval (in milliseconds). All members with the same name must specify the same value for this parameter. The value must be strictly positive.
Schedule Weight	Weight represents the ability of this member to fulfill the role of scheduler, relative to other members with the same name. Cooperating distributed queue transports use relative scheduler weight values to elect one transport as the scheduler; members with higher scheduler weight take precedence. Acceptable values range from 1 to 65535.
Schedule Activation	When the heartbeat signal from the scheduler has been silent for this interval (in milliseconds), the member with the greatest scheduler weight takes its place as the new scheduler. All members with the same name must specify the same value for this parameter. The value must be positive.
Worker Weight	Relative worker weights assist the scheduler in assigning tasks. When the scheduler receives a task, it assigns the task to the available worker with the greatest worker weight. The default worker weight is 1. The scheduler applies a round-robin ordering to distribute tasks among several workers equivalent with equal weight.
Worker Tasks	Sets the task capacity for the worker (this pertains to the worker processing queue requests, not to BusinessWorks process engines). Task capacity is the maximum number of tasks that a worker can accept. When the number of accepted tasks reaches this maximum, the worker cannot accept additional tasks until it completes one or more of them.

For further information on quality of service, see TIBCO Rendezvous documentation.

Setting Global Variables

By default a number of global variables are defined for any TIBCO Designer project. These global variables configure the values for parameters such as session, subject, and so on. In most cases, it is recommended that developers keep and make use of these default configuration variables since they provide a flexible way of changing the adapter configuration at run-time without having to load, reconfigure and save the adapter configuration instance every time.

However, depending on your requirements, it may not be desirable to change an adapter configuration at run-time. In this case, an adapter can define fixed values for these parameters.

See the *TIBCO Designer User's Guide* for information about modifying, adding and deleting global variables.

Appendix C Error Codes

This appendix explains the trace messages used by TIBCO ActiveMatrix BusinessWorks SmartMapper.

Topics

Overview of Trace Messages

Trace messages provide information about plug-in activities. The messages are logged to a log file and the console where the runtime plug-in is started.

Introduction of Trace Message Roles

The roles of trace messages in TIBCO ActiveMatrix BusinessWorks SmartMapper are shown in Table 84.

Table 84 Trace Message Roles

Role Name	Description
Info	Indicates normal plug-in operation. No action is necessary. A tracing message tagged with Info. indicates that a significant processing step is reached and logged for tracking or auditing purposes. Only info. messages preceding a tracking identifier are considered significant steps.
Warn	An abnormal condition is found. Processing will continue, but special attention from an administrator is recommended.
Error	An unrecoverable error occurs. Depending on the error severity, the adapter may continue with the next operation or may stop altogether.
Debug	A developer-defined tracing message. In normal operating conditions, debug messages should not display.

Error Codes

BW-SMARTMAPPER-100000 Participant is not specified.

Role: errorRole

Category: BW_Plug-in

Description: No participant is selected.

Resolution: Please select a participant to use.

BW-SMARTMAPPER-100001 Entity is not specified.

Role: errorRole

Category: BW_Plug-in

Description: No entity is selected.

Resolution: Please select an entity to use.

BW-SMARTMAPPER-100002 Relationship is not specified.

Role: errorRole

Category: BW_Plug-in

Description: No relationship is selected.

Resolution: Please select a relationship to use.

BW-SMARTMAPPER-100003 Input participant is not specified.

Role: errorRole

Category: BW_Plug-in

Description: No participant is specified in the Input Participant field.

Resolution: Please specify a participant to use in the Input Participant field.

BW-SMARTMAPPER-100004 Output participant is not specified.

Role: errorRole

Category: BW_Plug-in

Description: No participant is specified in the Output Participant field.

Resolution: Please specify a participant to use in the Output Participant field.

BW-SMARTMAPPER-100010 Could not create the SmartMapper service for: %1.

Role: errorRole

Category: BW_Plug-in

Description: Cannot create the SmartMapper storage service.

Resolution: Please check the configurations of the storage service.

BW-SMARTMAPPER-100011 Could not initialize Participant, %1. Perhaps the ER Model has not been synchronized with the data store.

Role: errorRole

Category: BW_Plug-in

Description: Cannot initialize the specified participant.

Resolution: Please check whether the ER Model has been synchronized.

BW-SMARTMAPPER-100012 Could not initialize Relationship, %1. Perhaps the ER Model has not been synchronized with the data store.

Role: errorRole

Category: BW_Plug-in

Description: Cannot initialize the specified relationship.

Resolution: Please check whether the ER Model has been synchronized.

BW-SMARTMAPPER-100013 This SmartMapper Wizard operation is not fully configured: %1.

Role: errorRole

Category: BW_Plug-in

Description: An error occurred in the SmartMapper Wizard activity when running the process.

Resolution: Please check the configurations of the SmartMapper Wizard activity.

BW-SMARTMAPPER-100014 Wizard Operation Exception: %1.

Role: errorRole

Category: BW_Plug-in

Description: An error occurred in the SmartMapper Wizard activity when running the process.

Resolution: Please check the configurations of the SmartMapper Wizard activity.

BW-SMARTMAPPER-100015 Wizard Exception: %1.

Role: errorRole

Category: BW_Plug-in

Description: An error occurred in the SmartMapper Wizard activity when running the process.

Resolution: Please check the configurations of the SmartMapper Wizard activity.

BW-SMARTMAPPER-100016 SmartMapper Exception: %1.

Role: errorRole

Category: BW_Plug-in

Description: An error occurred when running the process.

Resolution: Please check the configurations.

BW-SMARTMAPPER-100017 Entity key mismatch: %1 Perhaps the ER Model has not been synchronized.

Role: errorRole

Category: BW_Plug-in

Description: No value is specified for the required fields.

Resolution: Please check whether the ER Model has been synchronized.

BW-SMARTMAPPER-100018: Invalid input data.

Role: errorRole

Category: BW_Plug-in

Description: The input data is invalid.

Resolution: Please check your input configurations.

BW-SMARTMAPPER-100019 Unsupported operation: Entity [%1] has non-keys. Non key attributes are not supported for File based service.

Role: errorRole

Category: BW_Plug-in

Description: The entity has no key field specified. The file-based service does not support non-key attributes.

Resolution: Please change a SmartMapper storage service.

BW-SMARTMAPPER-100020 No active service configured.

Role: errorRole

Category: BW_Plug-in

Description: No SmartMapper storage service is active.

Resolution: Please activate a SmartMapper storage service to store the cross-referencing data.

BW-SMARTMAPPER-100021 Participant, %1 not found, please make sure participant name matches with the one defined in the ER Model.

Role: errorRole

Category: BW_Plug-in

Description: Cannot find the specified participant.

Resolution: Please check the name of the participant. Ensure that the name is the same with the one defined in the ER Model.

BW-SMARTMAPPER-100022 Relationship, %1 not found, please make sure relationship name matches with the one defined in the ER Model.

Role: errorRole

Category: BW_Plug-in

Description: Cannot find the specified relationship.

Resolution: Please check the name of the relationship. Ensure that the name is the same with the one defined in the ER Model.

BW-SMARTMAPPER-100023 Invalid subsetsize value : %1, value must be greater than 0. Invalid subsetsize value : %1, value must be greater than 0.

Role: errorRole

Category: BW_Plug-in

Description: The value for the subset is invalid.

Resolution: Please enter an integer greater than 0.

BW-SMARTMAPPER-100025 %1 format cannot be used with %2 relationship type.

Role: errorRole

Category: BW_Plug-in

Description: The data format does not match with the relationship type.

Resolution: Please check your configurations. If the relationship has one participant, the identity relationship should be selected; if the relationship has only two participants, the association relationship should be selected.

BW-SMARTMAPPER-100026 Participant, %1 not found for relationship %2 please make sure participant name matches with the one defined in the ER Model.

Role: errorRole

Category: BW_Plug-in

Description: Cannot find the relationship with the specified participant.

Resolution: Please check the name of the participant. Ensure that the name is the same as the one defined in the ER Model.

BW-SMARTMAPPER-100027 ER Model cannot be empty.

Role: errorRole

Category: BW_Plug-in

Description: The ER Model cannot be empty.

Resolution: Please check your configuration.

BW-SMARTMAPPER-100028 Unsupported operation: File based service is not supported for this activity.

Role: errorRole

Category: BW_Plug-in

Description: The file-based service does not support the current operation.

Resolution: Please change a SmartMapper storage service.

BW-SMARTMAPPER-100029 Schema is not defined.

Role: errorRole

Category: BW_Plug-in

Description: No schema is defined for the specified entity.

Resolution: Please define a schema for the entity.

BW-SMARTMAPPER-100030 Schema must contain at least one key attribute.

Role: errorRole

Category: BW_Plug-in

Description: No key field is specified.

Resolution: Please add a key field.

BW-SMARTMAPPER-100031 Attribute index value, %1, is out of valid indexes range (1-%2) for this participant.

Role: errorRole

Category: BW_Plug-in

Description: The entered value is invalid.

Resolution: Please enter a valid value.

BW-SMARTMAPPER-100032 Relationship instance creation requires at least two participants, only %1 is provided in the input.

Role: errorRole

Category: BW_Plug-in

Description: Only one participant is specified when creating the relationship instance.

Resolution: Please check your input configuration.

BW-SMARTMAPPER-100033 At least one participant, provided in the input, needs to be of cardinality one.

Role: errorRole

Category: BW_Plug-in

Description: No cardinality participant is specified.

Resolution: Please specify a cardinality participant.

BW-SMARTMAPPER-1000034 Following required fields (name/index, value) are missing from input: %1.

Role: errorRole

Category: BW_Plug-in

Description: No value is specified for the required fields in the Input tab.

Resolution: Please check your input configurations.

BW-SMARTMAPPER-1000035 ER Model is not specified.

Role: errorRole

Category: BW_Plug-in

Description: No ER Model is specified.

Resolution: Please specify an ER Model to use.

BW-SMARTMAPPER-1000036 Unsupported operation: Rendezvous Distributed Queue is not supported for this activity.

Role: errorRole

Category: BW_Plug-in

Description: Rendezvous Distributed Queue is not supported for the current operation.

Resolution: Please check your configuration.

BW-SMARTMAPPER-1000037 Connection Pool size specified [%1] is not an integer. Using the default value of 10.

Role: warnRole

Category: BW_Plug-in

Description: The specified connection pool size is invalid and use the default value 10.

Resolution: No action required.

BW-SMARTMAPPER-1000038 Skipping duplicate entry.

Role: warnRole

Category: BW_Plug-in

Description: The duplicate entity is ignored.

Resolution: No action required.

BW-SMARTMAPPER-1000039 One and only one participant can be flagged as lookup participant, currently [%1] participants are flagged, please check lookupParticipant flag value in input data.

Role: errorRole

Category: BW_Plug-in

Description: Only one participant is needed.

Resolution: Please check your configurations.

BW-SMARTMAPPER-1000040 Attempt to update one cardinality participant [%1], with many cardinality input; please check the value of unlinkParticipant flag in the input .

Role: errorRole

Category: BW_Plug-in

Description: The specified value is invalid in the Input field.

Resolution: Please check your configurations.

BW-SMARTMAPPER-1000041 ER Model [%1] not found, please provide a valid value for ER Model.

Role: errorRole

Category: BW_Plug-in

Description: Cannot find the specified ER Model.

Resolution: Please check your configuration.

BW-SMARTMAPPER-1000042 The schema [%1] is not declared in Entity definition.

Role: errorRole

Category: BW_Plug-in

Description: The input value is invalid.

Resolution: Please enter a valid value.

BW-SMARTMAPPER-1000043 No value is provided for participant [%1].

Role: errorRole

Category: BW_Plug-in

Description: No value is provided for the participant.

Resolution: Please enter a value.

BW-SMARTMAPPER-1000044 At least one field should be provided for participant [%1].

Role: errorRole

Category: BW_Plug-in

Description: No field is provided for the participant.

Resolution: Please specify a field for the participant.

BW-SMARTMAPPER-1000045 Incorrect use of flags in input data : [%1].

Role: errorRole

Category: BW_Plug-in

Description: The input value is invalid.

Resolution: Please enter a valid value.

BW-SMARTMAPPER-1000046 At least one field should be provided for entity [%1].

Role: errorRole

Category: BW_Plug-in

Description: No field is provided for the entity.

Resolution: Please specify a field for the participant.

BW-SMARTMAPPER-100047 Could not initialize Entity, %1. Perhaps the ER Model has not been synchronized with the data store.

Role: errorRole

Category: BW_Plug-in

Description: Cannot initialize the specified entity.

Resolution: Please check whether the ER Model has been synchronized.

BW-SMARTMAPPER-101999 Unexpected error occurs %1.

Role: errorRole

Category: BW_Plug-in

Description: An error occurred when initializing the process.

Resolution: Please check your configurations.

BW-ACTIVESPACE-100003 Could not connect to the Metaspace.[%1]

Role: errorRole

Category: BW_Plug-in

Description: Cannot connect to the specified metaspace.

Resolution: Please check your configurations.

Appendix D Relationships in TIBCO ActiveMatrix BusinessWorks SmartMapper

This appendix explains in detail the relationship concepts used in TIBCO ActiveMatrix BusinessWorks SmartMapper.

Topics

Identity Relationship

The identity relationship is used to keep different applications' corresponding keys. For example, after a company merger involving several applications, you are given the task to create a cross-reference of keys for three data file dumps, one for SAP, one for Siebel, and one for Oracle. A particular customer may be in one or more of the applications. Also, because of the imperfect nature of the system design, some customers may have more than one record in the application, although the majority would have just one.

An identity relationship should be used in each of the following cases.

In the simplest case described in Table 85, a customer record appears in each application *once and only once*. The table is as follows (the relationship cardinality is 1-1-1).

Table 85 Identity Relationship: Customer Record Appears Once

SAP_Customer	Siebel_Account	Oracle_Customer	
A	1	1243	J
B	2	1222	K
C	3	1433	L
D	4	4123	M

In the case described in Table 86, a customer record is not in all the applications, and if the record is in one application, it would be found *only once*. The table is as follows (the relationship cardinality is 1-1-1).

Table 86 Identity Relationship: Customer Record Not in the Application

SAP_Customer	Siebel_Account	Oracle_Customer	
NULL	1	1243	J
B	NULL	1222	K
C	3	NULL	NULL
D	4	4123	M

In the case described in Table 87, one customer appears *zero or more times* in the applications. This is a practical reality when dealing with such data files. The table is as follows (the relationship cardinality is M-M-M).

Table 87 Identity Relationship: Customer Record Appears Zero or More Times

SAP_Customer	Siebel_Account	Oracle_Customer	
NULL	1	1243	J
B	NULL	1222	K
		1122	M
C	3	NULL	NULL
F	5		
D	4	4123	M
E			

Common Participant Added

In the case described in Table 88, a common participant is added to simplify application usage. This is the typical arrangement of the master data reference system, where there is a single master customer table that is referenced by all users (the relationship cardinality is 1-M-M-M).

Table 88 Identity Relationship: Common Participant Added

Common_Customer	SAP_Customer	Siebel_Account	Oracle_Customer	
C001	NULL	1	1243	J
C002	B	NULL	1222	K
			1122	M
C003	C	3	NULL	NULL
	F	5		
C004	D	4	4123	M
	E			

Association Relationship

This kind of relationship can have *two and only two* participants. They can be used to model relationship of two typically orthogonal entities, like teacher and class; or a relationship of the same entity, like a part-subpart containment relationship.

Table 89 Association Relationship

Teacher	Class (grade, class number)	
Jody	2	1
Jody	2	2
Dover	2	1
Dover	1	1
McDonald	1	2

This kind of relationship is very different from the identity relationship type. An identity relationship is for participants of same kind, such as customer, product, order, etc. An association relationship is used to model a relationship between an order and a product, rather than different order IDs of different systems.

An association relationship can also be used for one-to-many relationships where the many-side can grow very large.

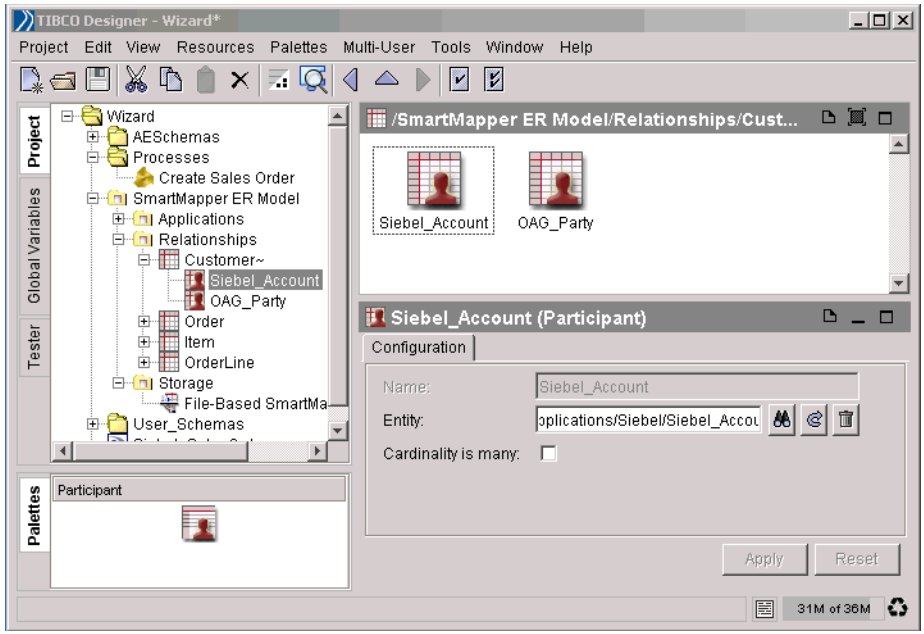
Participant

A participant is the role that an entity plays in a relationship. For the purposes of this software, the role that an entity plays is that of itself.

A relationship may require a collection of entities to be a peer with the other entities, so there is a Cardinality is many check box in the Participant resource. Note that Cardinality is many check box is only available for the identity relationship, not for the association relationship.

For example, the Customer relationship may contain an entity called Siebel_Account. In this case, Siebel_Account is a participant in the Customer relationship.

Figure 95 Participant Defined in the ER Model



Index

A

B

C

D

E

F

G

I

J

L

M

O

P

R

S

T

U