TIBCO iProcess® Modeler

Advanced Design

Software Release 11.4
July 2013

Two-Second Advantage®

TIBCO®

Important Information

Contents

Preface

This guide explains and demonstrates how to add more advanced functionality to your iProcess procedures using features such as sub-procedures, dynamic sub-procedure calls, case prediction, and scripts.

This guide assumes that all users have attended a TIBCO training course before starting to define procedures and design forms and this is strongly recommended. It is aimed at the following types of iProcess user:

- iProcess managers/supervisors.
- Business analysts and IT specialists.
- IT consultants specializing in business process management/workflow and the iProcess Suite.

Topics

Related Documentation

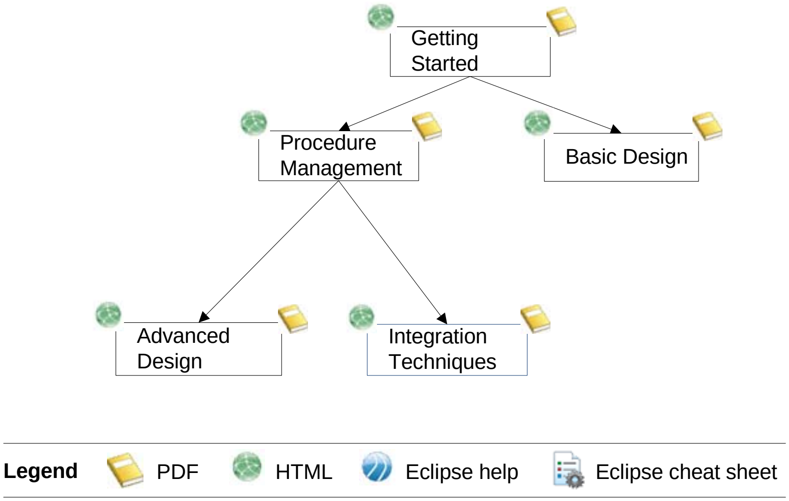
This section lists documentation resources you may find useful.

TIBCO iProcess Modeler Documentation

The following documents form the TIBCO iProcess Modeler and TIBCO iProcess® Workspace (Windows) documentation set, which are supplied with the TIBCO iProcess Workspace (Windows) software:

- *TIBCO iProcess Workspace (Windows) Installation* Read this manual for instructions on site preparation and installation.
- *TIBCO iProcess Workspace (Windows) 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.
- TIBCO iProcess Suite Documentation This documentation set contains all the manuals for TIBCO iProcess Modeler, TIBCO iProcess® Workspace (Windows), and other TIBCO products in TIBCO iProcess® Suite. The manuals for TIBCO iProcess Modeler and TIBCO iProcess Workspace (Windows) are as follows:
  - — *TIBCO iProcess Workspace (Windows) User's Guide*
  - — *TIBCO iProcess Modeler Getting Started*
  - — *TIBCO iProcess Modeler Procedure Management*
  - — *TIBCO iProcess Modeler Basic Design*
  - — *TIBCO iProcess Modeler Advanced Design*
  - — *TIBCO iProcess Modeler Integration Techniques*
  - — *TIBCO iProcess Expressions and Functions Reference Guide*
  - — *TIBCO iProcess Workspace (Windows) Manager's Guide*

If you are new to iProcess procedure development, you are advised to follow the reading path shown next. The documentation road map shows the relationships between the books and online references in this product's documentation set.



Other TIBCO Product Documentation

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

- TIBCO ActiveMatrix BusinessWorks™
- TIBCO Business Studio™
- TIBCO Enterprise Message Service™
- TIBCO Hawk®
- TIBCO Rendezvous®

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
SWDIR	TIBCO iProcess Engine installs into a directory. This directory is referenced in documentation as *SWDIR*. The value of *SWDIR* depends on the operating system. For example, • on a Windows server (on the `C:` drive) if *SWDIR* is set to the `C:\swerver\staffw_nod1` directory, then the full path to the `swutil` command is in the `C:\swerver\staffw_nod1\bin\swutil` directory. • on a UNIX or Linux server if *SWDIR* is set to the `/swerver/staffw_nod1` directory, then the full path to the `swutil` command is in the `/swerver/staffw_nod1/bin/swutil` directory or the `$`*SWDIR*`/bin/swutil` directory. **Note**: On a UNIX or Linux system, the environment variable $*SWDIR* should be set to point to the iProcess system directory for the *root* and *swadmin* users.
`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 **`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 (Cont'd)

Convention	Use
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix 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 param1 | param2 | param3`

Table 2 Syntax Typographical Conventions (Cont'd)

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 the pair `param1` and `param2`, or the pair `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. It is 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 TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, 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 Defining and Using Sub-Procedures

This chapter describes the use of sub-procedures and how to create them. You should read this chapter before working with sub-procedures, sub-procedure parameter templates, array fields and multiple sub-procedures.

Topics

What are Sub-Procedures

A sub-procedure is a type of iProcess procedure that can be called from a step in one or more other procedures. When you design a procedure to fit your business process, you can think of it as the main procedure and then you can call any number of sub-procedures at any point along your main procedure. Sub-procedures can receive case data from the main procedure and can return case data to it.

A sub-procedure is designed in the same way as an iProcess main procedure, but you select **Sub-procedure** in the **New Procedure** dialog instead of **Main Procedure**.

See Defining a Sub-Procedure on page 9 for information about defining a sub-procedure.

You can also create a sub-procedure that is based on a sub-procedure parameter template. This means that the input and output parameters for the sub-procedure are inherited from the template. See Defining Sub-Procedure Parameter Templates on page 49 for more information about templates. All sub-procedures that are to be called from a dynamic sub-procedure call *must* be based on the same template.

Why Use Sub-Procedures

The following are some example reasons why you might want to use sub-procedures as part of your procedure design:

- Your procedure is becoming too large to manage.

  Any procedure over 50-100 steps can be difficult to manage. In this case, it can be more efficient to break up the procedure and create sub-procedures to split it up into more manageable chunks.

- You have procedures that carry out identical tasks.

  When you create a number of iProcess procedures for your business, it is often the case that several procedures have sections that carry out identical tasks. To prevent duplication of effort in both writing and maintaining your procedures, you can use sub-procedures that can be called any number of times. Typical examples of common business processes are:

  — getting approval for payment, and updating an external accounting system.

  — common back office processing behind a variety of front-office procedures.

- You need multiple people to work on a procedure.

  If you have a number of procedure developers who need to work on parts of the business process independently, you can split up the procedure into sub-procedures so that each person can work on their allocated procedures.

- You need to call a dynamic number of sub-procedures when the case is run depending on what data is entered during a case of a procedure.

  If you define a dynamic call to your sub-procedures, it is only at run-time when iProcess determines which sub-procedures need to be run. For example, if a case is started and the first few steps collate details about a customer’s order, iProcess can determine which sub-procedures to run depending on what case data has been entered. Therefore, if a stock check and an account credit check are required, two sub-procedures can be run to perform these extra processes.

- You are using an external application that controls what processes need to be run. Sub-procedures can be started by the application as required (via TIBCO iProcess Objects or SAL application calls) and attached to the main procedure via a graft step.

  For example, if you have a financial application that determines what procedures are to be run based upon case data gathered by an iProcess procedure, it can start the required sub-procedures (such as DEBIT, CREDIT and AUDIT). These are attached to the main procedure via a graft step. See "Using Graft Steps" in *TIBCO iProcess Modeler Integration Techniques* for more information about graft steps.

Calling Sub-Procedures from the Main Procedure

After creating your sub-procedure, you need to define a call from the main procedure to run it.

> Sub-procedures can only be started from within another procedure. They cannot be started from the **Case Start** dialog.

You can define a step in your main procedure to call:

- a static sub-procedure

  Use the static sub-procedure step tool when you know at procedure definition time the sub-procedure you want to run. You may have a specific approval process that you have created as a sub-procedure, which can be called from many of your procedures. See Defining a Static Call to a Sub-Procedure on page 23.

- a dynamic number of sub-procedures

  Use the dynamic sub-procedure call step tool when you have multiple sub-procedures that can be called in your procedure. The procedure can be designed so that from the case data entered, an array field can be populated with the required sub-procedures to run. See Using Array Fields on page 39, Defining Sub-Procedure Parameter Templates on page 49 and then Calling a Dynamic Number of Sub-Procedures on page 57.

In each call, the sub-procedure(s) perform a number of steps before returning to the main procedure.

Nesting Sub-Procedures

You can nest sub-procedures so that one sub-procedure calls another sub-procedure. The default value for the maximum depth of sub-procedures is 100 but you can change this value within the *SWDIR***etc****staffcfg** file. See “Tuning iProcess Engine using SWDIR\etc\staffcfg Parameters” in *TIBCO iProcess Engine Administrator’s Guide* for more information.

How Do Sub-Procedures Work

At run-time, when a case in the main procedure reaches the sub-procedure call step, the following sequence of events occurs:

1. iProcess starts a new case of the sub-procedure(s) and links it to the main procedure's case.
2. The case data in the main procedure is transferred to the sub-case(s), which progress according to the rules defined in the sub-procedure. The transfer of case data is achieved by mapping fields in the main procedure to fields or parameters in the sub-procedure.

> Sub-procedure parameters can be defined to restrict the list of sub-procedure fields that can be mapped to main procedure fields - see Why Define Sub-Procedure Parameters? on page 16.

3. When the sub-case(s) reaches its conclusion, the case data is transferred back to the main case via the sub-procedure field/parameter mapping.
4. When the sub-case(s) has completed, the actions of the step that called the sub-procedure(s) are executed and the main case proceeds to the next step(s).

> To run a Transaction Control step as the first step of the sub-procedure before processing the predefined first step of the sub-procedure, set the value of the `SUBCASE_START_AUTOCOMMIT` process attribute to `1` or `2` by using the `swadm` server configuration utility. For more information about this attribute, see "SUBCASE_START_AUTOCOMMIT" in *TIBCO iProcess Engine Administrator's Guide*.

The audit trail for the sub-case(s) can be seen within the audit trail for the main case. See "Viewing an Audit Trail for a Selected Case" in *TIBCO iProcess Workspace (Windows) Manager's Guide* for more information.

Mapping Case Data Between the Main Procedure and Sub-Procedures

When you define a sub-procedure call in your main procedure, you need to define what case data from the main procedure is going to be transferred to the sub-procedure(s), and what case data will be returned from the sub-procedure(s) when it terminates.

The transfer of case data is determined by mapping sub-procedure fields (or parameters if you have defined sub-procedure parameters) to fields in the main procedure.

If you are defining a static call to a sub-procedure, you can pre-define the parameter mapping using parameters from the sub-procedure you are calling. This is because you know the sub-procedure that will be started.

If you are defining a dynamic sub-procedure call, you can still pre-define the parameter mapping but the parameters are taken from the sub-procedure parameter template on which the call is based. This is because the sub-procedures that are run will be decided at run-time.

For example, if you have defined a dynamic call to a number of mortgage application sub-procedures that are only run if certain conditions have been met in the case, the following process occurs:

1. When the case is run, the array field that determines what sub-procedures to start is populated with data. Also, array fields that have been set up to pass case data to the sub-procedures are populated with case data as the case progresses.
2. iProcess starts the required sub-procedures based upon the names of the sub-procedures in the sub-procedure array field.
3. iProcess uses the parameter mapping definition defined in the sub-procedure parameter template and transfers the field/value pairs contained in the arrays to the appropriate sub-procedures.

There are two types of mapping that you can use:

- Field mapping

  This is the default method used. This is where you map one field from the main procedure to another field in the sub-procedure. See Defining Input and Output Field Mappings on page 35.

- Parameter mapping

  This method extends the field mapping capability by allowing expressions or scripts in the mapping. To use parameter mapping, you have to define what sub-procedure fields will be available for mapping. You define a parameter for each sub-procedure field you want to include in the sub-procedure mapping. You can define these parameters for each sub-procedure or you can create a sub-procedure parameter template containing the sub-procedure parameters you require:

- See Defining Sub-Procedure Parameters Available for Mapping on page 16 if you just want to define the parameters for a single sub-procedure.
- See Defining Sub-Procedure Parameter Templates on page 49 if you want to create a template containing the parameters so that you can use multiple sub-procedures.

  You can then use the Parameter mapping method for mapping sub-procedure parameters to fields in the main procedure - see Defining Input and Output Parameter Mappings on page 29.

Defining a Sub-Procedure

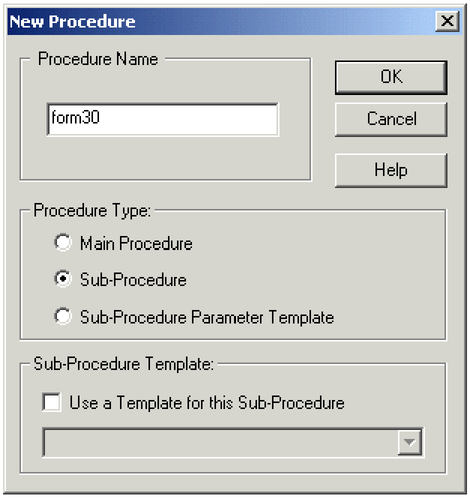
You define a sub-procedure in exactly the same way as any other procedure, but you must choose the sub-procedure option when entering the name for your new sub-procedure.

There are two ways you can define a new sub-procedure:

- *Standalone* - the sub-procedure is created with no input or output parameter information.
- *Based on a sub-procedure parameter template* - the sub-procedure will inherit the input/output parameters defined in the template. If you are going to be calling this sub-procedure as part of a dynamic sub-procedure call or graft step, it needs to be based on a template.

To define a new sub-procedure:

1. In the Procedure Management window, click **Procedure Management** > **New Procedure**.
2. The **New Procedure** dialog is displayed.



New Procedure

Procedure Name: form30

Procedure Type:
○ Main Procedure
● Sub-Procedure
○ Sub-Procedure Parameter Template

Sub-Procedure Template:
□ Use a Template for this Sub-Procedure

3. Enter a name for your sub-procedure.

   The name must be unique and can be up to 8 alpha-numeric characters. If the entered name is not unique or contains invalid characters you are warned and asked to re-specify the name.

4. Click the **Sub-procedure** radio button.

5. (Optional) To base this sub-procedure on a sub-procedure parameter template, click the **Use a Template for this Sub-Procedure** check box. Use the drop-down list box to select the template you want to use.

   You can only base a sub-procedure on a released template. Unreleased templates are grayed out.

   You must base sub-procedures on a template if you are calling a set of sub-procedures via a dynamic call step or graft step. See Chapter 5 on page 57 for more information about dynamic sub-procedure call steps.

6. Click **OK** and TIBCO iProcess Modeler opens ready for you to define your sub-procedure.

   See “Placing Procedure Objects” in *TIBCO iProcess Modeler Basic Design* for more information about using the procedure design tools.

7. Save your sub-procedure.

A procedure can be changed from being a main procedure to a sub-procedure and vice versa. See “Changing the Procedure Type of a Version” in *TIBCO iProcess Modeler Procedure Management* for more information about how to do this.

Editing a Sub-Procedure

Any user with Edit permissions and access to TIBCO iProcess Modeler can edit a sub-procedure. However, any procedure definer can make calls to any sub-procedure. See “Setting Access Controls” in *TIBCO iProcess Modeler Procedure Management* for more information about access permissions.

iProcess Administrator users can edit all sub-procedures and procedures.

If a released procedure calls an unreleased sub-procedure, the work items for the sub-procedure will go to the sub-procedure owner’s test queues.

Summary of Calling Sub-Procedures

This section summarizes the steps involved to define a call to your sub-procedure(s).

Use the following table to determine the tasks you need to perform depending on how you are setting up your procedure using sub-procedures.

After performing these steps, it is advisable to run a case of the main procedure to check that the sub-procedure call is made and check that the expected field values returned from the sub-procedure are in your main procedure. View the audit trail for the case to trace the flow of events and make sure it works as designed.

What Do You Want to Do?	Tasks to Perform	Refer To
Dynamically call multiple sub-procedures from your main procedure depending on case data that is entered.	1. Define a sub-procedure parameter template.	Defining a Sub-Procedure Parameter Template on page 52.
	2. Define the sub-procedures you want to call. These need to be based on the same sub-procedure parameter template.	Defining Sub-Procedure Parameters Available for Mapping on page 16.
	3. Define the dynamic sub-procedure call step and select the same template used to base the sub-procedures on.	Defining a Dynamic Call to Multiple Sub-Procedures on page 62.
	4. Map sub-procedure parameters to main procedure fields.	Defining Input and Output Mappings Using a Sub-Procedure Parameter Template on page 64.
	5. Define appropriate array fields in the main procedure so that multiple sub-procedures can be called and multiple elements of case data can be passed to and from them.	Using Array Fields on page 39.

What Do You Want to Do?	Tasks to Perform	Refer To
Call a single sub-procedure that is static every time a case is run for that procedure.	1. Define the sub-procedure.	Defining Sub-Procedure Parameters Available for Mapping on page 16.
	2. (Optional) Define IO Parameters.	Defining Sub-Procedure Parameters Available for Mapping on page 16.
	3. Define the static sub-procedure call.	Adding a Static Sub-Procedure Call on page 24.
	4. Map sub-procedure fields/parameters to main procedure fields.	Defining Input and Output Parameter Mappings on page 29 if you have defined sub-procedure parameters or Defining Input and Output Field Mappings on page 35 if you have not.

What Do You Want to Do?	Tasks to Perform	Refer To
Use an external application to start multiple sub-procedures	1. Define a sub-procedure parameter template.	Defining a Sub-Procedure Parameter Template on page 52.
	2. Define array fields in the main procedure.	Using Array Fields on page 39.
	3. Define a graft step.	"Using Graft Steps" in *TIBCO iProcess Modeler Integration Techniques*.
	4. Map output parameters.	"Using Graft Steps" in *TIBCO iProcess Modeler Integration Techniques*.
	5. Define TIBCO iProcess Objects calls in your application to start the required sub-procedures.	See TIBCO iProcess Objects Workspace (Windows) Online Help.

Defining Sub-Procedure Parameters Available for Mapping

iProcess enables you to define which sub-procedure fields you will be able to map to main procedure fields.

When defining the sub-procedure fields available for mapping, you create a set of *parameters* that can be used as inputs and outputs to your sub-procedures. A parameter is the combination of the field name and its description and optionally its allowed values.

Depending on your use of sub-procedures, there are two methods of defining sub-procedure parameters: per sub-procedure or for a set of sub-procedures using a sub-procedure parameter template.

Why Define Sub-Procedure Parameters?

There are two benefits to defining the list of available sub-procedure parameters for mapping:

- Ease of mapping fields.

  When defining a call to your sub-procedure, you need to define the field mappings between the main procedure and the sub-procedure. A sub-procedure can contain a large number of fields so it can take time to sort through the field list and find which fields you need to map. If you only need to map a small sub-set of these fields, it is advisable to define a list of parameters to map.

  For example, if there are only two fields in the sub-procedure that you need to map data to, you can create two parameters for these fields and prevent all of the other fields being displayed (including system fields) when you perform the parameter mapping.

- Mapping parameters to scripts and expressions.

  By defining the sub-procedure parameters available for mapping, you can use the parameter mapping functionality that includes being able to map parameters to expressions and scripts. (If you do not define sub-procedure parameters, you can only use the field to field mapping functionality.)

Methods of Defining Sub-Procedure Parameters

To define the sub-procedure parameters that will be available for mapping to the main procedure, you have two options:

- Use a sub-procedure parameter template. You *must* use this method if you are going to define dynamic calls to multiple sub-procedures or define graft steps. See Chapter 4 on page 49 for more information about defining a template. See Automatically Defining Input/Output Parameters for a Sub-Procedure on page 17 for information about automatically defining the parameters available for mapping.
- Manually select the sub-procedure fields that will be available for mapping using the **Input/Output Parameter Definition** dialog and create parameters for them. See Manually Defining Sub-Procedure Input Parameters on page 18 and Manually Defining Sub-Procedure Output Parameters on page 20.

You can also define values for the input parameters to the sub-procedure. This means that the field data is not derived from the main procedure but is defined, at design time, only for the sub-procedure parameter. See Defining Valid Values for a Sub-Procedure Input Parameter on page 19.

Automatically Defining Input/Output Parameters for a Sub-Procedure

There are a number of ways you can associate a parameter template with your sub-procedure:

- If you have created a sub-procedure based upon a template, the input/output parameters will be inherited from the template.
- If you have created a sub-procedure that has not been based on a sub-procedure parameter template, you can choose a template to base it on after its initial creation.
- You can also change the template that is associated with the sub-procedure.

If you have created a template containing sub-procedure parameters, you can automatically pre-fill the sub-procedure's input/output parameters by assigning the template to it. Template parameters appear in blue so you can have a mixture of template parameters and manually inserted parameters (black).

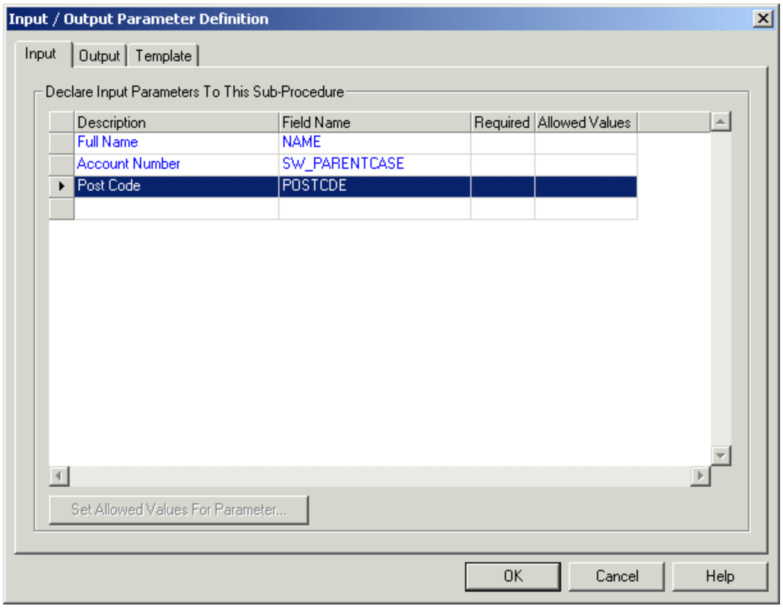
Manually Defining Sub-Procedure Input Parameters

You define the sub-procedure input parameters when you define your sub-procedure.

1. In TIBCO iProcess Modeler, open your sub-procedure and click **Procedure** > **IO Parameters**.

If the menu option is grayed out, it means that your procedure has not been defined as a sub-procedure.

2. The **Input/Output Parameter Definition** dialog is displayed. The **Input** tab is displayed by default.


Input / Output Parameter Definition
Input
Output
Template
Declare Input Parameters To This Sub-Procedure
Set Allowed Values For Parameter...
OK
Cancel
Help

Description	Field Name	Required	Allowed Values
Full Name	NAME		
Account Number	SW_PARENTCASE		
Post Code	POSTCDE		

3. In the **Description** column, enter a name (up to 32 characters) for the first input parameter you want to define.

4. In the **Field Name** column, use the drop-down list box to select a sub-procedure field that you want to make available for sub-procedure mapping.

5. If you want this field to always be defined in any call to this sub-procedure, click in the **Required** column so that a tick is displayed. This field must then be included in the Input mappings for the step to be complete. See Defining Input Parameter Mappings on page 29.

6. In the **Allowed Values** column, you can define a set of values for this parameter. When the **Allowed Values** column is selected, you can click **Set Allowed Values For Parameter**. See Defining Valid Values for a Sub-Procedure Input Parameter on page 19 for more information.

7. Press **Return** to save the new parameter.

8. Continue to define more parameters as required. Click **OK** to save your settings. To define the output parameters, see Manually Defining Sub-Procedure Output Parameters on page 20.

   The entries are added to the list of declared sub-procedure parameters for your sub-procedure.

To change the settings of an input parameter:

1. Click in the row that requires changing and update the details as required.

2. Click **OK**.

To remove an input parameter so that it does not appear in the list of valid sub-procedure parameters, select the parameter from the list and press the **Delete** key.

If the parameter is from a template (i.e. it is blue), you cannot delete it.

Defining Valid Values for a Sub-Procedure Input Parameter

When defining your sub-procedure input parameters, you can also define a list of allowed values for any of your declared parameters so that the procedure definer can only select from a restricted list of values.

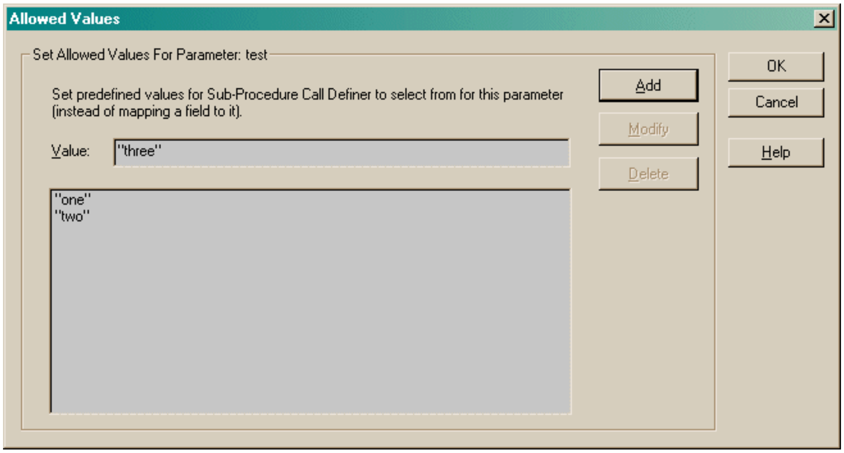
For parameters that have been defined in a template, the allowed values are set in the template and are inherited by the sub-procedure(s) using the template.

For example, if you use a sub-procedure for a generic medical checkup process, you can make the sub-procedure more specific to the calling procedure by using different forms in the sub-procedure. If you define a **Form Type** field with two valid values (in this example, **Generic** and **Diabetic**), the procedure definer can select one of these values and the correct form will be used in the sub-procedure. If **Diabetic** is selected by the procedure definer, additional fields in a form can be activated using conditional fields.

To define a list of values for a parameter, do the following:

1. On the **Input** tab of the **Input/Output Parameter Definition** dialog, select the parameter you want to define values for by clicking the gray box next to the parameter. Click **Set Allowed Values for Parameter**.

   The **Allowed Values** dialog is displayed.



Allowed Values

Set Allowed Values For Parameter: test

Set predefined values for Sub-Procedure Call Definer to select from for this parameter (instead of mapping a field to it).

Value: "three"

"one"
"two"

2. Enter a value in the **Value** field and click **Add**.

For text fields, you must enter the values in quotes.

3. To remove a value, select it in the list and click **Delete**. To change a value, select it in the list, change the value in the **Value** field and click **Modify**.

4. Click **OK** to confirm your allowed values.

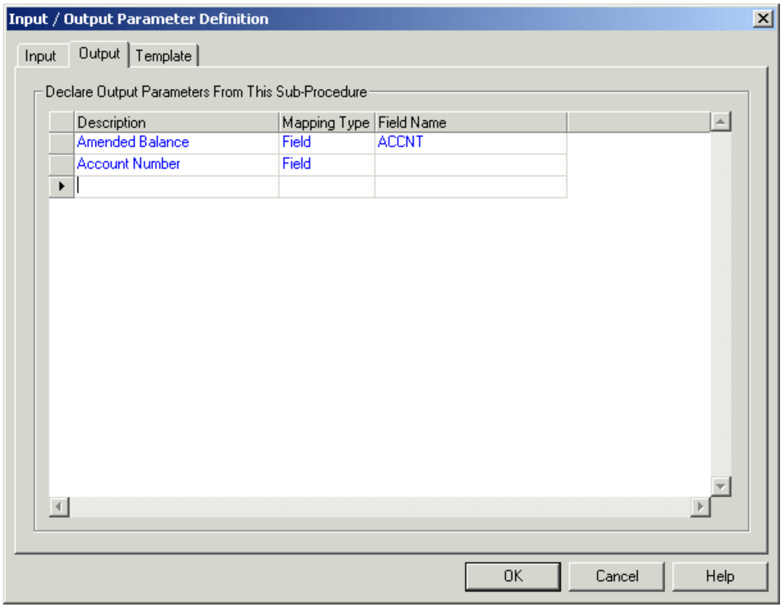
Manually Defining Sub-Procedure Output Parameters

You define the sub-procedure output parameters when you define your sub-procedure.

1. In TIBCO iProcess Modeler, open your sub-procedure and click **Procedure** > **IO Parameters**.

If the menu option is grayed out, it means that your procedure has not been defined as a sub-procedure.

2. The **Input/Output Parameter Definition** dialog is displayed showing the **Input** tab by default. Click the **Output** tab.



Declare Output Parameters From This Sub-Procedure

Description	Mapping Type	Field Name
Amended Balance	Field	ACCNT
Account Number	Field	

3. In the **Description** column, enter a name (up to 32 characters) for the first output parameter you want to define.

4. In the **Mapping Type** column, use the drop-down list to select how you want to map the parameter to a main procedure parameter:

   — **Field** - you will be able to choose a sub-procedure field from the drop-down list in the **Field Name** column.

   — **Expression** - You will be able to enter a valid expression in the **Field Name** column. See *TIBCO iProcess Expressions and Functions Reference Guide* for more information about using functions and expressions.

The SELECTVAL and SWITCHVAL expressions can be used to create conditional expressions.

5. Click **OK** to save your settings and then save your sub-procedure.

Chapter 2 Defining a Static Call to a Sub-Procedure

This chapter describes how to define a static call to a sub-procedure from your main procedure. Use this type of sub-procedure call when you know at procedure definition time which sub-procedure you want to call.

Topics

Adding a Static Sub-Procedure Call

This section describes how to define a static call to a sub-procedure from your main procedure.

If you want to define a call to multiple sub-procedures instead where the sub-procedures to run are defined when the case is run, you need to define a dynamic sub-procedure call step or graft step - see Calling a Dynamic Number of Sub-Procedures on page 57. See “Using Graft Steps” in *TIBCO iProcess Modeler Integration Techniques* for more information about graft steps.

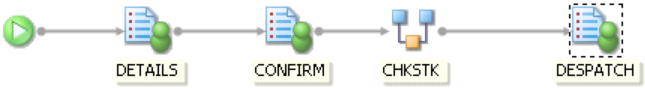
The static sub-procedure call defines the:

- sub-procedure to start
- parameters that are passed to it from the main procedure
- parameters that are returned to the main procedure when the sub-procedure completes.

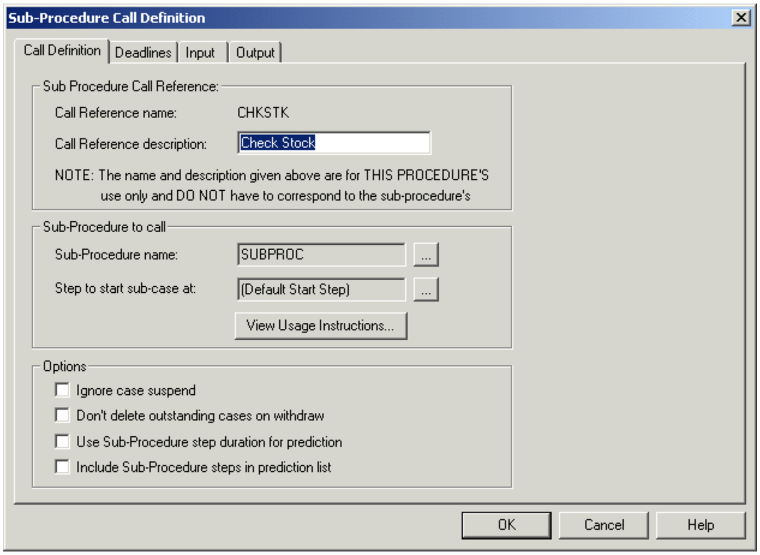
To add a static sub-procedure call to your main procedure:

1. Open your main procedure in TIBCO iProcess Modeler. Use the Procedure Browser to find the sub-procedure you want to call and drag ‘n’ drop the sub-procedure onto your procedure chart window.

   Alternatively, select the sub-procedure tool and click where you want to place the object in the Process Definer layout.



The **Sub-Procedure Call Definition** dialog is displayed.



Sub Procedure Call Reference:

Call Reference name: CHKSTK

Call Reference description: Check Stock

NOTE: The name and description given above are for THIS PROCEDURE'S use only and DO NOT have to correspond to the sub-procedure's

Sub-Procedure to call

Sub-Procedure name: SUBPROC ...

Step to start sub-case at: (Default Start Step) ...

View Usage Instructions...

Options

- ☐ Ignore case suspend
- ☐ Don't delete outstanding cases on withdraw
- ☐ Use Sub-Procedure step duration for prediction
- ☐ Include Sub-Procedure steps in prediction list

2. Enter a name (up to 8 characters) for the sub-procedure call in the **Call Reference Name** field. (Optional) Enter a description (up to 24 characters) for the call in the **Description** field.

These names do not have to match the sub-procedure's name and description as they are only for use in this procedure.

If you have dragged a sub-procedure onto your procedure from the procedure browser, the **Call Reference Name** and **Sub-Procedure Name** are pre-filled for you.

3. Enter the name of the sub-procedure you want to call in the **Sub-Procedure Name** field. If you have used the sub-procedure call object to define your call i.e. you have not used the drag 'n' drop method, do the following:

   a. Click the button to the right of the **Sub-Procedure Name** field to see a list of existing sub-procedures on your iProcess installation. This can be either a new, undefined, sub-procedure or an existing sub-procedure. If you enter a new name you are warned that the sub-procedure does not exist and asked if you want to continue.

   b. In the **Select Sub-Procedure to Call** dialog, use the procedure browser to find your sub-procedure, then click **OK**.

4. (Optional) Enter the name of the step that you want the sub-procedure to start at in the **Step to start sub-case** at field.

The default start step is the first step in the procedure but you can select any other step as appropriate. Click the button to the right of the field to see a list of the steps in the sub-procedure you have selected.

5. (Optional) Click **View Usage Instructions** to display any usage information that has been set up for this sub-procedure. This can be information at a specific URL or a text file. See “Usage Instructions” in *TIBCO iProcess Modeler Procedure Management* for more information about defining usage instructions.

6. (Optional) Click **View Sub-Procedure** if you want to view the sub-procedure you are calling.

The basic call reference information is now defined. You can now do the following:

7. (Optional) Define the call definition options such as step prediction. See Defining Call Definition Options on page 27.

8. (Optional) Define a deadline for the sub-procedure call step. See “Defining a Deadline” in *TIBCO iProcess Modeler Basic Design*.

9. Define the input parameter mappings. See Defining Input Parameter Mappings on page 29 if you have pre-defined the sub-procedure input and output parameters or see Defining Input Field Mappings on page 35 if you haven’t.

10. Define the output parameter mappings. See Defining Output Parameter Mappings on page 32 if you have pre-defined the sub-procedure output parameters or see Defining Output Field Mappings on page 37 if you haven’t.

11. Click **OK**. TIBCO iProcess Modeler checks if the defined sub-procedure exists. If it doesn’t (this may be the case if a new sub-procedure has not yet been saved) then TIBCO iProcess Modeler issues a warning message. If the sub-procedure name is blank then the step is marked as incomplete and the main procedure also has a status of incomplete.

Defining Call Definition Options

When defining a call definition, you can choose to select one or more of the following options:

Option	Description
Ignore Case Suspend	Use this when you want the sub-procedure to still be processed as normal while a case is suspended by a TIBCO iProcess Objects or SAL application. This means that work items generated by the sub-procedure can still be opened, and deadlines on work items generated by the sub-procedure are still processed. If Ignore Case Suspend is not checked (the default option), the sub-procedure is not processed while the case is suspended. This means that: • work items generated by the sub-procedure are marked as unavailable and cannot be opened (until the case is re-activated). • deadlines on work items generated by the sub-procedure are not processed. The date and time at which deadlines are due are not affected, and deadlines continue to expire. However, no actions are processed when a deadline expires. When the case is re-activated, any expired deadlines are immediately processed. **Note**: Cases can only be suspended and re-activated from a TIBCO iProcess Objects or SAL application. Audit trail messages indicate whether a case is active or suspended. See TIBCO iProcess Objects documentation for more information about suspending cases.

Option	Description
Don't delete outstanding cases on withdraw	Select the **Don't Delete Outstanding Cases On Withdraw** option. If this option is selected, and the deadline on an outstanding step expires or it is withdrawn as an action (release or deadline expire) of another step: • the deadline actions are processed. • the step remains outstanding (the step remains in the work queue or the sub-procedure case is not purged). • when the step is released (or the sub-procedure case completes) the normal release actions are not processed but the case field data associated with the release step (e.g. the field values set in a normal step whilst in a work queue or the output parameters of a sub-case) is applied to the main case data.
Use sub-procedure step duration for prediction	Select the **Use Sub-procedure Step Duration For Prediction** option. If this option is selected, the duration of the sub-step, which is set in the main procedure, is displayed. If this option is not selected, the duration of the sub-step is regarded as 0 or 1 microsecond. The sub-procedure is displayed in the outstanding step list but its steps are not displayed. See Chapter 7 on page 89 for more information about case prediction.
Include sub-procedure steps in prediction list	Includes all of the sub-case steps in the outstanding step list. See Using Case Prediction to Forecast Outstanding Work Items on page 89 for more detailed information about using case prediction.Select this option to include all of the sub-case steps in the outstanding step list.

Defining Input and Output Parameter Mappings

This section describes how to define input and output mappings for case data if you have defined the sub-procedure parameters you want to use for mapping (see Defining Sub-Procedure Parameters Available for Mapping on page 16).

If you have *not* defined the sub-procedure parameters you want to use for mapping, see Defining Input and Output Field Mappings on page 35 instead.

Defining Input Parameter Mappings

When defining the sub-procedure call, the **Sub-procedure Call Definition** dialog enables you to select the sub-procedure you want to call and then define the input and output case data mappings.

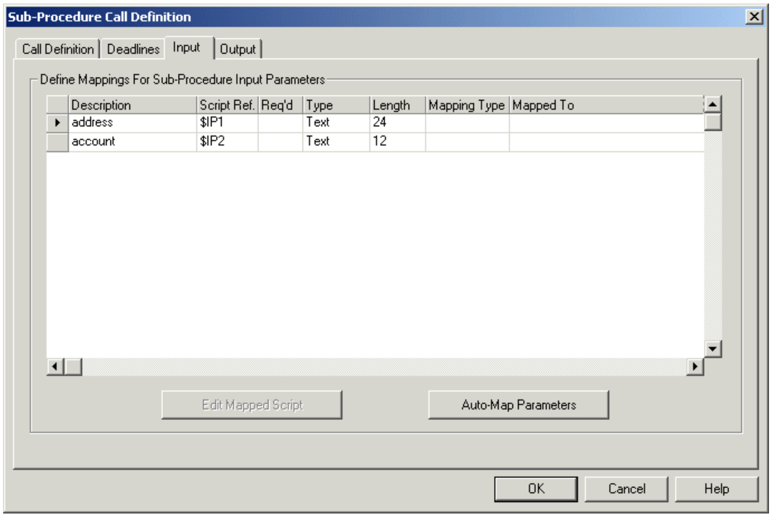
To automatically map sub-procedure fields with calling procedure fields, use the Auto Map feature. See Using Automatic Parameter Mapping on page 34 for more information.

To define which fields from the main procedure are mapped to the sub-procedure fields, do the following:

1. On the **Sub-Procedure Call Definition** dialog, click the **Input** tab.

The **Input** and **Output** tabs are only displayed after a sub-procedure has been selected on the **Sub-Procedure Call Definition** dialog.

The list of previously defined sub-procedure parameters is displayed. You can map these to appropriate main procedure fields, expressions or scripts as required.



Description	Script Ref.	Req'd	Type	Length	Mapping Type	Mapped To
address	$IP1		Text	24		
account	$IP2		Text	12		

Parameter to Field Mapping:

To perform a simple parameter to field mapping, do the following:

1. All pre-defined input parameters will be displayed automatically. For the parameter you want to map to a field, click in the **Mapping Type** column and select **Field** from the drop-down list.

2. In the **Mapped To** column, select the main procedure field you want to map it to. Only fields of the same field type are listed e.g. **Numeric** or **Date** except for parameters whose field type is either **Text** or **Memo**. Parameters whose field type is either **Text** or **Memo** allow selection from a list of fields including both **Text** and **Memo** field types. .

If a memo is passed into a text field, then the data is truncated at the size of the text field.

To unmap a parameter, select the entire row by clicking the gray button next to the row and press Delete.

Mapping a Parameter to an Expression:

To map a sub-procedure parameter to an expression, do the following:

1. For the parameter you want to map to an expression, click in the **Mapping Type** column and select **Expression** from the drop-down list.
2. In the **Mapped To** column, enter a valid iProcess expression to map the field to. For example, if the main procedure contains two fields for a person's name (FORENAME and SURNAME), you can create an expression to join the two values so that the single name is mapped to a NAME sub-procedure parameter. See *TIBCO iProcess Expressions and Functions Reference Guide* for more information about expressions and functions.

Mapping a Parameter to a Script:

To perform a parameter to script mapping, do the following:

1. For the parameter you want to map to a script, choose **Script** in the **Mapping Type** column.
2. In the **Mapped To** column, select a script name or choose **<Private Script>**. Private scripts can be created for use only by this sub-procedure mapping. See Creating Scripts on page 79 for more information about creating scripts and private scripts.
3. If you want to edit the script, click **Edit Mapped Script** and the Script Editor is displayed.

Mapping a Field With Pre-Defined Values:

If a parameter has had values previously defined for it, the following procedure applies:

1. Select the parameter and click in the **Mapping Type** column. If it has values pre-defined, the **Preset Values** string is automatically entered into this column.
2. In the **Mapped To** column, select one of the pre-defined values from the drop-down list box.

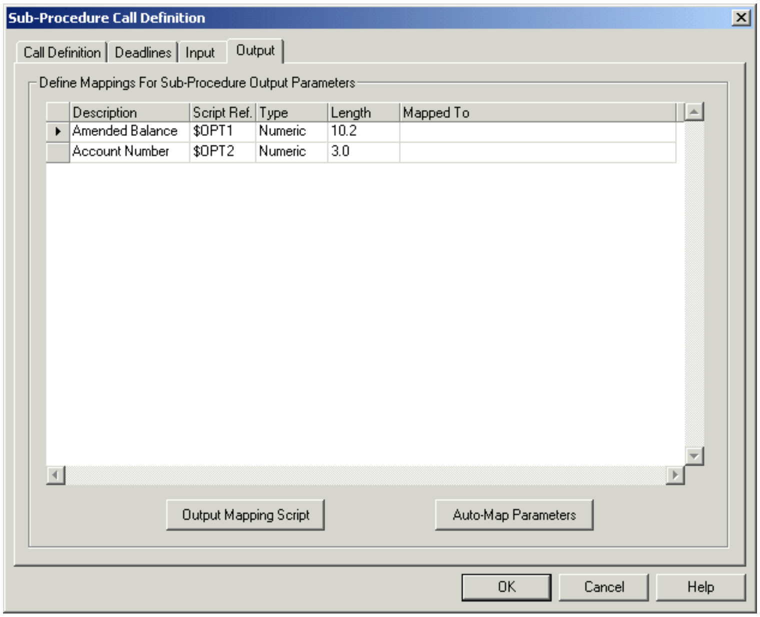
Defining Output Parameter Mappings

When defining the sub-procedure call, the **Sub-procedure Call Definition** dialog enables you to select the sub-procedure you want to call and then define the input and output mappings. To define the output mappings:

1. On the **Sub-procedure Call Definition** dialog, click the **Output** tab.

> The **Input** and **Output** tabs are only displayed after a sub-procedure has been selected on the **Sub-Procedure Call Definition** dialog.

The **Output** tab is displayed. A list of the defined sub-procedure output parameters is displayed. You can map these to appropriate main procedure fields or expressions as required.



Sub-Procedure Call Definition

Define Mappings For Sub-Procedure Output Parameters

Description	Script Ref.	Type	Length	Mapped To
Amended Balance	$OPT1	Numeric	10.2	
Account Number	$OPT2	Numeric	3.0	

2. For each parameter, select a main procedure field to map it to from the **Mapped To** column. Only fields of the same type as the one selected are displayed e.g. numeric or text.

   Alternatively, you can automatically create the output mappings using the Auto-Map feature. See Using Automatic Parameter Mapping on page 34.

3. Click **OK** to save your settings.

Running an Output Mapping Script

Output mapping consists of mapping the sub-procedure output parameters to main procedure fields and optionally the ability to run a private script after all mappings have been performed. This can be used to perform additional field manipulation. You can use the $OP*n* and $OPT*n* names in your script to see the parameter values in your script. These are displayed in the parameter list.

1. Click **Output Mapping Script** to define a private script.

   The Script Editor is displayed.

2. Define your script and save it. See Creating Scripts on page 79 for more information and an example about creating private scripts.

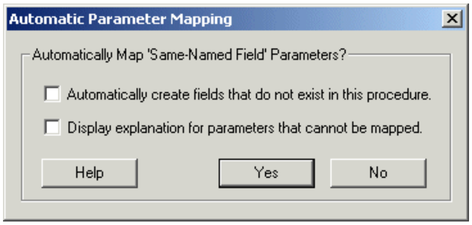
Using Automatic Parameter Mapping

To speed up the process of sub-procedure parameter mapping, you can use the auto-map feature to automatically map fields that have the same name. This facility can also create any fields in the main procedure that are not already created.

Automapping does not work for fields that have allowed values defined for them. See Defining Valid Values for a Sub-Procedure Input Parameter on page 20.

To automatically map sub-procedure parameters to calling fields, do the following:

1. Open the **Sub-Procedure Call Definition** dialog, click the **Input** tab if you want to automap input fields or click the **Output** tab to automap output fields.
2. Click **Auto-Map Parameters**. The **Automatic Parameter Mapping** dialog is displayed:



Automatic Parameter Mapping

Automatically Map 'Same-Named Field' Parameters?

☐ Automatically create fields that do not exist in this procedure.

☐ Display explanation for parameters that cannot be mapped.

3. You can select either or both of the following options:

— **Automatically create fields that do not exist in this procedure**

Select this option if you want any sub-procedure fields that do not exist in the calling procedure to be created when performing the parameter mapping.

If you click **Cancel** in either the **Input** or **Output** tabs, any fields that have already been created are not undone.

— **Display explanation for parameters that cannot be mapped.**

Select this option so that a message is displayed for any parameters that cannot be automatically mapped. For example, a field with the same name may exist in the main procedure, but it is a different type to the sub-procedure field.

4. Click **Yes**.

For example, if the sub-procedure has an input parameter called CUSTOMERNAME and the main procedure has a field called CUSTOMERNAME, the fields will be automatically mapped. If the CUSTOMERNAME field in the main procedure does not exist, the field will be created if you select the **Automatically create fields that do not exist in this procedure** option.

Defining Input and Output Field Mappings

This section describes how to define input and output mappings for case data if you have not defined sub-procedure parameters to use for sub-procedure mapping - see Defining Sub-Procedure Parameters Available for Mapping on page 16.

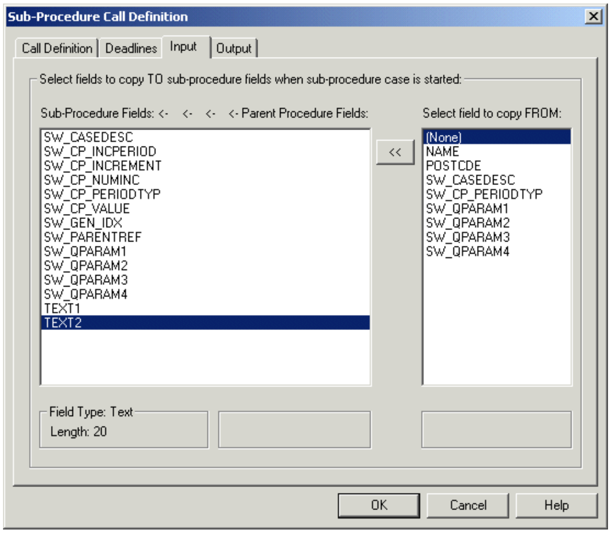
If you have defined the sub-procedure parameters you want to use for mapping, see Defining Input and Output Parameter Mappings on page 29 instead.

Defining Input Field Mappings

To specify the input field data to be mapped from the main procedure to the sub-procedure when the sub-case is started, do the following:

1. Click the **Input** tab from the **Sub-Procedure Call Definition** dialog.

The **Input** and **Output** tabs are only displayed after a sub-procedure has been selected on the **Sub-Procedure Call Definition** dialog.



Select fields to copy TO sub-procedure fields when sub-procedure case is started:

Sub-Procedure Fields: <- <- <- <- Parent Procedure Fields:

SW_CASEDESC
SW_CP_INCPERIOD
SW_CP_INCREMENT
SW_CP_NUMINC
SW_CP_PERIODTYP
SW_CP_VALUE
SW_GEN_IDX
SW_PARENTREF
SW_QPARAM1
SW_QPARAM2
SW_QPARAM3
SW_QPARAM4
TEXT1
TEXT2

Select field to copy FROM:

(None)
NAME
POSTCDE
SW_CASEDESC
SW_CP_PERIODTYP
SW_QPARAM1
SW_QPARAM2
SW_QPARAM3
SW_QPARAM4

Field Type: Text
Length: 20

The dialog shows the available fields in the sub-procedure (as per the last save of that procedure) on the left and the main procedure fields on the right. The main procedure fields shown are those that match the type of field in the sub-procedure that is currently highlighted. In the above example the sub-procedure field ITEM2 is a text field, so only text fields are displayed in the main list. If a date field was highlighted then only date fields would be displayed in the main field list.

2. Click the required sub-procedure field.
3. Click the main procedure field you want to map from.
4. Click << and the name of the main procedure field is displayed alongside the sub-procedure field name.

Double-clicking the main procedure field has the same effect.

The names of the fields in the main procedure and the sub-procedure do not need to be the same. However, you should ensure that the size of the field you are copying to is big enough to hold the copied data. A warning message will be displayed if the destination field is not big enough to hold the maximum possible value of the field being copied. If truncation occurs, this will lead to data loss and the field may be set to SW_NA.

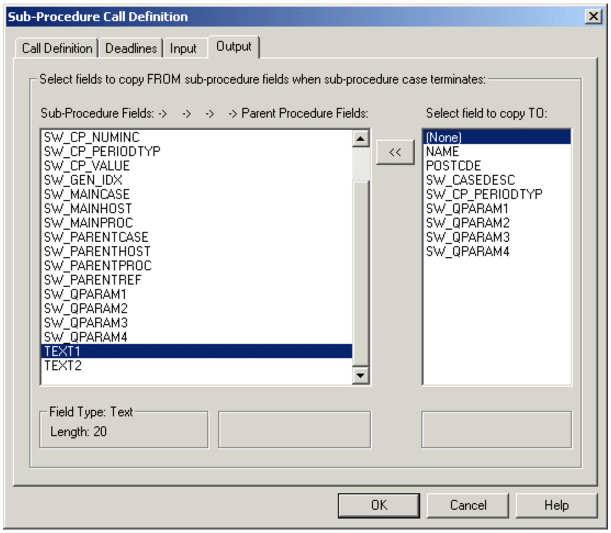
To remove a field mapping, select the sub-procedure field, select **(None)** in the **Select field to copy FROM** list and click <<.

Defining Output Field Mappings

To specify the output field data to be mapped from the sub-procedure to the main procedure when the sub-case terminates, do the following:

1. Click the **Output** tab from the **Sub-Procedure Call Definition** dialog.

The **Input** and **Output** tabs are only displayed after a sub-procedure has been selected on the **Sub-Procedure Call Definition** dialog.



Sub-Procedure Call Definition

Select fields to copy FROM sub-procedure fields when sub-procedure case terminates:

Sub-Procedure Fields: -> -> -> Parent Procedure Fields:

SW_CP_NUMINC
SW_CP_PERIODTYP
SW_CP_VALUE
SW_GEN_IDX
SW_MAINCASE
SW_MAINHOST
SW_MAINPROC
SW_PARENTCASE
SW_PARENTHOST
SW_PARENTPROC
SW_PARENTREF
SW_QPARAM1
SW_QPARAM2
SW_QPARAM3
SW_QPARAM4
TEXT1
TEXT2

Select field to copy TO:

(None)
NAME
POSTCDE
SW_CASEDESC
SW_CP_PERIODTYP
SW_QPARAM1
SW_QPARAM2
SW_QPARAM3
SW_QPARAM4

Field Type: Text
Length: 20

This dialog operates in exactly the same way as the Input tab with the exception that you are specifying which fields to map *from* the sub-procedure *to* the main procedure when the sub-case has terminated.

2. Click the required sub-procedure field.
3. Click the main procedure field you want to map to.
4. Click << and the name of the main procedure field is displayed alongside the sub-procedure field name.

Double-clicking the main procedure field has the same effect.

The names of the fields in the main procedure and the sub-procedure do not need to be the same. However, you must ensure that the size of the field you are copying to is big enough to hold the copied data otherwise the data will be truncated.

To remove a field mapping, select the sub-procedure field, select **(None)** in the **Select field to copy TO** list and click <<.

Chapter 3 Using Array Fields

This chapter describes the use of array fields in iProcess and how to reference the data elements in an array. Array fields are similar to standard (single instance) iProcess fields but can contain multiple data elements instead of just one element.

Topics

What are Array Fields

An array field is similar to a standard iProcess field but instead of storing just one data element (for example, a customer name), it can store multiple data elements (for example, 10 customer names). An array field can contain up to 99,999 elements where each element has the same field type and length/decimal place specifications.

An array field can be used in the same way as iProcess single instance fields, so you can add them to forms and use them in scripts.

You can either map an array field from a parent case to an array field in a sub-case or only transfer the currently indexed item in the array field to the sub-case. For more information, see Defining Input and Output Mappings Using a Sub-Procedure Parameter Template on page 64.

Array fields enable you to set up dynamic calls to sub-procedures because they are used to define which sub-procedures to start and which start step to start at. For example, if an array field gets populated during a case with two sub-procedure names, iProcess will start two sub-procedures. When the next case is run, the array field might be populated with three sub-procedure names so three sub-procedures are run. Similarly, you need to use array fields when using graft steps. For more information about using graft steps, see “Using Graft Steps” in *TIBCO iProcess Modeler Integration Techniques*.

Array fields are also used to pass multiple data elements to the sub-procedures. For example, if you are passing data to multiple sub-procedures and the data in the array field is only determined when the case is run, the array can pass each sub-procedure its own element of data.

Each data element in the array is uniquely identified by an index. This means that a set of data in the array can be passed around iProcess and each element is tracked using the index. When using multiple sub-procedures, you pass a set of data to the sub-procedure using array fields so that each sub-procedure case is passed one of the elements. If the first element has an index of 1, one sub-procedure is started using the element associated with index 1 and any data returned to the main procedure is associated with that index.

Why Use Array Fields

The following are the main reasons for using array fields in iProcess:

- Calling a dynamic number of sub-procedures.

  If you define a procedure where a number of sub-procedures can be run depending on what data has been entered into the case, you need to set up an array field to define what sub-procedures are run.

- Transferring the correct data element from a field in the parent case to each sub-case that is started from a dynamic sub-procedure call or graft step.

  The field in the parent case needs to be set up as an array field so that it can contain multiple data elements.

- Grafting multiple sub-procedures to a procedure.

  You can use grafts steps to attach sub-procedures (that have been started by an external application) to the main procedure. See *TIBCO iProcess Modeler Integration Techniques* for more information about using Graft steps.

See Example of Using Arrays on page 44 for a detailed example of using array fields.

How Do Array Fields Work in iProcess

Each array field that is defined in iProcess has a corresponding numeric array element index field automatically created for it. This index is given the name IDX_*arrayfieldname*. Therefore, if you
create an array field called CUSTNAME, the index field is called IDX_CUSTNAME.

There is also a generic index field called SW_GEN_IDX, which is used if an array field's individual index is unassigned.

In a form, an array field is identified by its name only (i.e. no specific index is specified). When a case is run and the form is displayed, the field will show the value according to the current array element index. When the array is referenced by its name, iProcess will take the array element index number to be used from the array field's corresponding index field (IDX_*arrayfieldname*). If this is not assigned, it uses the SW_GEN_IDX field index. You can set the array field index to SW_NA and use the SW_GEN_IDX index for all those array fields.

You do not need to specify the maximum size of the array because it will grow to the size of the highest index used.

Using array fields, it is possible to design a form so that you can step through record sets. If you have two array fields on a form called account and name, using a script you can increment the array index to display each data element in the array. See Example of Using Arrays on page 44 for more information about how this can be done.

Using Composite Array Fields

You can define composite fields as arrays so that each element in the array stores a value for each of the fields in the composite field. This can be an effective way of organizing your data sets because it is easy to see what data elements belong together.

Before creating your composite array field, you need to create a table containing the fields you want to use using the iProcess Table Manager. See "Using Table Manager" in *TIBCO iProcess Workspace (Windows) Manager's Guide* for more information.

For example, if a procedure requires a set of fields to store customer details that are passed to a multiple number of sub-procedures, a composite array field is defined. A table called CUSTDETL with NAME and ACCOUNT fields is also created. In TIBCO iProcess Modeler, an array field called CUSTDETL is created based on this table, which contains the NAME and ACCOUNT sub-fields. iProcess automatically creates the index field for this array field (IDX_CUSTDETL).

There are limitations on the field types and sizes used in iProcess tables. For example, the maximum size of a table text field is 30 characters so you would not be able to use an array of 50 character text fields in a composite array field. Also, you cannot include memo fields in tables even though you can create an array of memos.

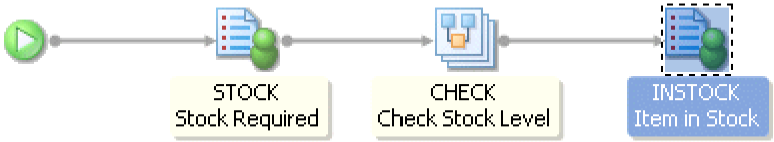
Example of Using Arrays

The following example describes how multiple data elements can be transferred to a set of sub-procedures using array fields.

Brief

A procedure is designed so that a stock check can be performed on items that are selected for order from an online ordering system. The stock check is performed as a sub-procedure with each stock item being checked as a separate sub-procedure case.

Example of an iProcess Procedure



The procedure is set up as follows:

- A dynamic sub-procedure call step called **check** is set up to perform a stock check at the distribution center. This returns a value of Y or N to the main procedure for each item placed on the order list.
- The first step of the main procedure allows the user to enter the stock number and quantity of each stock item they require. It contains two array fields called **stockno** and **quantity**.
- An array field called **sprocnam** has been defined which is used to start the required sub-procedures. This gets calculated as a result of SW_GEN_IDX being incremented when a new stock item is added - for example, if 3 stock items are entered, then 3 elements of the **sprocnam** array will be set to “check” so 3 cases of the **check** sub-procedure will be started. An **Index** field has been added to the example form below to show the current value of SW_GEN_IDX.
- A **Next Record** button is added to the form to enable the user to enter further product items that they wish to order. This button is set up as an application field with a call to the following script, which increments the array index value:

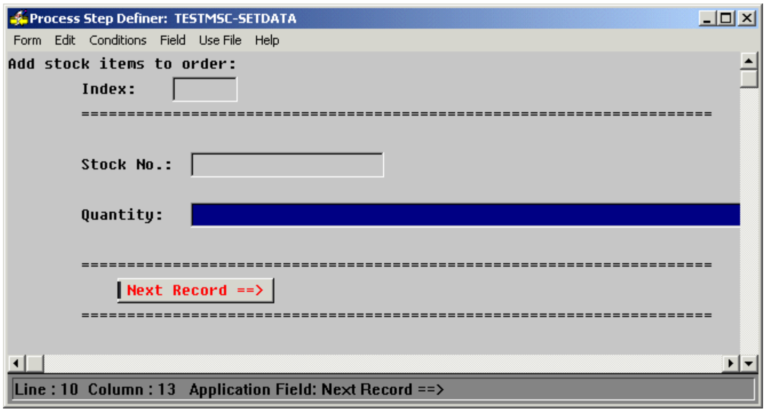
```
if (sw_gen_idx = SW_NA)
sw_gen_idx:=0
else
```

```
sw_gen_idx:=sw_gen_idx +1
endif
sprocnam:="check"
```

Flow of a Case of STOCKCHK

1. When the case is started, the user is presented with a form in which they enter the stock item number they want to order and the quantity they require of that item. Because these are array fields, they can contain more than one data element and the user can enter multiple stock items and quantities.

   The user can press the **Next Record** button to step through the records and enter data for each one. This increments the array element index and displays the next data element in the array.


Process Step Definer: TESTMSC-SETDATA
Form Edit Conditions Field Use File Help

```
Add stock items to order:
        Index:
        ======================================================================

        Stock No.:

        Quantity:

        ======================================================================
            Next Record ==>
        ======================================================================
```

Line : 10 Column : 13 Application Field: Next Record ==>

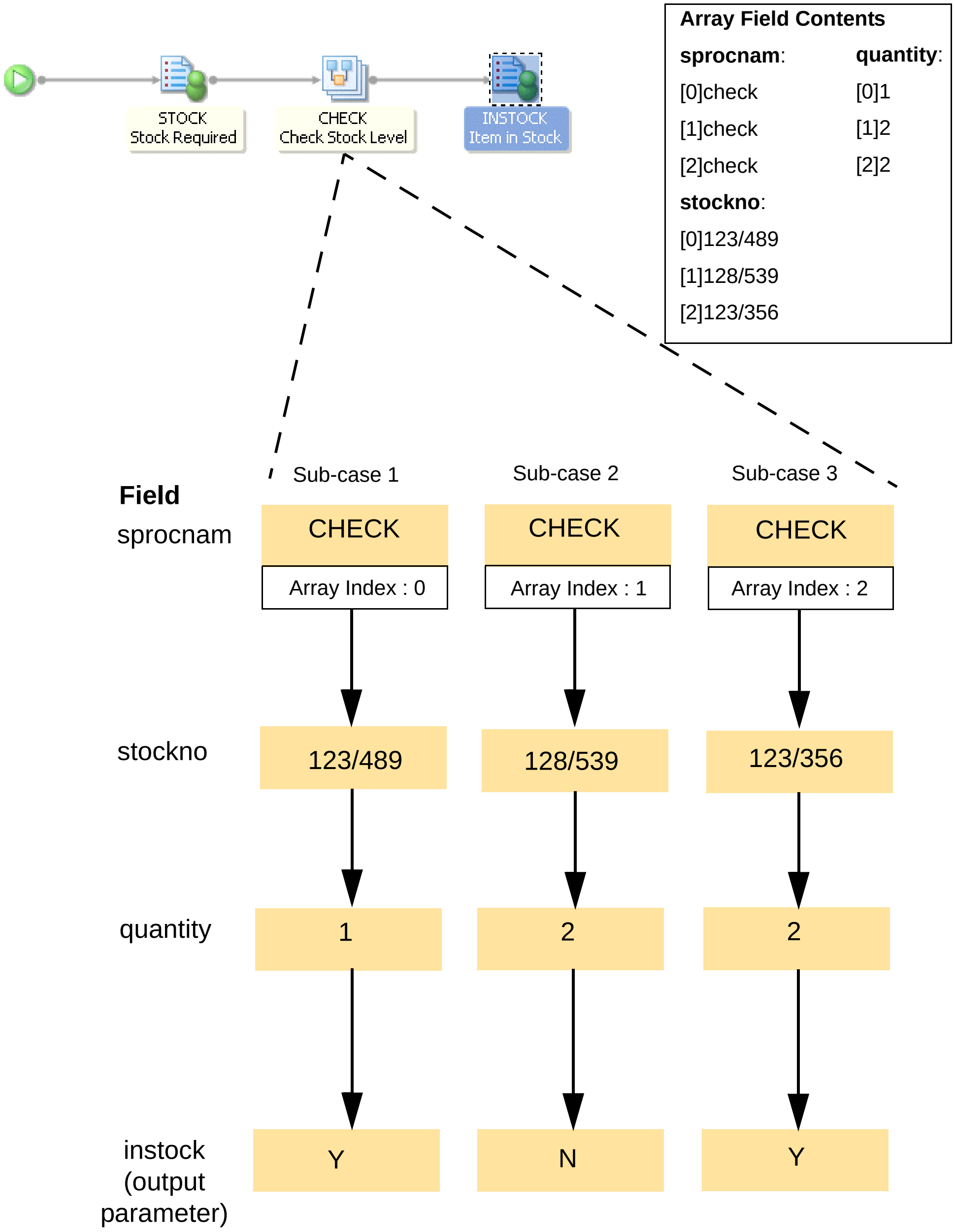
2. When the step is released, a dynamic sub-procedure call is made and the **sprocnam** array field is used to start the required sub-procedures. This contains 3 elements each called **stockchk** so 3 cases of the **check** sub-procedure are started.

3. The sub-procedure mapping is set up so that the **stockno** and **quantity** parameters in the sub-procedure are mapped to the same named array fields in the main procedure.

4. The first **check** sub-procedure case that starts has an index of **0**. The **stockno** array field contains multiple items and the element with the matching index of **0** is passed to this sub-procedure. Similarly, the **quantity** array field passes the element with an index of **0** to the sub-procedure.

5. The second sub-procedure that is started has an index of **1**. The array fields pass the data elements that correspond to the element index of **1**. This process continues for all the sub-processes that are started. See the diagram on the following page to see how the array elements are passed to the correct sub-procedures.

6. When the sub-case completes, the return value is passed back using the same index number. This means that the correct return values are kept with their respective sub-cases and sub-case data.



Array Field Contents

sprocnam:	**quantity**:
[0]check	[0]1
[1]check	[1]2
[2]check	[2]2

stockno:

[0]123/489

[1]128/539

[2]123/356

Referencing Elements in Array Fields

The following sections describe the ways you can reference and find the individual elements in an array field:

- Referencing an Array by Name Only on page 47
- Referencing Arrays in an abox file on page 47
- Referencing Array Elements in Expressions on page 48
- Iterating Through and Finding Array Elements on page 48.

For a description of defining array fields, see“Defining Fields” in *TIBCO iProcess Modeler Basic Design*.

Referencing an Array by Name Only

An array field can be identified by just its name without providing any index identifier. This method is used when adding arrays to iProcess forms. When the array field is referenced by its name, iProcess automatically uses the array element index number to be used from the array fields corresponding index field (IDX_*arrayfieldname*).

If the index field is not currently assigned, the array element index is taken from the generic SW_GEN_IDX field. If both the index field and SW_GEN_IDX are set to SW_NA, iProcess defaults to an array element index of **0**.

Referencing Arrays in an abox file

When naming a field in an abox file or SPO interface call, you can specify the array element index number directly in the field name rather than presetting the corresponding index field value first. Use the *FLDNAME*[*array element index number*] syntax. For example, an abox file could be specified as follows:

```
ACCOUNT[0], 123678
ACCOUNT[1], 56389
ACCOUNT[2], 32789
```

If you have defined composite array fields, the abox file could be specified as follows:

ACCOUNT[0]:CURRENT, 123678
ACCOUNT[0]:SAVINGS, 123679
ACCOUNT[1]:CURRENT,56389
ACCOUNT[1]:SAVINGS, 56390
ACCOUNT[2]: 32789

Specifying the index number directly overrides the default use of the IDX_ACCOUNT index or the generic SW_GEN_IDX index.

Referencing Array Elements in Expressions

When using iProcess expressions, you can use the [...] element referencing convention with any valid expression that results in a numeric return. This means you can use any of the following methods to define the array element index:

- a constant hard-coded value, for example:

  ```
  ACCOUNT[0]:=”123678”
  ```

- a numeric field, for example:

  ```
  ACCOUNT[NUM1]:=”123678”
  ```

- or a numeric expression, for example:

  ```
  ACCOUNT[n1+n2+n3]:=”123678”
  ```

Iterating Through and Finding Array Elements

To iterate through the elements in an array field, you can use the expression function **NextArrElement**. This allows you to provide a starting element and then iterate through each element in the given array field. For example, you could set up a script to cycle through the values in an array field (such as customer names) and display them in a message box on the screen. For more details about using this function see “NextArrElement” in *TIBCO iProcess Expressions and Functions Reference Guide*.

You can also find the first array element that matches a given value using the **FindArrElement** function. For example, in a customer name array field, you can find all elements that contain “J Smith”. For more details about using this function see “FindArrElement” in *TIBCO iProcess Expressions and Functions: Reference Guide*.

These functions will also help you if your procedure does not ensure the use of consecutive array element indexes because you can find an element that is not next in the index sequence.

Chapter 4 Defining Sub-Procedure Parameter Templates

This chapter describes how to create *sub-procedure parameter templates*. A parameter template enables you to keep a consistent set of input and output data for multiple sub-procedures that are called from a dynamic sub-procedure call step or graft step.

You have to define and use a parameter template when you intend to use dynamic calls to sub-procedures using the dynamic sub-procedure call object or when you use graft steps. When creating sub-procedures which you will call from graft steps or dynamic sub-procedure calls, you must base them on a template so that the input/output parameters are consistent for each sub-procedure.

What are Sub-Procedure Parameter Templates

A sub-procedure parameter template is used to define a consistent set of input and output data for multiple sub-procedures. With graft steps and dynamic sub-procedure call steps, it is not known which sub-procedures will be run when the procedure is being defined. It is only when the case is run that the exact sub-procedures to run are determined. Therefore, each sub-procedure must use the same parameter set.

The parameter template enables you to define a set of parameters that you can use to map to a group of sub-procedures. For example, if you have 5 sub-procedures relating to the process of ordering a new mobile phone (e.g. transfer number, buy insurance, etc.), you can set up a template to use for this group. Without a template, the main procedure will not know how to call the sub-procedures

You associate a parameter template with a sub-procedure that will be called from a dynamic sub-procedure call. This defines the available list of the sub-procedure's input and output parameters.

For example, a dynamic sub-procedure call using a template called **Set1** can only call sub-procedures that have been associated with the same template (**Set1**).

The template associated with the sub-procedure can be changed at any time if required - see Editing a Sub-Procedure Parameter Template on page 54.

Editing a template may invalidate out of date dynamic call steps and sub-procedure combinations (depending on the error handling options that are defined in the call step - see Troubleshooting Dynamic Sub-Procedure Calls on page 76).

Why Use Sub-Procedure Parameter Templates

When using dynamic calls to sub-procedures and graft steps, iProcess decides which sub-procedures to run during a case of the procedure (depending on what case data is entered to populate the array fields). Instead of defining separate parameter mappings for each possible sub-procedure that may be started, you use a parameter template to define the input and output parameters for a given set of sub-procedures. Therefore, for each sub-procedure that can be run, only the parameters specified in the template can be passed into the sub-procedure and passed back to the main procedure.

You can have one set of sub-procedures using one template and another set using a different template i.e. you can define multiple templates and have different sub-procedures assigned to them.

When you define a call to multiple sub-procedures or a graft step, you select the parameter template to use and then map the input and output parameters using the parameters from the template.

Defining a Sub-Procedure Parameter Template

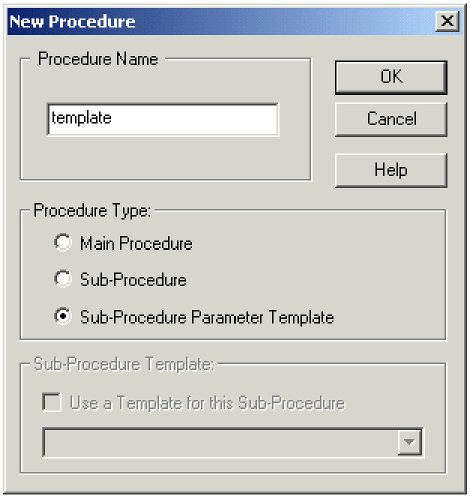
Parameter templates are displayed as objects in the Procedure Manager along with main procedures, sub-procedures and libraries.

Create a folder in which to store your parameter templates so you can easily find them.

To create a new sub-procedure parameter template:

1. Using the Procedure Manager, select a library in which to create your template.
2. Click **Procedure Management** > **New Procedure.**

   The **New Procedure** dialog is displayed.



New Procedure

Procedure Name: template

Procedure Type:

○ Main Procedure

○ Sub-Procedure

● Sub-Procedure Parameter Template

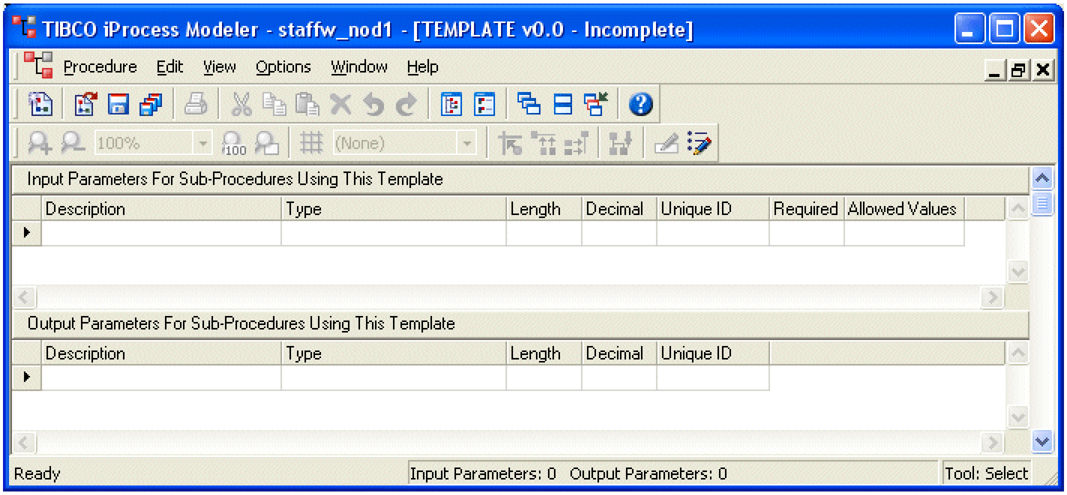
Sub-Procedure Template:

□ Use a Template for this Sub-Procedure

3. In the **Procedure Name** section, enter a name for your parameter template.

   The name must be unique and can be up to 8 alpha-numeric characters. If the entered name is not unique or contains invalid characters you are warned and asked to re-specify the name.
4. Select **Sub-Procedure Parameter Template** and click **OK**.

TIBCO iProcess Modeler is displayed with a template chart.


TIBCO iProcess Modeler - staffw_nod1 - [TEMPLATE v0.0 - Incomplete]
Procedure
Edit
View
Options
Window
Help
100%
(None)

Input Parameters For Sub-Procedures Using This Template

Description	Type	Length	Decimal	Unique ID	Required	Allowed Values

Output Parameters For Sub-Procedures Using This Template

Description	Type	Length	Decimal	Unique ID

Ready | Input Parameters: 0 Output Parameters: 0 | Tool: Select

5. In the **Input Parameters for Sub-Procedures Using this Template** table, declare all the input parameters for sub-procedures that will use this template. Do the following:

   a. In the **Description** column, enter the name of the first parameter you want to define.

   b. In the **Type** column, use the drop-down list box to select the field type e.g. text, numeric, date, etc.

   c. In the **Length** column, enter the field length if you have chosen **Text** or **Numeric** field types.

   d. In the **Decimal** column, enter the number of decimal places allowed for the field (only required for numeric type fields).

   e. The **Unique ID** column is automatically completed when you press **Return** at the end of the row. You cannot edit this ID. This is an internal ID that iProcess uses to identify the parameter.

   f. In the **Required** column, click the cell to display a check if you always want this parameter to be mapped.

   g. In the **Allowed Values** column you can pre-define a list of values that can be selected when the sub-procedure mapping is performed. Click **Edit > Allowed Values** to define your values. See Defining Valid Values for a Sub-Procedure Input Parameter on page 20 for more information.

   h. Press **Return** to start a new row and enter details about any other parameters you require.

6. Define the Output parameters in the same way using the **Output Parameters for Sub-Procedures Using this Template** table.

The **Required** and **Allowed Values** columns do not appear for Output Parameters because they are not needed for Output parameter mapping.

7. Click **Procedure** > **Save** to save the template.

Before you can create any sub-procedures that are based on this template or define sub-procedure calls based on this template, you must release it. See "Releasing a Version of a Procedure" in *TIBCO iProcess Modeler Procedure Management* for more information about releasing a template.

Editing a Sub-Procedure Parameter Template

To edit a parameter template:

1. Double-click the parameter template from the Procedure Manager window list.
2. Using the tables in the chart display, make any changes you want to the parameter list and save the parameter template.

If you are using version control for your procedures, see "Using Version Control" in *TIBCO iProcess Modeler Procedure Management* for more information about managing versions of procedures and templates.

Removing a Sub-Procedure Parameter Template

If you no longer use a parameter template, you can delete it from the list of procedures. The effects of deleting a template being used by sub-procedures, dynamic sub-procedure calls or graft step calls are:

- Dynamic sub-procedure calls and graft steps will continue to work. This is because the parameter definitions are copied from the parameter template into the step's IO parameters at definition time. They are not referenced at run-time.

However, when you next edit the call step, a warning message is displayed informing you that the template cannot be found. You will need to choose another template and base the sub-procedures on the new template to define another valid call.

- Sub-procedures based on a deleted template display a warning message on opening but the IO parameters are left intact.

To delete a sub-procedure parameter template:

1. Select the parameter template in the Procedure Manager window.
2. Click **Procedure Management** > **Delete**.

Chapter 5 Calling a Dynamic Number of Sub-Procedures

This chapter describes how to define a dynamic call to multiple sub-procedures from the main procedure. Defining a dynamic call to sub-procedures is useful for when you do not know which sub-procedures need to be run at procedure definition time. It is only when a case of the procedure is run that iProcess knows which sub-procedures will be started.

Before creating a dynamic call to multiple sub-procedures, you need to have created a *sub-procedure parameter template* and one or more *array fields* - see Defining Sub-Procedure Parameter Templates on page 49 and Using Array Fields on page 39.

Topics

Why Use a Dynamic Call to Sub-Procedures

You can create dynamic procedures where specific sub-procedures are started according to case data that is entered as a case of a procedure progresses. The following are some business reasons for using dynamic calls to sub-procedures:

- An order fulfillment process.

  For example, an order is received for a range of products. In some cases multiple requests for the same product are made. For each product there might be a different sub-process that is required to manage the production/procurement of the product. So for each product type a different sub-procedure might be started or in some cases multiple instances of the same sub-procedure might be started (or a mixture of the two).

- A telecoms process.

  You may have a process for ordering or upgrading a mobile phone where you have sub-processes that can be run such as order phone accessories, determine the mobile operator, organize insurance, cancel previous phone details, transfer number etc.

- A mortgage application process.

  You can have the main procedure related to the actual mortgage application and numerous sub-procedures that are called along the way for processes such as credit checks and supervisor approvals. However, at some point while the application is progressing, you might need to start other sub-processes to offer other products like life insurance, property insurance and buildings insurance. In this case a dialog with the customer might result in a number of sub-procedures being started.

How Does a Dynamic Call to Multiple Sub-Procedures Work

When iProcess processes a dynamic sub-procedure call step, it looks at the array field that has been defined for the sub-procedures to start (defined in the **Sub-Procedure Name Array** field on the **Call Definition** tab). This array field may contain no data (i.e. no sub-procedures need to be started) or multiple data elements (i.e. multiple sub-procedures need to be started).

If multiple sub-procedures are started, iProcess keeps track of what data elements need to be passed to each sub-procedure field in each sub-procedure. It does this using the same index from the sub-procedure name array for each array field mapped as an input or output.

Example of Designing a Procedure with Dynamic Multiple Sub-Procedures

The following example describes how a procedure is designed so that one or more sub-procedures are run depending on information that is entered on the first few work items of the case.

Brief

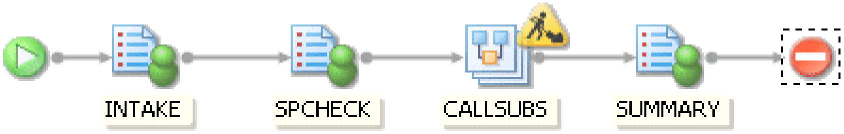
Design a procedure so that a hospital patient admission process can start a number of sub-processes depending on the information supplied by the doctor/patient. The following sub-processes may need to be started:

- x-ray
- blood test
- book a bed

Not all of the sub-processes need to be run for every patient.

Example iProcess Procedure

The procedure might be defined as follows:



Running a Case of the Intake Procedure

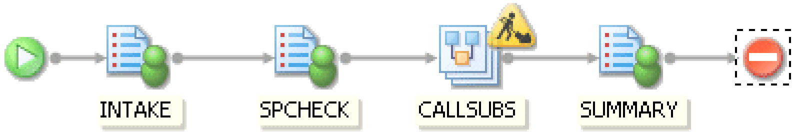
1. The first step (**intake**) requires the nurse to enter some basic information about the new patient such as name, address and the medical condition(s) they have.
2. The next step (**spcheck**) relies on a supervisor to check the details and make a choice as to what else needs to be done as part of the patient’s intake. They may require an x-ray or the booking of a bed.
3. The details from this step populate an array field called SPROCS with sub-procedure names e.g. if an x-ray is required, the sub-procedure name for the x-ray process is added.

4. The next step is the dynamic call to the sub-procedures. The call looks at the array field to find out what sub-procedures to start. In this case, just an x-ray is required so there is only one data element in the array (**xray**). The **xray** sub-procedure is started which delivers work items to the x-ray department and results in a booking for the patient. The booking date and time is returned back to the main procedure. The last work item displays a summary of the patient's admission information along with any new booking information that has been entered.

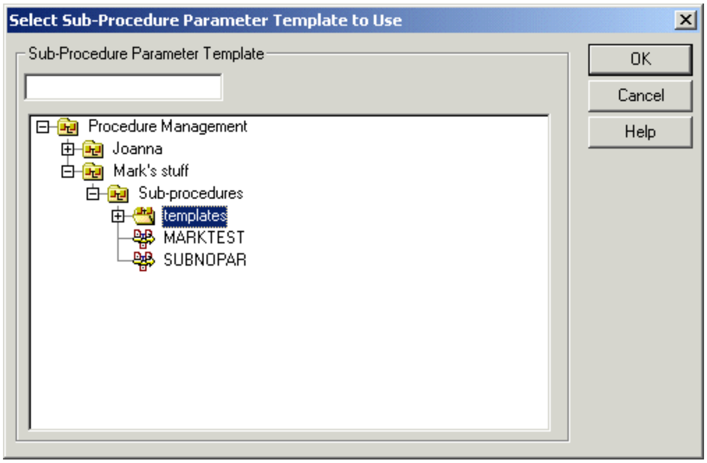
Defining a Dynamic Call to Multiple Sub-Procedures

To add a dynamic call to multiple sub-procedures:

1. Select the dynamic sub-procedure tool and click where you want to place the object in the TIBCO iProcess Modeler chart.



The **Select Sub-Procedure Parameter Template To Use** dialog is displayed.



Select Sub-Procedure Parameter Template to Use

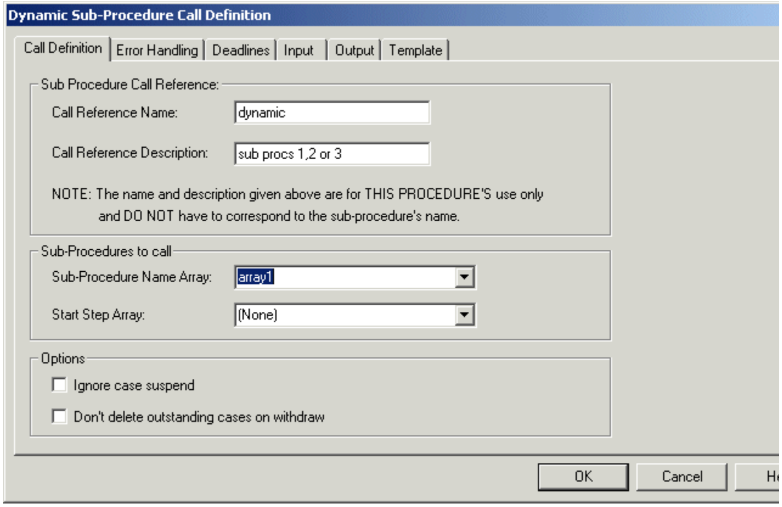
Sub-Procedure Parameter Template

```
⊟ Procedure Management
   ⊞ Joanna
   ⊟ Mark's stuff
      ⊟ Sub-procedures
         ⊞ templates
           MARKTEST
           SUBNOPAR
```

2. Select a sub-procedure parameter template to base the dynamic sub-procedure call on. This determines what sub-procedures will be valid to call from this step at run-time. Click **OK**.

Only released parameter templates are available for selection. Unreleased templates are grayed out.

The **Dynamic Sub-Procedure Call Definition** dialog is displayed.



Call Reference Name: dynamic

Call Reference Description: sub procs 1,2 or 3

NOTE: The name and description given above are for THIS PROCEDURE'S use only and DO NOT have to correspond to the sub-procedure's name.

Sub-Procedure Name Array: array1

Start Step Array: (None)

□ Ignore case suspend

□ Don't delete outstanding cases on withdraw

3. In the **Call Reference Name** field, enter a name for the dynamic sub-procedure call.
4. In the **Call Reference Description**, enter a description for the call.
5. From the **Sub-Procedure Name Array** drop-down list, select the array field that will be used to determine the sub-procedure names to call. These will be defined at run-time when the case is run because the array field will contain dynamic data. One sub-case for each element in the array will be started.
6. (Optional) From the **Start Step Array** drop-down list, select the array field that determines which step each sub-procedure will start at.

   The default start step for all the sub-procedures called is the first step in the procedure.
7. Click the **Ignore case suspend** check box if you want the step to be processed as normal while a case is suspended by an SPO or SAL application.

   If **Ignore case suspend** is not checked (the default option), the step is *not processed* while the case is suspended.

Cases can only be suspended and re-activated from an TIBCO iProcess Objects or SAL application. Audit trail messages indicate whether a case is active or suspended. See *TIBCO iProcess Objects Programmer's Guide* for more information about suspending cases.

8. (Optional) Select the **Don't delete outstanding cases on withdraw** option. If this option is selected, and the deadline on an outstanding step expires:

   — the deadline actions are processed.

   — the step remains outstanding (the step remains in the workqueue or the sub-procedure case is not purged).

   — when the step is released (or the sub-procedure case completes) the normal release actions are not processed but the case field data associated with the release step (e.g. the field values set in a normal step whilst in a work queue or the output parameters of a sub-case) is applied to the main case data.

9. (Optional) Click the **Error Handling** tab to define the error handling conditions that are applied if there are problems with the dynamic sub-procedure call. See Troubleshooting Dynamic Sub-Procedure Calls on page 76 for more information.

10. (Optional) Click **Deadlines** to set a deadline on this step. See "Defining a Deadline" in *TIBCO iProcess Modeler Basic Design*.

11. To define the input and output parameter mappings for this call, see Defining Input and Output Mappings Using a Sub-Procedure Parameter Template on page 64. The sub-procedure parameter template pre-fills the parameters that you can map.

12. Click **OK** to save your settings. To edit your dynamic sub-procedure call settings at any time - see Editing a Dynamic Sub-Procedure Call Step on page 75.

    If the sub-procedure array field is not selected, the step will be marked as incomplete and so the main procedure will also have a status of incomplete.

Defining Input and Output Mappings Using a Sub-Procedure Parameter Template

This section describes how to define the input and output mappings for a dynamic sub-procedure call. The parameter template defines the parameters that can be used for mapping.

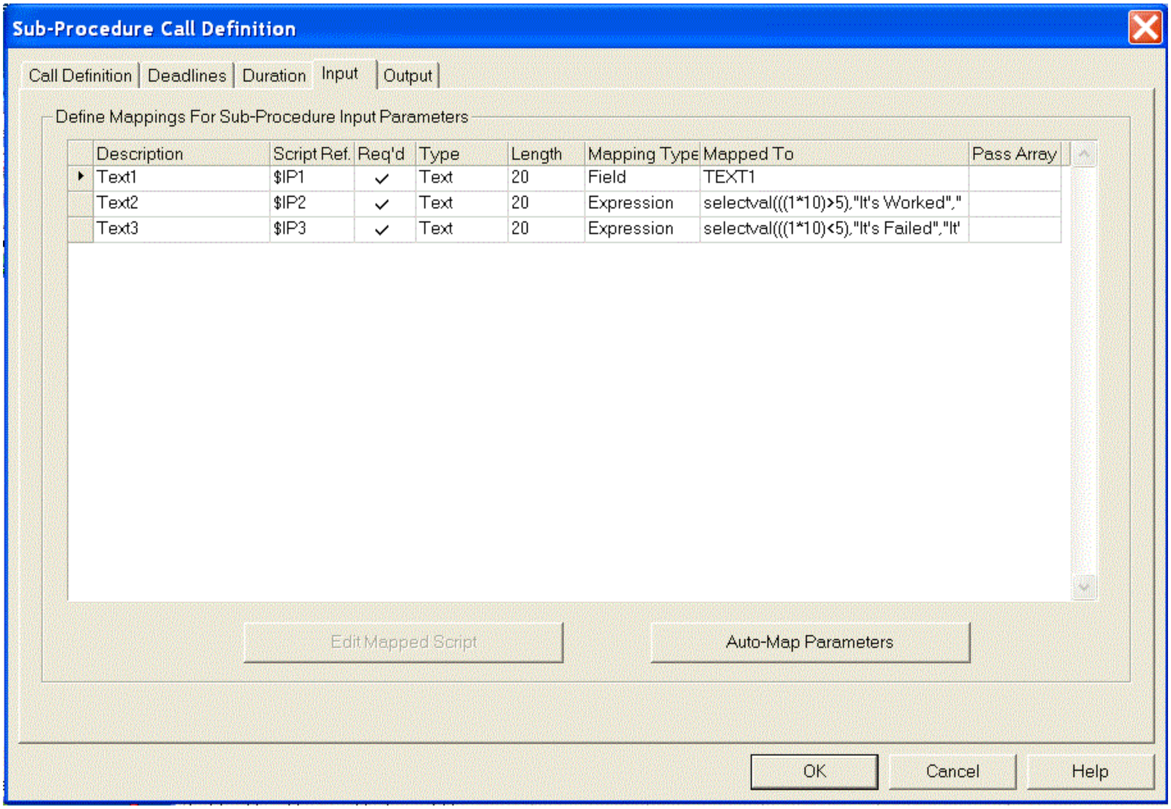
Defining Input Parameter Mappings

After defining the dynamic sub-procedure call details (see Defining a Dynamic Call to Multiple Sub-Procedures on page 62), you need to define how the sub-procedure input parameters will be mapped to the main procedure.

To map sub-procedure input parameters to the main procedure:

1. On the **Dynamic Sub-Procedure Call Definition** dialog, click **Input**.

The Input tab is displayed.



Define Mappings For Sub-Procedure Input Parameters

Description	Script Ref.	Req'd	Type	Length	Mapping Type	Mapped To	Pass Array
Text1	$IP1	✓	Text	20	Field	TEXT1	
Text2	$IP2	✓	Text	20	Expression	selectval(((1*10)>5),"It's Worked","	
Text3	$IP3	✓	Text	20	Expression	selectval(((1*10)<5),"It's Failed","It'	

The list of sub-procedure input parameters defined in the parameter template is displayed. You can map these to appropriate main procedure fields or define an expression or script to map the parameter to.

Parameter to Field Mapping:

To perform a simple parameter to field mapping, do the following:

1. Select the sub-procedure input parameter. In the Type column, select **Field**.
2. In the Mapped To column, select a field from the main procedure to map the sub-procedure parameter to.

Mapping Complete Arrays Into Sub-procedures:

To pass complete arrays into sub-procedures, do the following:

1. Click the blank area in the Pass Array column corresponding to the sub-procedure input parameter row. A checkbox appears. Check the checkbox.
2. In the Mapped To column, select a field from the main procedure to map the sub-procedure parameter to.

If you uncheck the checkbox in the Pass Array column in the Output tab, you can only transfer the currently indexed item in the array field to the sub-case.

To pass complete arrays into and out of sub-procedures, see Example of Passing Complete Arrays into and out of Sub-procedures on page 71 for more information.

Mapping a Parameter to an Expression:

To map a sub-procedure parameter to an expression, do the following:

1. Select the sub-procedure input parameter. In the Mapping Type column, select **Expression**.
2. In the Mapped To column, enter a valid iProcess expression to map the parameter to. For example, if the main procedure contains two fields for a person’s name (FORENAME and SURNAME), you can create an expression to join the two values so that the single name is mapped to a NAME sub-procedure input parameter. See *TIBCO iProcess Expressions and Functions Reference Guide* for more information about expressions and functions.

Mapping a Parameter With Pre-Defined Values:

If a parameter has had values previously defined for it, the following procedure applies:

1. Select the sub-procedure parameter. Click in the **Mapping Type** column and the Preset Values string are inserted if there are pre-defined values.
2. In the Mapped To column, select a value from the list.
3. Click the **OK** button to exit the Sub-Procedure Call Definition dialog.

Mapping a Parameter to a Script:

To perform extra manipulation of the contents of a parameter, you can map a parameter so that its content is calculated as a result of running the script.

To perform a parameter to script mapping, do the following:

1. Select the sub-procedure input parameter. In the Mapping Type column, select **Script**.
2. In the Mapped To column select a public script if any have been created or choose **<private script>** to create a script private to this sub-procedure mapping. You can then click **Edit Mapping Script** to open the Script Editor and define your script. See Creating a Private Script on page 88 for more information about creating and using private scripts.
3. If you want to edit a script, select the parameter containing the script and click **Edit Mapping Script.** The Script Editor is displayed and you can modify the script.

Defining Output Parameter Mappings

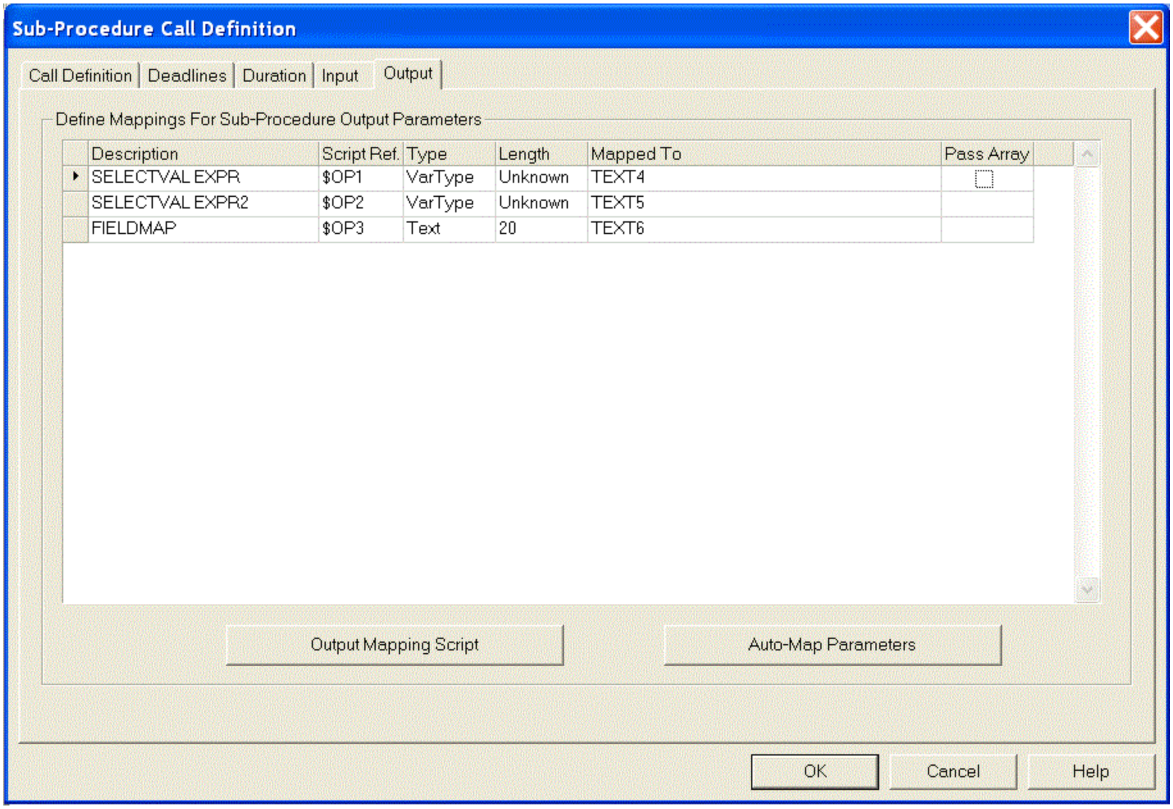
After defining the dynamic sub-procedure call details (in Defining a Dynamic Call to Multiple Sub-Procedures on page 62), you need to define what output parameters will be mapped to the main procedure.

Sub-procedure output parameters can be mapped to fields in the main procedure and you can also run a private script after the data mappings have been processed to perform further manipulation of the output values.

To map sub-procedure output parameters to the main procedure:

1. On the Dynamic Sub-procedure Call Definition dialog, click **Output.**

The Output tab is displayed.



Description	Script Ref.	Type	Length	Mapped To	Pass Array
SELECTVAL EXPR	$OP1	VarType	Unknown	TEXT4	□
SELECTVAL EXPR2	$OP2	VarType	Unknown	TEXT5	
FIELDMAP	$OP3	Text	20	TEXT6	

A list of the sub-procedure output parameters that have been pre-defined in the sub-procedure parameter template is displayed. You need to map these to the appropriate fields in the main procedure.

Parameter to Field Mapping:

To perform a simple field to field mapping, do the following:

1. Click in the Mapped To column and select a main procedure field to map the parameter to. Only fields of the same field type are listed e.g. numeric or date.
2. Continue to map any other sub-procedure parameters to main procedure fields as required.

Mapping Complete Arrays out of Sub-procedures:

To pass complete arrays out of sub-procedures, do the following:

1. Click the blank area in the Pass Array column corresponding to the sub-procedure output parameter row. Then a checkbox appears. Check the checkbox.
2. In the Mapped To column, select a field from the main procedure to map the sub-procedure parameter to.

To pass complete arrays into and out of sub-procedures, see Example of Passing Complete Arrays into and out of Sub-procedures on page 71 for more information.

When a single field in a sub-procedure is mapped to an array field in a main procedure and you check the checkbox in the Pass Array column in the Output tab, the field in the sub-procedure returns nothing to the array field in the main procedure. This is because selecting Pass Array causes the non-array data to be discarded.

If you check the checkbox in the Pass Array column in the Output tab for a dynamic sub-call, the last returned array overwrites all the previous arrays returned from the sub-procedure.

Running an Output Mapping Script

To provide extra manipulation of output values, you have the option to execute a private script on completion of the output mappings. The script can see the sub-procedure output values using keywords with the format $OP*n* or $OPT*n* where:

- *n* is a positive integer that is automatically assigned by iProcess to each output parameter.
- **T** denotes that the parameter has been inherited from a template.

The script can also see, and assign values to, the main procedure fields. The following is an example of using a script to manipulate the output values and assign them to fields in the main procedure:

```
IF ($OPT1= “add”) THEN
SPVAL:=$OPT2+$OPT3
ELSE IF ($OPT1=“subtract”) THEN
SPVAL:=$OPT2-$OPT3
ELSE IF ($OPT1=“multiply”) THEN
SPVAL:=$OPT2 * $OPT3
ELSE IF ($OPT1= “divide”) THEN
SPVAL:=$OPT2 / $OPT3
ENDIF
FULLNAME:=SCRIPT (“BLDNAME”, 2, $OPT11, $OPT6, $OPT7)
```

Therefore, if the sub-procedure field associated with $OPT1 has a value of “subtract”, then the SPVAL output parameter will have a value of $OPT2-$OPT3.

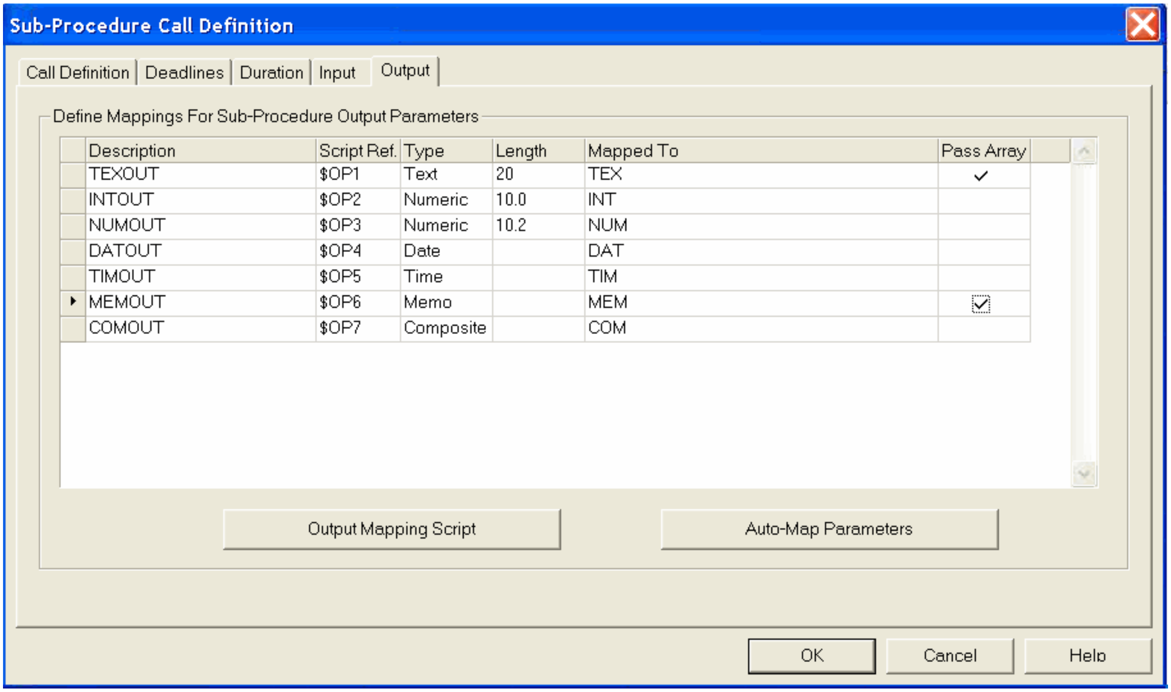
Example of Passing Complete Arrays into and out of Sub-procedures

Complete arrays can be passed into and out of sub-procedures. The following example shows how to pass complete arrays into and out of sub-procedures.

To pass complete arrays into and out of sub-procedures, do the following:

1. Prepare the MAINA main procedure embedded with the SUBNA sub-procedure:
   - the steps of MAINA and SUBNA are defined with Array fields, for example, TEX, INT, NUM, DAT, TIM, MEM , and COM.
   - the steps of MAINA and SUBNA are defined with the SW_GEN_IDX field.
   - the input and output mappings for case data are defined.
2. Open the main procedure in TIBCO iProcess Modeler, double-click the sub-procedure to open the Sub-Procedure Call Definition dialog.
3. Click the **Input** tab, and click the **Pass Array** field of the items (for example, TEXIN, MEMIN). Check the checkbox in the Pass Array column.

4. Click the **Output** tab, and click the **Pass Array** field of the items (for example, TEXOUT, MEMOUT). Check the checkbox in the Pass Array column.



Description	Script Ref.	Type	Length	Mapped To	Pass Array
TEXOUT	$OP1	Text	20	TEX	✓
INTOUT	$OP2	Numeric	10.0	INT	
NUMOUT	$OP3	Numeric	10.2	NUM	
DATOUT	$OP4	Date		DAT	
TIMOUT	$OP5	Time		TIM	
MEMOUT	$OP6	Memo		MEM	☑
COMOUT	$OP7	Composite		COM	

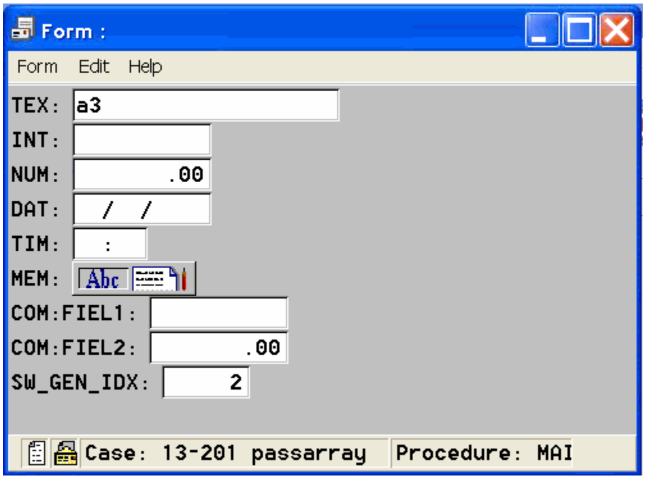
5. Click the **OK** button to exit the Sub-Procedure Call Definition dialog.

6. Click the **Case Start** button, and fill in the Case Start dialog. Click the **Start** button to start the case.

7. In the Form: dialog, type values for the Array fields, as shown in the following table.

SW_GEN_IDX	TEX Field	MEM Field
0	a1	q1
1	a2	q2
2	a3	q3

The following figure shows the Form: dialog when entering 2 in the SW_GEN_IDX field.

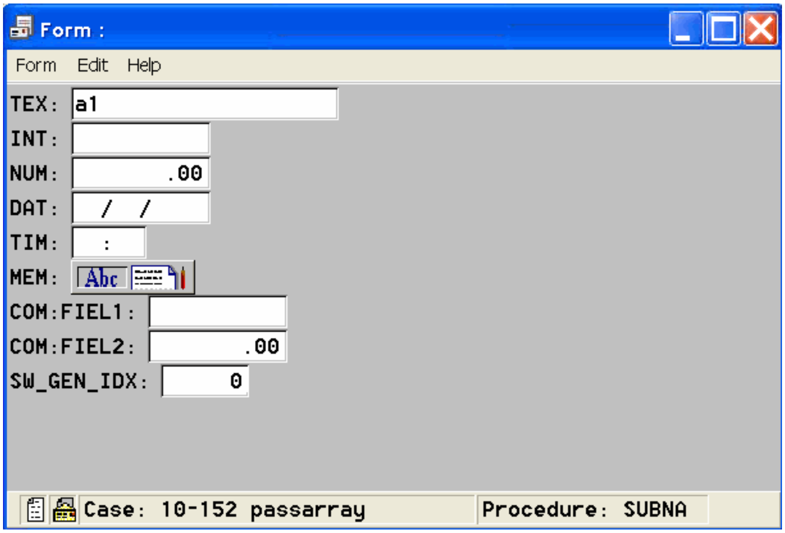


TEX: a3
INT:
NUM: .00
DAT: / /
TIM: :
MEM:
COM:FIEL1:
COM:FIEL2: .00
SW_GEN_IDX: 2

Case: 13-201 passarray Procedure: MAI

Click the **Release** button to continue the case.

8. Double-click the work item in the list to check the case data passed into the sub-procedure, as shown in the following figure.



TEX: a1
INT:
NUM: .00
DAT: / /
TIM: :
MEM:
COM:FIEL1:
COM:FIEL2: .00
SW_GEN_IDX: 0

Case: 10-152 passarray Procedure: SUBNA

The complete arrays are passed into the sub-procedure.

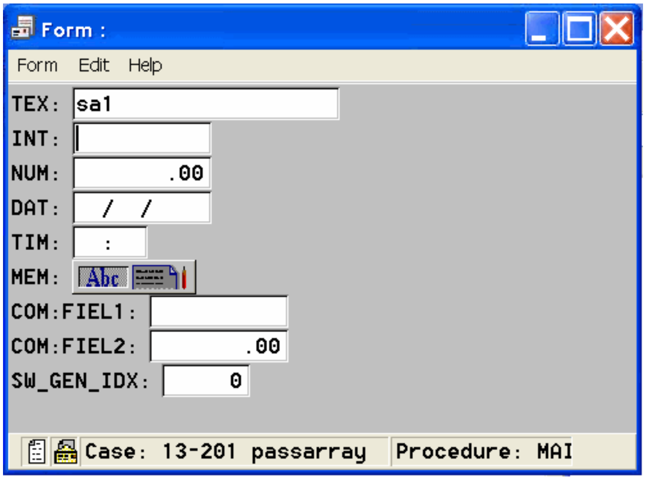
Type **1** in the SW_GEN_IDX field, the TEX field value changes to a2, and the MEM field value changes to q2. Type **2** in the SW_GEN_IDX field, the TEX field value changes to a3, and the MEM field value changes to q3.

9. Update the Array fields values as shown in the following table.

SW_GEN_IDX	TEX Field	MEM Field
0	sa1	sq1
1	sa2	sq2
2	sa3	sq3

Click the **Release** button to pass the arrays from the sub-procedure to the main procedure.

10. Double-click the work item in the list to check the case data passed into the main procedure, as shown in the following figure.



TEX: sa1
INT:
NUM: .00
DAT: / /
TIM: :
MEM:
COM:FIEL1:
COM:FIEL2: .00
SW_GEN_IDX: 0
Case: 13-201 passarray
Procedure: MAI

The complete arrays are passed into the main procedure.

Type **1** in the SW_GEN_IDX field, the TEX field value changes to sa2, and the MEM field value changes to sq2. Type **2** in the SW_GEN_IDX field, the TEX field value changes to sa3, and the MEM field value changes to sq3.

Editing a Dynamic Sub-Procedure Call Step

To edit the dynamic sub-procedure call details, do the following:

1. Double-click the dynamic sub-procedure call step in your procedure.
2. Make your changes to call details, deadline settings, and/or input and output settings. Click **OK** to save your new settings.

You can also select the call step and right-click to choose to edit the call details (**Multiple Sub-Procedure...**) or the deadline settings (**Deadlines...**)

Troubleshooting Dynamic Sub-Procedure Calls

This section describes some error handling parameters you can set to help troubleshoot problems with dynamic sub-procedure calls.

Stopping the Process if an Error Occurs

The dynamic sub-procedure call definition provides parameters that you can use to stop the business process if a specific error occurs. If these are not selected, the sub-cases and process will continue but you may have errors in the case data. On the **Dynamic Sub-Procedure Call Definition** dialog, click the **Error Handling** tab and choose one or more of the following:

— **Sub-procedure names are invalid**

Select this option to stop the process if iProcess cannot find one of the sub-procedures it needs to call.

— **Sub-procedures that do not use the same sub-procedure parameter template.**

Select this option to stop the process if iProcess finds parameters that are not in the sub-procedure parameter template being used by the dynamic sub-procedure call.

— **Sub-procedures that use different versions of the same sub-procedure parameter template.**

Select this option to stop the process if iProcess finds some parameters that are not valid for the version of the template being used for this call.

Returning an Error Status

You can use an array field to store error return values related to dynamic sub-procedure calls. You can use the return values to help troubleshoot a problem.

On the **Error Handling** tab, select a Return Status Array field which can be populated with a return value if an error occurs.

The return values you can get are:

SW_NA	Sub-case start is not attempted.
1	Sub-case has started successfully.
2	Sub-case has completed successfully.
-1	Error starting sub-case with invalid sub-procedure name.
-2	Error starting sub-case because the call step and sub-procedure use different parameter templates.
-3	When starting the sub-case, the call step and sub-procedure use different parameter templates.
-4	The same error as -3 but this applies to sub-case completion and therefore applies to output mapping.

If the call step fails for a reason which you have a halt on error option set but the transaction is aborted, the return status array is not set.

Chapter 6 Creating Scripts

This chapter describes the use of scripts within iProcess procedures. A script is a collection of statements that can be called from various places within iProcess (for example, when a field is opened). Scripts are useful when more than one iProcess expression is needed to achieve a command's requirements.

There are two types of scripts in iProcess:

- Public

  These are named scripts that are available throughout the entire procedure. For example, you can create a script called EURO to convert pounds to euros, which you can call at various points from your procedure.

- Private

  These are unnamed scripts only associated with the current sub-procedure parameter mapping. This type of script is created when you perform the sub-procedure parameter mapping. For example, you can concatenate a SURNAME and FORENAME field before assigning it back to the NAME field in the main procedure.

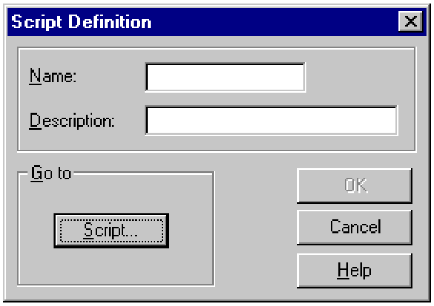
Topics

Creating a Script

To create a script:

1. Open your procedure in TIBCO iProcess Modeler and click the script tool .
2. Move the cursor to the position on TIBCO iProcess Modeler procedure layout where you want to place the script tool and click.

   The **Script Definition** dialog is displayed.



3. Enter your script name (up to 8 characters).
4. Enter your script description (up to 24 characters).
5. Click **Script** and you are taken to the script editor. See Using the Script Editor on page 81 for more information.
6. Enter your script and click **Script > Exit**. You are asked if you want to save your changes. Click **Yes**.

   The script editor then checks the syntax of your script and if any errors are found, you are asked if you want to correct them. See Script Error Checking on page 83 for more information.

Using the Script Editor

Using the script editor you can define the statements, conditions and loops that make up the script. You can also edit, save and delete scripts.

Defining Script Statements

Script statements are entered one per line, and most consist of assignment expressions and calls to functions which perform an action, such as SERVERRUN and WINRUN, which call external programs on the server and the client, respectively. Comments can be included; the rest of the line from a semicolon (;) is ignored. For example:

```
; The next line runs a program on the Server
serverrun ("/home/myprog", 0)
limit := 10000 ; set maximum estimate
```

See *TIBCO iProcess Expressions and Functions Reference Guide* for full details of allowable expressions.

Using Conditions in Scripts

You can include conditions in scripts so that whether or not statements are executed depends upon the values of fields or the results of function calls, etc. This is done as follows:

```
IF Condition 1
True Block
ELSEIF Condition 2
Elseif Block
ELSE
Else Block
ENDIF
```

where:

- *Condition 1* and *Condition 2* are expressions which return type boolean, such as `REPLY="Yes"`.
- *True Block* is a sequence of script statements which are executed if *Condition 1* is true.
- *Elseif Block* is a sequence of script statements which are executed if *Condition 1* is false and *Condition 2* is true.
- *Else Block* is a sequence of script statements which are executed if *Condition 1* and *Condition 2* are both false.

Note that:

- The sequence `ELSEIF...`*Elseif Block* is optional and may be repeated a number of times.
- The sequence `ELSE...`*Else Block* is optional.
- You may freely nest complete IF ... ENDIF or IF .. ELSE ..ENDIF constructs inside one another (up to a maximum of 20 deep).

Using Loops in Scripts

Using loops in scripts enables sequences of statements to be repeated over and over as long as a condition is satisfied. This is achieved as follows:

```
WHILE Condition
While Block
WEND
```

Condition is any expression which returns type boolean, such as `REPLY="Yes"`.

While Block is a sequence of script statements that are executed repeatedly for as long as *Condition* is true. (The conditional expression is re-evaluated each time.)

Using Parameters in Scripts

A script can contain iProcess variables with the names in the format $ARG*n* where *n* is a positive integer. To pass parameter values into a script, the SCRIPT function must be used to execute the script.

See “Script” in *TIBCO iProcess Expressions and Functions Reference Guide* for more information and an example.

Using Exit Statements

You can exit a script prior to the end by using a statement consisting of the single word EXIT. This would normally be used inside a condition. For example:

```
IF REPLY = "YES"
EXIT
ENDIF
```

Combining Constructs

Statements in conditions and loops may consist of other conditions and loops, i.e., structures may be nested freely.

Defining a Field

You can define fields from within the script editor in the same way as in a form. See “Defining Fields” in *TIBCO iProcess Modeler Basic Design* for further information on how to do this.

Saving Changes

To save the current script file, choose **Script > Save Changes**.

This does NOT update the Procedure Definition; to do this, choose **Procedure > Save** from the **iProcess Modeler** window.

Script Error Checking

On Saving, the script is checked for errors.

- **Structure** errors, for example an IF without a corresponding ENDIF, are reported in a message box. On clicking **OK**, you are returned to the **Script Editor** window to correct the error.

  You cannot leave the **Script Editor** window until you have corrected all structure errors.

- **Expression** errors are also reported, but in this case you have the option of returning to correct the error, ignoring it and continuing to check for others, and ignoring them all. If you choose to correct the error, you are shown the line on which the error occurred. Scripts with expression errors will run, but may not produce the desired result.

Exiting the Script Editor

To exit the **Script Editor** window, choose **Script > Exit**. If the current script has been changed, you are prompted to **Save** it first.

Editing a Script

To edit a script, right click on the script in your iProcess Modeler layout and select **Script**. You are taken to the script which you can then edit and amend as required.

Deleting a Script

To delete a script, select the script on the iProcess Modeler layout and press <**Delete**>.

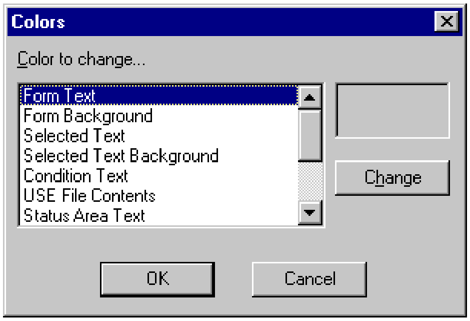
Configuring the Script Editor

The Script Editor can be configured in a similar way to the Step Definer. From within your script, click **Script > Setup** to see the following options:

- Colors
- Dynamic Scroll
- Tabs
- Nesting Level

Changing Colors

The colors that are used for the text and background in different parts of the step can be changed through the standard Windows dialog.



Click **Change** to display the color palette, make your selection and click **OK**.

Changes are implemented across all procedures and are saved between iProcess Modeler sessions.

Using Dynamic Scroll

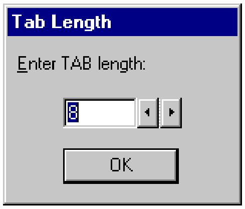
When **Dynamic Scroll** is on, the contents of the window moves as you drag the scroll bar up or down. When it is off, the contents of the window will not move until you release the scroll bar.

Click on **Dynamic Scroll** to select it and a check mark appears to the left of the list. Click again to de-select it.

This option is per script only and is not saved between iProcess Modeler sessions.

Setting Tabs

To change the distance, in characters, between tabs, select **Tabs** from the **Setup** menu. Enter the distance you want and click **OK**.



Tab Length

Enter TAB length: 8

The maximum tab length is 16 characters and the change is implemented across all procedures and is maintained between iProcess Modeler sessions.

Displaying Nesting Levels

When you use conditions in your script you can nest up to 20 levels. To make it easier to follow when working on your script, you can choose to show the nesting levels either numerically or graphically.

Select **Nesting Level** from the **Setup** menu on the script and choose **Numeric** or **Graphical** (or both). See “Nesting Level” in *TIBCO iProcess Modeler Basic Design* for examples of how this will appear.

This option is per script and is not saved between iProcess Modeler sessions.

Calling Scripts

A script is run by a Form Command or Field Command using the CALL or SCRIPT functions - see *TIBCO iProcess Expressions and Functions Reference Guide*.

You can also call a script from within a script using the CALL and SCRIPT functions. This enables you to recursively call other scripts up to a maximum limit defend by the MAX_SCRIPT_CALL_DEPTH parameter in the *SWDIR*\etc\staffcfg file. See “Tuning iProcess Engine Using SWDIR\etc\staffcfg Parameters” in *TIBCO iProcess Engine Adeministrator’s Guide* for more information about this parameter.

Creating a Private Script

Private scripts are used for manipulating input or output parameters during sub-procedure parameter mapping. A private script can only be created from the sub-procedure mapping dialog - see Defining Input and Output Parameter Mappings on page 29.

1. On the **Input** or **Output** tab of the **Sub-Procedure Call Definition** dialog, click **Edit Mapping Script** or **Output Mapping Script** respectively to open the Script Editor.
2. Type in your script as required. Use *TIBCO iProcess Expressions and Functions Reference Guide* as a reference to the functions and expressions you can use.
3. Click **Script** > **Exit** to save and close your private script. It is automatically assigned an internal name such as $PS000000.

Example of Using a Private Script

The following example demonstrates how to manipulate data from a sub-procedure before transferring it to the main procedure i.e. in an output mapping script. The main procedure fields SPVAL (numeric) and FULLNAME (text) are assigned from various combinations of sub-procedure output parameters $OPT1, $OPT2, $OPT3 (numeric) and $OPT6, $OPT7 and $OPT11 (text, numeric, string, etc.). The $OPT*n* values are keywords that relate to sub-procedure template output parameters. $OP*n* relate to sub-procedure output parameters that have been manually pre-defined.

```
IF ($OPT1= “add”) THEN
SPVAL:=$OPT2+$OPT3
ELSE IF ($OPT1=“subtract”) THEN
SPVAL:=$OPT2-$OPT3
ELSE IF ($OPT1=“multiply”) THEN
SPVAL:=$OPT2 * $OPT3
ELSE IF ($OPT1= “divide”) THEN
SPVAL:=$OPT2 / $OPT3
ENDIF
FULLNAME:=SCRIPT (“BLDNAME”, 2, $OPT11, $OPT6, $OPT7)
```

Therefore, if the sub-procedure field associated with $OPT1 has a value of “subtract”, then the SPVAL output parameter will have a value of $OPT2-$OPT3.

See Running an Output Mapping Script on page 69 for more information about the $OP*n* and $OPT*n* values.

Chapter 7 Using Case Prediction to Forecast Outstanding Work Items

This chapter describes how to set up case prediction.

Topics

About Case Prediction

Case prediction enables you to accurately forecast:

- Outstanding work items
- Expected work items

For example, if a new patient is admitted to hospital, you can forecast the work that is outstanding for the current shift and the expected work that needs to be performed during the patient's stay in hospital.

Therefore, the benefit of using case prediction in this example is improved resource handling. You can make sure the correct number of staff are available to perform the work.

Overview of Setting Up Case Prediction

The following list provides an outline of the steps you need to perform to enable case prediction:

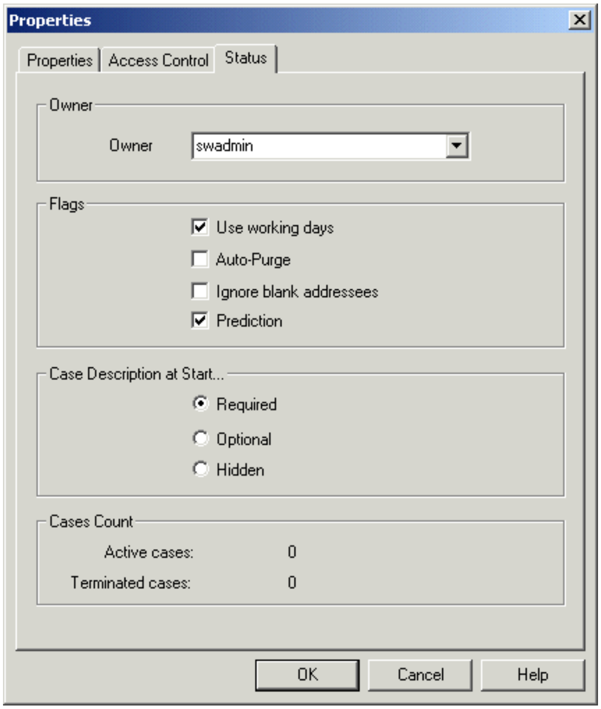
1. Enable case prediction on your iProcess Engine, see *TIBCO iProcess Engine Administrator's Guide* for more information.
2. Enable a procedure or sub-procedure to use prediction, see Enabling a Procedure or Sub-Procedure to use Case Prediction on page 92.
3. Define duration of your procedure or sub-procedure, see Defining Duration of a Procedure or Sub-Procedure on page 94.
4. Define duration of steps in your procedure, see Defining Duration of Procedure Steps on page 96.
5. Define prediction settings for conditional steps, see Defining Predicted Routes for Condition Steps on page 98.
6. Define the application that will use the prediction results, see *TIBCO iProcess Objects Programmer's Guide.*

Enabling a Procedure or Sub-Procedure to use Case Prediction

You can define which procedures/sub-procedures you want case prediction enabled for. Once it is enabled, you can use the OEM lock feature to prevent it being modified.

To enable a procedure or sub-procedure to use case prediction, you need to set the **Prediction** flag on the **Status** tab of the **Procedure Properties** dialog.

To view the **Procedure Properties** dialog, select your procedure from the Procedure Manager and select **Procedure Management > Properties**. Click the **Status** tab.



Owner

Owner: swadmin

Flags

☑ Use working days

□ Auto-Purge

□ Ignore blank addressees

☑ Prediction

Case Description at Start...

● Required

○ Optional

○ Hidden

Cases Count

Active cases: 0

Terminated cases: 0

Using Case Prediction with Sub-Procedures

If you enable case prediction for a sub-procedure, all the steps in the sub-procedure are displayed in the outstanding step list.

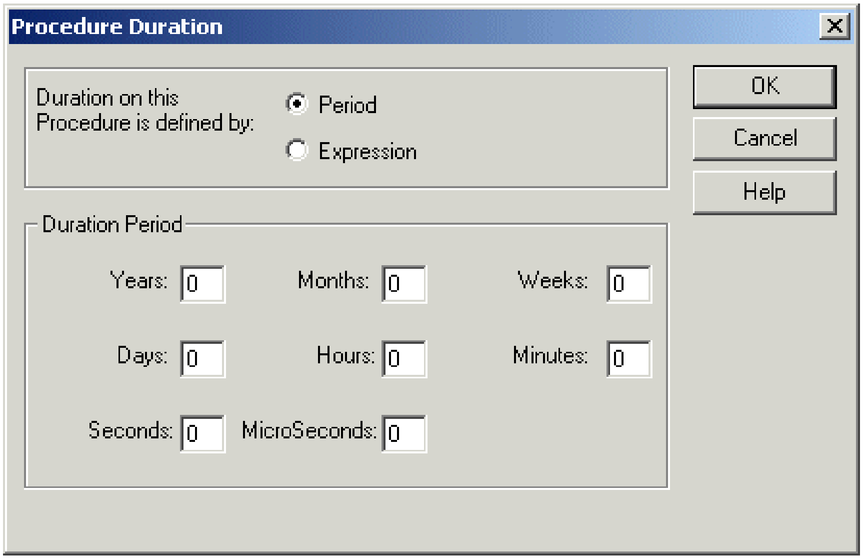
If you do not enable case prediction for a sub-procedure, iProcess calculates the total duration of all the steps in the sub-procedure. Therefore, although the sub-procedure is displayed in the outstanding step list, the steps in the sub-procedure are not displayed.

Defining Duration of a Procedure or Sub-Procedure

For iProcess to calculate the expected time of future work items, you need to define the expected duration of the procedure or sub-procedure.

1. From the **Procedure Properties** dialog, select **Prediction** and click **OK**. See Enabling a Procedure or Sub-Procedure to use Case Prediction on page 92.

2. Open the procedure and click **Procedure** > **Duration**. The **Procedure Duration** dialog is displayed.



Procedure Duration

Duration on this Procedure is defined by: ● Period ○ Expression

Duration Period

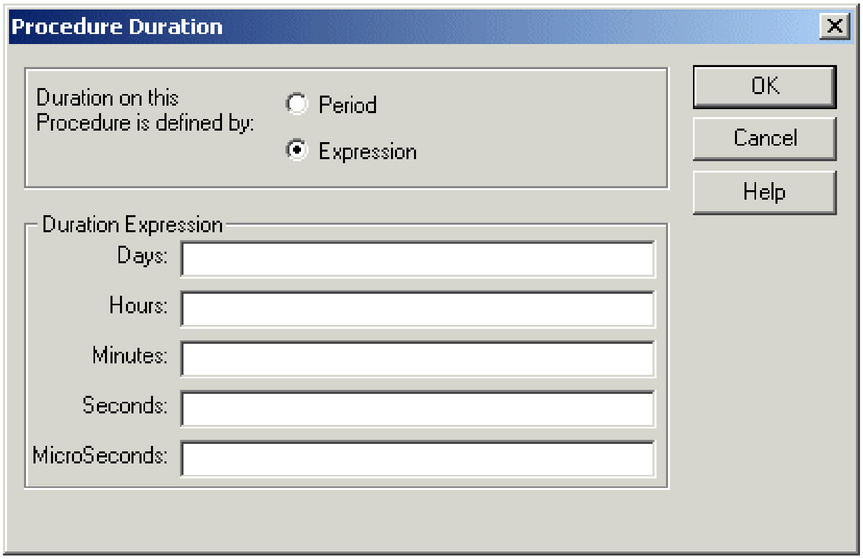
Years: 0 Months: 0 Weeks: 0

Days: 0 Hours: 0 Minutes: 0

Seconds: 0 MicroSeconds: 0

3. Select whether duration is to be based on a **Period**, such as 2 weeks, or on an **Expression**.

   If you select **Expression**, this section automatically changes to allow the input of the expression, as shown below.



Duration on this Procedure is defined by:

○ Period

● Expression

Duration Expression

Days:

Hours:

Minutes:

Seconds:

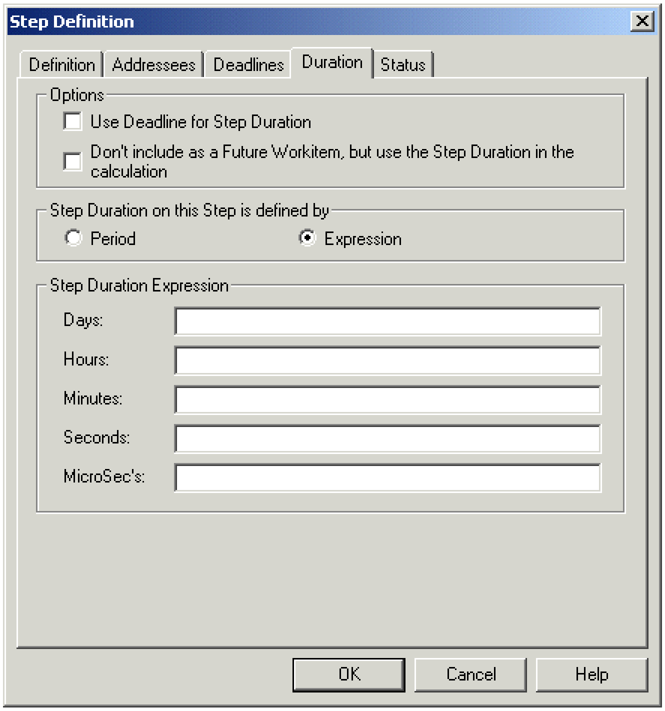
MicroSeconds:

Defining Duration of Procedure Steps

For iProcess to calculate the expected time of future work items, you need to define the expected duration of each step in your procedure/sub-procedure. Note that:

- You need to set the duration for each step in your procedure/sub-procedure.
- Only the following steps can be used in prediction:
  - Normal step
  - Event step
  - Sub-Procedure step
  - Dynamic sub-procedure call step
  - Graft step
  - EAI step

1. From the **Step Definition** window, click the **Duration** tab.



Step Definition

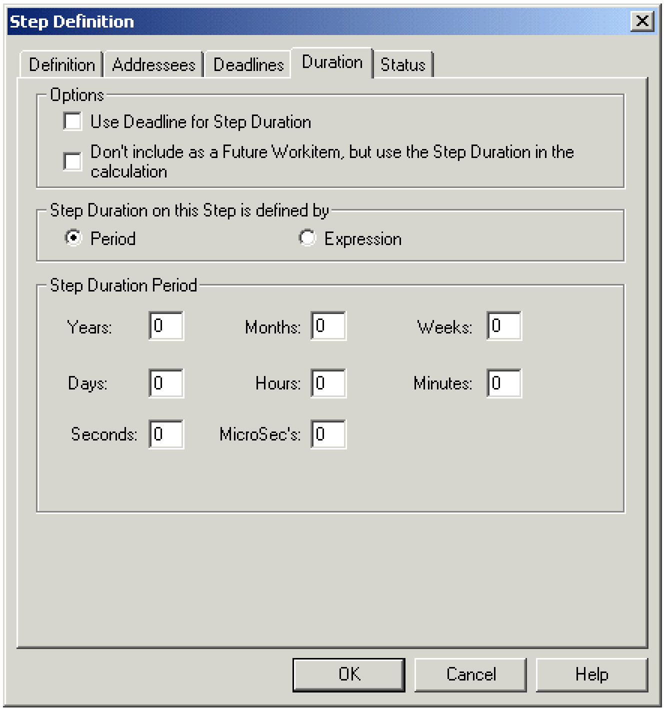
Options
☐ Use Deadline for Step Duration
☐ Don't include as a Future Workitem, but use the Step Duration in the calculation

Step Duration on this Step is defined by
○ Period ● Expression

Step Duration Expression
Days:
Hours:
Minutes:
Seconds:
MicroSec's:

2. Select whether duration is to be based on a **Period**, such as 2 weeks, or on an **Expression**.

   If you select **Period**, this section automatically changes to allow the input of the period, as shown below.



Step Definition

Options

□ Use Deadline for Step Duration

□ Don't include as a Future Workitem, but use the Step Duration in the calculation

Step Duration on this Step is defined by

● Period ○ Expression

Step Duration Period

Years:	0	Months:	0	Weeks:	0
Days:	0	Hours:	0	Minutes:	0
Seconds:	0	MicroSec's:	0		

3. (Optional) Select the **Use Deadline for Step Duration** check box. If this box is checked it enables you to set the step duration to be the same as the deadline. Therefore, if a deadline is set then the step duration will be the same as the deadline. This means you do not have to manually set the duration to be the same as the deadline.

4. (Optional) Select the **Don’t include as a future WorkItem, but use the Step Duration in the calculation** check box. If this box is checked you can exclude the step from prediction. For example, you may want to exclude broker and EAI steps because these steps are processed automatically. This means that although these steps are taken into account as part of the case prediction, they do not appear in the outstanding step list.

Defining Predicted Routes for Condition Steps

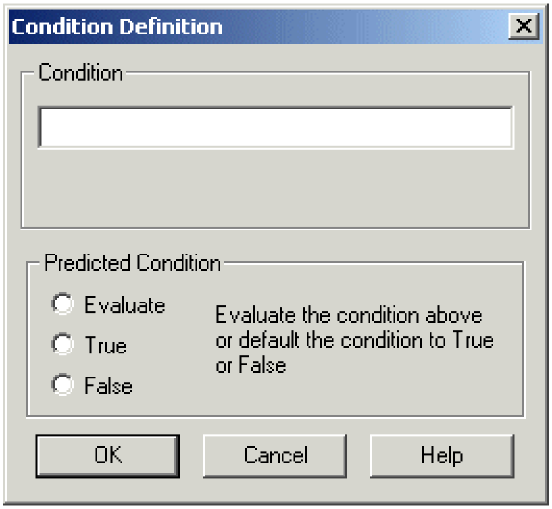
When iProcess predicts the duration of a procedure, it needs to know what path to follow through the procedure. If your procedure contains conditional steps, you can define a default or expected path that the case will follow.

Predicted conditions allow you to more accurately predict the duration of the outstanding steps. If you have a condition that usually evaluates as true then you should select the **True** check box. This means that Case Predict will always evaluate the condition to be **True** when calculating the duration of the outstanding steps.

To view the **Conditional Definition** dialog:

1. Click the condition object in the Process Definer.

   The **Condition Definition** dialog appears. See "Defining a Conditional Action" in *TIBCO iProcess Modeler Basic Design* for more information about defining conditional actions.



Evaluate the condition above or default the condition to True or False

2. Select one of the following flags:

   — **Evaluate**

   Case prediction will always evaluate the condition.

   — **True**

   Case prediction will always default the condition to true.

   — **False**

   Case prediction will always default the condition to false.

Chapter 8 Using Work Queue Parameter Fields

Work queue parameter fields are system fields that allow you to:

- define application-specific data in your procedure, such as customer name or invoice number, and
- make that data available in Work Queue Manager to display, sort or filter work queues.

This chapter explains how to use work queue parameter fields in procedures and in Work Queue Manager.

Topics

Using Work Queue Parameter Fields in Procedures

There are four work queue parameter fields available. The following table shows their default characteristics:

Name	Type	Length	Identifier
SW_QPARAM1	Text	24	WQ Parameter1
SW_QPARAM2	Text	24	WQ Parameter2
SW_QPARAM3	Text	12	WQ Parameter3
SW_QPARAM4	Text	12	WQ Parameter4

You can assign values to work queue parameter fields in the normal way, using forms, scripts or abox files.

The work queue parameter fields are defined automatically in new procedures. You can change the type and/or length of the work queue parameter fields as required, or delete them if you do not use them.

The field lengths shown in the table above are the maximum lengths that can be displayed in Work Queue Manager. You can use longer field lengths if you wish, but fields whose contents are longer than these lengths will be truncated in Work Queue Manager.

These same field lengths are also the maximum lengths that can be published in the JMS messages used for Work Queue Delta publication via JMS. (See *TIBCO iProcess Engine Architecture Guide* for a description of Work Queue Delta publication via JMS.)

Using Work Queue Parameter Fields in Work Queue Manager

Work queue parameter fields can be used in Work Queue Manager from:

- the **Work Queue Sort Criteria** dialog, to sort the order in which work items are displayed based on the contents of the field.
- the **Work Item List Filter** dialog, to filter the work items which are displayed based on the contents of the field.
- the **Display Criteria** dialog, to display the contents of the field for each work item.
- the **Find Work Item** dialog, to search for work items based on the contents of the field.

See *TIBCO iProcess Workspace (Windows) User's Guide* for more information.

Work Queue Parameter Field Identifiers

By default, the work queue parameter fields are referenced in the Work Queue Manager dialogs using the identifiers listed in the table on page 102.

However, you can redefine these identifiers, on either a system wide or per-queue basis, to provide users with a more meaningful description of the field. See "Setting Work Queue Field Identifiers" in *TIBCO iProcess Workspace (Windows) Manager's Guide* for more information about how to do this.

Making Sure Work Queue Parameter Field Definitions are Unique

iProcess does not prevent you from using the same work queue parameter fields to represent different pieces of data in different procedures. However, you should be aware that if you do this, you may confuse Work Queue Manager users if work items from these different procedures are sent to the same work queues.

For example, a company has a mortgage application procedure which uses the SW_QPARAM1 field to store **Customer Name**. Staff in the Finance department receive work items from this procedure in a **Finance** group queue, and use the WQ Parameter1 identifier to both display and sort work items by **Customer Name**.

The company now implements an expenses claim procedure which uses SW_QPARAM1 to store the expenses **Claim Number**. Staff in the Finance department also receive work items from this procedure in the same group queue. However, where they are expecting to see a **Customer Name**, they will now see an expenses **Claim Number**.

The simplest way to ensure that work queue field definitions remain unique is to use them for corporate-wide data. For example, you could stipulate that SW_QPARAM1 should be used only for **Customer Name** in any procedure that uses it. If you do use multiple definitions of work queue parameter fields it is your responsibility to resolve any potential confusions which may arise.

Chapter 9 Using Work Item Priorities and Escalation

Every step, and therefore every work item, has a priority attached to it which, when used to sort a work queue, can determine where it appears in the queue. You can change the priority of a step so that it appears above or below other items in the work queue.

You can set priorities so that the item is given a high or low priority immediately after it is processed to a queue, or it can increase or decrease in priority by a given amount after a certain period of time.

Topics

About Priorities

Every time a step is processed, iProcess automatically calculates the priority of the work item according to the information it has been given. It takes this information from:

- the **Status** tab on the **Step Definition** dialog

or if nothing is specified there, from:

- the **SW_CP_xxx** system field values

or if these are not assigned, from:

- the *SWDIR***\etc\staffcfg** file.

In addition, the priority of a work item may be changed whilst it is in a work queue using the **SW_IP_xxx** system fields.

To use priorities successfully, you need to sort the work queue by priority. See *TIBCO iProcess Workspace (Windows) User's Guide* for more information.

The following sections look at each of these and provide examples of how they can be used.

Changing the Default Priority

Every iProcess installation has a default step priority level which is used when no other priority setting has been defined. This is a configurable value and is defined by the following entry in the *SWDIR***etc****staffcfg** file. The initial value is 50.

```
WQS_DEFAULTPRIORITY,50; Default priority level
```

No escalation values can be set in the *SWDIR***etc****staffcfg** file. These must be set within the procedure as discussed in the following paragraphs.

Changing Priority at Case Level

When defining your procedure you can set the base priority level and escalation criteria by using a series of special system fields. By assigning values to the priority system fields in a case, you can ensure that all steps in that case default to the same priority in the work queue and will escalate at the same rate. These system fields take the form of **SW_CP_xxx** (CP meaning Case Priority).

There are five fields that can be assigned. These are:

Field	Description
SW_CP_VALUE	This is the Base Priority Value that each step will be given. This can be between 1 and 999, where 1 is the highest priority.
SW_CP_INCREMENT	This is the Increment, the amount that will be added to the item’s Priority Value whenever the Increment Period expires. (Assigning a negative number causes the priority to increase, i.e., move toward 1.)
SW_CP_NUMINC	This is the number of increments that will be added to the item’s Priority Value.
SW_CP_INCPERIOD	This is the time period, in the units specified in Period Type, which must expire before the item’s Priority Value is incremented.
SW_CP_PERIODTYP	This is the unit of measure of the Increment Period, either: • “M” or “m” for minutes • “H” or “h” for hours • “D” or “d” for days

You can assign values to these fields in the same ways as any other field, for example:

- by marking them as input fields on the form (not usually used)

- by assigning the values in a script to be run:
  - as a form command (initial, keep or release)
  - as a field command
  - as an application field.

Values can be assigned as whole numbers or as expressions. See *TIBCO iProcess Expressions and Functions Reference Guide* for information on valid expressions.

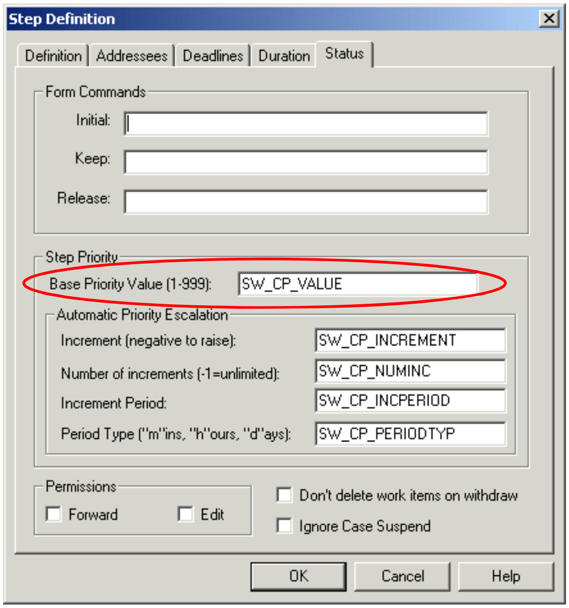
Setting the **SW_CP_xxx** values sets the priority for **all** subsequent steps in that case unless the **Step Status** settings have been changed for a particular step, in which case the **Step Status** settings take precedence.

Setting Priority at Step Level

To change the step priority and escalation values and override the **SW_CP_xxx** values for a particular step, open the **Status** tab dialog by right clicking on the step object and selecting **Status**.

Changing Priority

On the **Status** tab, amend the **Base Priority Value** to change the initial priority given to the step when it is processed.



Form Commands

Initial:

Keep:

Release:

Step Priority

Base Priority Value (1-999): SW_CP_VALUE

Automatic Priority Escalation

Increment (negative to raise): SW_CP_INCREMENT

Number of increments (-1=unlimited): SW_CP_NUMINC

Increment Period: SW_CP_INCPERIOD

Period Type ("m"ins, "h"ours, "d"ays): SW_CP_PERIODTYP

Permissions

□ Forward □ Edit

□ Don't delete work items on withdraw

□ Ignore Case Suspend

Click **OK** to save your changes.

Changing Priority Escalation

You can use priority escalation to ensure that a step increases (or decreases) in priority at certain time periods. For example, you might want a step to increase in priority by 10 every half hour it is waiting in a queue.

In the **Automatic Priority Escalation** section of the **Step Status** tab:

1. Enter the **Increment** by which you want the step to increase in priority each time. (A negative number causes it to increase in priority.)

2. Enter the **Number of Increments** that can occur. Enter -1 for unlimited.
3. Enter the **Increment Period**, e.g., 30 (for 30 minutes).
4. Enter the **Period Type**. This can be minutes, hours or days. Click **OK** to save your changes.

Once a work item is in a work queue, it cannot be affected by changes to the **Status** tab or the SW_CP_xxx system fields.

Changing Work Item Priority in a Work Queue

You can allow the supervisor of a queue to change the priority of a work item when it is in a queue by using the **SW_IP_xxx** system fields (IP meaning item priority). These are as described for Case Priority (see Changing Priority at Case Level on page 108), but substituting *IP* for *CP*.

To do this, set up conditional text on the form to allow the supervisor to either enter values for the **SW_IP_xxx** system fields or to run a pre-defined script to assign values.

Any changes made at run-time are for this work item only and will not affect any other item in the work queue.

Where can Priorities be Used

Priorities can be used for all cases of all procedures. Below are some example scenarios of how they can be used.

You must include Priority as part of the Sort criteria of the work queue for items to be displayed in priority order.

Priority Relative to Other Cases

It may be that some cases of a procedure need to be dealt with sooner than others and should have a higher priority, for example, you want to set priority at case level. An example of this might be a mortgage application procedure where applications with a value greater than `100,000` should have a higher priority than those of a lower value.

To achieve this, assign the `SW_CP_VALUE` field to `50` where the mortgage value is less than `100,000` and to `20` where the mortgage value is `100,000` or greater. This can be done by running a script on the first step of the case. The entries in the Status tab should not be altered as they will override any `SW_CP_xxx` values.

Priority Relative to the Life of the Case

If you want to increase (or decrease) the priority of a case depending on how long it has been in progress, then in a script at the beginning of the procedure you can set the initial `SW_CP_VALUE` and also include escalation parameters. Again, the entries in the Status tab must not be altered as they will override any `SW_CP_xxx` values.

Priority of Each Step Relative to a Baseline Value

It can happen that some steps in a procedure need to have a higher priority than others and this can be achieved by using a combination of both `SW_CP_xxx` values and the Status tab.

Set the `SW_CP_xxx` value at the start of the case to a base value of `30`. Set the base priority value of each step in the Status tab to be an offset of this value. For example, Step1 might be set to `SW_CP_VALUE + 10` and Step2 might be set to `SW_CP_VALUE - 10`. The more important steps in cases will then appear higher in the queue than the less important steps of other cases.

Priority Relative to Messages

A case or work item’s priority is also propagated to the internal message queues when passing messages between iProcess processes, for example, from the background and the WISes, or from SSOLite to the BG processes. Its default value is 50.

Messages are processed in the order of SW_CP_VALUE or SW_IP_VALUE when forwarding or releasing work items for queue.

For example, if a case sets its value for SW_CP_VALUE to 25, then all the messages for that case being sent around the system will have a value of 25, and automatically be processed before any messages with the default value 50. The value of SW_IP_VALUE is only valid while a work item is in a queue. If priority escalation is enabled, SW_IP_VALUE can increase while in the queue. When a work item is released if SW_IP_VALUE has changed, it will be used to set the priority for the release from the queue (but only once for that work item, for example, a single RELEASE or FORWARD instruction). After that, the case priority SW_CP_VALUE is then used. To propagate that value to subsequent messages, a release script which copies the value of SW_IP_VALUE into the field SW_CP_VALUE can keep the case running at the new priority level.

When using SSOLite stored procedures to start a case or to trigger an event, the following rules determine which message queue priority settings should be used for processing messages:

- If the value of the SW_CP_VALUE field is set, the message will be processed in the order of SW_CP_VALUE regardless of the message queue priority that is set by using the SW_SET_PRIORITY control procedure.
- If the SW_CP_VALUE field is not set, the message will be processed in the order of the message queue priority that is set using the SW_SET_PRIORITY control procedure.
- If both the SW_CP_VALUE field and the SW_SET_PRIORITY control procedure are not set for the message priority, the message priority will be set to the default value of the SW_CP_VALUE field, 50.

See "Message Prioritizing" in the appropriate *TIBCO iProcess Engine Database Administrator’s Guide* for more information about SSOLite stored procedures.

Index