TIBCO iProcess™ Engine

Architecture Guide

Software Release 11.1
September 2009

Important Information

Contents

Back to Library

Back to Library

Back to Library

Back to Library

Back to Library

Back to Library

About This Guide

This guide describes the architecture of the TIBCO iProcess Engine, and how each server process works and interacts with other processes. It also provides information about how the TIBCO iProcess Engine interacts with other applications such as databases and transaction control applications. You can use this information to help plan and implement your system, debug system problems and trace performance problems to specific processes.

The point-to-point models on page 131 describe the full flow of one case of a procedure through each of the server processes. This shows what happens when a case is released through to it appearing in the next user's work queue.

Back to Library

How to Use This Guide

Refer to the following chapters for the information you need:

- Chapter 1, Introduction to TIBCO iProcess Products provides an overview of the products in the TIBCO iProcess™ Suite and other TIBCO products that interact with the TIBCO iProcess Suite.
- Chapter 2, Introduction to the TIBCO iProcess Engine provides an overview of the TIBCO iProcess Engine architecture and the server processes that are involved to process business process information.
- Chapter 3, Using the TIBCO iProcess Suite in a Multilingual Environment describes how internationalization is supported by TIBCO iProcess Suite and what are the implications of deciding to support UTF-8 encoding in your iProcess database.
- Chapter 4, *TIBCO iProcess Engine Processes* describes all of the server processes in more detail.
- Chapter 5, *Introduction to Transactional Business Process Automation* provides an overview of how the TIBCO iProcess Engine provides transaction control.
- Chapter 6, *iProcess Mbox Sets* describes how the TIBCO iProcess Engine uses messaging to provide reliable, transactional message queuing.
- Chapter 7, *Monitoring Activities* describes how messages are processed between the **BG** process, the **IAPJMS** process and the external application when activity monitoring is enabled on the TIBCO iProcess Engine.
- Chapter 8, describes how the TIBCO iProcess Engine handles database failure or failover.
- Chapter 9, *Process Management* describes the role of the TIBCO iProcess Engine Process Sentinels and how they manage the starting, stopping and running of all the server processes.
- Chapter 10, *Network Communication* explains how the TIBCO iProcess Engine and TIBCO iProcess™ Workspace communicate with each other and how servers in a cluster installation communicate with each other.
- Chapter 11, *Point-to-Point Data Flow Models* explains how case data is processed by all the TIBCO iProcess Engine processes for a number of different scenarios such as starting a case and releasing a case.

Back to Library

Target Audience

This guide is aimed at the following types of user of the TIBCO iProcess Suite:

- System integrators
- Application developers
- Server administrators
- Support.

Back to Library

Changes from the Previous Issue of this Guide

Major technical changes from the information presented in the previous issue of this guide are:

- TIBCO iProcess Engine now supports UTF-8 encoding. See Chapter 3, Using the TIBCO iProcess Suite in a Multilingual Environment, on page 45.
- Messages produced by the IAPJMS process can optionally be generated in an extended format to provide more information. See The Monitor Event Detail Message (MED) on page 98.

Back to Library

Where You Can Find More Information

You can find more information about the TIBCO iProcess Engine from the following sources:

- The installation guide, supplied with the software, explains how to install the software.
- A **Readme** file, supplied with the software, provides any last-minute and version-specific information that could not be included in the main documentation.
- Detailed information about using the TIBCO iProcess Suite™ can be found on the TIBCO iProcess Suite: Documentation Library CD.
- For more information about iProcess database tables, see the following guides:
  - *TIBCO iProcess Engine (SQL) Administrator's Guide*
  - *TIBCO iProcess Engine (Oracle) Administrator's Guide*
  - *TIBCO iProcess Engine (DB2) Administrator's Guide*
- For the latest TIBCO iProcess Suite product information, please refer to the TIBCO Support web site at **http://www.tibco.com/services/support**

Back to Library

Documentation Conventions

Because this guide covers both Windows, UNIX and Linux versions of the TIBCO iProcess Engine, this guide uses the Windows convention of a backslash (\). The equivalent pathname on a UNIX or Linux system is the same, but using the forward slash (/) as a separator character.

UNIX or Linux pathnames are occasionally shown explicitly, using forward slashes as separators, where a UNIX/Linux-specific example or syntax is required.

Any references to UNIX in this guide also apply to Linux unless explicitly stated otherwise.

The following conventions are used throughout this guide.

Convention	Description
SWDIR	Indicates the iProcess system directory where the TIBCO iProcess Engine is installed. For example, if *SWDIR* is set to **\swserver\staffw_nod1** then the full path to the **swutil** command would be: • on a Windows server (on the **c:** drive): **c:\swserver\staffw_nod1\bin\swutil** • on a UNIX or Linux server: **/swserver/staffw_nod1/bin/swutil** or $SWDIR**/bin/swutil** On a UNIX or Linux system, the environment variable $SWDIR should be set up to point to the iProcess system directory for the **root** and **swadmin** users.
italics	Indicates emphasis, variables and manual titles.
`monospace text`	Indicates code samples, commands and their options, directories and filenames. Any text that you must enter from the keyboard is displayed as monospace text.
monospace italic text	Indicates variables in commands.
{ }	Indicates a set of choices in a syntax line. The braces should not be entered.

Back to Library

Convention	Description
[ ]	Indicates optional items in a syntax line. The brackets should not be entered. For example: `SHOW_ALL_ATTRIBUTES [attribute]`
|	Indicates mutually exclusive choices in a syntax line i.e. you enter only one of the given choices. You should not enter the symbol itself.

Back to Library

Back to Library

Chapter 1 Introduction to TIBCO iProcess Products

This chapter provides an overview of the products in the TIBCO iProcess™ Suite and other TIBCO products that interact with the TIBCO iProcess Suite. It gives a brief description of each product and explains how they interact with each other. The following products are described:

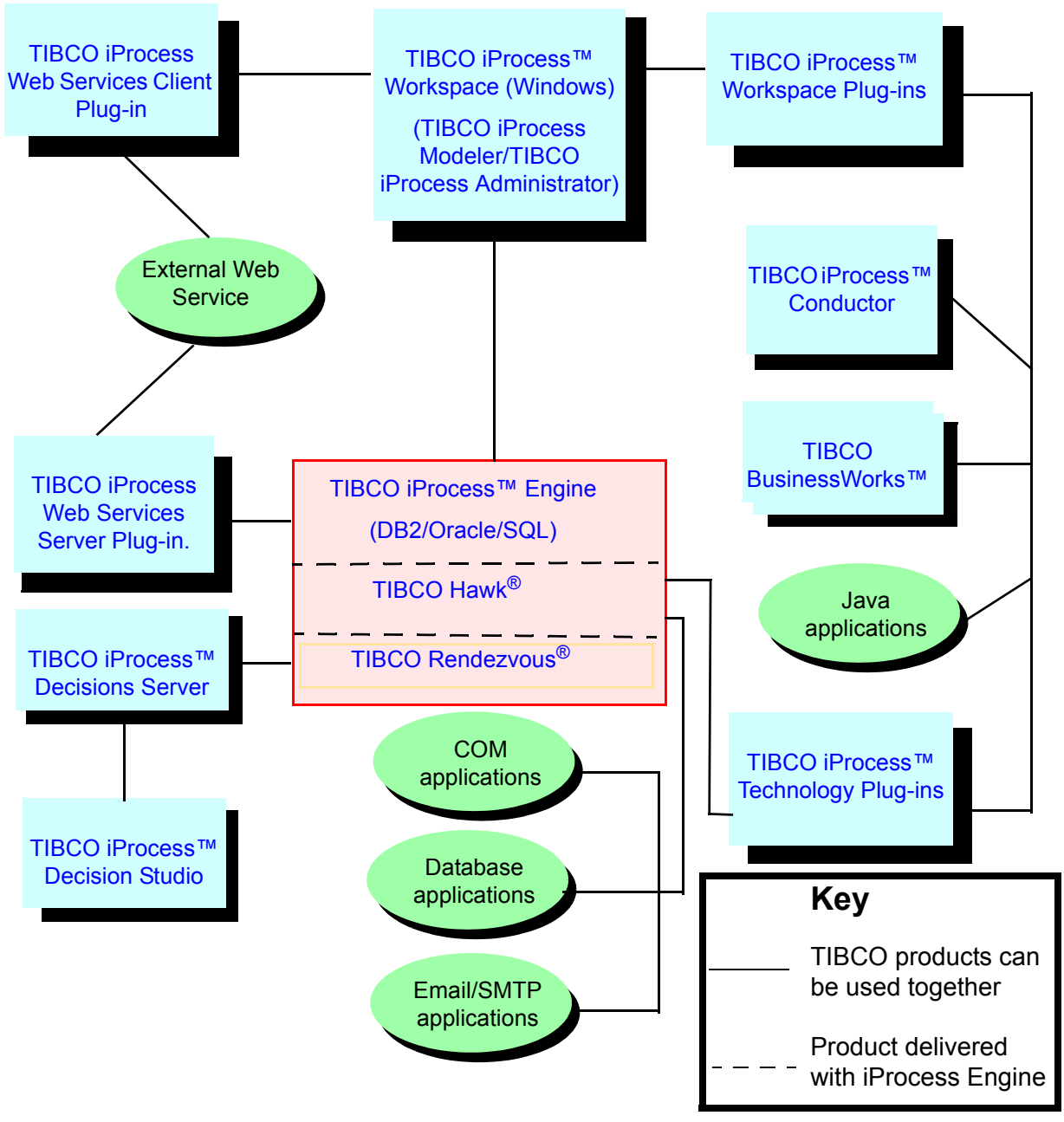
- TIBCO Business Studio™
- TIBCO BusinessWorks™
- TIBCO Enterprise Message Service™
- TIBCO Hawk®
- TIBCO iProcess™ Analytics
- TIBCO iProcess™ Analytics Export
- TIBCO iProcess™ Conductor
- TIBCO iProcess™ Decisions Server
- TIBCO iProcess™ Decision Studio
- TIBCO iProcess™ Engine
- TIBCO iProcess™ Insight
- TIBCO iProcess™ Objects Server
- TIBCO iProcess™ Objects Director
- TIBCO iProcess™ Server Objects
- TIBCO iProcess™ Technology Plug-ins
- TIBCO iProcess™ Web Services Plug-in
- TIBCO iProcess™ Workspace (Browser)
- TIBCO iProcess™ Workspace Plug-ins
- TIBCO iProcess™ Workspace (Windows)
- TIBCO Rendezvous®

All Java-based TIBCO iProcess Suite components (for example, TIBCO iProcess Java Server Plug-in and TIBCO iProcess BusinessWorks Server Plug-in) must be run using a 32-bit JVM.

Back to Library

TIBCO Product Overview 1

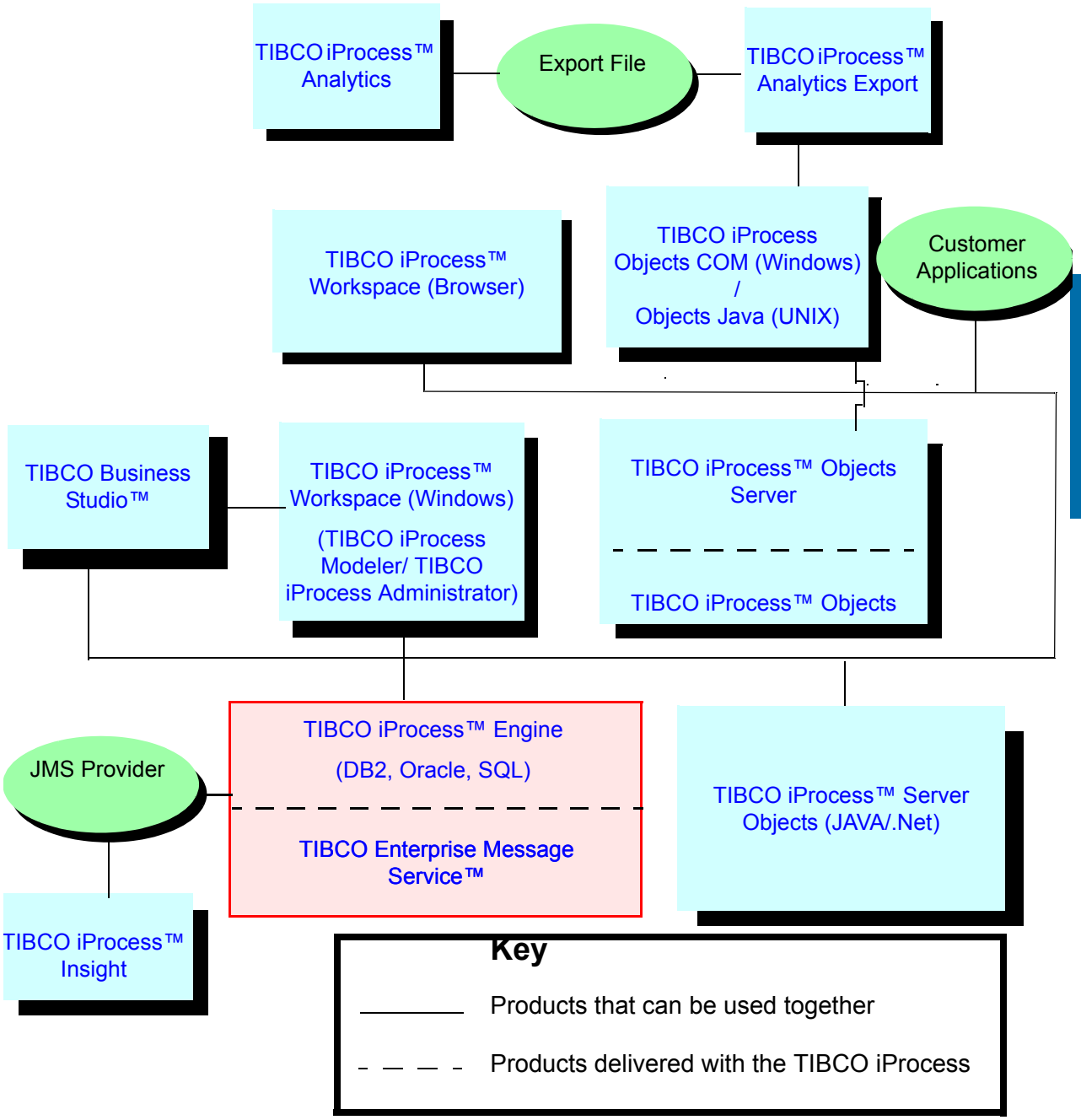
Click on a product to find out more information



Back to Library

TIBCO Product Overview 2

Click on a product to find out more information



Back to Library

TIBCO Business Studio™

Description

TIBCO Business Studio enables business analysts to implement business processes. It provides a standards-based modelling environment that supports both Business Process Modeling Notation (BPMN) and XML Process Definition Language (XPDL).

If you are using native service calls (database or email) or general service calls (such as web services), you can augment the process with execution details in TIBCO Business Studio and deploy it directly to the TIBCO iProcess Engine. However, if you need to make other types of service calls (for example, EAI Java), you must augment and implement the Process using another product such as TIBCO iProcess Modeler.

TIBCO Business Studio also enables business analysts to simulate processes that have been developed in Business Studio. This is useful to identify areas of the process that can be improved such as bottlenecks and areas of high cost or reduced service levels.

TIBCO Business Studio can be used instead of iProcess Workspace (Windows) to design business processes for use with the iProcess Engine.

For Information

For information about TIBCO Business Studio, see the following:

- *TIBCO Business Studio Modelling Guide*
- *TIBCO Business Studio Simulation Guide*

Back to Library

TIBCO BusinessWorks™

Description

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

TIBCO BusinessWorks also works with TIBCO Administrator, a web-based GUI for monitoring and managing run-time components.

The TIBCO iProcess BusinessWorks Connector enables the TIBCO iProcess Engine to interact with TIBCO BusinessWorks and vice versa. For example, BusinessWorks processes can be invoked from iProcess Engine procedures and iProcess Engine cases can be started from BusinessWorks.

For Information

For information about TIBCO BusinessWorks, see the TIBCO Technical Publications Document Library.

For information about the TIBCO iProcess BusinessWorks Connector, see the *TIBCO iProcess BusinessWorks Connector User's Guide*.

Back to Library

TIBCO Enterprise Message Service™

Description

TIBCO Enterprise Message Service implements JMS and integrates support for connecting other message services, such as TIBCO Rendezvous and TIBCO SmartSockets.

Java Message Service 1.1 (JMS) is a Java framework specification for messaging between applications. Sun Microsystems developed this specification, in conjunction with TIBCO and others, to supply a uniform messaging interface among enterprise applications.

Using a message service allows you to integrate the applications within an enterprise. For example, you may have several applications: one for customer relations, one for product inventory, and another for raw materials tracking. Each application is crucial to the operation of the enterprise, but even more crucial is communication between the applications to ensure the smooth flow of business processes. Message-oriented-middleware (MOM) creates a common communication protocol between these applications and allows you to easily integrate new and existing applications in your enterprise computing environment.

TIBCO Enterprise Message Service is delivered with the TIBCO iProcess Engine to enable the iProcess Engine to communicate with other TIBCO products (for example, TIBCO BusinessWorks and TIBCO iProcess Insight) using a JMS protocol.

For Information

For information about TIBCO Enterprise Message Service, see the TIBCO Technical Publications Document Library.

Back to Library

TIBCO Hawk®

Description

TIBCO Hawk is a monitoring system that monitors systems and applications on a specific computer. It consists of the following components:

- **TIBCO Hawk Agent**. A TIBCO Hawk agent is an autonomous process that resides on each computer and monitors systems and applications on that computer. Agents run independently of the TIBCO Hawk Display. Agents operate autonomously and are active whenever the operating system they monitor is active. Agents use sets of rules, called rulebases, to configure system management, status, and automation tasks. A TIBCO Hawk agent must be installed on each computer you wish to monitor.
- **TIBCO Hawk Microagent**. A TIBCO Hawk Microagent (HMA) is a partner process to the TIBCO Hawk agent and provides the local agent with methods for monitoring the host operating system. Like the agent, a TIBCO Hawk Microagent is generally installed on each computer you wish to monitor.
- **TIBCO Hawk Display Program**. The TIBCO Hawk Display program is used by system administrators to view network health and to create rulebases (sets of rules that automate monitoring activities). A TIBCO Hawk Display should be installed on any computers you wish to use for monitoring the network or for building rulebases.
- **TIBCO Hawk Event Service**. The TIBCO Hawk Event Service is a process that records TIBCO Hawk alerts and changes in agent status. When communication with an agent is lost, the Event Service can invoke a user-provided script. Alerts and notifications can be recorded to log files or a database. Typically, the TIBCO Hawk Event Service is installed on a minimal number of computers in the network.

TIBCO Hawk is delivered with the TIBCO iProcess Engine to enable you to use it to monitor the TIBCO iProcess Engine server processes. For example, you can use TIBCO Hawk to check what iProcess Engine server processes are running or stop and start them.

For Information

For information about TIBCO Hawk, see the TIBCO Technical Publications Document Library.

For information on configuring TIBCO Hawk for use with the TIBCO iProcess Engine, see the *TIBCO iProcess Engine Administrator's Guide*.

Back to Library

TIBCO iProcess™ Analytics

Description

TIBCO iProcess™ Analytics is a tool that enables organizations to analyze, evaluate and monitor business processes, providing insight into the performance of key processes and the knowledge needed to continuously improve them. iProcess Analytics combines current and historical process data and presents this data via intuitive dashboard views. An integrated early-warning system monitors all running instances and triggers alerts when deviations against planned values occur, thus enabling organizations to react quickly and take rapid corrective action.

iProcess Analytics uses a SQL RDBMS as the repository that saves all configurations and data. It is developed in Java as a client/server application.

TIBCO iProcess Analytics, in conjunction with TIBCO iProcess™ Analytics Export, is used to analyze the performance of iProcess procedures. iProcess Analytics Export is used to export the data from the iProcess Engine to prepare it for import into iProcess Analytics. Data from multiple sources can be analyzed providing data from iProcess is part of that data.

For Information

For information about iProcess Analytics, see the following:

- *TIBCO iProcess Analytics System Architecture Guide*
- *TIBCO iProcess Analytics LDAP User's Guide*
- *TIBCO iProcess Analytics Advanced User's Guide*
- *TIBCO iProcess Analytics ARIS Toolset Interface Guide*
- *TIBCO iProcess Analytics CTK Quick Start Guide*
- *TIBCO iProcess Analytics Performance Cockpit Customizing Guide*
- *TIBCO iProcess Analytics Delta Notes*
- *TIBCO iProcess Analytics Customization Guide*
- *TIBCO iProcess Analytics Data Import Guide*

Back to Library

TIBCO iProcess™ Analytics Export

Description

TIBCO iProcess Analytics Export is a utility that is used to export cases from iProcess procedures. It generates an XML file that can be imported into TIBCO iProcess™ Analytics.

TIBCO iProcess Analytics Export, in conjunction with TIBCO iProcess™ Analytics, is used to analyze the performance of iProcess procedures.

For Information

For information about iProcess Analytics Export, see the following:

- *TIBCO iProcess Analytics Export (Java) User's Guide*
- *TIBCO iProcess Analytics Export (Windows) User's Guide*

Back to Library

TIBCO iProcess™ Conductor

Description

The TIBCO iProcess Conductor supports complex business process management (BPM) by abstracting timing and resource requirements into a dependency mapping layer that maximizes reusability of process components. The TIBCO iProcess Conductor packages TIBCO iProcess Engine processes as independent process components that are combined to create an execution plan. Process components can be reused, coordinated, individually monitored, dynamically modified — even completely remodeled — during runtime to enable goal-oriented BPM.

This enables the flexibility and control of business processes. TIBCO iProcess Conductor coordinates business processes that are executed in TIBCO iProcess Engine. It enables business users to select templates or create processes – including resource requirements and timing dependencies – immediately prior to run-time from re-usable process components.

The TIBCO iProcess Conductor fosters a goal-oriented approach to BPM because processes are designed in terms of goals, with each step toward the goal achieved by process components. Processes are monitored at the sub-goal level, but within the context of the larger goal. If a sub-goal is not met in a timely manner and jeopardizes the overall goal, users are notified and can ameliorate the problem. Processes can be modified during execution. If a new goal is identified, a process can be rolled back to the last process component in common between the old and new goal.

For Information

For information about iProcess Conductor, see the following:

- *TIBCO iProcess Conductor Concepts Guide*
- *TIBCO iProcess Conductor Implementation Guide*
- *TIBCO iProcess Conductor User's Guide*
- *TIBCO iProcess Conductor Administrator's Guide*
- *TIBCO iProcess Conductor Utility Framework Guide*
- *TIBCO iProcess Execution Plan Interfaces Developer's Guide*

Back to Library

TIBCO iProcess™ Decisions Server

Description

The TIBCO iProcess Decisions Server enables you to define and use TIBCO iProcess Decisions steps in your iProcess procedures. iProcess Decisions Service steps extend the functionality of your business process by enabling you to integrate with a powerful rules engine. See TIBCO iProcess™ Decision Studio.

iProcess data can be sent to the TIBCO iProcess Decisions Server to be processed by a set of business rules and the resulting data can then be passed back to iProcess.

For Information

For information about the TIBCO iProcess Decisions Server, see the *TIBCO iProcess Decisions Plug-in: User's Guide*

Back to Library

TIBCO iProcess™ Decision Studio

Description

The TIBCO iProcess Decision Studio is used to build Rule Sets. Once deployed to the TIBCO iProcess™ Decisions Server, a rule set becomes a Decision Service.

A Decision Service automates a discrete decision-making task. It is implemented as a set of business rules and exposed as a Web Service (or Java Service). By definition, the rules within a Decision Service are complete and unambiguous; for a given set of inputs, the Decision Service addresses every logical possibility uniquely, ensuring "decision integrity".

Using an iProcess Decisions Service step in your iProcess procedure enables you to interact with a decision service in the TIBCO iProcess™ Decisions Server.

This means that you can:

- send iProcess case data to the TIBCO iProcess™ Decisions Server where it can process the case data against the decision service rules.
- capture the resulting data from the decision service back in iProcess.

For Information

For information about the TIBCO iProcess Decision Studio, see the following:

- *TIBCO iProcess Decisions Start Here*
- *TIBCO iProcess Decisions DB Access Tutorial*
- *TIBCO iProcess Decisions Studio Quick Reference Guide*
- *TIBCO iProcess Decisions Rule Language Guide*
- *TIBCO iProcess Decisions Rule Modeling Guide*
- *TIBCO iProcess Decisions Rule Modeling Tutorial*
- *TIBCO iProcess Decisions Sample Application Guide*
- *TIBCO iProcess Decisions Server Integration & Deployment Guide*
- *TIBCO iProcess Decisions Server Deployment Tutorial*

Back to Library

TIBCO iProcess™ Engine

Description

The TIBCO iProcess Engine is the core of iProcess and is responsible for the execution of business processes. In response to various input, for example. case start, completion of steps, events being triggered, deadlines expiring and data being modified, coupled with knowledge of the business process, it makes the next activity or step in the process available for execution. This might be the dispatching of a work item to a user queue or the invocation of an application or task via an EAI step.

When a case of a procedure is started by an individual from a TIBCO iProcess Workspace (Windows) computer, a number of processes are used to process the information contained in one step before the next step can be performed.

The TIBCO iProcess Engine maintains the list of work items in a user's work queue for all the active cases to be processed. If the first step was opened, completed and released, the TIBCO iProcess Engine determines the subsequent actions for the step and updates the necessary case data in the database.

The iProcess Engine enables dynamic processing and knowledge-based interaction with iProcess procedures. This means iProcess procedures can be processed based on events as they occur and that may not be known about at procedure definition time. This feature is available using graft steps. A graft step enables an external application to graft (attach) one or more sub-procedures to a particular point in your procedure at run-time. Therefore, when a case of the main procedure is started, the external application can start a number of sub-processes which are attached to the main procedure via the graft step. For example, a financial application determines that a credit check and a transfer of funds are required as part of the main procedure. When another case is started, it determines that only a transfer of funds is required. This means that the procedure is dynamic and cannot be decided at procedure definition time. One of the processes is an iProcess sub-procedure and the other is a process run by the financial system.

iProcess procedures can be defined in TIBCO Business Studio or TIBCO iProcess Modeler.

All the TIBCO iProcess Engine case data such as fields and their values are stored in a TIBCO iProcess Engine database instance. An iProcess Engine database instance can be a SQL Server, Oracle or DB2 database.

The TIBCO iProcess Engine is a 32-bit application but can be used on both 32-bit and 64-bit operating systems, and against a 32-bit or 64-bit database.

Back to Library

The iProcess Engine (in conjunction with other iProcess products, for example, the iProcess Technology Plug-ins, iProcess Server Objects, and iProcess Web Services Plug-in) enables connectivity to external applications. This means that iProcess procedures can integrate with other systems in your business process. iProcess case data can be sent to external applications using whatever communication method is required. Data can also be passed back from the application to iProcess.

For Information

For information about the TIBCO iProcess Engine, see the following:

- *TIBCO iProcess Engine: Architecture Guide*
- *TIBCO iProcess Engine Administrator's Guide*
- *TIBCO iProcess Engine (SQL) Administrator's Guide*
- *TIBCO iProcess Engine (Oracle) Administrator's Guide*
- *TIBCO iProcess Engine (DB2) Administrator's Guide*
- *TIBCO iProcess swutil and swbatch Reference Guide*
- *TIBCO iProcess Engine System Messages Guide*
- *TIBCO iProcess User Validation API: Reference Guide*

Back to Library

TIBCO iProcess™ Insight

Description

TIBCO iProcess Insight is a process monitoring product from TIBCO that adds BAM capabilities to TIBCO's business process management (BPM) systems to support proactive management of users, tasks, operations and exceptions. TIBCO iProcess Insight accomplishes this by introspecting processes to discover registered procedures/sub-procedures and steps.

Using TIBCO iProcess Insight, you can monitor the execution of the automated processes and hence identify whether or not the execution of business processes is efficient. You can specifically check for the following points:

- whether processes are efficient in real-time (Are our procedures doing what they are supposed to?)
- whether a resource bottleneck exists (Where can we optimize?)
- whether performance targets are being met (How are we doing as a company?)
- what volume of work is being performed?

After gathering the above information, you can optimize your business processes and deploy the resources in a better fashion. It will lead to an improved performance and efficiency which in turn means improved advantage and a reduced timeline to realize Return On Investment (ROI).

You can use the TIBCO Enterprise Message Service (which is delivered with the iProcess Engine) or an external JMS Provider to interact with TIBCO iProcess Insight. The TIBCO iProcess Engine can publish events with process data as XML documents on a JMS queue, as they occur. iProcess Insight can consume these XML documents real-time. It uses this information to perform its analysis. It does this using a wizard that allows you to select the processes and activities to be monitored, along with the data and reports that you are interested in. It includes a supervisor capsule that allows a supervisor to take remedial actions to react to events that occur during the execution of the processes.

For Information

For information about TIBCO iProcess Insight, see the TIBCO Technical Publications Document Library.

Back to Library

TIBCO iProcess™ Objects Server

Description

The TIBCO iProcess Objects Server acts as a gateway between iProcess client applications developed with TIBCO iProcess Server Objects (Java or .NET), and the TIBCO iProcess Engine.

After communication is established between the client application and the TIBCO iProcess Objects Server, the TIBCO iProcess Objects Server waits for request messages from the client. When the TIBCO iProcess Objects Server receives a request message, it in turn makes calls to the TIBCO iProcess Engine to perform functions such as locking work items, moving work items to other work queues, writing data to the database, etc.

The TIBCO iProcess Workspace (Browser) is an example client application that sends requests to the iProcess Engine through the iProcess Objects Server. For more information about this client application, see TIBCO iProcess™ Workspace (Browser).

For Information

For more information about the iProcess Objects Server, see the *TIBCO iProcess Objects Server Administrator's Guide*.

Back to Library

TIBCO iProcess™ Objects Director

Description

The TIBCO iProcess Objects Director is a standalone program that maintains a list of TIBCO iProcess Objects Servers that are configured in a node cluster. When a client application needs access to a TIBCO iProcess Objects Server, it first establishes a connection to the TIBCO iProcess Objects Director. The TIBCO iProcess Objects Director then decides, based on a "pick method," which TIBCO iProcess Objects Server the client should connect to.

The list of known TIBCO iProcess Objects Servers is updated dynamically as TIBCO iProcess Objects Server instances are started and stopped. The TIBCO iProcess Objects Director maintains this list by checking the **process_config** table of the iProcess Engine to which it is associated.

For Information

For information about using and configuring the TIBCO iProcess Objects Director, see the *TIBCO iProcess Objects Director Administrator's Guide*.

Back to Library

TIBCO iProcess™ Server Objects

Description

TIBCO iProcess Server Objects is an Application Programming Interface for iProcess. It provides access to all of the information and functionality required to write either a BPM user oriented client application or a batch oriented broker application.

It comprises a set of objects that are used to build applications that automate business processes. TIBCO iProcess Server Objects consists of an object model that provides access to the information and functionality needed in these applications.

The objects in the TIBCO iProcess Server Objects object model can be used to start cases, present information on screens to users, manipulate work items, remind users when actions need to be taken, and monitor and control the flow through the business process.

TIBCO iProcess Server Objects is designed to be used in server-side application architectures.

Client applications make use of the objects in the TIBCO iProcess Server Objects by making method calls that either retrieve or modify data. These method calls cause messages to be sent to a TIBCO iProcess Objects Server. The TIBCO iProcess Objects Server acts as a gateway between the client application created with TIBCO iProcess Server Objects, and the TIBCO iProcess Engine, where the actual processing and storage of data occurs. The TIBCO iProcess Engine manages all data, routing work items and updating the appropriate work queues.

There are two types of implementation available for iProcess Server Objects:

- TIBCO iProcess Server Objects (.NET)
- TIBCO iProcess Server Objects (Java)

Both 32-bit and 64-bit variants of TIBCO iProcess Server Objects (.NET) and TIBCO iProcess Server Objects (Java) are available. These variants allow customers to write applications that can interface with TIBCO iProcess Suite on 32-bit and/or 64-bit Windows platforms as required.

For Information

For information about TIBCO iProcess Server Objects, see the following:

- *TIBCO iProcess Server Objects (JAVA): Programmer's Guide*
- *TIBCO iProcess Server Objects (.NET): Programmer's Guide*

Back to Library

TIBCO iProcess™ Technology Plug-ins

The TIBCO iProcess Technology Plug-ins is a package that includes a number of individual products. These products have been grouped together to ease their installation.

The TIBCO iProcess Technology Plug-ins include the following individual plug-ins[1]:

- **TIBCO iProcess Java Server Plug-in** - This plug-in enables the iProcess Engine to process EAI Java steps that have been added to the iProcess procedure. An EAI Java step enables you to design an iProcess procedure so that you can call out to a custom Java object to perform some additional work. For example, you can create a custom Java object to call business methods in Enterprise Java Beans (EJBs), call a database via a JDBC connection, or post a message on a Java Message Service (JMS) queue.
- **TIBCO iProcess BusinessWorks Server Plug-in** - This plug-in provides a way of initiating a BusinessWorks process from the TIBCO iProcess Engine. It provides the communication mechanism that allows the iProcess Engine to make calls to TIBCO BusinessWorks.Enabling the iProcess Engine to integrate with BusinessWorks provides a highly performant and versatile way of integrating with other applications and technologies.
- **TIBCO BusinessWorks iProcess Plug-in** - This plug-in provides a set of resources that allow a TIBCO BusinessWorks process to communicate with the TIBCO iProcess Engine. For example, a BusinessWorks process can start or suspend a case of an iProcess Engine procedure. The BusinessWorks iProcess Plug-in includes facilities that enable you to create a form using TIBCO BusinessWorks FormBuilder, and then provide it to the iProcess Workspace (Browser).
- **TIBCO iProcess Conductor Server Plug-ins** - These consist of the following plug-ins:
    - **TIBCO iProcess Conductor Order Server Plug-in** - This plug-in provides an interface that notifies the TIBCO iProcess Conductor of the status of the order. It also enables you to update order data from the TIBCO iProcess Conductor in the TIBCO iProcess Engine and vice versa.

1. To improve the installation process, a new assembly-based combined installer was introduced in Version 10.6, giving a standard installation process for all the iProcess Technology Plug-ins. Previously, these products were installed separately.

Back to Library

  - **TIBCO iProcess Conductor Orchestration Server Plug-in** - This plug-in provides the communication mechanism that allows the iProcess Conductor to pass iProcess data to the TIBCO iProcess Conductor and for the TIBCO iProcess Conductor to process the data before sending the resulting data back to the TIBCO iProcess Engine.

- **TIBCO iProcess Conductor XML Transform Server Plug-in** - This plug-in provides the facility to transform XML to/from iProcess field data and/or a designated URL. For example, an EAI Transform step could be used to take XML data from an iProcess memo field, apply a transformation to the data, and pass the result to another memo field. Equally, XML data can be parsed and mapped onto discrete iProcess fields.

For Information For information about the TIBCO iProcess Technology Plug-ins, see the following:

- *TIBCO iProcess Java Plug-in: User's Guide*
- *TIBCO iProcess BusinessWorks Connector User's Guide*
- *TIBCO iProcess Conductor: Implementation Guide*
- *TIBCO Business Studio Modeling: User's Guide*

Back to Library

TIBCO iProcess™ Web Services Plug-in

Description

The main function of the iProcess Web Services Plug-in is to provide an interface for both inbound and outbound communication between the iProcess Engine and external applications:

- Outbound - iProcess procedures make calls to external applications to perform some operation.
- Inbound - External applications make calls to iProcess to perform operations such as starting cases, triggering events or suspending cases.

The iProcess Web Services Plug-in consists of three components:

- **TIBCO iProcess Web Services Server Plug-in**.

  This consists of:

  — Jetty - This is a Java HTTP Server and Servlet Container that runs on a Java Virtual Machine (JVM) and the components contained in Jetty handle communication for both inbound and outbound calls.

  — iProcess Engine Interface - This consists of an EAI Plug-in. It allows the iProcess background processes to communicate with Jetty.

- **TIBCO iProcess Web Services Client Plug-in**

  This plug-in needs to be installed on your client machine that hosts your iProcess Workspace (Windows) and iProcess Modeler. This plug-in enables you to define EAI Web Service steps in your iProcess procedures.

For Information

For information about the TIBCO iProcess Web Services Plug-in, see the *TIBCO iProcess Web Services Plug-in: User's Guide.*

Back to Library

TIBCO iProcess™ Workspace (Browser)

Description

The TIBCO iProcess Workspace (Browser) is a client application that enables users to participate in business processes. It enables users to view and perform the work that has been assigned to them. This means that the movement of data/information is automated through your business process, whatever your business may be. For example, it may automate the process of filing an insurance claim, a loan application, or any process that has multiple steps.

The TIBCO iProcess Workspace (Browser) application is run in a browser (e.g., Microsoft Internet Explorer). You connect to a TIBCO iProcess Engine over the internet or an intranet by entering a web address that is determined by the location at which your TIBCO software is installed.

The type of functions you can perform are:

- Open and view work queues
- Apply filter and/or sort parameters to a work queue so only the desired work items are listed in the desired order.
- Open and view work items
- Enter information into forms that are displayed when you open a work item.
- Either "keep" a work item, which closes it and places it back in the work queue, or "release" a work item, which causes the case to advance to the next step in the procedure.
- Forward work items to a different work queue (user or group).
- Start a case of a procedure.
- Suspend or reactivate a case.
- View the history of a case.
- Allow temporary access to your personal work queue to another user.
- Temporarily redirect work items from your work queue to another work queue.

For Information

For information about the TIBCO iProcess Workspace (Browser), see the following:

- *TIBCO iProcess Workspace (Browser) User's Guide*
- *TIBCO iProcess Workspace (Browser) Configuration and Customization Guide*
- *TIBCO iProcess Workspace (Browser) Action Processor Reference Guide*

Back to Library

TIBCO iProcess™ Workspace Plug-ins

The TIBCO iProcess Workspace Plug-ins is a package that includes a number of individual products. These products have been grouped together to ease their installation.

The iProcess Workspace Plug-ins include the following individual plug-ins[1]:

- **iProcess Java Client Plug-in** - This plug-in enables the iProcess Engine to define EAI Java steps to be added to an iProcess procedure. An EAI Java step enables you to design an iProcess procedure so that you can call out to a custom Java object to perform some additional work. For example, you can create a custom Java object to call business methods in Enterprise Java Beans (EJBs), call a database via a JDBC connection, or post a message on a Java Message Service (JMS) queue.
- **iProcess BusinessWorks Client Plug-in** - This plug-in enables you to define EAI BusinessWorks steps in an iProcess procedure. EAI BusinessWorks steps allow an iProcess Engine procedure to invoke a TIBCO BusinessWorks process definition.
- **iProcess Conductor Client Plug-ins (Order and Orchestration)** - The Orchestration plug-in enables you to define Orchestration steps in your iProcess procedures. The Order plug-in enables you to define Order steps in your iProcess procedures.
- **iProcess Conductor XML Transform Client Plug-in** - This plug-in enables you to define Transform steps in your iProcess procedures.

For Information: For information about the TIBCO iProcess Workspace Plug-ins, see the following:

- *TIBCO iProcess Java Plug-in: User's Guide*
- *TIBCO iProcess BusinessWorks Connector User's Guide*
- *TIBCO iProcess Conductor Implementation Guide*

1. To improve the installation process, a new assembly-based combined installer was introduced in Version 10.6, giving a standard installation process for all the iProcess Workspace Plug-ins. Previously, these products were installed separately.

Back to Library

TIBCO iProcess™ Workspace (Windows)

TIBCO iProcess Workspace (Windows) is provides the ability to define and manage procedures.

The TIBCO iProcess Workspace (Windows) consists of:

- The Work Queue Manager
- The Procedure Manager

The Work Queue Manager displays users' queues and work items and the Procedure Manager displays all of the procedures currently available. From the Procedure Manager, you can:

- create and edit procedures. The TIBCO iProcess Modeler is started from TIBCO iProcess Workspace (Windows).
- organize and manage your procedures as a hierarchical structure of procedure libraries, in the same way as, for example, you manage files and directories.

TIBCO iProcess Workspace (Windows) includes:

- **TIBCO iProcess Modeler** - gives you a visual representation of your business process that is easy to follow and that can be enhanced or amended at any time. The TIBCO iProcess Modeler builds on the familiar flowchart metaphor to show in an unambiguous manner, the flow of work for a particular business process. The rules that you define graphically are stored by the iProcess Engine and can then be deployed across a wide ranging hardware architecture.

  The TIBCO iProcess Modeler is automatically started by TIBCO iProcess Workspace (Windows) when you want to create or edit procedures. From the TIBCO iProcess Modeler you can access the Step Definer, which enables you to design the forms for each step in your procedure. The forms are the part of the step seen by the person who receives the work item in their queue. The forms contain text and fields into which users can enter information for a particular case, or instance, of a procedure.

Business processes can also be modelled in TIBCO Business Studio. See TIBCO Business Studio™ on page 4 for more information.

- **TIBCO iProcess Administrator** - The TIBCO iProcess Administrator enables you to perform the tasks involved to administer iProcess Workspace (Windows), such as managing users, cases and work queues. It is a graphical utility that can be used on any iProcess Workspace (Windows) to perform common administration functions on the TIBCO iProcess Engine.

Back to Library

For Information For information about TIBCO iProcess Workspace (Windows), see the following:

- *TIBCO iProcess Workspace (Windows) User's Guide*
- *TIBCO iProcess Workspace (Windows) Manager's Guide*
- *TIBCO iProcess Modeler - Getting Started*
- *TIBCO iProcess Modeler - Basic Design*
- *TIBCO iProcess Modeler - Advanced Design*
- *TIBCO iProcess Modeler - Procedure Management*
- *TIBCO iProcess Modeler - Integration Techniques*
- *TIBCO iProcess Expressions and Functions Reference Guide*

Back to Library

TIBCO Rendezvous®

Description

Rendezvous software makes it easy to create distributed applications that exchange data across a network. You get software support for network data transport and network data representation. Rendezvous software supports many hardware and software platforms, so programs running on many different kinds of computers on a network can communicate seamlessly.

From the programmer's perspective, the Rendezvous software suite includes two main components—a Rendezvous programming language interface (API) and the Rendezvous daemon.

Rendezvous software includes several programming language interfaces, which are efficient, easy to use, and compatible with most other libraries (including window systems).

The Rendezvous daemon runs on each participating computer on your network. All information that travels between program processes passes through the Rendezvous daemon as the information enters and exits host computers. The daemon also passes information between program processes running on the same host.

Rendezvous programs are programs that use Rendezvous software to communicate over a network.

A Rendezvous distributed application system is a set of Rendezvous programs that cooperate to fulfill a mission.

TIBCO Rendezvous is delivered with the TIBCO iProcess Engine to enable the iProcess Engine to communicate with other TIBCO products, for example, TIBCO BusinessWorks and TIBCO iProcess Insight.

For More Information

For more information about TIBCO Rendezvous, see the TIBCO Technical Publications Document Library.

For information on configuring TIBCO Rendezvous for use with the TIBCO iProcess Engine, see the *TIBCO iProcess Engine Administrator's Guide*.

Back to Library

Chapter 2 Introduction to the TIBCO iProcess Engine

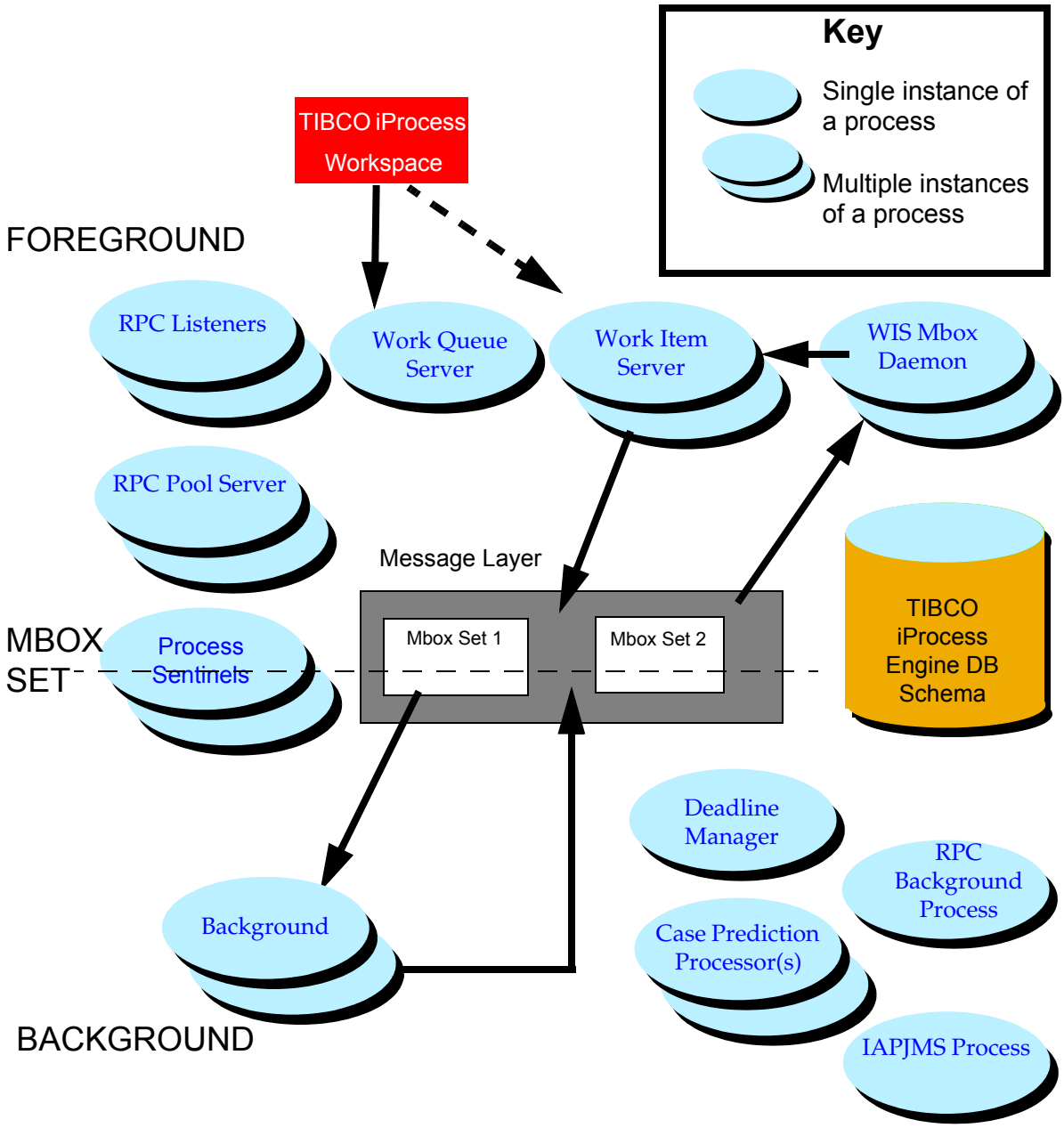
This chapter provides an overview of the TIBCO iProcess Engine architecture and the server processes that are involved to process business process information. It also contains some important TIBCO iProcess Engine concepts, which you need to be familiar with when administering the TIBCO iProcess Engine. This chapter describes:

- the role of the TIBCO iProcess Engine - see page 29.
- the physical architecture of the TIBCO iProcess Engine - see page 30.
- how TIBCO iProcess Workspace communicates with the TIBCO iProcess Engine - see page 31.
- the basic process structure of the TIBCO iProcess Engine - see page 33.
- the role of the TIBCO iProcess Engine Process Sentinels -see page 33.
- the concepts of work items and messages - see page 38.

Back to Library

TIBCO iProcess Engine Architecture

Click on a process to find out more information



Back to Library

The Role of the TIBCO iProcess Engine

The following example demonstrates the role of the TIBCO iProcess Engine in the business process automation environment. In a typical business process where an item is purchased from an office supplies company you would go through a number of steps:

1. Obtain authorization.
2. Allocate requisition number.
3. Submit expense claim.

When a case of the procedure is started by an individual from a computer running TIBCO iProcess Workspace, a number of processes are used to process the information contained in one step before the next step can be performed.

The TIBCO iProcess Engine maintains the list of work items in a user's work queue for all the active cases to be processed. If the first step was opened, completed and released, the TIBCO iProcess Engine determines the subsequent actions for the step and updates the necessary case data in the database.

In this example, after the Authorization step is released, the TIBCO iProcess Engine reads the procedure definition and determines that the next step is the Allocate Requisition Number. The addressee of this step is determined and a work item with the relevant form is sent back to the appropriate work queue for the TIBCO iProcess Workspace to display.

Back to Library

iProcess Physical Architecture

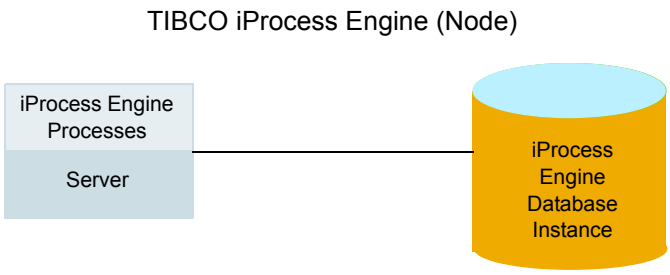
The following section describes:

- installing the TIBCO iProcess Engine on a single server
- installing the TIBCO iProcess Engine on mulitple servers

Installing the iProcess Engine on a Single Server

A TIBCO iProcess Engine installation on one server is known as a **TIBCO iProcess Engine node**. All the iProcess Engine processes are run on a single server.

The following diagram shows a TIBCO iProcess Engine (node) that comprises of one server.

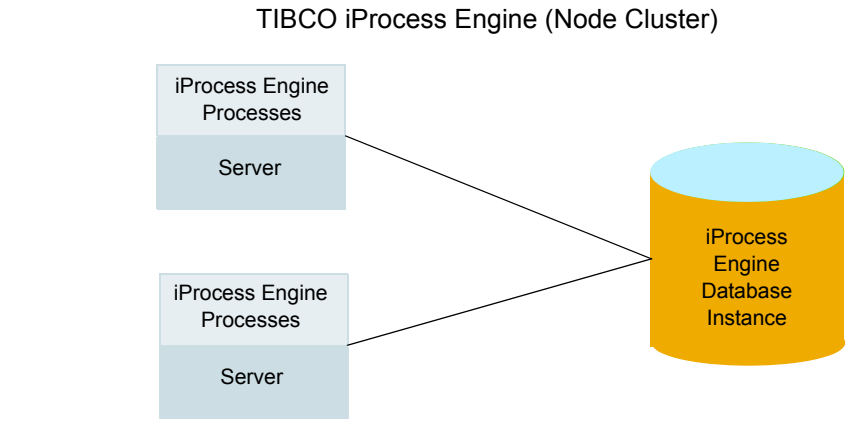


Installing the iProcess Engine on a Node Cluster

Installing the TIBCO iProcess Engine on multiple servers that all use the same database instance is known as a **node cluster** architecture. You can convert from a single-server to a node cluster at any time simply by adding another server to the installation. In the node cluster architecture, you can have a number of TIBCO iProcess Engine processes running on different servers, but they act as though they are a single iProcess Engine. Node clusters improve load balancing and performance. However, all the TIBCO iProcess Engine case data such as fields and their values in forms will be stored in one TIBCO iProcess Engine database instance.

Back to Library

The following diagram illustrates a TIBCO iProcess Engine (node cluster) that uses two servers to run TIBCO iProcess Engine processes. They can all access iProcess case data from the same TIBCO iProcess Engine database instance.



Note that:

- A complete installation of the iProcess Engine needs to be performed on each machine in the node cluster.
- All the iProcess Engines point to a single database instance. Within the database there is a single engine configuration that specifies which executables are configured to run on each server.
- The iProcess sentinels are configured to run on each machine. One server in the cluster is designated as the master server and the iProcess Sentinels on this server co-ordinate the startup, shutdown and management of the processes on all machines in the cluster.
- The iProcess Sentinels on the slave servers manage the executables designated to run on that machine.
- All servers that form part of the node cluster are required to provide full operation. See The iProcess Engine and Hardware Clustering on page 42 for information about failover capabililty for the iProcess Engine.

TIBCO iProcess Workspace and TIBCO iProcess Engine Communication

The TIBCO iProcess Engine uses a client/server model for communication where there is a two way communication path between each TIBCO iProcess Workspace client and TIBCO iProcess Engine server. The communication protocol used for this communication is called **Remote Procedure Call** (RPC).

Back to Library

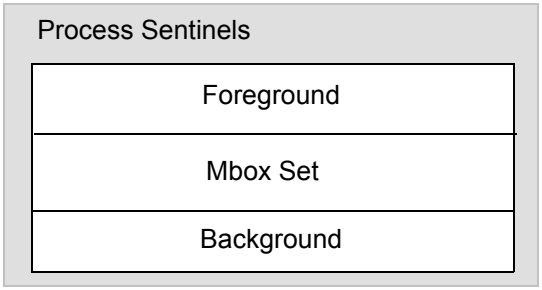
Information keyed in at the client needs to be passed to the TIBCO iProcess Engine for processing, for example when completing an insurance claim form, the new or updated data needs to be processed, stored, or updated so that subsequent steps can use the information. Information also needs to be passed from the TIBCO iProcess Engine to one of the TIBCO iProcess Workspaces, for example to tell the client what the next step in the business process is or show previous details about a work item that has already been entered.

Refer to Network Communication on page 123 for more information about TIBCO iProcess Workspace and TIBCO iProcess Engine communication.

Back to Library

TIBCO iProcess Engine Process Structure

In order to understand the flow of data through the TIBCO iProcess Engine and the relationship of TIBCO iProcess Engine processes, you can think of the TIBCO iProcess Engine as being split into the following four areas.



An TIBCO iProcess Engine process operates in either the foreground or background layer. The Mbox set provides the communication link between the foreground and background processes. This is the Messaging layer where instructions are sent from the foreground to the background and vice versa.

The foreground area deals with the user interaction, list of queues and their contents. The background processes perform the case processing.

The Process Sentinels keep control of all the processes to make sure they are always running.

Process Sentinels

The Process Sentinels (*SWDIR***etc****procmgr**) are responsible for controlling all the TIBCO iProcess Engine processes. If a node cluster architecture is used, then the Process Sentinels will exist on each server to manage the processes running on that server. This process operates in different ways depending on whether you are using a third party transaction processing monitor.

In a node cluster, the Process Sentinels have a master process on one of the servers, which manages and controls slave Process Sentinels on all the other servers in the node cluster.

The Process Sentinels control the safe start up and shut down of processes on one or multiple servers i.e. making sure processes are started in the correct sequence.

Back to Library

The Process Sentinels can also monitor processes to check that they are running and can automatically restart them if they fail. It can also detect if *SWDIR***logs****sw_error** and *SWDIR***logs****sw_warn** files have been created and sends a work item to the administrator to inform them that these files have been created.

Refer to Process Management on page 109 for more information about the Process Sentinels.

Foreground Processes

These processes are responsible for communicating with the TIBCO iProcess Workspaces and for passing any TIBCO iProcess Workspace requests such as released work items to the background area for processing.

The following table shows the processes that operate in the foreground:

Process Description	Logical Process Name
Work Queue Server	WQS
Work Item Server	WIS
WIS Mbox Daemon	WISMBD
RPC listeners	RPC_TCP_LI and RPC_UDP_LI
RPC pool server	RPC_POOL

Refer to Foreground Processes on page 60 for more information about what these processes do.

All of the foreground processes *must* operate on the master server. See Determining Where Processes Run on page 116 for more information.

Mbox Sets

The Mbox Sets are responsible for communicating information between the foreground and background processes. It is essentially the messaging system that stores and processes messages from TIBCO iProcess Workspace or TIBCO iProcess Engine requests. Mbox sets comprise of one or more message queues.

The messaging system used is:

- *in the UNIX Oracle and Windows Oracle versions*, Oracle AQ. This is a transactional messaging system provided by the Oracle database.

Back to Library

- *in the UNIX DB2 and Windows SQL Server versions*, the iProcess Suite's own transactional messaging system, which uses database tables provided by the DB2 or Microsoft® SQL Server database.

The messaging system provides the ability to store messages in tables. This means that message queues are highly scalable and have a guaranteed delivery mechanism.

See *iProcess Mbox Sets* on page 87 for more information about the Mbox sets.

Background Processes

The background processes are responsible for processing message instructions received from the clients such as releasing a step or forwarding a step. They also monitor and process any deadlines that have been set up in the procedure and manage case prediction.

The following processes operate in the background:

Process Description	Logical Process Name
Background	BG
Case Prediction	BGPREDICT
Database Queue Daemon	DBQD[a]
Deadline Manager	DLMGR
IAPJMS	IAPJMS
RPC Background	RPCBG

a. This process is only present on the DB2 version of the TIBCO iProcess Engine.

Refer to Background Processes on page 70 for more information about what these processes do.

Event Handling

The TIBCO iProcess Engine uses a publish/subscribe event mechanism to handle the following inter-process tasks:

- notifying processes to update caches.
- synchronization of process startup and shutdown.

Back to Library

The event handling mechanism used depends on the TIBCO iProcess Engine variant used, as follows:

- *UNIX Oracle version*. Events are handled using Oracle AQ's publish/subscribe interface.
- *Windows Oracle and Windows SQL Server versions*. Events are handled using the **Staffware Events** COM+ application, which is installed with the TIBCO iProcess Engine. All processes that want to subscribe to events register with the **Staffware Events** COM+ application.

If you are using a node cluster architecture, the **Staffware Events** COM+ application runs by default on the machine on which you installed the master server. If performance is or becomes an issue, TIBCO recommend that you dedicate a separate server, which is not running any TIBCO iProcess Engine processes, as the *event server*. This will reduce the load on the master server.

The name of the event server is stored in the Windows registry, so you will need to change this if you move the COM+ application to a new server. (See the *TIBCO iProcess Engine Administrator's Guide* for more information.)

- *UNIX DB2 version*. Events are handled using iProcess event/notification daemons.

  The event daemon is a single thread created by the Process Sentinel watcher process. The event daemon:

  — tracks which processes have subscribed to which events. (It maintains this information both in memory and in the **sw_subscription_list** database table.)

  — receives events when they are published.

  — forwards notification of an event to the notification daemon in each subscribed process.

  A notification daemon is a thread created by any TIBCO iProcess Engine process that wishes to receive event notifications.

Back to Library

Where is TIBCO iProcess Engine Case Data Stored?

All iProcess case data (the information provided by users in forms such as customer details, orders, etc.) is stored in the TIBCO iProcess Engine database instance. This means that all important iProcess data can be backed up and restored as part of your regular database backup policy.

Back to Library

How Do Work Items Appear in Work Queues?

For any work item to appear in a user's work queue, a work item record has to exist on the server for the TIBCO iProcess Workspace to know about it. The record contains information about the work item such as its case number, where in the database the fields and values are stored, the addressee and procedure, etc.

The WIS process contains work item records that define what work items are in the queue. The WIS reads the **staffo** database table which contains all the work item records where one row in the table represents one work item record. Each work item entry in the table contains specific information about the work item such as its case number, unique identifier, case description, name and node of case starter, etc. The TIBCO iProcess Workspace also calls the RPC server which makes requests to the database tables to retrieve fields and their values and any memos and attachments.

Sending Instructions From the TIBCO iProcess Workspace to the TIBCO iProcess Engine

When a user processes a work item by either Keeping or Releasing it, the TIBCO iProcess Workspace creates a package of instructions that are sent to the TIBCO iProcess Engine. This package contains all the information about the work item, such as what fields and data are shown on the form, what fields may have changed, and the instruction to be performed.

The message is sent from the TIBCO iProcess Workspace to the TIBCO iProcess Engine in packets via RPC calls. The package is split up into its various components so that an instruction message is sent to the background and any case data updates are made by updating the database. The instruction message is enqueued to a Mbox message queue where it waits to be processed by one of the background processes.

The Background process reads the message queues for new messages so it dequeues the message from the queue and then processes the case instructions. The Background performs the necessary actions such as update the changed fields in the database and determine what the next step is and who to send it to.

Back to Library

Mbox Sets

The message system is able to distribute messages across physical Mbox queues, which means that high volumes of messages can be handled. The Mbox queues are grouped into logical sets (Mbox sets) with each set configured for a specific purpose. Different processes require access to different Mbox sets, for example the instruction processor requires write access to a background Mbox, and WIS Mbox.

Message Queues

All TIBCO iProcess messages are posted to a central queue repository (Mbox). This is hosted by:

- *in the UNIX Oracle or Windows Oracle versions*, a set of tables using Oracle Advanced Queues (Oracle AQ).
- *in the UNIX DB2 or Windows SQL Server versions*, a set of iProcess queue tables in the database.

The SQL Server database queue tables can be created in the same database as iProcess, or a different database on the same SQL Server. Refer to the TIBCO iProcess Engine Installation Guide for more information about the location of database queue tables.

The Mbox can be configured to be a number of queues in a number of database tables, and even split across physical disks. Therefore, even though the Mbox appears as a single queue, it could consist of multiple queues in multiple tables across multiple disks.

The messaging system provides an assured delivery mechanism for instruction and request messages between the foreground and background.

Back to Library

User Access to iProcess Engine Work Queues

The total number of work queues a single iProcess user can have access to is **32,767**.

If the number of work queues a iProcess user has access to exceeds 32,767, the iProcess user is not able to login to the iProcess Engine and the following error message is displayed:

Sal Queues Interface 'sal_queues_count()' returned fatal error -*n*. Called by wqsess_lcreat_folder.

Abort WorkQs?

---

where *n* is the total number of work queues.

This only affects the single user who has access to too many work queues. All other iProcess users will be able to log in to the iProcess Engine without any problems.

An iProcess user can have access to the following work queues:

- Their **Personal Queue** which only they can access.
- Any **Group Queues** that they are a member of.
- Any **Test User Queues** or **Test Group Queues**, if the user is allowed to define procedures.

An iProcess user can also be given access to work queues in the following ways:

- By being a supervisor of a work queue.
- By being a participant of a work queue.

Back to Library

For example, the following table illustrates the total number of work queues a user called **swusr001** has access to:

Work Queues that swusr001 has access to:	
Personal Queue	swusr001
Group Queue	group1
Test User Queue	swusr001
Test Group Queue	group1
Supervisor of	swusr002, swusr003
Participant of	swusr002
Total number of work queues that swusr001 has access to:	7

From the example above, you can see that the number of work queues a user can have access to can quickly increase if that user is a member of several groups and is a supervisor or participant of several queues.

This means it is important to plan in advance the number of work queues you want to create and how many supervisors and participants you are going to need. For example, if you want to create 17,000 work queues on your TIBCO iProcess Engine, you could not make a single user both a supervisor and a participant of all 17,000 work queues. This is because that user would then have access to 34,000 work queues which would exceed the limit.

If you do want to create a large number of work queues and you want to have supervisors and participants for them, you would have to split them between more than one user. For example, if you had 17,000 work queues you could make:

- **swusr001** a participant of all 17,000 work queues
- **swusr002** a supervisor of all 17,000 work queues.

Back to Library

24*7 TIBCO iProcess Engine Operation

The TIBCO iProcess Engine is designed to continue running non-stop even when administration tasks need to be performed. This is achieved by:

- all case data being stored in a database. Online backup and restore is performed via your database tools.
- automatically restarting TIBCO iProcess Engine processes if they fail at any time. This is achieved by the TIBCO iProcess Engine Process Sentinels monitoring all the processes that are running to make sure they are running and if not, restart them.
- TIBCO Hawk, supplied with the iProcess Engine, a monitoring system that enables you to monitor the TIBCO iProcess Engine server processes. For example, you can use TIBCO Hawk to check what iProcess Engine server processes are running or stop and start them.
- New or updated procedures and processes can be deployed without affecting work in progress.

The iProcess Engine and Hardware Clustering

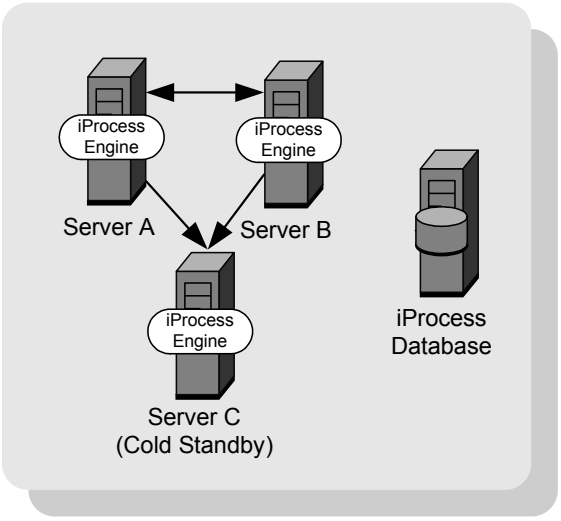
To cater for the loss of a physical server, the iProcess Engine can be deployed with a hardware cluster. The following section describes:

- Deploying the iProcess Engine in a hardware cluster for UNIX
- Deploying the iProcess Engine in a hardware cluster for Windows

Back to Library

Deploying the iProcess Engine in a Hardware Cluster for UNIX

There are two methods of deploying a hardware cluster, active-active and cold standby. The iProcess Engine is not supported in an active-active hardware cluster. However, it is supported in a cold standby hardware cluster. An example of a cold standby hardware cluster is shown below:



Each server is built in exactly the same way. Servers A and B are up and running and server C is a cold standby. If server A or B fails, server C is started and takes the place of the failed server.

To deploy the iProcess Engine in a cold standby hardware cluster:

- An iProcess Engine should be installed on each server in the cluster.
- If Server A or B fails, start the cold standby server and use *SWDIR***swadm move_server** to move the iProcess Engine processes from the failed to server to the cold standby server. See "Administering Servers" in the *TIBCO iProcess Engine Administrator's Guide* for more information about the *SWDIR***swadm move_server** command.

This can be done automatically by writing a script that uses the *SWDIR***swadm move_server** command.

Back to Library

Deploying the iProcess Engine in a Hardware Cluster for Windows

The iProcess Engine can be used with Microsoft Server Clustering when deployed in a Windows environment. Microsoft define a cluster as "a group of independent computers that work together collectively to provide a common set of services". When combining an iProcess node cluster with a Microsoft Server Cluster, each iProcess Engine node is contained within its own virtual server. Each virtual server is run on its own physical server. The iProcess Engine is installed on a shared disk that is accessible to all servers in the cluster. It is configured to work within the virtual server at installation by specifying a Cluster Network Name. If a physical server fails, the Microsoft Server Cluster automatically relocates the virtual server onto a secondary physical server and restarts the iProcess Engine.

To install the iProcess Engine in a Microsoft Windows Cluster, see either the *TIBCO iProcess Engine (Oracle) for Windows Installation* or the *TIBCO iProcess Engine (SQL) for Windows Installation* guide, depending on the version of the database you are using.

Back to Library

Chapter 3

Using the TIBCO iProcess Suite in a Multilingual Environment

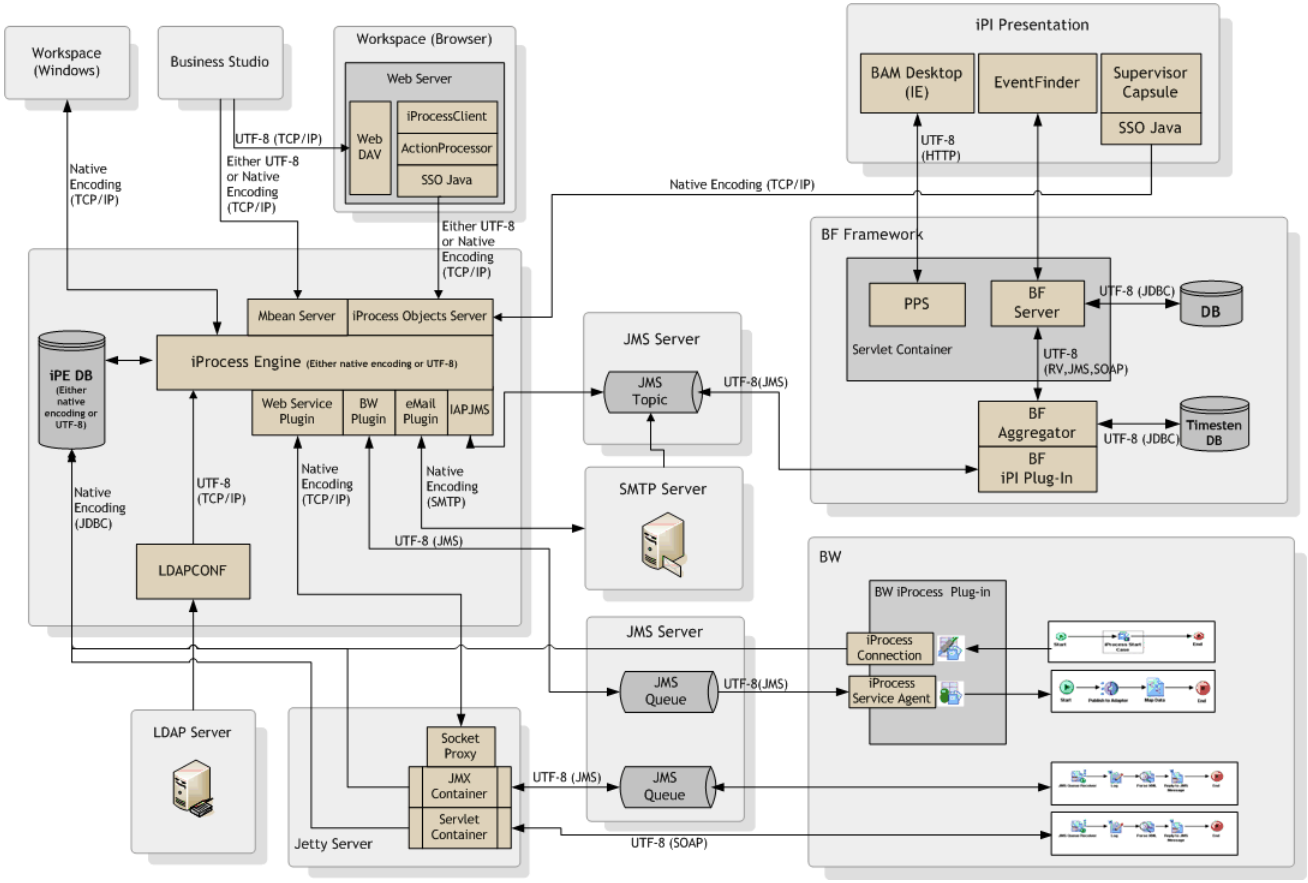
This chapter describes how globalization is supported by TIBCO iProcess Suite. It also describes the steps you should follow to implement TIBCO iProcess Suite in an international environment and the advantages and costs of the different ways in which TIBCO iProcess Suite can support globalization.

Back to Library

Overview

TIBCO iProcess Suite consists of many different products. A typical environment could be made up of any number or combination of iProcess products.

The diagram below illustrates how data is transmitted between products in the iProcess Suite.



Note that:

- The diagram indicates that some components, including the iProcess Engine can use either native encoding or UTF-8. Depending on the encoding used by other components in the iProcess Suite, the iProcess Engine performs the necessary data conversions. For information about setting the encoding attributes for iProcess components, refer to the documentation supplied with the components.

Back to Library

- The iProcess database can use either UTF-8 or native encoding. This is a choice that you can specify when you install or upgrade the iProcess Engine.
  - If you choose to use UTF-8 for the iProcess database, the iProcess Engine also operates in UTF-8.
  - If you choose not to use UTF-8 for the iProcess database, the iProcess Engine uses native encoding.

  See the *TIBCO iProcess Installation* guide for your operating system for setting this option. Before deciding on an encoding, make sure you understand the issues discussed in this chapter.
- Each component that communicates with an external application must use the same character set as the external application. See Implementing iProcess Suite in an International Environment with Native Encoding on page 58.

Back to Library

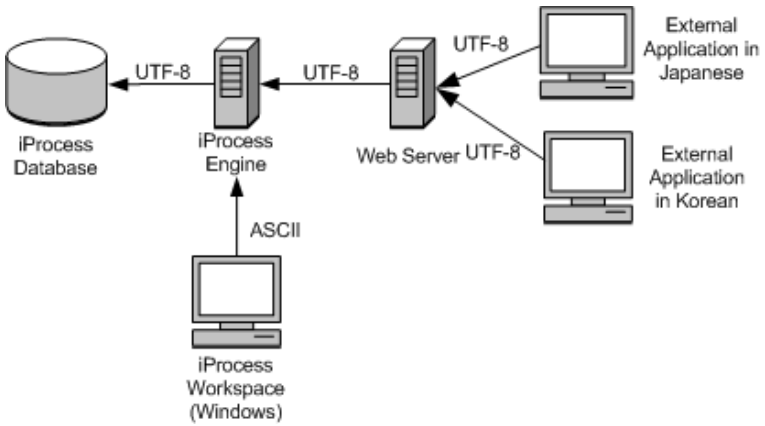
Globalization Options in the iProcess Suite

From version 11.1, the TIBCO iProcess Suite supports Unicode (UTF-8) character encoding. At installation time, you can enable UTF-8 support, or continue to use native encoding. See the TIBCO iProcess Engine Installation guide for your operating system for information on setting this option.

Advantages of UTF-8 Encoding

TIBCO iProcess Suite's support for UTF-8 (or UCS-2 in the case of SQL Server) enables you to work not simply in an international, but in a multi-national environment, using more than one multi-byte character set simultaneously.

This means that one TIBCO iProcess site can host multiple regions. For example, an enterprise operating globally could support data in Chinese, Japanese, Russian and English character sets in the same database. Work can be partitioned so that an end user working in this installation will normal see only work in his or her language:



In this example, you must ensure that users of the Japanese and Korean application do not receive each other's work items because the characters will not display correctly.

This is not possible without UTF-8 support because an iProcess Engine installation can support only one native character set. This option also simplifies using external UTF-8 applications. The conversions described in Globalization Support Using Native Encoding on page 56 are not necessary if the iProcess Engine supports UTF-8.

Back to Library

An additional advantage of using UTF-8 is that WIS searches using wildcard characters are also more consistent. For example, the expected results for filtering work items with the expression **a?c** would be items such as **aac**, **abc**, **acc**, and so on. However, searches are performed by a comparison of bytes, not characters, so actual results could include **a**, followed by a Japanese character that includes **c** in its byte representation, followed by **c**. This situation is avoided when the iProcess Engine uses UTF-8.

Issues with UTF-8 Encoding

While UTF-8 encoding is more versatile than alternatives, there are some issues of which you should be aware when deciding whether to use this encoding:

- Some other multi-byte character encoding systems are more efficient than UTF-8, in terms of the number of bytes they use per character. This means that converting from such a system, for example from Big5 encoding of Chinese characters to UTF-8, will result in a larger database. Perhaps more importantly, some items may overflow the space designed for them. For example, an increase from using 2 bytes to represent a character to 3 bytes means that a case description limited to 24 bytes can only contain 8 characters rather than 12 characters.
- The TIBCO iProcess Workspace (Windows) client does not itself use UTF-8 encoding. It converts data from the native character encoding for its locale to UTF-8 to send to the iProcess Engine, and similarly converts data it receives from the iProcess Engine from UTF-8. It is therefore possible that the same issue of space mentioned in the previous point could arise. Data entered in a native character encoding could end up being stored in the iProcess database as a larger number of bytes.

  For these reasons, if you are using UTF-8 encoding, TIBCO recommends using TIBCO Business Studio for modeling processes, and either TIBCO iProcess Workspace (Browser), or an application that utilizes SSO for processing work items.
- With multiple TIBCO iProcess Workspace (Windows) clients that use different locales, you must ensure that you do not address work items to users in different locales as unpredictable results can occur.

iProcess Names

Some internal names are supported in iProcess Engine only as 7-bit ASCII. These names are not normally seen by end-users, and generally do not exceed eight bytes in length. This affects the following items:

- Procedure name

Back to Library

- Field name
- Node name
- List name
- Table name
- Step name
- Attribute name
- Role name

If you access these internal names, do not use them to store data that exceeds 7-bit ASCII.

UserName is not affected by this limitation.

Back to Library

Recommendations

You should consider carefully whether you want to support UTF-8 encoding. While circumstances differ between individual cases, TIBCO recommends that:

- New iProcess Engine installations should always opt to use UTF-8 encoding. This enables your installation to convert to supporting multiple multi-byte character sets even if you don't need it now, at little cost.
- Existing iProcess Engine installations should upgrade based on the type of character encoding currently in use:
  - Existing iProcess Engine installations that use a single-byte character set should in most cases opt to use UTF-8 encoding. This ensures that you can convert to supporting multiple languages should it become necessary at a later date.
  - Existing iProcess Engine installations that already use a multi-byte native character set *should not* use UTF-8, because of the likely increase in database size and in the length of individual items, as well as the risk of data truncation in the conversion process (see Issues with UTF-8 Encoding on page 49).

Back to Library

Configuring the iProcess Suite Using UTF-8

When implementing an iProcess Suite installation in an multi-national environment using UTF-8 encoding, you need to consider the flow of data between the individual products that make up the iProcess Suite. Some products need particular settings configured in order to communicate with the UTF-8 character set.

Identify which parts of the iProcess Suite you are using and the physical locations of those products. The components that require consideration are:

- iProcess clients
- IAPJMS process
- LDAPCONF Utility
- iProcess Plug-ins
- TIBCO Business Studio

Back to Library

TIBCO iProcess Engine

The following configuration needs to be done on the TIBCO iProcess Engine:

- The IAPJMS process on the TIBCO iProcess Engine needs to be configured to use UTF-8 by setting the IAPJMS_LANGUAGE process attribute. Use the *SWDIR***util****swadm** utility as follows:

  ```
  swadm set_attribute 0 ALL 0 IAPJMS_LANGUAGE UTF-8
  ```

- On Oracle and DB2, the following environment variable must be set to specify the character set of the database:

  — **Oracle** If iProcess Engine is used with a database supporting UTF-8, set the variable **NLS_LANG** to **AL32UTF8**.

  — **DB2** If iProcess Engine is used with a database supporting UTF-8, set the variable **DB2CODEPAGE** to **1208**.

- Non-XML files (such as XFR or Abox files) are converted from the value specified in the environment variable **SW_FILE_ENCODING** or, if that is blank, from the encoding specified by the system locale. In most cases the encoding will conform to that specified by the system locale, so this environment variable can be safely left blank. However, in a situation where multiple iProcess Engine installations exist on the same machine and serve users with different language requirements, you may want override the locale

setting using the environment variable. For example, depending on your platform, you can set **SW_FILE_ENCODING** to UTF-8 as follows:

— In Windows

```
C:\> set SW_FILE_ENCODING=UTF-8
```

— In UNIX Korn Shell

```
$ export SW_FILE_ENCODING=UTF-8
```

— In UNIX C Shell

```
$ setenv SW_FILE_ENCODING UTF-8
```

— In UNIX Bourne Shell

```
$ SW_FILE_ENCODING=UTF-8;export SW_FILE_ENCODING
```

iProcess Clients

Any iProcess Clients, both iProcess Workspace (Windows) and iProcess Workspace (Browser), send to and receive from iProcess Engine data encoded in UTF-8. Any necessary conversions are carried out by the client.

TIBCO iProcess Workspace (Windows)

Using iProcess Workspace (Windows), no configuration is necessary. Conversions take place automatically, as follows:

- Sending data to iProcess Engine, iProcess Workspace (Windows) converts it from native encoding to UTF-8.
- Receiving data from iProcess Engine, iProcess Workspace (Windows) converts it from UTF-8 to native encoding.

TIBCO iProcess Workspace (Windows) determines the native encoding to use from the system locale to which it is set.

TIBCO iProcess Workspace (Browser)

If you are using iProcess Workspace (Browser), you must:

- On a UNIX system, set the **TISOUnicodeConverterName** environment variable to **UTF-8**.
- On a Windows system, set the **TISOUnicodeConverterName** registry entry (**HKEY_LOCAL_MACHINE\SOFTWARE\Staffware plc\Staffware SSO Client**) to **UTF-8**.

TIBCO iProcess Workspace (Browser) uses the **TISOUnicodeConverterName** value to determine what encoding to use for communications with iProcess Engine.

Back to Library

The LDAPCONF Utility

If you use the LDAPCONF Utility and the system default encoding is not UTF-8, ensure that option **[10] Enable Attribute Value Translation from UTF-8** option is deselected. If selected, this enables the conversion between the UTF-8 format used by LDAP and the system locale of the iProcess Engine. If iProcess Engine is using UTF-8, this conversion is redundant.

See "Setting up the Connection" in the *LDAPCONF Utility User's Guide* for details of this menu option.

TIBCO Business Studio

If you use TIBCO Business Studio, data sent from TIBCO Business Studio to iProcess Engine is converted to the encoding specified in the process attribute **DEPLOY_XSL_OUT_ENCODING**. Therefore you must:

- Set the iProcess Engine process attribute **DEPLOY_XSL_OUT_ENCODING** to **UTF-8** by using the SWDIR\util\swadm utility as follows:

  ```
  swadm set_attribute 0 ALL 0 DEPLOY_XSL_OUT_ENCODING
   UTF-8
  ```

  (The default is **ISO-8859-1**.)

iProcess Plug-ins

All iProcess Plug-ins send data to and receive it from the iProcess Engine in UTF-8 encoding. Any necessary conversion is carried out by the Plug-in.

Some iProcess Plug-ins do not require configuration for UTF-8 support; only those that require configuration are listed in this section.

iProcess BusinessWorks Plug-in

The **defaultEncoding** attribute in the *SWDIR*/**eaijava/encoding.properties** file must be set to the encoding used by the iProcess Engine. Therefore:

- Set the **defaultEncoding** attribute in the *SWDIR*/**eaijava/encoding.properties** file to **UTF-8**.

iProcess Web Services plug-in

The **WSDocumentHandler.Encoding** attribute in the **wsconfig.properties** file must be set to the encoding used by the iProcess Engine. Therefore:

Back to Library

- Set the attribute **WSDocumentHandler.Encoding** in the *WebServiceHome*/**jetty-6.1.1/staffware/wsconfig.properties** file to **UTF-8**.

iProcess Email Plug-in

The **Charset** attribute in the **eai_mail.cfg** file must be set to the encoding used by the iProcess Engine. The iProcess Email plug-in embeds this attribute in the encoded-words streams for the MIME header fields **subject**, **to address**, **cc address**, **from address** and others. Therefore:

- Set the attribute **Charset** in the *SWDIR*/**lib/eai_mail.cfg** file to **UTF-8** for the Windows platform.
- Set the attribute **Charset** in the *SWDIR*/**libs/eai_mail.cfg** file to **UTF-8** for UNIX platform

Back to Library

Globalization Support Using Native Encoding

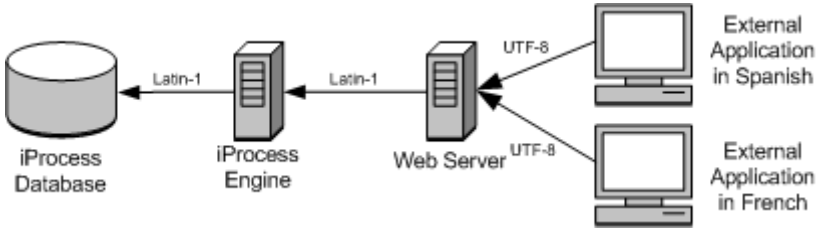
The following sections describe how an iProcess Suite installation that uses native encoding rather than UTF-8 encoding. This type of environment supports either of the following:

- a single-byte native character encoding environment
- a multi-byte native character encoding environment

Some native multi-byte character sets are referred to as double-byte character sets (such as BIG-5 or Shift-JIS).

Using iProcess Suite in a Single-Byte Native Encoding Environment

The following diagram shows iProcess Suite in an international environment that uses a single-byte native character encoding environment (in this example, Latin-1).



In this example:

- When text is entered, in either Spanish or French, into the external applications, the data is encoded using UTF-8. UTF-8 is used to send the data from the external applications to the web server.
- The web server converts the data from UTF-8 to Latin-1 and sends the data to iProcess Engine.
- iProcess Engine inserts the data into the iProcess Database.
- The iProcess Database stores the data as Latin-1.

Note that:

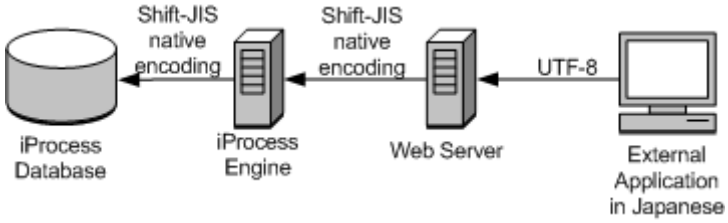
- iProcess Engine supports a variety of character sets as its native character encoding. However, iProcess Engine still needs to handle data in other encodings. Therefore, iProcess Engine provides the functionality to convert different character encodings into its native character encoding.

Back to Library

- It does not matter if the machines hosting the iProcess Database, iProcess Engine, and iProcess Web Services Server and Client Plug-ins are all in separate locations to the machines hosting the external applications.

Using iProcess Suite With a Multi-Byte Character Encoding Environment

The following diagram shows the iProcess Suite in an environment that uses one multi-byte native character set.



In this example:

- When text is entered in Japanese into the external application the data is encoded using UTF-8. UTF-8 is used to send the data from the external application to the web server.
- The web server converts the data from UTF-8 to Shift-JIS (native encoding) and sends the data to iProcess Engine.
- The iProcess Engine inserts the data into the iProcess Database.
- The iProcess Database stores the data as Shift-JIS (native encoding).

Note that:

- iProcess Engine supports a variety of character sets as its native character encoding. However, iProcess Engine still needs to handle data in other encodings. Therefore, iProcess Engine provides the functionality to convert different character encodings into its native character encoding.
- It does not matter if the machines hosting the iProcess Database, iProcess Engine, and iProcess Web Services Server and Client Plug-ins are all in separate locations to the machines hosting the external applications.

Back to Library

Implementing iProcess Suite in an International Environment with Native Encoding

When implementing the iProcess Suite in an international environment, you need to consider the flow of data between the individual iProcess products that make up the iProcess Suite. Each iProcess product that needs to communicate with an external application needs to be configured to use the same character set as the external application.

An overview of steps you need to follow is:

1. Identify which iProcess products you are using and the physical locations of those products. The components that require configuration for encoding are:
   - your host system locale
   - iProcess Engine database encoding
   - iProcess Email Plug-in
   - iProcess Workspace (Browser)
   - iProcess Technology Plug-ins
   - iProcess Web Services Plug-in
   - LDAPCONF Utility
2. On the machines that are hosting the iProcess product components, check that host system locales are using the correct character sets.
3. Set the encoding attributes for each component. Refer to the Installation or user guides supplied with each component for information on how to do this.
4. 4. Non-XML files (such as XFR or Abox files) are always converted from the encoding specified by the system locale. The environment variable SW_FILE_ENCODING does not work in native mode.

Back to Library

Chapter 4 TIBCO iProcess Engine Processes

This chapter provides detailed information about the TIBCO iProcess Engine processes and how they interact with each other. The iProcess Engine Process Sentinels are responsible for starting and stopping these processes and making sure that they keep running. The Process Sentinels are described in Chapter 9 on page 109.

Back to Library

Foreground Processes

The following table summarizes the function of each foreground process.

Process Description	Logical Process Name	Process Executable	Function
Work Queue Server	WQS	wqsrpc	The WQS is responsible for providing a complete list of work queues on the system, along with the RPC addresses (RPC number and machine name) of the WIS that handles each queue. It contains the list of all the users and groups and controls who has access to each queue. Refer to page 61.
Work Item Server	WIS	wisrpc	The WIS processes cache the contents of the work queues and provide lists of work items to TIBCO iProcess Workspace. The WIS processes concurrently produce messages (under transaction control) to be placed in the background Mbox set. Each WIS process receives messages from the WISMBD. These will be stored temporarily in an in-memory buffer (per physical work queue) for later retrieval when the WIS performs an update from the in-memory Mbox. Refer to page 65.
WIS MBOX Daemon	WISMBD	wismbd	These are responsible for dequeuing/reading the messages from the WIS Mbox sets and passing them to the appropriate WIS that is handling the item. Refer to page 71.
RPC Pool Servers	RPC_POOL	swrpcsvr	These manage and allocate the RPC connections. For each user that logs into iProcess using the TIBCO iProcess Workspace, a connection to a swrpcsvr pool process is allocated by the listener. Each pool process maintains a pool of connections.

Back to Library

Process Description	Logical Process Name	Process Executable	Function
RPC Listener (UDP)	RPC_UDP_LI	swrpcudp	For each user that logs into iProcess using the TIBCO iProcess Workspace, a connection to a swrpcsvr pool process is allocated by the listener.
RPC Listener (TCP)	RPC_TCP_LI	swrpcsvr	

All of the foreground processes *must* operate on the master server. See Determining Where Processes Run on page 116 for more information.

Work Queue Server

The iProcess work queues, which contain all the user's work items, are managed by the following processes:

- **Work Queue Server (WQS)**, which handles the listing of queues. This process is run by *SWDIR***etc****wqsrpc**. There is only a single **wqsrpc** process running at any time.
- **Work Item Server (WIS)**, which handles the listing of work items in the queues. This process is run by *SWDIR***etc****wisrpc**. The number of **wisrpc** processes running is configured by the Process Sentinels (**process_attribute** table).

The WQS process handles what is displayed in the left hand pane of the Work Queue Manager (the queue list) and the WIS process handles the contents of the right hand pane (the work items list).

The work queue processes are started automatically when the other TIBCO iProcess Engine processes (such as the RPC pool servers) are started, and stopped when the other TIBCO iProcess Engine processes are stopped. The Process Sentinels start (and stop) processes in a specific sequence to make sure that processes have all their dependent processes running.

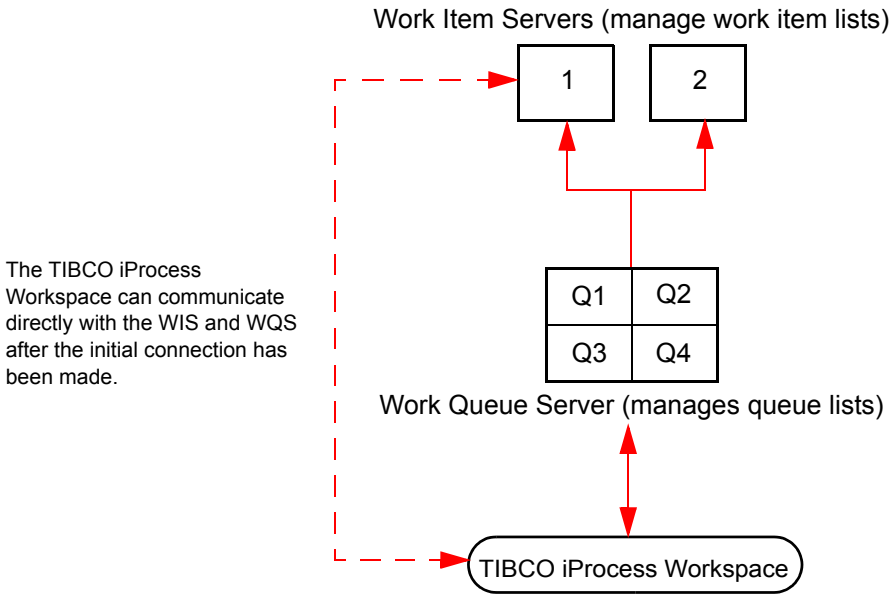
When a TIBCO iProcess Workspace logs in to the TIBCO iProcess Engine, the following sequence of events occur:

1. The TIBCO iProcess Workspace communicates with the RPC listener process (**RPC_TCP_LI** or **RPC_UDP_LI**).
2. The RPC listener process provides the RPC number of a pool server (it can start a new pool server if required).
3. The TIBCO iProcess Workspace connects to the pool server.

Back to Library

4. The RPC listener provides the RPC number of the Work Queue Server that the TIBCO iProcess Workspace then connects to.
5. The TIBCO iProcess Workspace communicates with the WQS.
6. The WQS provides the RPC numbers of the Work Items Servers that serve the work queues in the TIBCO iProcess Workspace.

The TIBCO iProcess Workspace communicates with the WQS and the WIS via RPC.



Refer to TIBCO iProcess Workspace and TIBCO iProcess Engine Network Communication on page 124 for more information about RPC calls and how the TIBCO iProcess Workspace and TIBCO iProcess Engine communicate over the network.

The Work Queue Server controls which user and group queues a user will see on their TIBCO iProcess Workspace and which WIS process is handling the queue. The WQS provides a mapping to the appropriate work queues that are held in the WIS processes. Each TIBCO iProcess Engine instance runs with one WQS process.

Shared memory is used for caching this information. This memory is released when the instance has finished.

Back to Library

The WQS allocates work queues to the WIS processes that are running using either *round robin* or *on-demand* allocation. You can configure which allocation method is used by modifying the WQS_ROUND_ROBIN parameter in *SWDIR***etc****staffcfg**.

Allocation of Work Queues to WIS Processes

The WQS process performs the work queue allocation. The WQS reads the list of users and groups from the database and sorts them alphabetically and allocates them to a particular WIS. The WQS allocates which WIS a work queue is sent to in one of two ways.

Round Robin Queue Allocation

The WQS allocates a work queue to each WIS alphabetically, cycling round until all the work queues are allocated. For example, if a system has 5 WIS processes and 15 work queues (A-O) then the following allocation is performed:

- Queues A,F,K are allocated to WIS 1
- Queues B,G,L are allocated to WIS 2
- Queues C,H,M are allocated to WIS 3
- Queues D,I,N are allocated to WIS 4
- Queues E,J,O are allocated to WIS 5

This method of allocation takes no account of queue size so it is best used when queues are evenly distributed with work items and user access is evenly spread.

On-Demand Queue Allocation

This method allocates work queues alphabetically but only to the first available WIS. Therefore if a WIS is allocated a large work queue, it will take some time before it is ready to accept another queue. This means that other WIS processes that have smaller queues can accept more queues.

The effect of using on-demand allocation is that a more even distribution of work is achieved. However, the initial allocation is based upon the initial loading size of each queue so it may not be representative of the amount of requests to that work queue.

As both methods allocate work queues alphabetically, it is possible to have some control over queue allocation by careful naming of your queues. For example, if all the larger queues are first alphabetically, the early WIS processes will start getting these and the later WIS processes can pick up the smaller queues.

Back to Library

Controlling the Assignment of Queues to WIS Processes

There are two additional methods you can use to customize the assignment process to better reflect your system requirements, and so optimize performance.

- Use different WIS processes to handle user queues and group queues.

  User queues and group queues frequently have different characteristics, in terms of the amount of load they carry. For example, if group queues are far more active than user queues on your system, you may want to give them higher priority for WIS allocation.

- Assign a queue explicitly to a WIS process

  If you have certain queues that are very large or very busy, you may find it useful to dedicate specific WIS processes to handling *only* those queues (leaving the remaining queues to be dynamically assigned to the remaining WIS processes).

Refer to "Administering the Work Queue Server and Work Items Server" in the *TIBCO iProcess Engine: Administrator's Guide* for more information.

RPC Pool Server

This process is responsible for handling RPC requests from a TIBCO iProcess Workspace to access and update data in the iProcess Engine instance.

A number of RPC pool servers can be created by the RPC Listener when the TIBCO iProcess Engine is started and each RPC server will be responsible for a configured pool of TIBCO iProcess Workspace connections. You can set up a number of pool servers that are pre-loaded if you have lots of TIBCO iProcess Workspaces logging in quickly. You do this by defining the PRE_LOAD_POOL_SERVERS parameter in the *SWDIR***etc****staffcfg** file. TIBCO iProcess Workspaces can be allocated to pool servers using either a round robin or load balanced method.

The RPC Pool servers are started by the RPC TCP Listener. The number of users that each pool server can support is configured using the MAX_USERS_PER_PROCESS parameter in the *SWDIR***etc****staffcfg** file.

RPC Listeners

The RPC Listeners are started by the Process Sentinels and are the first TIBCO iProcess Engine foreground server processes to be started. A listener is started for both TCP and UDP protocols. The RPC number for the Listener process is the same for TCP and UDP and is a start-up configuration parameter for the TIBCO iProcess Engine. Line 11 of the *SWDIR***swdefs** file defines this RPC number.

Back to Library

This RPC number is the published initial connection port for a TIBCO iProcess Engine for use by any client applications built using a TIBCO iProcess Workspace interface (e.g. the iProcess Applications Layer).

Work Item Server

The WIS handles the listing of work items in the queues. The process executable is *SWDIR***etc****wisrpc**. A number of WIS processes can be run and this is controlled by the Process Sentinels.

A WIS process is one instance of a Work Item Server and caches work queues that the WQS has allocated to it. Every iProcess work queue is hosted by a WIS process and each WIS can process more than one work queue. The iProcess Engine runs two WIS processes by default but you can increase or decrease this number using the *SWDIR***util****swadm** utility.

The WIS processes are multi-threaded processes. Different threads are used to perform different tasks - for example, responding to RPC requests, caching queue information, filtering queues or updating CDQP information.

Version 10.4 of the TIBCO iProcess Engine introduced multi-threaded WIS processes. This improved WIS performance significantly, making it possible to reduce the default number of WIS processes from 6 (the pre-Version 10.4 default) to 2.

There are many work queue performance issues related to the number of WIS processes you have, how many work queues they process, how many threads they use for different tasks and so on. Refer to "Administering the Work Queue Server and Work Item Servers" in the *TIBCO iProcess Engine: Administrator's Guide* for more information.

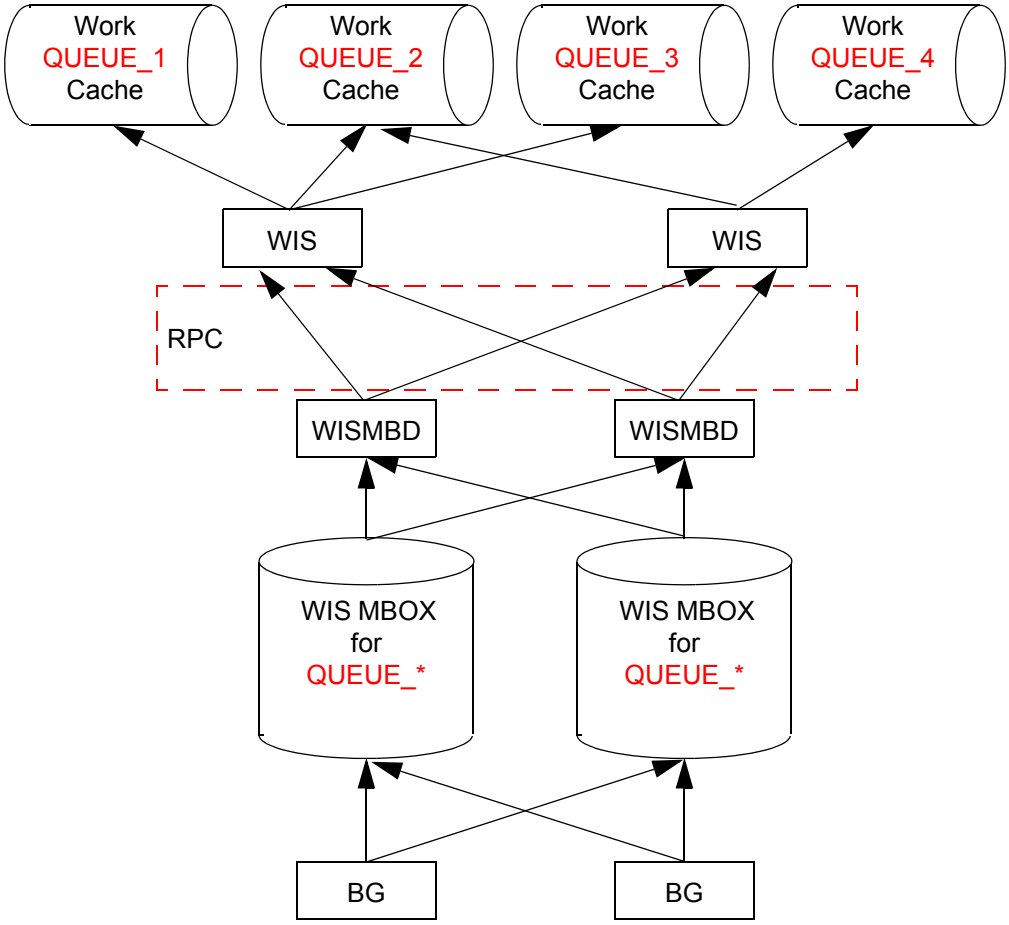
The WIS processes maintain a cache of the information they contain (which is the user's work queue). This cache is synchronized with the same information stored in the user or group's work queue (**staffo** database table). You can view the information in this table using *SWDIR***util****plist -m**.

WIS Mbox Daemon

This process (**WISMBD**) operates between the WIS Mbox set and the **WISRPC** processes forwarding messages from one to the other. The executable is *SWDIR***etc****wismbd**.

The WISRPC processes do not write outgoing messages via the WISMBD, they go straight to the Background Mbox set.

Back to Library



The **WISMBD** process is configured to read from a configurable number of physical WIS Mbox sets in a round-robin manner and it will deliver the messages to the appropriate WIS process.

Back to Library

When the WIS Mbox sets are empty, the **WISMBD** will sleep for a configurable amount of time. This is defined by the EMPTYMBOXSLEEP, EMPTYMBOXSLEEP_INC and EMPTYMBOXSLEEP_MAX process attributes. Refer to "Administering Process Attributes" in the *TIBCO iProcess Engine: Administrator's Guide*.

The WQS and WIS processes are started up before the WISMBD processes so that no work queues will receive messages until ALL the WIS processes have started.

The **WISMBD** sends synchronous RPC requests to the WIS that maintains the work queue to which the message is addressed.

The **WISMBD** is initially set to read from the following Mbox sets:

- MBSET_READ_WIS - This is used to read Mbox messages
- MBSET_WRITE_BG - This is used to forward undelivered work items back to the local background for locally hosted procedures.

Back to Library

Mbox Sets and Message Queues

The foreground and background processes communicate with each other using messages. Messages contain information about the iProcess case and instructions about what to do with the case (such as release, keep or forward it.) A message is processed by the case instruction processors. **Mbox set** is the generic name used for a container in which these messages are stored. Processes can dequeue messages from a Mbox set as and when resources are available.

The messages are stored and managed in the following way:

- *in the UNIX Oracle and Windows Oracle versions*, using Oracle AQ. Oracle AQ uses message queues, which are defined as Oracle AQ tables.
- i*n the UNIX DB2 and Windows SQL Server version*, using iProcess database tables that are managed by the database server.

Please see the appropriate TIBCO iProcess Engine Database Administrator guide for more information about the format of the AQ or database tables that are used to hold the message queues.

The TIBCO iProcess Engine uses a logical and physical grouping for queues to help improve message throughput. The physical message queues are grouped together as logical Mbox sets.

An Mbox set can be used for different purposes, for example two Mbox sets can be created for the Backgrounds. The Deadline Manager (**DLMGR**) can write to one set while other processes could write to the second set. The relative priority of processing the work can be changed by assigning different numbers of background processes to dequeue the messages in the different Mbox sets.

Processes can be configured to enqueue or dequeue messages to/from a specific Mbox set rather than to/from a single Mbox queue. The sets can be changed as the performance demands change in your system environment. Multiple queues in an Mbox set enable processes to distribute messages in a round robin manner resulting in an even distribution of messages on the queues.

Refer to Chapter 6 on page 87 for more information about Mbox sets.

Back to Library

Transaction Control of Messages

The Mbox provides a message repository so that processes can post messages and continue with their own processing without having to wait for a reply. The messages are stored persistently and are processed only once. Messages have to be processed exactly once to preserve the data integrity of a transaction. For example, if a message instruction is to debit an amount from a bank account then this has to happen once only even if the systems fail. This is achieved using transaction control.

The messages in the database tables provides a reliable messaging system because the message queues are under transaction control. For example, if a message is delivered to the background case instruction processor but the server goes offline for a while, the message instruction is still persisted in the queue and is retried at a later time.

Back to Library

Background Processes

The following table summarizes the background processes:

Process Description	Logical Process Name	Process Executable	Function
Background	BG	swbgmd	This is the core background process that interprets the business rules that have been defined in the iProcess Modeler. It performs the case instructions such as working out what the next step is in the procedure, updating the database with new information and checking for deadlines. This process is also responsible for dequeuing messages from the Mbox sets. There can be multiple background processes concurrently dequeuing messages. This process can be configured to read a specific Mbox set as defined by the **process_attribute** table.
Case Prediction	BGPREDICT	swbgmd	This process is responsible for updating prediction data (stored in the database).
Database Queue Daemon	DBQD[a]	n/a	This process: • caches a configurable number of messages from the database for each available queue. • processes RPC requests to dequeue messages for the **BG** and **WISMBD** processes.
Deadline Manager	DLMGR	dlmgr	This process is responsible for monitoring deadlines in cases of procedures.

Back to Library

Process Description	Logical Process Name	Process Executable	Function
IAPJMS	IAPJMS	iapjms	This process is responsible for receiving messages containing activity monitoring information from the **BG** process, and Work Queue Delta publication messages from the **WIS** process, and routing these to the specified JMS topic.
RPC Background	RPCBG	staffrpcbg	This process handles the Jump To and Case Suspend features in the iProcess Suite.

a. This process is only present in the DB2 version of the TIBCO iProcess Engine.

Back to Library

Background

This process retrieves messages (containing case instructions) from the Mbox sets and then processes the case instructions in the messages. There can be multiple Background processes all concurrently dequeuing messages from an Mbox set and processing case instructions

The number of Background processes is controlled by the Process Sentinels. The Process Sentinels read the **process_config** table to see how many instances of the process to run and on which computers.

Each message contains the case instructions from which the Background can determine what actions to take. The Background interprets the business rules defined in the procedure (such as the addressee of the next step) and routes work items to the necessary work queues or external applications. The process makes decisions based upon the iProcess data and procedure definition instructions as to what happens in the business process next.

Case Prediction Processor(s)

This process (*SWDIR***etc****bgpredict**) receives messages from the iProcess Background processes. When a case instruction that results in a change to a case has been processed, the iProcess Background processes notify the Case Prediction processes so that the prediction data (stored in the database) can be updated. There can be multiple case prediction processes running concurrently.

Each message contains information about the case that has changed and the procedure and instruction that caused the change. The process will read messages from the queue(s) and update the **predict** table so that it contains the latest prediction information about the case that the queued message was for.

The process attempts to determine a valid addressee for each step, so that SW_USER:*attribute* (where attribute is a user attribute, for example, DESCRIPTION) resolves to something valid. If there are multiple addressees on a step then the first one is taken, which is resolved in the following order:

1. First the process checks the user and group addressees and uses the first one in the list. If it finds a group it gets the first member listed in the group.
2. If no user or group was found it next checks the field addressees.
3. Finally it checks the roles.

If no addressee can be determined, if a group has no members, or a field or role contains an invalid user, then the process defaults to using the system administrator user (by default, **swadmin**).

Database Queue Daemon

This process is only currently used on the DB2 version of the TIBCO iProcess Engine.

This process (*SWDIR***etc****swdbqd**) processes RPC requests from the **BG** and **WISMBD** processes to dequeue messages from the database queue tables.

It caches a configurable number of messages from the database for each available queue. When a request to dequeue a message arrives, the process returns a message from the cache. If the cache is empty, the process first refills the cache from the database queue tables.

The number of messages to be cached is determined by the DBQD_MAX_CACHED_MESSAGES process attribute.

Deadline Manager

This single process manages the deadlines that have been defined in a procedure using the iProcess Modeler.

At defined intervals (deadline processing interval), the Deadline Manager checks the **outstanding_addressee** table for expired deadlines. If deadlines have expired, the Deadline Manager sends an Mbox instruction to the background Mbox set so that the case instruction process can process the deadlines for the case.

You can define a limit to the number of Mbox instructions the Deadline Manager sends. This is to avoid the Deadline Manager sending out duplicate Mbox instructions for the same unprocessed, expired deadlines. There are two process attributes that enable you to configure this:

Back to Library

- The UNPROCESSED_DL_POST_LIMIT process attribute sets a limit on the number of messages for expired deadlines that the Deadline Manager allows in the mbox queue at any one time.
- The MAX_AGE_BEFORE_RESETPOST specifies the time period before the Deadline Manager resets its internal marked of the last deadline it has processed to 0 (beginning of time).

Refer to "Administering Process Attributes" in the *TIBCO iProcess Engine: Administrator's Guide* for more information about process attributes.

When all the deadlines have been processed, the Deadline Manager "sleeps" until the deadline processing interval has expired. This interval can be set as an absolute or repeating value, for example:

- Absolute intervals are used to process deadlines once a day at a specific time.
- Repeating intervals are used to process deadlines at regular intervals and at set times throughout the day relative to midnight each day. The time is defined on the local computer on which the deadline manager is running.
- For example, an interval of five hours would set deadlines to be processed at 5.00am, 10.00am, 3.00pm and 8.00pm on Day 1 and 5.00 am, 10.00 am etc. on Day 2.

The deadline processing interval is set in the **process_attribute** database table. If the table does not contain an initial value, the Deadline Manager defaults to one minute.

IAPJMS Process

If activity monitoring is enabled on your TIBCO iProcess Engine, the **BG** process sends out a message when any of the TIBCO iProcess Engine activities that you have configured to monitor occur. This process (*SWDIR***etc****iapjms**) is responsible for receiving messages from the **BG** process and routing these to the JMS topic or queue.

If Work Queue Delta publication is in use, the **WIS** process similarly publishes messages to the **IAPJMS** process giving details of changes in monitored work queues.

See Monitoring Activities on page 91 for more information and see "Activity Monitoring and Work Queue Delta Configuration" in the *TIBCO iProcess Engine: Administrator's Guide* for more information about configuring activity monitoring and Work Queue Delta publication.

Back to Library

RPC Background Process

This process (*SWDIR***etc****rpcbg*) handles synchronous RPC calls from the Jump To and Case Suspend features in the TIBCO iProcess Workspace.

Back to Library

Chapter 5 Introduction to Transactional Business Process Automation

This chapter provides an overview of the transaction management capabilities provided when using the iProcess Engine.

Back to Library

Overview

Typically, a business process involves a number of transactions such as looking up information from a customer database, taking an order, adding a new order to a database, updating prices, etc. To guarantee that all parts of the business process are completed, a resource manager is used to keep track of each step (or transaction) to make sure it is completed.

If all the steps can be completed successfully, then the complete process will be committed. This means that any updates to external systems and iProcess case data will be performed and committed in a single transaction.

However, if one or more parts of the process cannot be completed for any reason (such as a database being offline), no parts of the process are committed. This means that all the external systems such as databases, document management systems and legacy systems are left in the same state as when the process was started i.e. no data updates are performed.

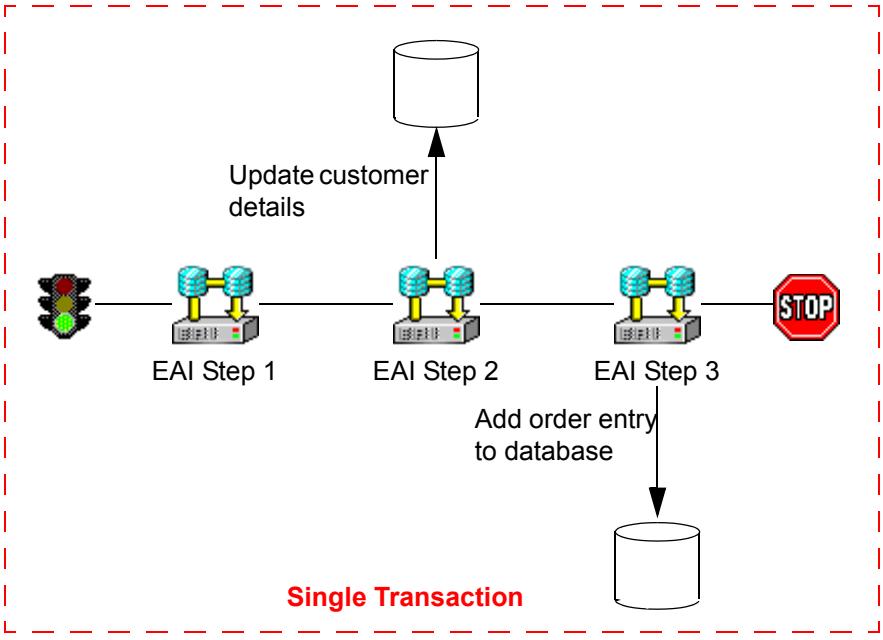
There are two types of transaction scope: local and distributed. The difference between them is the number of resource managers used to control the transaction. A local transaction uses just one resource manager and a distributed transaction involves using more than one resource manager. Both of these are described in the following sections.

Back to Library

What is a Local Transaction?

A local transaction is where a number of business operations are under the control of a single resource manager (typically the database resource manager). The TIBCO iProcess Engine process will be under the control of the resource manager so that the entire process will either be committed to the database or rolled back if one of the operations fail. The following diagram illustrates a procedure using EAI Database steps to update data in the local database.

Figure 1 A Local Transaction



Back to Library

Example of a Local Transaction

The following is an example of how a business procedure is designed so that all the steps are either committed or not committed using the local resource manager.

In a banking environment you want to make sure that all the accounts are updated correctly so that the balances are totalled. The ideal scenario is where all the processes are committed in one go so that money is deducted from one account and added to another. For example, if money is to be transferred from a savings account to a current account, two accounts will need updating. Both accounts need to be changed accordingly. It would cause serious problems if only one account was updated.

If the update to the savings account was successful but the update to the current account failed, there would be a discrepancy in the totals. However, because the step is part of the local transaction, the whole transaction will fail and the accounts are rolled back to their original state.

Back to Library

What is a Distributed Transaction?

A distributed transaction is where a number of related business operations (individual transactions) are grouped together so that they are put under the control of a transaction manager program (such as MSDTC). The integrity of the business data is maintained because all external systems being used (databases, document management systems, etc.) are either updated with information or left in their previous state if one or more of the steps fails.

If all the business operations (transactions) are successful, they can all be committed in one go. The information is stored in memory until the final transaction is reached when all the updates can be committed.

Back to Library

Transaction Scope

The following sections describe the scope of transactions on the different versions of the TIBCO iProcess Engine.

Oracle Server Transaction Scope

The Oracle version of the TIBCO iProcess Engine enables you to use one local resource manager (provided by the Oracle resource manager).

Because the TIBCO iProcess Engine uses Oracle Advanced Queues to store its internal instruction messages and Oracle tables to store case data, transaction management can be controlled by the internal Oracle database transaction manager.

This means that if updates to the Oracle database fail, the resource manager can roll back the updates and restore the database and case data to its original state.

DB2 Transaction Scope

The DB2 version of the TIBCO iProcess Engine enables you to use the local DB2 resource manager.

This means that if updates to the DB2 database fail, the resource manager can roll back the updates and restore the database and case data to its original state.

SQL Server Transaction Scope

When using the iProcess Suite with the Microsoft SQL Server, all transactions are managed by the SQL Server resource manager. SQL Server provides the following transaction functionality:

- Locking facilities that ensure transactional integrity.
- Logging facilities that ensure transaction durability. For example, if the hardware, operating system, or SQL Server fails, SQL Server uses the transaction logs, upon restart, to automatically roll back any uncompleted transactions to the point of the system failure.
- Transaction management features that enforce transaction atomicity and consistency. After a transaction has started, it must be successfully completed, or SQL Server undoes all of the data modifications made since the transaction started. Refer to the SQL Server documentation for more information about how SQL Server transactions work.

Back to Library

When all the processes use the local resource manager on the SQL Server, this is called a local transaction. If some processes use an external resource manager such as the EAI COM step, then this is known as a distributed transaction because more than one resource manager is used. A slightly different architecture is used for distributed transactions because a transaction manager program is used to control the resource managers. Refer to Using Distributed Transactions with MSDTC on page 81 for more information.

Using Distributed Transactions with MSDTC

Windows has a built in transaction manager component called Microsoft Distributed Transaction Coordinator (MSDTC). The MSDTC controls resource managers from distributed sources such as SQL Server and COM.

The transaction scope provided with this architecture allows distributed transactions involving many resource managers to be controlled by the MSDTC. Some server EAI plug-ins such as EAI COM+ may require a distributed transaction and therefore need to use MSDTC. You can set the TIBCO iProcess Engine to use MSDTC by setting the EAI_NEEDS_MSDTC process attribute - refer to “Administering Process Attributes” in the *TIBCO iProcess Engine: Administrator’s Guide* for more information about setting process attributes.

Back to Library

Using Enterprise Application Integration Steps in Procedures

To keep control of external transactions with third party applications and to keep the transactions under the control of the transaction manager, you can use EAI steps in your procedure definition. There are different types of EAI step used to communicate with different types of external system. For example:

- The EAI Database step is used to communicate with database stored procedures so you can use existing business logic in your procedures. Refer to the *TIBCO iProcess™ Modeler - Integration Techniques Guide* for more information about installing and using EAI Database steps.
- The COM+ EAI step is used so that iProcess can communicate with a COM+ application. Refer to the *TIBCO iProcess™ COM Plug-in: User's Guide* for more information about installing and using the COM EAI step.

An EAI step causes the iProcess background to call-out to an application defined sub-system in order to perform some work. In the MSDTC version, this work may be included within the current distributed transaction, hence combining data updates performed by that work with the data updates performed as part of the workflow case processing.

Refer to "Using Enterprise Application Integration Steps" in the *TIBCO iProcess Modeler - Integration Techniques Guide* for more information about using EAI steps.

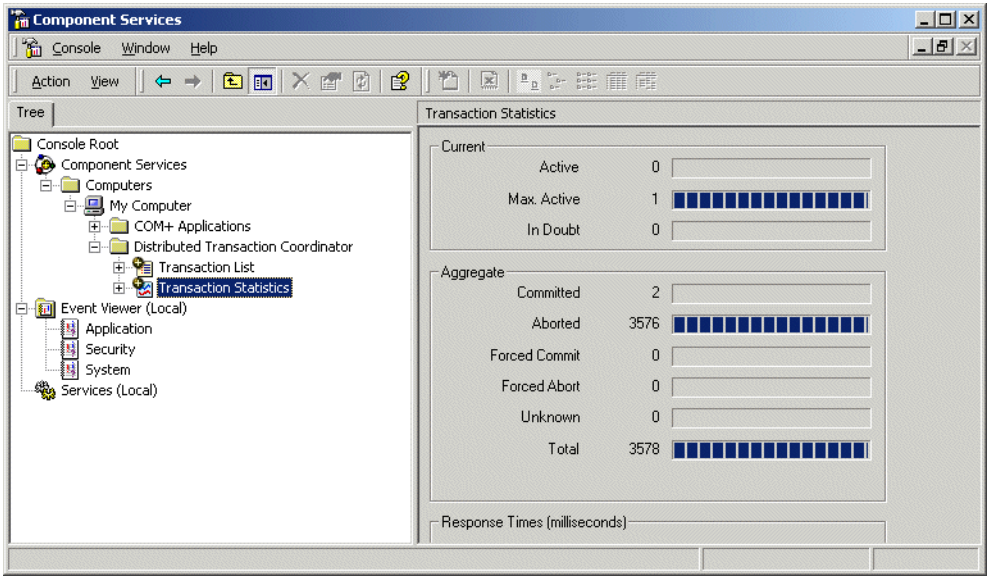
Back to Library

What is MSDTC?

The Microsoft Distributed Transaction Coordinator is the transaction manager program built into the Windows 2003 Server. This runs as a service on Windows and coordinates transactions between distributed data stores.

The iProcess Suite integrates with the MSDTC by creating and controlling distributed transactions. This means that iProcess can start a transaction by sending a request to the MSDTC and then other resources (such as databases) can enlist in the transaction as required by communicating with the MSDTC.

The MSDTC can be administered using the **Component Services Console** (as shown in the following example).



You can configure iProcess to use MSDTC by setting the EAI_NEEDS_MSDTC process attribute - refer to "Administering Process Attributes" in the *TIBCO iProcess Engine: Administrator's Guide* for more information about setting process attributes.This means that instead of the transaction being controlled by the local resource manager, the MSDTC is notified of the transaction and other distributed resource managers can enlist in the transaction.

Refer to the Microsoft documentation for more information about administering and monitoring MSDTC transactions.

Back to Library

Examples of Transaction Control

The following section provides some examples of iProcess transaction control.

Case Data Updates to the SQL Server using MSDTC

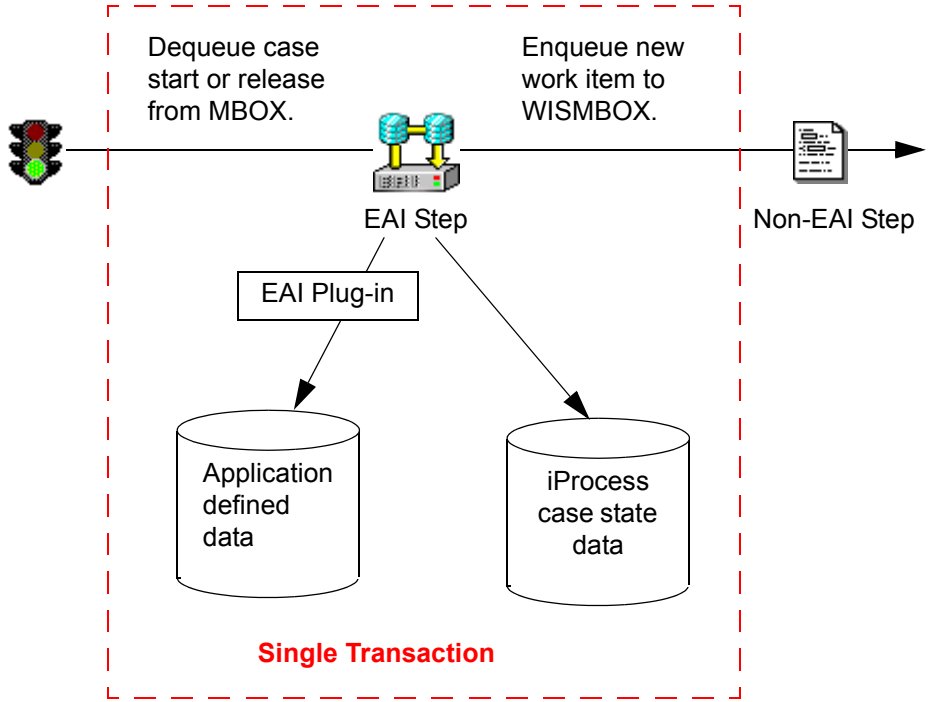
In a simple process where iProcess case data is being updated in the SQL database and an external database using an EAI plug-in, the transaction process involves the following:

1. The iProcess Engine starts a transaction by notifying the MSDTC.
2. The SQL Server resource manager enlists itself as part of the transaction with the MSDTC.
3. The external database resource manager also enlists itself as part of the transaction with the MSDTC.
4. If all the case updates are successful, the MSDTC performs a two phase commit operation with all resources that are enlisted as part of the transaction.

Back to Library

External Updates Using EAI Steps

The Enterprise Application Integration (EAI) steps can be used in procedures to control updates to third party applications and iProcess case data under transaction control. If the external applications operate in the same TPM environment as iProcess, all business operations in a single procedure can be completed as one global transaction.



Back to Library

Transaction Failures and Rollbacks

In the transaction example in Figure 1 on page 77, all the external data updates and iProcess case data updates are saved to memory until the case has been processed at which time all the updates are written to the database. This is known as the Commit process.

However, if one or more steps in the transaction are not possible (such as the database is not available), none of the updates are committed and the data is left in the same state as when the transaction started. This is known as the rollback process. Similarly, if the connection to the database is broken, any outstanding transactions are rolled back.

If the instruction is received by the Background but one of the steps it processes subsequently fails (such as a COM+ EAI step calling an application that is not running), then the resource manager returns an error to the Mbox daemon. The Mbox daemon aborts the transaction while the resource manager ensures that all of the steps involved in the transaction are rolled back to their initial state.

Poison Transactions

If a transaction is continually failing and being retried, this can have a serious impact on system performance. It is possible to restrict the number of times a transaction will be retried and to be notified of this failed transaction.

The iProcess installation process sets the message queues to have:

- a retry count of twelve (defined using the IQL_RETRY_COUNT process attribute).
- a retry delay of five minutes (defined using the IQL_RETRY_DELAY process attribute).

See "Administering Process Attributes" in the *TIBCO iProcess Engine: Administrator's Guide* for more information about setting process attributes.

Back to Library

Chapter 6 iProcess Mbox Sets

This chapter describes how:

- TIBCO iProcess Workspaces communicate with the Background process.
- Mbox sets are used to help in the distribution of messages on the system.
- the middleware message queuing system is implemented for the TIBCO iProcess Engine.

Back to Library

Overview

One of the core components of the iProcess Engine is the messaging system. This is used to deliver instructions from:

- all the TIBCO iProcess Workspaces to the Background process
- the Background process to the foreground processes

Because of the potentially high volumes of transactions in a TIBCO iProcess Engine system, there can be many thousands of instructions trying to be passed from TIBCO iProcess Workspaces to the Background to be processed. To prevent messages being lost or messages being held up for long periods of time, a message queuing system is used so that all messages sent from TIBCO iProcess Workspaces or the Background can be sent to a message queue. Once it is in the queue, the message has an assured delivery mechanism where it will be dequeued as system resources permit.

The message queues also makes sure that a message instruction is only processed once (unless a transaction rollback occurs - see Transaction Management of Messages on page 90) and is processed even if system resources fail because it is retried when the system is running again. For example, in the case for banking transactions, a message to update a bank account must only be performed once and must be performed at some point even if the system has failed.

The iProcess Suite currently supports two message queuing systems:

- iProcess database queues (for the UNIX DB2 and Windows SQL Server versions)
- Oracle AQ (for the UNIX Oracle and Windows Oracle versions)

Both control the delivery and storage of iProcess messages.

What are iProcess Messages?

A message consists of the business data captured by the information entered into iProcess forms such as what fields and values exist. A message also contains instructions so that the iProcess Engine knows what to do with the message such as release or forward the message. It can also contain control information used to manage the message. A message is delivered (enqueued) to the message queue and consumed (dequeued) by recipients.

The message queues and their messages are stored in database tables that operate as message queues.

The iProcess Engine is set up to use the Local message type, which is used for communication between the foreground and background processes.

Back to Library

Definition of Mbox Sets

The iProcess Engine uses a number of message queues to enable high volumes of transactions to be processed. iProcess uses a logical grouping of physical queues called Mbox sets in which processes read or write messages. An Mbox set can be defined as having one or more queues in which messages are posted in a round robin method. For example, an Mbox set can comprise a number of physical database tables operating as message queues.

There are many performance factors related to this design such as spreading queues over multiple disks. Instead of a process writing or reading messages from a specific queue, it can be configured to use an Mbox set. The Mbox set enables you to dynamically configure the queues used in a set as system resources change.

For example, two Mbox sets can be configured as follows:

```
MBOX_SET = queue_1, queue_2
MBOX_SET2 = queue_1
```

A physical queue can be included in multiple Mbox sets but can only be included once in the same Mbox set.

Each iProcess Engine process is configured to use a particular Mbox set and has a process attribute type set for it. This information is configured in the **process_attribute** table.

Configuring Mbox Sets

You can dynamically configure the Mbox sets using the *SWDIR***util****swadm** utility. For example, you can create Mbox sets, add queues to Mbox sets and remove queues.

Different processes require access to different types of Mbox set, for example, the Background process needs to have write access to a background Mbox set and a WIS Mbox set.

The **process_attribute** table specifies the access type for each process and which Mbox set is used.

The Mbox set tables contain the necessary information for each Mbox set such as the logical names of all the queues in the set. Multiple queues in a set enable iProcess to post messages on a round-robin basis through the queues.

Back to Library

Transaction Management of Messages

For a queuing technology to be transactional it must be under the control of a Resource Manager. The resource manager ensures that its resources, the queues, are always in a consistent state. Any updates, i.e. the sending or receiving of messages, made during a transaction will either all succeed or all fail.

UNIX Oracle Transaction Implementation

When using the UNIX Oracle version of an iProcess Engine, all resources are updated in an iProcess transaction as one unit of work. This is because the message queues are maintained in the database and can therefore be controlled by the Oracle Resource Manager. Transactions are limited to the scope of this database as the resource manager only controls this database.

Refer to the Oracle Database documentation for more information about how Oracle handles resource management.

Windows SQL Server Transaction Implementation

The TIBCO iProcess Engine uses the Microsoft SQL Server database to store iProcess messages. The SQL Server makes sure the queues are in a consistent state and will make sure that all resources updated during a transaction are committed as one unit of work.

UNIX DB2 Transaction Implementation

When using the UNIX DB2 version of an iProcess Engine, all resources are updated in an iProcess transaction as one unit of work. This is because the message queues are maintained in the database and can therefore be controlled by the DB2 Resource Manager. Transactions are limited to the scope of this database as the resource manager only controls this database.

Back to Library

Chapter 7 Monitoring Activities

The TIBCO iProcess Engine can be enabled to publish:

- TIBCO iProcess Engine activity information, and
- TIBCO iProcess Engine Work Queue Deltas,

to external applications. This chapter describes how messages are processed between the TIBCO iProcess Engine and the external application. It:

- provides an overview of activity monitoring - see page 92
- describes how activity monitoring messages are processed from the **BG** process to the **IAPJMS** process - see page 94.
- describes how activity monitoring messages are processed from the **IAPJMS** process to the external application - see page 97.
- describes how Work Queue Delta messages are processed from the **WIS** process to the **IAPJMS** process - see page 96
- describes the types and possible formats of the messages - see page 98.

Back to Library

Overview

The TIBCO iProcess Engine can be enabled to publish both TIBCO iProcess Engine activity information and TIBCO iProcess Engine Work Queue Deltas to external applications, using JMS topics.

Both types of publication use the **IAPJMS** (Introspection Activity Publication JMS) process, but they differ in detail.

Activity Publishing

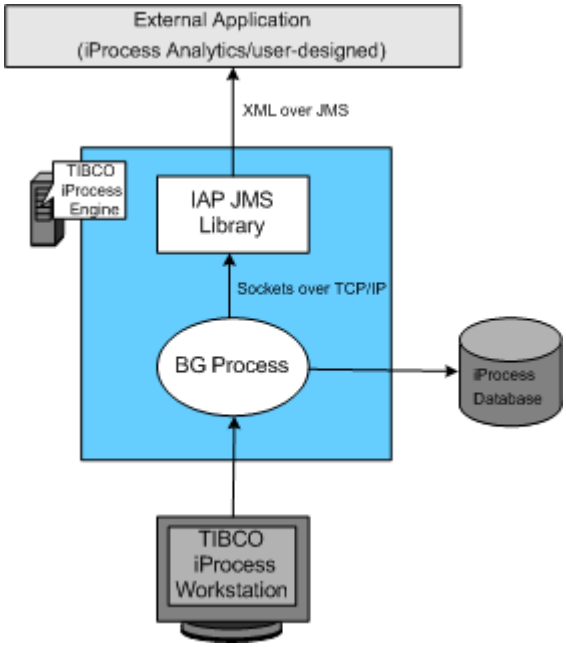
An activity is any instruction in the TIBCO iProcess Engine that creates an audit trail entry, for example, **Case started** or **Event issued**. You can configure any combination of step and/or activity to be monitored. This enables an external application to monitor important business events during the processing of cases.

The **BG** process can identify if a step is being processed and if activity monitoring has been configured for it. The **BG** process then sends details of the configured activities, in XML format to the IAPJMS Process.

The **IAPJMS** process sends the XML message to a specified JMS topic, from which an external application can receive the XML format message.

Back to Library

The following diagram demonstrates Activity Publishing message processing between the TIBCO iProcess Engine and the **IAPJMS** process.



Work Queue Delta Publication

Work Queue Delta Publication enables an external application to monitor an iProcess work queue and to retrieve only those work items in a given work queue that have changed since monitoring began.

If any data in a monitored work queue changes - for example if an existing work item is modified, an existing work item is removed from the work queue or a new item is added to the work queue - the **Work Item Server (WIS)** process sends the modified data in XML format to the **IAPJMS** process. The **IAPJMS** process then sends the XML message on to a specified JMS topic (either a default topic, or one specified by the subscribing application), from which an external application can receive the XML message.

Work Queue Delta Publication functionality can only be accessed by using iProcess Server Objects (Java) or iProcess Server Objects (.NET).

Back to Library

How Messages are Processed From the BG Process to the IAPJMS Process

Once the **BG** process has been configured to enable audited activities to be published (defined in the IAPJMS_PUBLISH process attribute), it sends out messages whenever an audited activity occurs.

The messages are published to the **IAPJMS** process using a configurable port number (defined in the IAPJMS_PORTNO process attribute). The **BG** process has two delivery methods for publishing messages, synchronous and asynchronous (defined in the IAPJMS_SYNCHRONOUS process attribute).

- SYNCHRONOUS - When the message is sent, a receipt is requested. The **BG** process waits until the **IAPJMS** process has confirmed the message has been published. If the message is not published, an error is written to the *SWDIR*/**logs**/**sw_error** file. With the SYNCHRONOUS method the **BG** process has to block until a response/timeout occurs which could impact throughput. This means the SYNCHRONOUS method is slower than the ASYNCHRONOUS method.
- ASYNCHRONOUS - The message is assumed to have been processed correctly if the message was sent successfully to the **IAPJMS** process. The ASYNCHRONOUS method is faster than the SYNCHRONOUS method because it does not have to wait for a receipt.

Back to Library

How Activity Messages are Processed From the IAPJMS Process to the External Application

The **IAPJMS** process listens for activity messages from the **BG** process. When it receives an activity message from the **BG** process it routes this to a configurable JMS topic (defined in the IAPJMS_TOPIC process attribute). The **IAPJMS** process does not need to know anything about the contents of the message. It simply passes it to the JMS topic.

The external application is configured to listen to the same JMS topic that is defined in the IAPJMS_TOPIC process attribute.

For information on how to configure and administer activity monitoring see "Monitoring Activities" in the *TIBCO iProcess Engine: Administrator's Guide*.

Back to Library

How Messages are Processed From the WIS Process to the IAPJMS Process

A user application - either using iProcess Server Objects (Java) or iProcess Server Objects (.NET) - subscribes to a work queue, creating an initial connection to the published queue. The application specifies a JMS topic name and subscribes to that topic (or, if no topic is specified at subscription time, the WQDJMS_TOPICNAME process attribute is used; see the *TIBCO iProcess Engine Administrator's Guide* for details of this attribute and of other process attributes involved), and then can start receiving JMS messages.

The first message sent is always a Work Queue Synchronization (DSY) Message. This informs the listening application that it now needs to begin processing on other messages received from this Work Queue.

The data passed in JMS messages may be restricted by other components of the system. For example, the maximum length of work queue parameter fields that can be published is limited by the **WIS** process: see the section "Using Work Queue Parameter Fields in Procedures" in the *TIBCO iProcess Modeler - Advanced Design* guide.

Messages are published to the **IAPJMS** process using a configurable port number (defined in the WQDJMS_PORTNO process attribute).

Back to Library

How Work Queue Delta Messages are Processed From the IAPJMS Process to the External Application

The **IAPJMS** process listens for Work Queue Delta messages from the **WIS** processes. When it receives a message from a **WIS** process it sends this to the appropriate JMS topic. By default, this is the topic defined in the WQDJMS_TOPICNAME process attribute; but this default can be overridden if the subscribing application specifies a different topic. The **IAPJMS** process does not need to know anything about the contents of the message, it simply passes it to the JMS topic.

The external application must be set up and configured to listen to the same JMS topic to which the IAPJMS process sends the messages.

The first time that a subscribing application opens a work queue, it uses iProcess Server Objects interfaces to open the queue and to retrieve all the existing work items for that queue.

Whenever any change is made to the work queue after that - a new work item added, an existing work item modified or removed - the change is published to the appropriate JMS topic. If more than one application is subscribed to a given work queue, and they do not use the default topic specified by WQDJMS_TOPICNAME, the JMS message must be published to each topic used by a subscribing application.

Back to Library

Understanding the Message Types

Activity Monitoring (IAP) uses two message types, and Work Queue Delta publication (WQD) uses three types.

IAP Message Types

There are two types of message:

- The Monitor Event Detail message (MED). A **BG** process sends this message. The message is addressed to the JMS topic configured in the IAPJMS_TOPICNAME process attribute.
- The Monitor Event Request message (MER). The MER message is sent to the iProcess database to update the activity monitoring configuration information. It can be sent by SSO or by using the *SWDIR***bin****swutil* **IMPMONITOR** command.

The Monitor Event Detail Message (MED)

Every MED message sent from the **IAPJMS** process to the external application consists of:

- a message header
- message properties
- message body. The message body contains the detailed XML for the events that are being monitored. The MED XML format is defined by the **SWAuditMessage.xsd** schema.

These messages can be sent in either a basic format or an extended format. You can specify which format is used by setting the IAPSCHEMA parameter in the *SWDIR***etc****staffcfg** configuration file. See the section "Tuning the TIBCO iProcess Engine Using SWDIR\etc\staffcfg Parameters" in the *TIBCO iProcess Engine: Administrator's Guide* for details of this parameter.

The extended format provides the following additional information in the message body:

- Description and type of the audit user. The basic format identifies the audit user; in the extended format the description and type of the audit user can be specified.
- Addressee. The basic format contains no addressee information. The extended format identifies the name, description and type of the addressee of the step for OPEN, KEEP, RELEASE and FORWARD actions.

Back to Library

- Main procedure of a sub-case. The basic format contains no information about the main procedure of a sub-case. The extended format identifies the procedure name and description.
- Parent procedure of a sub-case. The basic format identifies the parent procedure. The extended format adds a procedure description.

For examples of these IAPJMS messages, see the section "Examples of Configuring Activity Monitoring Information" in the *TIBCO iProcess™ Modeler Integration Techniques* guide.

Filtering Message Event Request (MER) Messages

Every MER message sent to the iProcess database to update the activity monitoring configuration information consists of XML requesting the events to monitor. The MER XML format is defined by the **SWMonitorList.xsd** schema.

WQD Message Types

There are three types of message:

- The Work Queue Delta message (WQD). The **WIS** process sends this message.
- The Work Queue Synchronization message (DSY). This contains the unique synchronization ID for the queue.
- The Work Queue Invalid message (DER). This informs the subscribing application that the work queue should be closed and re-opened.

Work Queue Delta (WQD) Message

Every WQD message sent from the **WIS** process consists of:

- a message header
- message properties
- the message body containing details of the Work Queue Delta. The WQD XML format is defined by the **apSSOTypes.xsd** schema. See the section "Schemas" in the *TIBCO iProcess™ Client (Browser) Action Processor Reference* guide for more information on this schema.

Whenever an item is added, modified or deleted from a monitored work queue, a WQD message is published by the **WIS** to the **IAPJMS** process.

Work Queue Synchronization (DSY) Message

Every DSY message consists of.

Back to Library

- a message header
- message properties
- the message body containing the unique synchronization ID, which will be published when the work queue has been fully loaded and JMS publication is to begin.

A DSY message is sent once the initial queue has been built. It is published to subscribing applications so that the subscriber knows it needs to begin processing any Work Queue Deltas for the specified Work Queue Delta ID.

Work Queue Invalid (DER) Message

Every DER message consists of.

- a message header
- message properties
- an empty message body.

The message type is an exception message sent to all applications subscribing to the work queue on which an error occurs. It tells the subscribing applications that the Work Queue Deltas for this queue are now invalid and that this work queue must be closed and re-opened.

See the *TIBCO iProcess Server Objects (.NET) Programmer's Guide* and the *TIBCO iProcess Server Objects (Java) Programmer's Guide* for more detail on Work Queue Delta publication via JMS and how it works.

Back to Library

Chapter 8 Database Failure and Failover

This chapter describes how the TIBCO iProcess Engine handles database failure or failover. It describes:

- an overview of database failure/failover - see page 102.
- TIBCO iProcess Engine behavior in the event of a failure/failover - see page 103.
- TIBCO iProcess Workspace behavior in the event of a failure/failover - see page 105.
- TIBCO iProcess™ Objects and TIBCO iProcess™ Server Objects behavior in the event of a failure/failover - see page 106.
- TIBCO iProcess Engine configuration requirements for failure/failover support - see page 107.

Back to Library

Overview

The TIBCO iProcess Engine stores all iProcess data in its database (Oracle, SQL Server or DB2). It therefore relies on the availability of that database to be able to process work. If the database fails, the TIBCO iProcess Engine can do nothing until the database returns.

Database **failover capability** exists when the database is configured in such a way that if a failure occurs, a standby instance of the database takes over immediately with no or minimal impact on service. Failover capability can be provided in many ways, depending on the hardware, operating system and database systems you are using. Failover capability is therefore an essential component of any high-availability, production-level system involving the TIBCO iProcess Engine.

If a database failure or failover occurs, the TIBCO iProcess Engine is able to:

- detect that the failure has occurred.
- guarantee the integrity of any in-progress transactions.
- seamlessly continue working when the database connection returns (whether that connection is to the original database or a failover database).

Back to Library

TIBCO iProcess Engine Behavior

The following table shows how different TIBCO iProcess Engine components behave in the event of a database failure or failover.

TIBCO iProcess Engine Component	Behavior When Failure or Failover Occurs
Server processes	**On SQL and DB2**: Each TIBCO iProcess Engine process clears down any database connections it already has (as they are used) and attempts to re-establish a new connection. If necessary, the process retries the connection attempt until the database becomes available again, when the attempt succeeds. **On Oracle**: The server processes use the Oracle Client to establish and maintain database connections. The Oracle Client therefore attempts to re-establish the database connection. If necessary, it retries the connection attempt until the database becomes available again, when the attempt succeeds.
Database transactions	Any in-progress Write transaction is rolled back. When the database becomes available again: • Read-only transactions continue and complete as normal. • Write transactions are retried. This means that transactional integrity is maintained; no transaction or workflow operation is left in an indeterminate state. Operations either succeed and are committed or fail and are rolled back.
Process Sentinels	Any processing that requires a connection to the database hangs until the database connection is restored. For example, if you use *SWDIR***util****swsvrmgr** to issue a Restart event for a process when the database has failed, the process is not restarted until the database connection is restored. **Note - On the Oracle (UNIX/Linux) TIBCO iProcess Engine the event system used by the Process Sentinels is provided by the database. This means that the use of Oracle's Transparent Application Failover (TAF) feature is required to provide failure or failover support. See page 107 for more information.**

Back to Library

TIBCO iProcess Engine Component	Behavior When Failure or Failover Occurs
System messages	Any TIBCO iProcess Engine process that attempts to contact the database will fail (until the connection is restored), causing it to write a database connection failure error to the *SWDIR***logs****sw_error** file. For example: `2005/04/25 15:41(BG:1:2496:pro::7.1:4352): 0-lcase_add() Unable to execute statement (Statement ID - 49) (S1002 - [Microsoft][ODBC SQL Server Driver]Invalid Descriptor Index) lcase_add() Unable to execute statement (37000 - Statement failed due to database FAILOVER )`
Command line utilities	Any command that requires a connection to the database hangs until the database connection is restored. For example, if you have a *SWDIR***util****swadm** session open when the database fails, and you then try to add a **BG** process, the command hangs until the database connection is restored. As soon as the database is available again, the command completes and the new **BG** process is added.

Back to Library

TIBCO iProcess Workspace Behavior

Existing TIBCO iProcess Workspace sessions may be unaffected by a database failure or failover, as many TIBCO iProcess Workspace operations - such as retrieving information about a work item from the WIS cache - do not require access to the database.

TIBCO iProcess Workspace behavior is only affected when an operation that requires access to the database is performed - for example, opening a procedure. The TIBCO iProcess Workspace then passes a request to the TIBCO iProcess Engine, which in turn tries to access the database. The request therefore hangs until the database connection returns. If this period exceeds the TIBCO iProcess Workspace's timeout period (by default, 25 seconds), a timeout dialog is displayed to the user. The user has the option to abort or retry the operation.

Back to Library

iProcess Objects and iProcess Server Objects Behavior

The following table shows how different TIBCO iProcess Objects and TIBCO iProcess Server Objects components behave in the event of a database failure or failover.

iProcess Objects/iProcess Server Objects Components	Behavior When Failure or Failover Occurs
iProcess Objects Server	From the viewpoint of a database failure or failover, the iProcess Objects Server is a TIBCO iProcess Engine server process. As such, its behavior is the same as any other TIBCO iProcess Engine server process - see Server processes on page 103.
iProcess Objects Clients and iProcess Server Objects Clients	Existing iProcess Objects/iProcess Server Objects client sessions may be unaffected by a database failure or failover, as many client operations - such as retrieving information about a work item from the WIS cache - do not require access to the database. iProcess Objects/iProcess Server Objects client behavior is only affected when an operation that requires access to the database is performed - for example, opening a procedure. The client passes a request to the iProcess Objects Server, which passes it to the TIBCO iProcess Engine, which in turn tries to access the database. The request therefore hangs until the database connection returns. If this period exceeds the iProcess Objects Server's RPC timeout period (by default, 25 seconds), a timeout exception is returned to the iProcess Objects/iProcess Server Objects client. It is the iProcess Objects/iProcess Server Objects client's responsibility to handle that exception.

Back to Library

TIBCO iProcess Engine Configuration Requirements

The following sections describe what you need to do to provide failure/failover support on the TIBCO iProcess Engine.

Oracle (UNIX or Linux)

To provide failure/failover support on the Oracle (UNIX/Linux) variant of the TIBCO iProcess Engine, you MUST configure the use of Oracle's Transparent Application Failover (TAF) feature. Oracle TAF enables an application user (such as the iProcess Engine) to automatically reconnect to a database if the connection fails.

The TIBCO iProcess Engine Process Sentinels are event driven. On the Oracle (UNIX/Linux) variant of the TIBCO iProcess Engine, the event system is provided using Oracle AQ's publish/subscribe mechanism. This means that if the database connection fails, the Process Sentinels cannot receive events, and so appear to hang. To avoid this problem you need to configure TAF so that the connection can be restored. Once the connection is restored, all connections are recovered and normal operation is resumed.

If you **are** running parallel servers, using TAF allows iProcess to switch to an alternative instance if the instance that it is currently using fails.

If you **are not** running parallel servers, using TAF still means that although iProcess will not function while the database is down, it can recover immediately and automatically when the database is recovered.

To enable the use of TAF with the TIBCO iProcess Engine, you need to configure TAF support for the service name that you intend to use to connect to the Oracle database. See the *TIBCO iProcess Engine (Oracle9i) for UNIX or Linux Installation Guide* for more information about how to do this.

TAF involves manual configuration of a net service name that includes the FAILOVER_MODE parameter included in the CONNECT_DATA section of the connect descriptor. For more information about TAF, and how to set up and configure it, please see your Oracle documentation.

Back to Library

Oracle (Windows), SQL Server and DB2 (UNIX or Linux)

Database **failure** support is provided automatically. You do not need to configure the TIBCO iProcess Engine in any way - it recovers immediately and automatically when the database connection is restored.

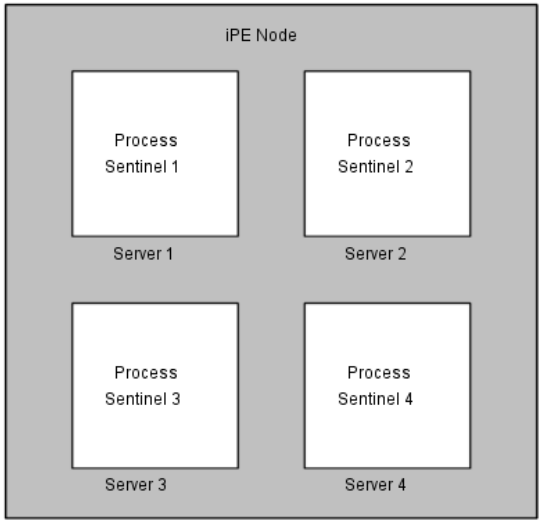
Database **failover** support is provided automatically **provided that you have configured your database system to provide failover capability**. For example, by implementing failover clustering in SQL Server, or by using DB2's high availability disaster recovery (HADR) database replication feature. (Failover capability can be provided in many ways, depending on the hardware, operating system and database systems you are using. Please see your SQL Server or DB2 documentation for more information about how to do this.) You do not need to configure the TIBCO iProcess Engine in any way - it recovers immediately and automatically when the failover completes and the database connection is restored.

Event handling is provided by the **Staffware Events** COM+ application on Windows variants, and by iProcess event/notification daemons on the UNIX DB2 variant. Consequently, the event system has no dependency on the database as it does in the UNIX/Linux Oracle variant.

Back to Library

Chapter 9 Process Management

This chapter explains the concepts of how the TIBCO iProcess Engine processes are managed by the Process Sentinels to make sure that the TIBCO iProcess Engine can operate on a 24*7 basis. If the TIBCO iProcess Engine is installed across multiple servers (a node cluster), processes can operate on the various servers within the node cluster. This means that each server in the cluster requires its own Process Sentinels to control the processes that are running on that server - see below.



Back to Library

Responsibilities of the Process Sentinels

The following list describes the responsibilities of the Process Sentinels:

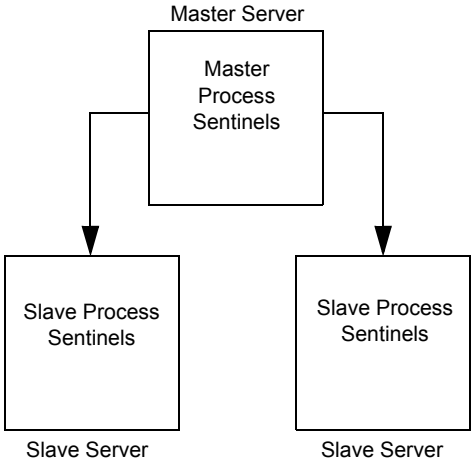
- Start processes during a server start up or upon a system administrator request. It will control the order in which the processes are started.
- Detect failed processes and restart them automatically (or manually using the command line *SWDIR***util****swsvrmgr** utility or the iProcess Server Manager).
- Shut down processes when the system is shut down or when the administrator requests a sub-set of the system to be stopped.
- Pause the entire system, a group of processes or a single process.
- Detect *SWDIR***logs****sw_error** and *SWDIR***logs****sw_warn** files and send a work item to the system administrator (**swadmin**) informing them that the file has been created and on which server.
- Check that there is enough disk space using the *SWDIR***fspart** file (you can configure which partitions to check). If there is not enough space, a work item is sent to the administrator informing them of the situation.
- Monitor other Process Sentinels and restart them if they fail - see Restarting Failed Process Sentinels on page 117.
- Provide interfaces for integration with TIBCO Hawk® - See the *TIBCO iProcess Engine: Administrator's Guide* for more information.
- Maintain the list of all active user logins.

Distribution and Hierarchy of Process Sentinels

In a distributed TIBCO iProcess Engine node cluster where there are a number of servers running, it is necessary to have Process Sentinels running on each server. To keep the system processes synchronized, the Process Sentinels are designed so that there are master Process Sentinels running on one server and slave Process Sentinels running on all the other servers. See *TIBCO iProcess Engine to TIBCO iProcess Engine Network Communication in a Node Cluster* on page 127 for more information about how process sentinels communicate between servers.

The following diagram shows that the Master process controls the slave processes in a hierarchical model. The master and slave processes have different responsibilities, which are described in Master and Slave Responsibilities on page 111.

Back to Library



Master and Slave Responsibilities

The master and slave processes have different responsibilities for tasks that they can perform. There are also some tasks that are common across master and slave processes.

Common Process Sentinel Tasks

The following operations are performed by both the master and slave Process Sentinels:

- Starting and stopping processes on the local computer
- Detecting the failure of processes on the local computer and restarting them
- Maintaining the **process_config** table to store the status of processes.
- Check that there is enough disk space using the *SWDIR***fspart** file (you can configure which partitions to check). If there is not enough space, a work item is sent to the administrator informing them of the situation.
- Control the sequence of processes during a system start-up or shutdown.
- Check for *SWDIR***logs****sw_error** and **sw_warn** files. If any are detected, a work item is sent to the system administrator.

Back to Library

Master Process Sentinel Tasks

The master Process Sentinels can perform all of the above tasks but the following task is only performed by the master Process Sentinels.

- Report the status of the node/node cluster by publishing an event, for example, if the server processes are running or have been stopped.

The startup and shutdown requests for server processes are broadcast by the SWDIR**util****swsvrmgr** utility. The master Process Sentinels will record the new system status when the shutdown or startup is complete.

Back to Library

How Processes are Controlled by the Process Sentinels

The following table lists all the processes that are controlled by the Process Sentinels and show how they operate.

Process Description	Startup	Shutdown	Restart	Can Be Paused Manually	Paused Automatically By Process Sentinels
Background	x	x	x	x	
Database Queue Daemon	x	x	x	x	
Deadline Manager	x	x	x	x	
IAPJMS[a]	x	x	x	x	
Prediction	x	x	x	x	
RPC Background	x	x	x		
RPC Pool Server	x	x	x		
iProcess Objects Server	x	x	x	x	
WIS Mbox Daemon	x	x	x	x	x
Work Item Server	x	x	x	x	x
Work Queue Server	x	x	x		x

a. This process is disabled unless you have chosen to enable it when installing the TIBCO iProcess Engine.

Back to Library

Starting the TIBCO iProcess Engine Processes

The Process Sentinels are responsible for starting all the TIBCO iProcess Engine processes. This can either be starting the complete system or starting individual processes as requested. The Process Sentinels make sure that the processes are started in the correct sequence.

Many processes require that dependent processes are already running, for example, the WQS needs to be started before the WIS processes can be started. The Process Sentinels make sure that all process dependencies are satisfied before starting a process.

All Process Sentinels (master and slaves) maintain a complete list of all processes running in the TIBCO iProcess Engine node cluster. Each server hosts their own Process Sentinels, which are only responsible for processes running on its computer. When the Process Sentinels attempt to start processes, they know if all the process dependencies are satisfied across all the TIBCO iProcess Engines in the node cluster.

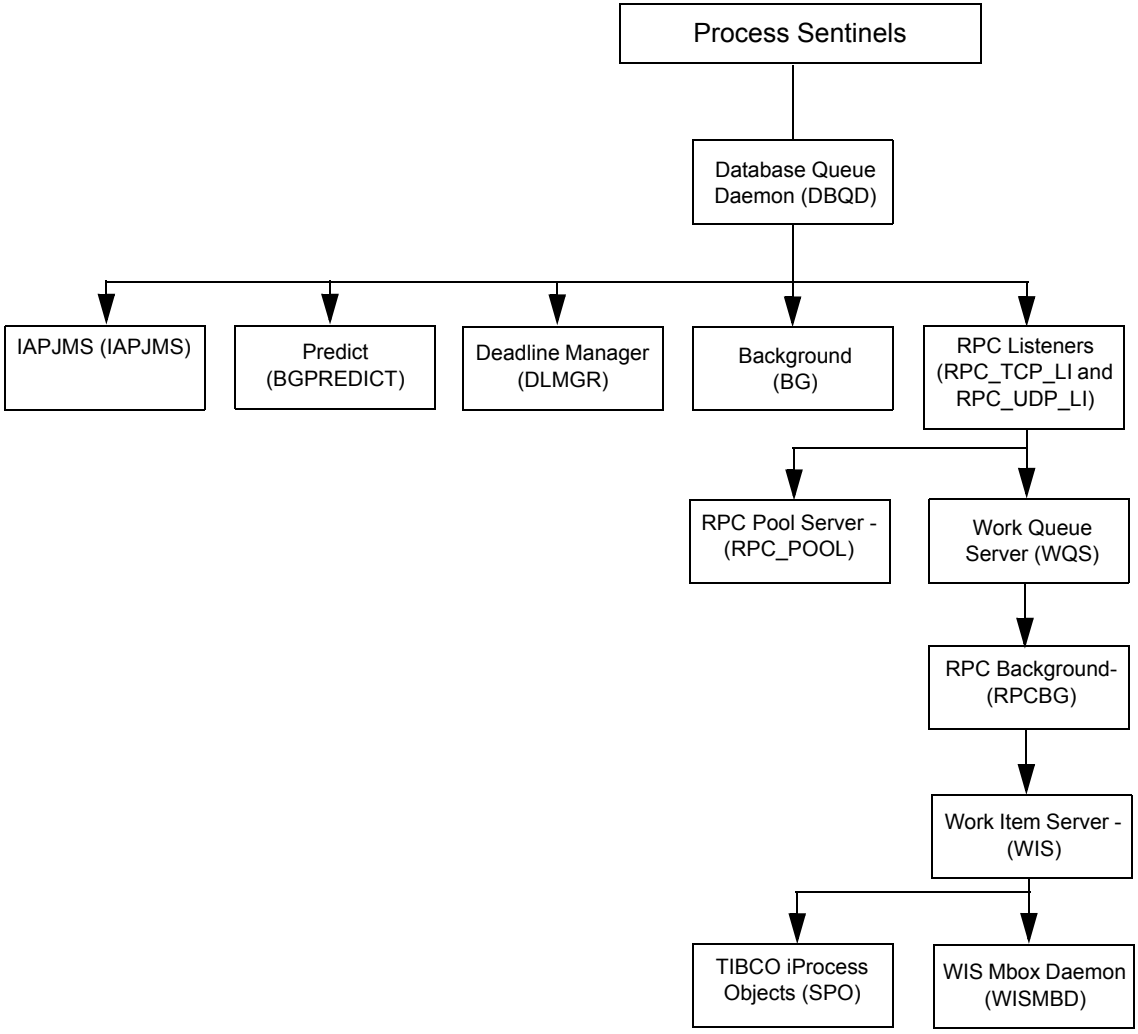
For example, if a system is configured to have 1 WQS, 5 WIS, 5 WISMBD, the processes would be started in the following order:

- Start the WQS processes
- Start all the WIS processes
- Start all the WISMBD processes.

The following diagram illustrates the order in which the Process Sentinels start the TIBCO iProcess Engine processes. All processes are started by the Process Sentinels apart from the RPC pool servers, which are started by the RPC TCP Listener process

The DBQD process is currently only present on the DB2 version of the TIBCO iProcess Engine.

Back to Library



Back to Library

Determining Where Processes Run

The **process_config** table determines on which computers individual processes will run and how many instances of those processes are started. There are a number of rules about where processes can be run:

- All of the foreground processes must operate on the master server. See *Foreground Processes* on page 34 for a list of these processes.
- The background processes can operate on any server and can be distributed across several servers, for example you might run a background server instance on each of your servers if you are using a node cluster architecture. See *Background Processes* on page 35.

Back to Library

Restarting Failed Processes

To make sure that the TIBCO iProcess Engine is always running, the Process Sentinels are constantly monitoring for any processes that fail. Each of the Process Sentinels operating in the node will monitor all processes running on the same server. If a process fails, the Process Sentinels will start a new copy of the process. In many cases, there can be many instances of a process running such as 5 Background processes and the Process Sentinels will make sure that the correct instance of the process is restarted.

The **process_attribute** table is responsible for the configuration of how processes are restarted. You can specify whether you want the process to be manually restarted, restarted a set number of times, and the minimum time a process must run before it can be restarted. Settings can be applied to the node, a single server in the node, a type of process, or an individual instance of a process.

Back to Library

Restarting Failed Process Sentinels

Process Sentinels consist of two processes: a worker process and a watcher process. On each server in an TIBCO iProcess Engine cluster, these two processes are started. This architecture is designed so that each process can monitor the other one and restart it if it fails. This makes sure that the Process Sentinels are always running. The watcher and worker processes perform different tasks:

Watcher Process

After starting the worker process and establishing a two way communication channel with the worker process, the watcher process monitors the worker process and can restart it upon a failure. If any errors occur, an error message is logged in the *SWDIR***logs****sw_error** file.

On the DB2 version, the watcher process also runs the iProcess event daemon (see Event Handling on page 35).

Worker Process

The worker process performs the process monitoring of the TIBCO iProcess Engine processes and monitors and restarts them if they fail. The worker process also monitors the watcher process to make sure that it is always running and will restart it if it fails. The following is the start-up routine for the worker process:

1. Initialize process by processing its command line arguments.
2. Establish a communication channel with its watcher process.

3. Connect to the database and read the **node_cluster** table to determine if the server is configured to be part of the TIBCO iProcess Engine. It can also determine if it needs to operate as a master or slave.

   If there is no entry for this server in the **node_cluster** table, an error message is logged to the operating system event log (**syslogd**). The worker process will shut down and also shutdown the watcher process.

4. Read the **process_config** table to build an in memory process hierarchy model.

5. Start process monitoring.

Back to Library

Shutting Down Processes

Processes are shut down by the Process Sentinels in the opposite sequence to when they were started.

Back to Library

Configuring the Process Sentinels

There are several areas of configuration information related to the Process Sentinels:

- The Process Sentinels command line utility (*SWDIR***util****swsvrmgr**)

  This utility enables system administrators to control the functions of the Process Sentinels such as setting which computer runs the master process.

  Refer to “Using swsvrmgr to Administer Server Processes” in the *TIBCO iProcess Engine: Administrator’s Guide* for more information.

- The iProcess Server Manager

  This graphical tool allows you to control the functions of the Process Sentinels in much the same way as with *SWDIR***util****swsvrmgr**. Refer to "Using the iProcess Server Manager to Administer Server Processes" in the *TIBCO iProcess Engine: Administrator’s Guide* for more information.

- *SWDIR***etc****staffpms** file

  When the Process Sentinels start, it uses the database connection information in this file to establish the database connection.

  Refer to “Using the TIBCO iProcess Engine Configuration Files” in the *TIBCO iProcess Engine: Administrator’s Guide*.

- **node_cluster** table

  After the Process Sentinels have connected to the database, it reads this table to see if the computer on which it has been started is part of the TIBCO iProcess Engine node cluster. If it is not, the Process Sentinels will shut down.

  Refer to the appropriate *TIBCO iProcess Engine Database Administrator’s Guide*.

- **process_config** table

  The Process Sentinels read this table to determine which processes should be running on which computer. A process hierarchy model is created in memory, which is used to start and stop the processes in the correct sequence.

  Refer to the appropriate *TIBCO iProcess Engine Database Administrator’s Guide*.

- **process_attribute** table

  The Process Sentinels read this table to find out how processes should be restarted, for example if they should be restarted automatically or manually. This table also defines how long the Process Sentinels are idle for before checking for *SWDIR***logs****sw_error** and **sw_warn** files and disk space.

  Refer to the appropriate *TIBCO iProcess Engine Database Administrator’s Guide*.

Back to Library

- You can use the TIBCO iProcess Engine Configuration utility *SWDIR***util****swadm** to administer the Process Sentinels and server processes. You can perform tasks such as setting the server on which to run the master Process Sentinels, adding a new server to the node cluster or enabling and disabling processes. This utility directly reads and updates the iProcess database tables.

  Refer to "Administering Servers" in the *TIBCO iProcess Engine: Administrator's Guide*.

Back to Library

Back to Library

Chapter 10 Network Communication

This chapter describes how:

- TIBCO iProcess Workspace communicates with the iProcess Engine. See TIBCO iProcess Workspace and TIBCO iProcess Engine Network Communication on page 124.
- TIBCO iProcess Engines communicate when they are in a node cluster. See TIBCO iProcess Engine to TIBCO iProcess Engine Network Communication in a Node Cluster on page 127.
- The iProcess Engine works when a firewall is implemented on your network. See Using the TIBCO iProcess Engine in a Firewalled Environment on page 128.

Back to Library

TIBCO iProcess Workspace and TIBCO iProcess Engine Network Communication

This section provides a brief overview of how the TIBCO iProcess Workspace and TIBCO iProcess Engine communicate with each other via the network.

The TIBCO iProcess Workspace makes *Remote Procedure Calls* (RPC) to the server using ports. This means the TIBCO iProcess Workspace makes calls to the TIBCO iProcess Engine to find out information such as what work queues and work items it needs to display. It also means that procedures can be started from one computer but actually be running on another computer on the network. There are a number of reasons why this might happen:

- The local computer is not able to provide the functionality required.
- The requested service is shared amongst many clients but centralized arbitration, synchronization and communication is required.
- You want to use the superior performance of a remote computer.
- You want to balance the processing load across the network.

Any program that offers functions that can be remotely accessed via RPC must have a unique RPC number. The RPC number for a service is either pre-defined (fixed) or allocated dynamically.

All RPC numbers need to be bound to network ports before remote clients can communicate with the RPC servers. This is because the port is the fundamental method of communication between computers.

The allocation of ports, similar to the allocation of RPC numbers, can be fixed or dynamic and is determined by the program providing the RPC service and the operating system. Therefore, it is important to know the distinction between RPC ports and numbers.

Function of a Portmapper

The connection between the RPC client computers and RPC servers is serviced by a program known as the **portmapper** (or **rpcbind**). The portmapper provides a directory for the RPC services on the computer on which it runs.

A client application can request a connection to an RPC service by passing the portmapper the RPC number. The portmapper responds by sending back the actual port number that the service is bound to. The client can then connect directly to the service using the service's port.

Back to Library

The TIBCO iProcess Engine RPC Service

The process that monitors the RPC calls that request a connection to the TIBCO iProcess Engine is known as the RPC listener. This has a pre-defined RPC number - by default it is 391875. If a TIBCO iProcess Engine is communicating with another TIBCO iProcess Engine, the default port is 391870. The *SWDIR***swdefs** file contains the RPC numbers.

When the iProcess Engine starts:

1. The RPC TCP listener process starts and requests the operating system to allocate it a port.
2. It binds a TCP socket to this port.
3. It then contacts the portmapper and requests that its RPC number (391875) is registered on the port that the operating system has allocated it.
4. The portmapper makes an entry in an internal table of the RPC number and port.
5. The iProcess Engine starts a number of other RPC servers but with dynamically allocated RPC numbers. These servers bind to a port dynamically allocated by the operating system and then registers the relationship between the RPC number and port with the portmapper.

The following describes what happens when an TIBCO iProcess Workspace connects to an iProcess Engine:

1. The TIBCO iProcess Workspace connects to the portmapper program.
2. The TIBCO iProcess Workspace requests details of the port to which the client RPC listener is bound. The request specifies the RPC number (the default is 391875).
3. The TIBCO iProcess Workspace connects to the client RPC listener using the port details and then requests details of the RPC numbers of the other RPC servers the TIBCO iProcess Workspace needs.
4. For each of the RPC numbers obtained, the TIBCO iProcess Workspace:
   a. Contacts the portmapper.
   b. Requests details of the port to which the RPC number is bound.

TCP/IP

The RPC listeners and TIBCO iProcess Workspace use the TCP/IP protocol to communicate. In order for the TIBCO iProcess Workspace to communicate with the RPC listener processes, the TIBCO iProcess Workspace must be able to perform HOST name resolution. This means that data packets are sent to a host

Back to Library

name such as HERCULES rather than to an IP address such as 10.10.56.202 and the TCP/IP stack looks up the correct IP address. A typical way of setting this up is to use a HOSTS file on each computer which provides a simple look up table for a host name and its IP address.

Back to Library

TIBCO iProcess Engine to TIBCO iProcess Engine Network Communication in a Node Cluster

This section provides a brief overview of how TIBCO iProcess Engines communicate with each other when configured as part of a node cluster.

The only processes that communicate with each other between nodes in a cluster are the Process Managers and the *SWDIR***util****swsvrmgr** process. The **swsvrmgr** process tells the process managers what actions to perform. They communicate via the TIBCO iProcess Engine event mechanism by publishing and subscribing to events.

For example when you start the system by running *SWDIR***bin****swstart**, the swsvrmgr START utility publishes an event to say that the system needs to start up something. All the process managers on all the nodes in the cluster should receive that event and start up any processes that are configured to run on that node. Similarly on shutdown, a shutdown event is published and each machine's process manager should receive the event and shutdown any appropriate processes.

Back to Library

Using the TIBCO iProcess Engine in a Firewalled Environment

In many enterprise network models, a firewall is used to link logical networks together to provide access security into the protected network. The following section describes how the iProcess Engine can be configured to work in a firewall environment.

What is a Firewall?

A firewall is a computer that links two logical networks together and re-routes data between the two networks as required. The firewall computer also contains a filter. This filter only allows data to pass through it that is requesting a particular service (using a variety of filtering methods defined by the firewall administrator).

A typical use of a firewall is for Web servers. A Web server needs to be accessed by remote computers outside of the logical network so they can access the web service. However, these computers should not be able to access other services on that server that are more likely to be a security risk.

A firewall can vary in the way that they restrict access to the network, they can:

- only allow access to and from certain computers.
- analyze the port number the client is requesting and compare it to a list of port numbers allowed.
- analyze the data that is being sent and only allow it through if it conforms to some pre-determined rules that have been set up.

Within the data being sent (known as *packets*), many firewalls can obtain the RPC number requested for RPC calls and only allow data through if it is requesting a particular RPC number and therefore a particular RPC service.

iProcess RPC and Firewall Access

When an iProcess Workspace and the iProcess Engine are separated by a firewall, the iProcess Suite can fail because its communication method (remote procedure calls - RPC) is stopped by the firewall filter. Because iProcess Engine RPC services are allocated dynamically, the firewall filter is not set up to open all ports that the iProcess Engine is using. Not all the ports will be open because the firewall administrator has set up certain restrictions to enable security on the network.

Back to Library

The RPC numbers are allocated dynamically so there is no fixed set of RPC numbers for a firewall administrator to add to the filter. If the ports used are not opened up on the firewall, the iProcess Workspace and iProcess Engine cannot communicate because data requests are denied by the firewall. In order for the iProcess Engine to operate in this environment, the firewall administrator needs to know what ports the iProcess Engine is using so that iProcess RPC calls can be filtered through.

You can set up the iProcess Engine to use a specific range of ports and/or RPC numbers so that the firewall administrator has a range of port numbers to add to the firewall filter. You can use one or both of the following methods to do this:

- Port range filtering
- RPC number filtering.

You use the *SWDIR***util****swadm** utility to configure port range and/or RPC number filtering. Refer to "Administering Firewalls" in the *TIBCO iProcess Engine: Administrator's Guide* for more information.

Port/RPC Number Resource Logging

When Port/RPC numbering is enabled, a log file containing resource allocation and release operations is stored in *SWDIR***logs****rpcport.log**. This is a text file containing entries for the following events:

- Start-up of the Port/RPC resource allocation service
- Shutdown of the Port/RPC resource allocation service
- Allocation of a Port/RPC number
- Release of a Port/RPC number
- Failure to re-bind a released port
- Successful re-bind of a previously failed port
- Errors in allocations/release of port/RPC number.

Using Oracle Events Through a Firewall

You can specify a range of AQ port numbers to be used in iProcess for communication through the firewall with Oracle Events.

You must ensure that you specify enough AQ ports for iProcess system processes. On a typical system, you should specify a minimum of 7 ports for iProcess (sentinel and utility processes), plus an additional port for each process defined in the process_config database table.

Back to Library

In addition to configuring the AQ port range in iProcess, you must also ensure that your firewall is configured to allow access for the iProcess AQ port range as well as the iProcess RPC port range, and any required Oracle ports (such as the default port 1521).

The following two tables are provided:

aq_port_range_conf: this records port ranges including the start port number and the count of the ports.

aq_port_range: this records the state of every port in every range.

You use the *SWDIR***util****swadm** utility to configure the aq port range. Refer to "Administering Firewalls" in the *TIBCO iProcess Engine Administrator's Guide* for more information.

Using JMX Through a Firewall

JMX relies on a JAVA technology called RMI. which uses dynamic ports to be able to communicate between a client and a server. Firewalls cannot handle dynamic ports as they need to know the port number. iProcess overcomes this problem by statically assigning a listening port for the RMI server.

Refer to the Post Installation tasks in *TIBCO iProcess Installation* for more information.

Back to Library

Chapter 11 Point-to-Point Data Flow Models

This chapter explains in detail what happens to workflow data when users:

- start a new case of a procedure - see page 132.
- open a work item - see page 135.
- keep a work item - see page 138.
- release a work item - see page 140.

A point-to-point data flow model is shown for each example to identify the flow of information between processes in the iProcess system.

These models can help you understand the TIBCO iProcess Engine so that you can identify any problems to specific areas of the system.

The following descriptions apply to the TIBCO iProcess Workspace accessing the TIBCO iProcess Engine. It does not describe how the TIBCO iProcess Objects Server accesses the TIBCO iProcess Engine.

Back to Library

User Starts a New Case

This section describes a step-by-step example of the data flow between processes when starting a new case using the TIBCO iProcess Workspace. The flow is different if the user who is starting the case:

- is the addressee of the first step
- is not the addressee of the first step.

Both examples are described in the following sections.

Case Starter is Addressee of the First Step

If the user starting the case is the addressee of the first step:

1. From the TIBCO iProcess Workspace, the user selects a procedure and starts a new case.

   The TIBCO iProcess Workspace queries the form definition to see what form to display. This is either read from:

   — memory (if the procedure has previously been accessed).

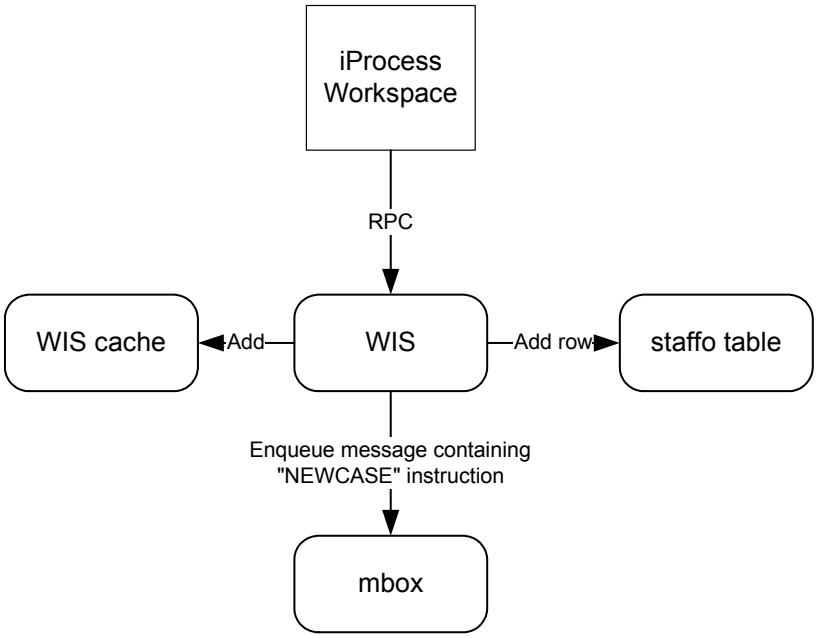
   — the TIBCO iProcess Workspace cache (for example, on Windows 2003 this is in c:**Documents and Settings***user***Application Data****iProcess***nodename*.**n***procname*.**p**

   — the server database procedure definition table (**proc_defn**).

2. The TIBCO iProcess Workspace calls the WIS process via an RPC call to generate the NEWCASE instruction.
3. The WIS process adds a new entry to the **staffo** database table. Because a new row is added, a case number and reqid number are generated by the database sequencing feature.
4. A message containing the NEWCASE instruction, case number and reqid number is posted to the appropriate Mbox set.
5. The BG process polls the Mbox set, reads the new message and sends it to one of the BG processes.
6. The BG processes the case instruction and starts the case including the generation of audit messages. Work items are only sent to queues if the start step is addressed to other users besides the case starter.

Back to Library

The following diagram shows this data flow.

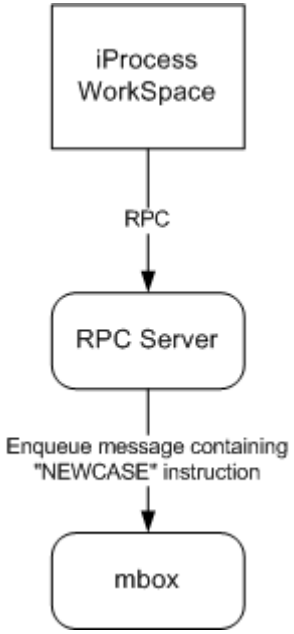


Back to Library

Case Started by Non-Addressee of the First Step

If the case starter is not the addressee of the first step, the TIBCO iProcess Workspace calls the RPC Pool server instead of the WIS. This is because the WIS is responsible for managing Work Queue lists. If the first step is not the user, the BG handles the case start and sends a request to the appropriate queue. The RPC Pool server is used to send the request to reduce the load on the WIS.

In this case, the RPC server sends a NEWCASE instruction to the Mbox set. A case number is added to the instruction but no reqid is required.



Back to Library

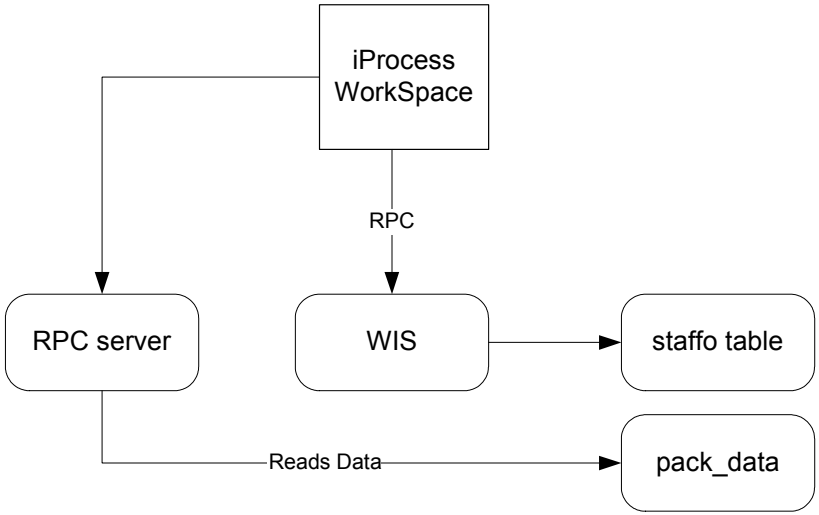
User Opens a Work Item

When a TIBCO iProcess Workspace requests the WIS to open a work item it:

- requests, via RPC, that the appropriate WIS should lock the work item. This is performed in the in-memory caches (and if configured using the **WIS_WRITELOCKS** parameter in *SWDIR***etc****staffcfg**, the lock is written to the **staffo** table). The WIS returns the information about the work item in its cache to the TIBCO iProcess Workspace.
- reads the procedure definition from the RPC Pool server. The TIBCO iProcess Workspace queries the form definition to see what form to display. This is either read from:
  - memory (if the procedure has previously been accessed).
  - the TIBCO iProcess Workspace cache (for example, on Windows 2003 this is in c:**Documents and Settings***user***Application Data****iProcess***nodename*.**n***procname*.**p**
  - the server database procedure definition table (**proc_defn**).
- reads pack data from the pool server.

Back to Library

When a user opens a work item in the Work Item Server, the WIS locks the work item on behalf of the TIBCO iProcess Workspace by setting a lock flag in the **staffo** database table. The WIS contains the skeleton details of the work item and gets the field/value data from the **pack_data** table. The WIS accesses the case data by communicating directly with the database. The following diagram shows the flow of information.



Back to Library

Accessing Memos

If the work item has a memo in the form, the TIBCO iProcess Workspace accesses the database to retrieve the memo data from **pack_memo**. The TIBCO iProcess Workspace does this via RPC and converts the data to a local file on the TIBCO iProcess Workspace. Any further access to the memo data can then be performed via this file.

For example, if a user opens a work item from the **Fred1** queue and opens a memo field in the form, the memo file would be located in the following location:

```
c:\Documents and Settings\user_name\Application
Data\iProcess\nodename.n\procname.p
```

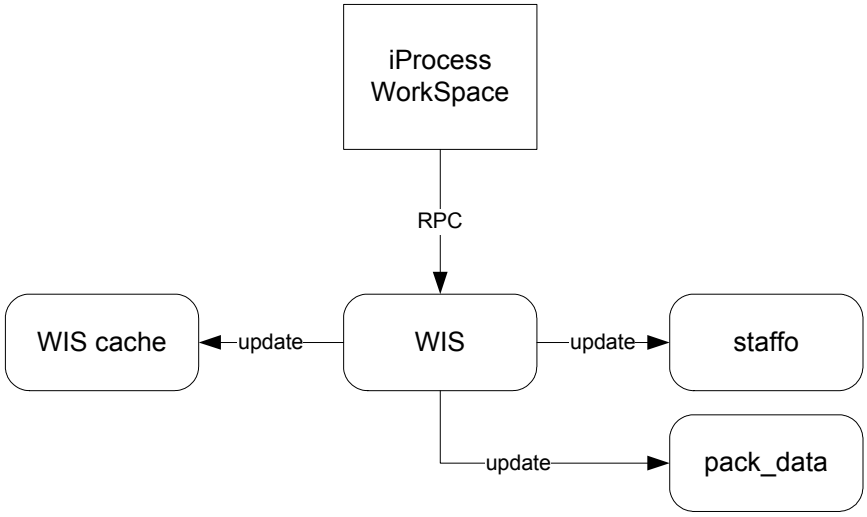
where:

- *user_name* is the name of the Windows user currently logged in
- *nodename* represents the name for the procedure's host node.
- *procname* is the name of the procedure.

Back to Library

User Keeps a Work Item

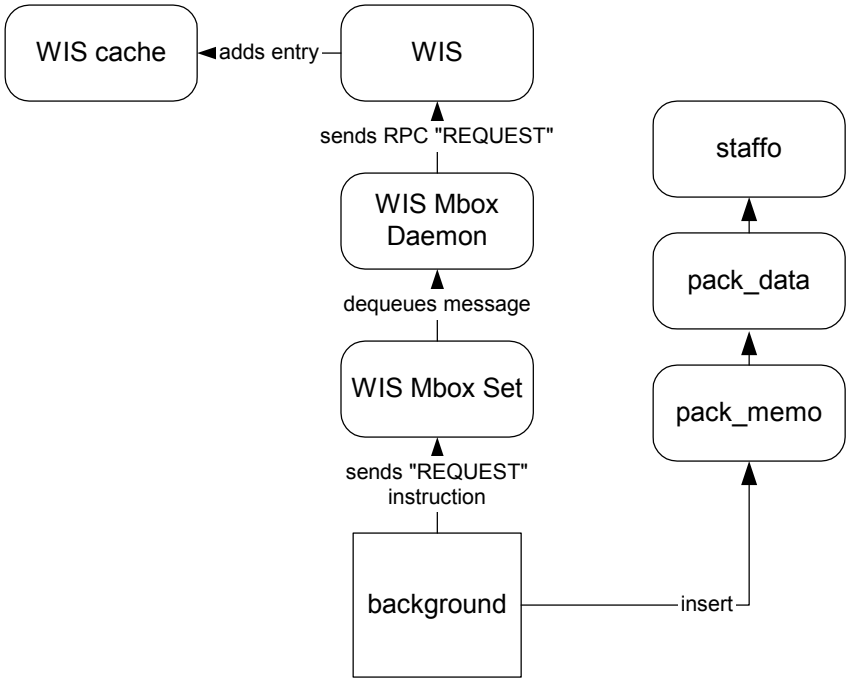
If a user keeps a work item rather than releasing it, the TIBCO iProcess Workspace stores any changed information (changed fields and their data and the operation to perform i.e. KEEP) in memory. This is stored in an in memory package that is sent via RPC calls to the WIS process. The WIS processes the changed information and updates the **staffo**, **pack_memo** and **pack_data** database tables in a single transaction.



Back to Library

Background Sends a Work Item to a Work Queue

When the Background process sends a new request, a new entry is added to the **staffo** table and any pack data is added to the **pack_data** table and memos to the **pack_memo** table. The REQUEST instruction is sent to the WIS Mbox set, which is dequeued by the WIS Mbox Daemon. This process sends the message via RPC to the relevant WIS process. The WIS process adds an entry to the cache for the queue that the instruction was sent to.



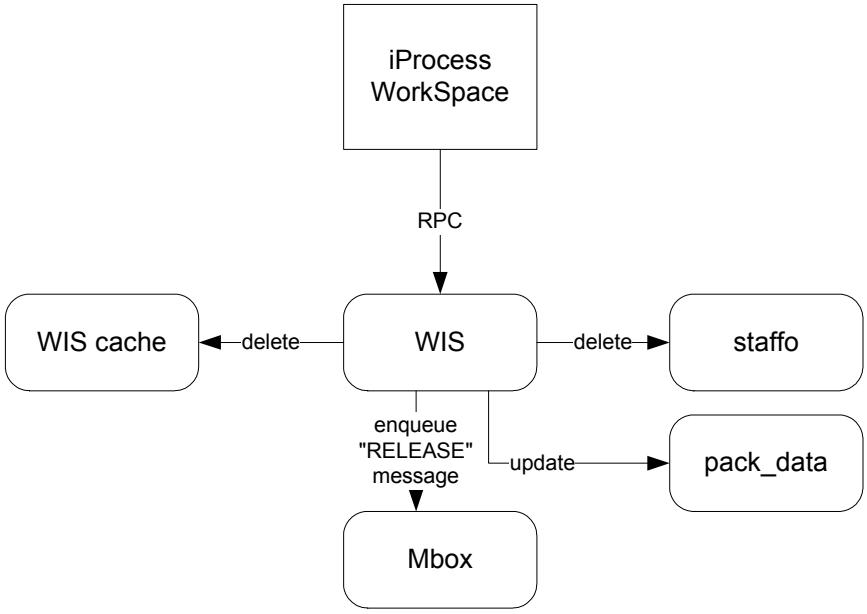
Back to Library

User Releases a Work Item

The data flow for this process is similar to what happens when a user keeps a work item. The TIBCO iProcess Workspace identifies any changes made to the form and creates a data package of the changes (stored in memory). This is sent via RPC calls to the relevant WIS process. The WIS breaks up the package and processes it.

The WIS updates the pack data table, deletes the work item entry in the **staffo** table, deletes the relevant work item in the WIS's cache and sends a RELEASE instruction to the Background Mbox set for a Background to process.

The Background process works out what to do next in the procedure and either terminates the case or sends the next step to the work queue.



Back to Library

The RELEASE instruction informs the Background that the step has been released and the background can update the case data in the database with the new data. The audit trail is updated to show that it has been released. The Background can read the procedure definition to determine what will happen next i.e. terminate the case or process the next step to the foreground.

When the Background sends out a new work item it writes:

- a record to the **staffo** database table
- field and value name pairings to the **pack_data** database table.
- memos to the **pack_memo** table.

Each message references the field/value name pairings in the table by a unique identifier (reqid).

Back to Library

Back to Library