JDBC Adapter Guide

Windchill 8.0

June 2005

Patents of Parametric Technology Corporation or a Subsidiary

Additionally, equivalent patents may be issued or pending outside of the United States. Contact PTC for further information.

GB2363208	25-August-2004	6,545,671 B1	08-April-2003	5,423,02305-June-1990
GB 2365567	10-March-2004	GB2354685B	18-June-2003	4,310,61521-December-1998
6,665,569 B1	16-December-2003	GB2354683B	04-June-2003	4,310,61430-April-1996
GB 2353115	10December-2003	6,608,623 B1	19 August 2003	4,310,614 22-April-1999
6,625,607 B1	23-September-2003	6,473,673 B1	29-October-2002	5,297,05322-March-1994
		GB2354683B	04-June-2003	5,513,31630-April-1996
6,580,428 B1	17-June-2003	6,447,223 B1	10-Sept-2002	5,689,71118-November-1997
GB2354684B	02-July-2003	6,308,14423-October-2001		5,506,95009-April-1996
GB2384125	15-October-2003	5,680,52321-October-1997		5,428,77227-June-1995
GB2354096	12-November-2003	5,838,33117-November-1998		5,850,53515-December-1998
GB2354924	24-September-2003	4,956,77111-September-1990		5,557,17609-November-1996
		5,058,00015-October-1991		5,561,74701-October-1996
6,608,623 B1	19 August 2003	5,140,32118-August-1992		
GB2353376	05-November-2003			
GB2354686	15-October-2003			

Third-Party Trademarks

Adobe, Acrobat, Distiller and the Acrobat Logo are trademarks of Adobe Systems Incorporated. Advanced ClusterProven, ClusterProven, and the ClusterProven design are trademarks or registered trademarks of International Business Machines Corporation in the United States and other countries and are used under license. IBM Corporation does not warrant and is not responsible for the operation of this software product. AIX is a registered trademark of IBM Corporation. Allegro, Cadence, and Concept are registered trademarks of Cadence Design Systems, Inc. Apple, Mac, Mac OS, and Panther are trademarks or registered trademarks of Apple Computer, Inc. AutoCAD and Autodesk Inventor are registered trademarks of Autodesk, Inc. Baan is a registered trademark of Baan Company. CADAM and CATIA are registered trademarks of Dassault Systemes. COACH is a trademark of CADTRAIN, Inc. DOORS is a registered trademark of Telelogic AB. FLEX*lm* is a trademark of Macrovision Corporation. Geomagic is a registered trademark of Raindrop Geomagic, Inc. EVERSYNC, GROOVE, GROOVEFEST, GROOVE.NET, GROOVE NETWORKS, iGROOVE, PEERWARE, and the interlocking circles logo are trademarks of Groove Networks, Inc. Helix is a trademark of Microcadam, Inc. HOOPS is a trademark of Tech Soft America, Inc. HP-UX is a registered trademark Hewlett-Packard Company. I-DEAS, Metaphase, Parasolid, SHERPA, Solid Edge, and Unigraphics are trademarks or registered trademarks of UGS Corp. InstallShield is a registered trademark and service mark of InstallShield Software Corporation in the United States and/or other countries. Intel is a registered trademark of Intel Corporation. IRIX is a registered trademark of Silicon Graphics, Inc. LINUX is a registered trademark of Linus Torvalds, MainWin and Mainsoft are trademarks of Mainsoft Corporation. MatrixOne is a trademark of MatrixOne, Inc. Mentor Graphics and Board Station are registered trademarks and 3D Design, AMPLE, and Design Manager are trademarks of Mentor Graphics Corporation. MEDUSA and STHENO are trademarks of CAD Schroer GmbH. Microsoft, Microsoft Project, Windows, the Windows logo, Windows NT, Visual Basic, and the Visual Basic logo are registered trademarks of Microsoft Corporation in the United States and/or other countries. Netscape and the Netscape N and Ship's Wheel logos are registered trademarks of Netscape Communications Corporation in the U.S. and other countries. Oracle is a registered trademark of Oracle Corporation. OrbixWeb is a registered trademark of IONA Technologies PLC. PDGS is a registered trademark of Ford Motor Company. RAND is a trademark of RAND Worldwide. Rational Rose is a registered trademark of Rational Software Corporation. RetrievalWare is a registered trademark of Convera Corporation. RosettaNet is a trademark and Partner Interface Process and PIP are registered trademarks of RosettaNet, a nonprofit organization. SAP and R/3 are registered trademarks of SAP AG Germany. SolidWorks is a registered trademark of SolidWorks Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and in other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Sun, Sun Microsystems, the Sun logo, Solaris, UltraSPARC, Java and all Java based marks, and “The Network is the Computer” are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and in other countries. TIBCO, TIBCO Software, TIBCO ActiveEnterprise, TIBCO Designer, TIBCO Enterprise for JMS, TIBCO Rendezvous, TIBCO Turbo XML, TIBCO BusinessWorks are the trademarks or registered trademarks of TIBCO Software Inc. in the United States and other countries. WebEx is a trademark of WebEx Communications, Inc. Certain PTC software products contain licensed third-party technology: Rational Rose 2000E is copyrighted software of Rational Software Corporation. RetrievalWare is copyrighted software of Convera Corporation. VisTools library is copyrighted software of Visual Kinematics, Inc. (VKI) containing confidential trade secret information belonging to VKI. HOOPS graphics system is a proprietary software product of, and is copyrighted by, Tech Soft America, Inc. G-POST is copyrighted software and a registered trademark of Intercim. VERICUT is copyrighted software and a registered trademark of CGTech. Pro/PLASTIC ADVISOR is powered by Moldflow technology. Moldflow is a registered trademark of

UNITED STATES GOVERNMENT RESTRICTED RIGHTS LEGEND

Contents

Change Record

This section details major changes applied to this book.

Changes for Windchill 8.0

Change	Description
Entire guide	Removed all references to Info*Engine 5.1 and 6.x, because the adapter is no longer supported on these releases.
Chapter 2, Installing and Configuring the Adapter	Updated the software platform matrix URL.
Chapter 2, Installing and Configuring the JDBC Adapter	Updated the installation procedure.
Chapter 2, JDBC Adapter Properties	Added a new property named Result Set Scrolling Capability.
Chapter 3, The Webject Library	Updated descriptions for ATTRIBUTE, FIELD and CLASS webject parameters.
Chapter 3, Execute-Procedure	Added two new examples: Executing a Stored Procedure that takes a STRUCT type OUT argument Executing a Stored Procedure that takes a STRUCT type IN argument
Chapter 3, Put-Bulk-Stream	Updated PutBulkStream.jsp.

About This Guide

This *JDBC Adapter Guide* documents the use of the PTC Info*Engine JDBC adapter software. It contains the following chapters:

- *About This Guide* is the section you are currently reading. It details the helpful documentation associated with this adapter, the contact information for technical support and documentation comments, as well as the documentation conventions specific to this manual.
- *Architectural Overview* describes the general Info*Engine architecture.
- *Installing and Configuring the Adapter* describes how to install and configure the Info*Engine JDBC adapter.
- *The JDBC Webject Library* details each of the webjects available for use with the adapter.

This guide assumes you are familiar with the basics of HTML, XML, and JSP as defined by the World Wide Web Consortium (http://www.w3c.org).

To take advantage of the advanced functionality of Info*Engine, you must have expert knowledge of HTML, XML, and JSP.

Related Documentation

The following Info*Engine documents may be helpful to you:

- The *Info*Engine Installation and Configuration Guide* details the procedure for installing and configuring the Info*Engine Server.
- The *Info*Engine User's Guide* details the main functionality of the Info*Engine Server.
- The *Info*Engine Java Adapter Development Kit Programming Reference* describes how to develop your own native Info*Engine adapters using the Java programming language if the PTC adapter library does not contain an adapter that suits your needs. The Info*Engine Java Adapter Development Kit (including the documentation) is sold as a separate product. See your sales representative for more information

Technical Support

Contact PTC Technical Support via the PTC Web site, phone, fax, or email if you encounter problems using this adapter.

For complete details, refer to Contacting Technical Support in the *PTC Customer Service Guide* enclosed with your shipment. This guide can also be found under the Support Bulletins section of the PTC Web site at:

`http://www.ptc.com/support/index.htm`

The PTC Web site also provides a search facility that allows you to locate Technical Support technical documentation of particular interest. To access this page, use the following link:

`http://www.ptc.com/cs/search.htm`

You must have a Configuration ID before you can receive technical support. If you do not have an ID, contact PTC License Management using the instructions found in your *PTC Customer Service Guide* under Contacting License Management.

Documentation for PTC Products

PTC provides documentation in the following forms:

- Help topics
- PDF books

To view and print PDF books, you must have the Adobe Acrobat Reader installed.

The adapter documentation is included on the CD. In addition, books updated after release (for example, to support a hardware platform certification) are available from the Reference Documents section of the PTC Web site, at the following URL:

`http://www.ptc.com/appserver/cs/doc/refdoc.jsp`

Comments

PTC welcomes your suggestions and comments on its documentation. You can submit your feedback through the online survey form at the following URL:

`http://www.ptc.com/go/wc_pubs_feedback`

Please include the name of the application and its release number with your comments. For online books, provide the book title.

Documentation Conventions

Info*Engine documentation uses the following conventions:

Convention	Item	Example
Bold	Names of elements in the user interface such as buttons, menu paths, and dialog box titles.	Click **OK**. Select **File > Save**. **License File** dialog box
Italic	Variable and user-defined elements in syntax formats.	**create**_*tablename*.**sql**

Monospace	Examples
	
	Messages
		`</ie:webject>`
		
		`Processing completed.`

"Quotation marks"	Strings	The string "UsrSCM" . . .

1

Info*Engine Architecture

In order to understand the operation of Info*Engine adapters, you must first understand how adapters work within the Info*Engine architecture. This chapter describes each component of the Info*Engine architecture and details how those components work in concert.

Topic **Page**

Identifying the Info*Engine Components

The following components make up the Info*Engine architecture:

- The ***Info*Engine servlet*** provides an interface between the Web server and Info*Engine.
- The ***Info*Engine server*** provides a mechanism for retrieving and manipulating the data that users or custom applications want to view or receive.
- The ***Naming Service*** is the software that supports the operation of Info*Engine components. In the Info*Engine Naming Service, you can identify the LDAP directory servers where entries for the network addresses of Info*Engine components and entries for configuration properties reside.
- The ***Info*Engine Service Access Kit*** *(SAK)* is an application program interface (API) that facilitates the development of Java applications, including JSP pages, that directly utilize the functions and features of Info*Engine. For example, high-level Info*Engine components such as the Info*Engine Servlet, the Info*Engine Server, and the E-Mail Broker use the SAK to invoke tasks and individual webjects.
- The ***native adapters*** provide a direct interface between Info*Engine and information systems.
- The ***non-native adapters*** provide an indirect interface between Info*Engine and information systems. These adapters use a different protocol from the protocol used by Info*Engine and therefore cannot connect directly to Info*Engine.
- ***Gateways*** provide an interface between Info*Engine and non-native adapters.
- The ***Info*Engine SOAP (Simple Object Access Protocol) RPC servlet*** catches and processes Info*Engine SOAP requests that are made over the Web. SOAP is a lightweight protocol that can be used by non-Java applications. By using this protocol, non-Java applications can send requests to execute Info*Engine code and return the output that is generated.
- The ***E-Mail Broker*** provides a process by which users can e-mail Info*Engine requests to a mailbox. Using the SAK, the messages in the mailbox are then passed on to the Info*Engine Server for processing.

The remainder of the chapter describes the relationships among the components.

Identifying Basic Configurations

Info*Engine components can be used in many different software and hardware configurations to meet your business requirements for accessing, managing, and presenting data from many different information systems.

Setting up your Info*Engine environment can be accomplished by:

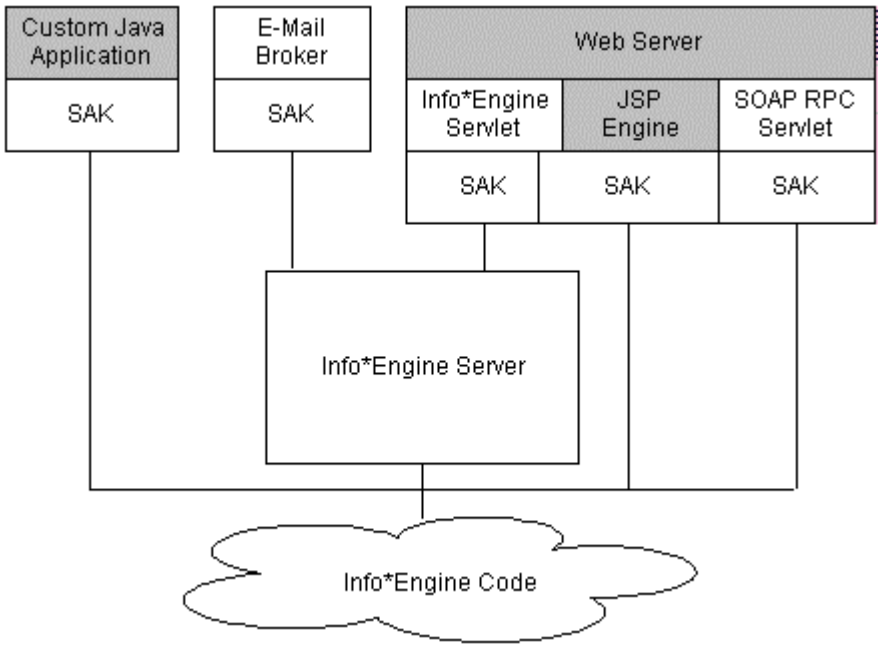
- Establishing interactions with Info*Engine.
- Managing the execution of Info*Engine tasks.
- Starting and managing Info*Engine components.
- Managing connections to the information systems where the data of interest resides.

Interacting with Info*Engine

Initiating an interaction with Info*Engine can be accomplished by using one or more of the following:

- Custom Java applications, including JavaServer Pages (JSP).
- Web Servers that process Info*Engine requests. The requests can come from applications, Web browsers, or wireless devices such as cell phones and personal digital assistants (PDAs).
- E-mail requests that contain formatted messages sent to a predefined Info*Engine mailbox.
- Java Message Service (JMS) events and messages that queue Info*Engine tasks for execution.
- Custom non-Java applications that make requests to execute Info*Engine tasks. These applications use the Info*Engine SOAP Servlet.

The following diagram shows how the Info*Engine components and other customer software components can interact to execute Info*Engine code.



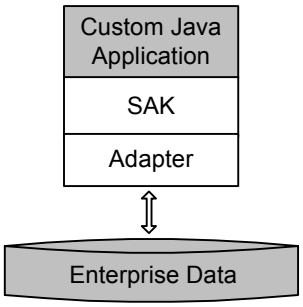
Info*Engine code consists of Java classes that are accessed through the Info*Engine API. The API is available through the SAK and externalizes predefined functions called webjects and tasks. The webjects and tasks can be easily instantiated and invoked as Java objects from a Java application or in a text file. Info*Engine text files can be accessed using requests or code within an application.

The following sections provide more details about how to use the Info*Engine components with your software.

Using a Custom Java Application

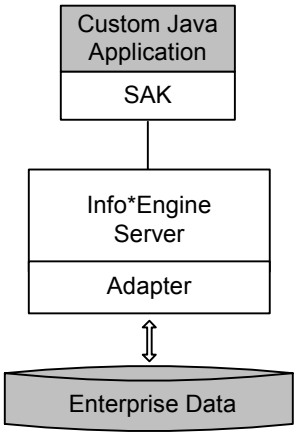
By coding a custom application in Java, you can have quick and easy access to Info*Engine without the added complexity of a Web server. By using the API defined in the SAK, you can execute Info*Engine webjects, tasks, and other Info*Engine code in the Java Virtual Machine (JVM) where the application resides.

The following diagram shows the SAK and adapter classes being used in the application to access data in a remote database.



Within a Java application, you also have the flexibility of executing Info*Engine tasks that are maintained outside of the application. An Info*Engine task consists of a set of webjects and surrounding code that supports the processing of the webjects. These tasks can then be processed either in the JVM of any Info*Engine Server or in the JVM of the application.

The following diagram shows the Info*Engine components that are used when an application executes a task in an Info*Engine Server. In this case the application requests that a task be executed in the server that accesses data in a remote database.



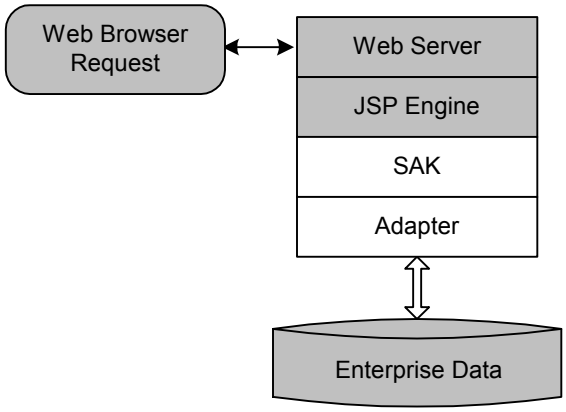
Using a Web Server to Process Info*Engine Requests

The installation process steps you through a procedure that deploys Info*Engine as a Web application. Going through the installation process sets up your Web server and its servlet engine to identify Info*Engine requests and pass those requests on to Info*Engine components for processing. After the installation is complete, your Info*Engine environment is set up so that Info*Engine requests to execute JSP and HTML pages coming from Web browsers are processed correctly.

By doing some additional Info*Engine configuration steps, you can set up your Info*Engine environment to process requests from the following additional sources:

- If you configure Info*Engine to identify wireless communication protocols, requests can come from wireless communication devices such as a cell phone.
- If you configure the Info*Engine SOAP RPC Servlet, Info*Engine SOAP requests can come from non-Java applications.

The following diagram shows the relationships among the components that process Web browser requests for JSP pages.



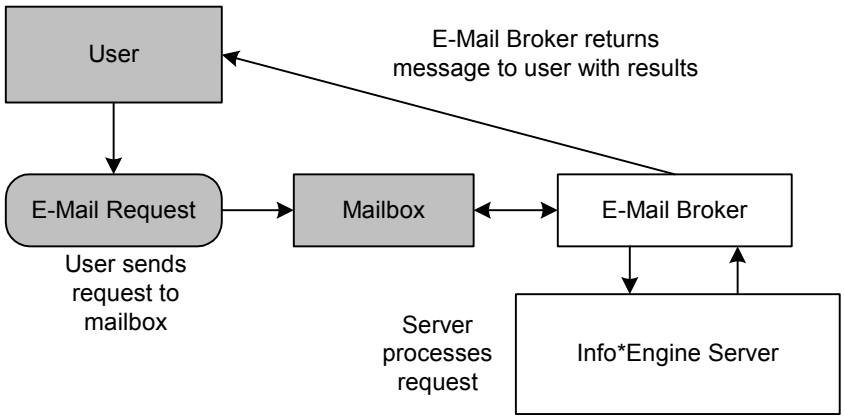
This diagram shows the components that are used when the request specifies that Info*Engine execute a JSP page. By default, Info*Engine and Web server configuration specifies that JSP pages are processed in the JSP engine of the servlet engine installed on your Web server. The JSP engine creates an instance of the SAK, which is then used to execute the Info*Engine-specific code on the page. For example, if a user clicks a link or uses a URL in a browser window that serves as a JSP request for information from Info*Engine, the JSP engine and the SAK work together to manage the request.

The SAK processes the request and, as needed, connects to specialized Info*Engine adapters that communicate with external applications such as Oracle databases, PDM systems, various legacy systems, and ERP systems. After the requested information is obtained from the external applications, the process reverses itself and ultimately displays information in the user's browser window.

Making E-Mail Requests to Info*Engine

The E-Mail Broker allows users to make Info*Engine requests by e-mail.

The E-Mail Broker provides a process that monitors a mailbox for requests to execute Info*Engine templates and tasks. When a request arrives in the mailbox, the E-Mail Broker connects to the server and passes the request to the Info*Engine server for processing. It also captures output from the processed template or task, and returns the output in an e-mail message to the address specified in the From or Reply-To heading of the original request.



Managing the Execution of Info*Engine Tasks

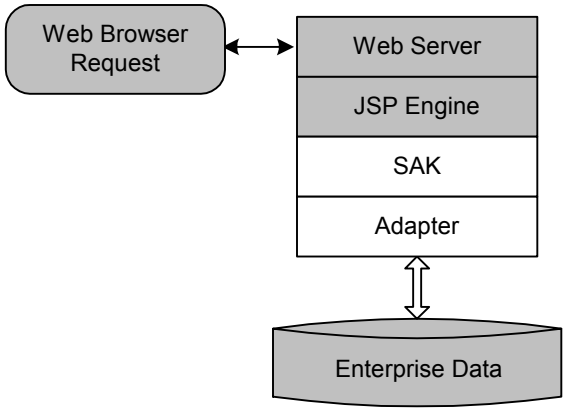
Info*Engine tasks control the retrieval and manipulation of data. Tasks consist of the following:

- Info*Engine webjects that retrieve and manipulate data.
- Surrounding Info*Engine custom tags that manage the execution of the webjects.

There are two basic ways to execute tasks:

- Incorporate tasks directly into any Java application, including JSP pages, using Info*Engine custom tags.
- Put the tasks in individual text-based documents, specify which tasks to execute in the Info*Engine custom tags within a Java application (or JSP page).

The decisions about how and where to execute Info*Engine tasks depend on your system requirements. For example, if you have a dedicated environment where one system contains both your Info*Engine application and all of the required software components, and the tasks to execute do not require any complex processing, you may choose to execute your tasks from within JSP pages that are also used to display the results. In this case, the environment used could be similar to the following:



The JSP engine depicted in the diagram instantiates an instance of the SAK within the JVM of the JSP engine. The SAK is then used to process the Info*Engine custom tags. Some of the Info*Engine tags can execute webjects that extract data from enterprise systems through an adapter, while others can display the data. In this example, all of the webjects are contained in the same JSP page.

In a more complex environment where you have a large Java application that executes complex tasks, you can manage the tasks more efficiently by separating them into individual documents, rather than coding them directly into the application. When a task is contained in its own document, it is called a standalone task. For a standalone task, the following processing options are available:

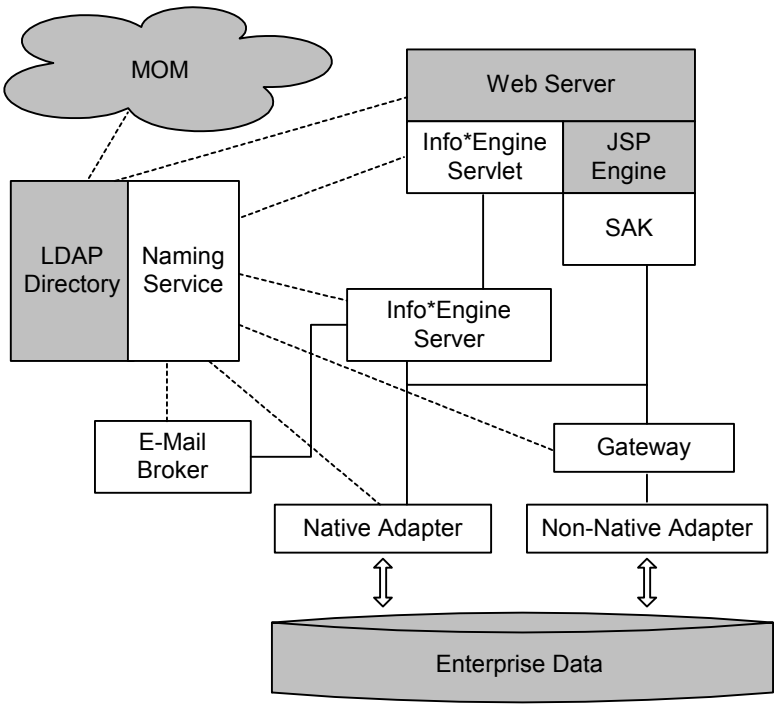
- You can specify where you want a standalone task to execute, whether it is in the same JVM as the application or in the JVM of any Info*Engine Server that is part of your environment.
- You can specify how you want to execute standalone tasks that do not execute in the same JVM as the application. There are three ways to execute these standalone tasks:
  - Requesting, through a TCP/IP connection, that the task executes in a specific Info*Engine server. Each Info*Engine server listens for task requests and executes them upon arrival.
  - Implementing a specific event that executes tasks. Establishing events through an Info*Engine Web Event Service allows you to execute tasks based on specific actions that can occur in your environment.
  - Queuing a task for execution. After you queue a task, you can disconnect from your application. Any results are queued for later retrieval either by you or others. By queuing a task, you can also guarantee that the task will be completed, even if it is interrupted due to a system problem.

  By performing the basic Info*Engine installation, the Info*Engine server is set up to receive task requests. To use either queues or events for executing tasks, you must install and configure additional Message-Oriented Middleware (MOM) software and then update your Info*Engine configuration.

Starting and Locating Info*Engine Components

The Naming Service uses an LDAP directory to provide the Info*Engine Servlet, the Info*Engine server, the native adapters, and the Info*Engine gateways with a means of locating each other, acting as a traffic director of sorts.

In the following diagram, dashed lines represent the communication between the Naming Service, Info*Engine components, and third party software that could be installed.



Additionally, if you configure the Info*Engine SOAP RPC servlet, there will be an entry in the Naming Service for this servlet.

The Naming Service can be used to automatically start Info*Engine components residing on the same hardware system. By default, the Naming Service is set up during the installation to start the Info*Engine Server and the E-Mail Broker. Depending on where you install adapters and gateways, you may want to configure the Naming Service to start them as well.

Setting Up Connections Through Adapters

Adapters provide a connection between the Info*Engine Server and information systems. One side of the adapter communicates with the Info*Engine Server and the other side communicates with the information system. The adapter translates Info*Engine Server requests into information system requests.

Info*Engine provides two types of adapters:

- **Native adapters** are implemented in the Java language and conform to the formal Info*Engine interface specification. For example, the JNDI and JDBC adapters are native adapters.
- **Non-native adapters** are implemented in a non-Java language or do not conform to the formal Info*Engine interface specification. Because the implementation is different from Info*Engine, you must also define a gateway for each non-native adapter you install. Gateways translate Info*Engine requests so that the adapters can process them. After an adapter receives a request, the adapter sends it to the associated database or data repository. The adapter also returns any information obtained from the data repository to the gateway where it is translated and passed back to the Info*Engine Server.

The adapters you must use are determined by the information systems from which you want to retrieve information. Info*Engine provides a unique adapter for each information system. For example, to retrieve information from a Metaphase database, you must install and configure the Metaphase adapter.

Native adapters can be installed as follows:

- Residing in the same Java Virtual Machine as the Info*Engine webject that accesses the adapter (known as the in-process adapter).
- Distributed in their own Java Virtual Machine on the same hardware system or on remote hardware systems (known as out-of-process adapters).

How to install native adapters is determined by your site.

Gateways usually reside in the same Java Virtual Machine as the calling webject since the code for gateways is installed as part of Info*Engine.

Non-native adapters are always distributed in their own environment and are run as out-of-process adapters.

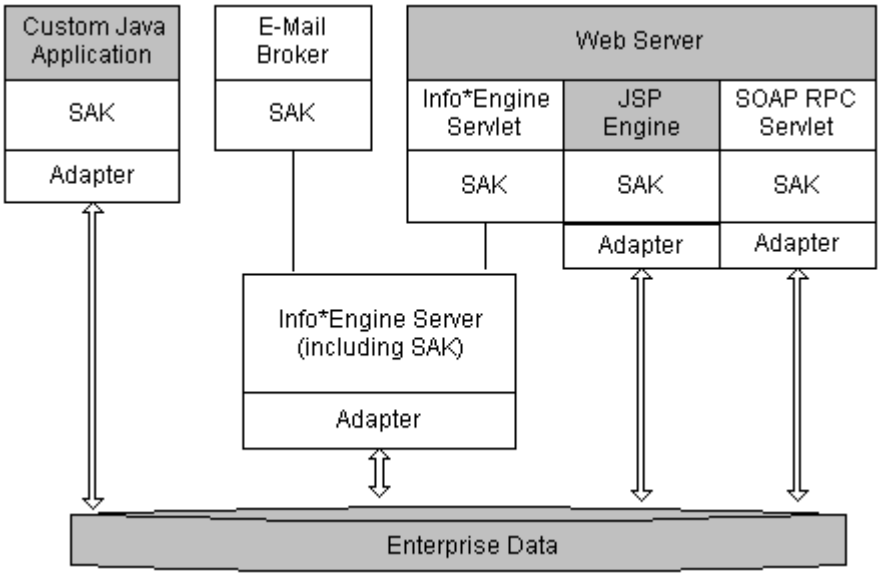
The following sections expand on the installation options.

Using In-Process Adapters and Gateways

In-process adapters and gateways are installed and run in the same Java Virtual Machine as the calling webject. Only native adapters and gateways can be configured to run in the same JVM as the calling webject. The SAK determines which classes are required when processing webjects for an in-process adapter or gateway, and instantiates the classes in the JVM. Therefore, the communication between the webject and the adapter or gateway is very efficient.

Configuring in-process adapters and gateways minimizes communication delays and resource usage; however, the total resource usage of the machine hosting the Info*Engine code may be increased because of the additional load burden of running the adapter or gateway.

When an adapter is configured to be an in-process adapter, the adapter classes can be instantiated by any SAK that executes adapter webjects. The following diagram shows adapter classes residing in the JVM of a custom Java application, the Web server, and the Info*Engine Server:



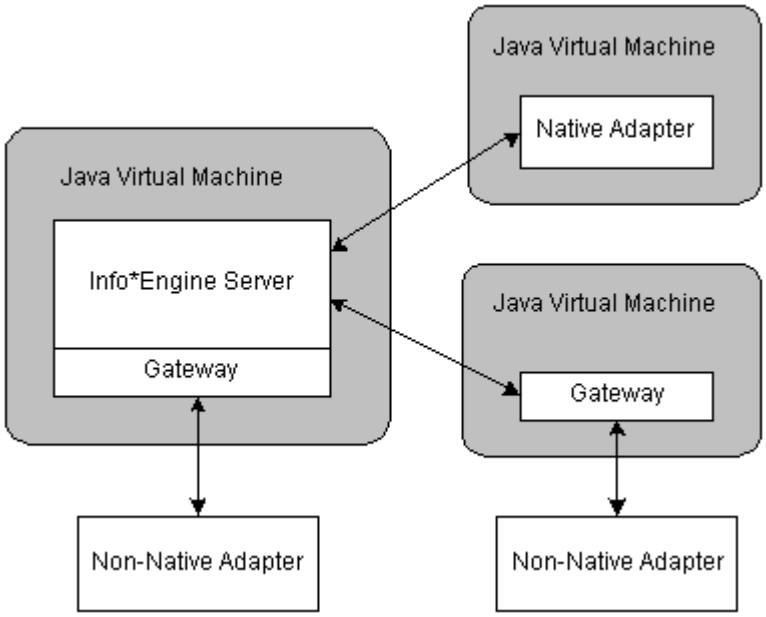
As shown in the diagram, no external communication is needed between the SAK and the adapter when the adapter is in the same process.

Running in-process native adapters and gateways is generally the preferred configuration if the resource usage on a single system is not excessive.

Using Out-of-Process Adapters and Gateways

Distributing adapters across multiple hardware systems reduces the overall resource usage on the machine hosting the Info*Engine code; however, it does introduce some delay and resource usage associated with using a TCP/IP connection for communicating between Info*Engine components and each adapter.

The following diagram shows the communication lines that are used when three adapters and one gateway are distributed.



Distributed native adapters and gateways are installed and run in their own Java Virtual Machine. These virtual machines can be on the same hardware system as the Info*Engine Server or on a different hardware system. Non-native adapters can only be configured as out-of-process adapters, and they always run as separate processes. Although gateways for non-native adapters are typically configured as in-process gateways to minimize the communication delays, they do not need to be in the same process.

The deployment of distributed adapters at your site may be determined by a company policy that requires the adapter to be located near the application it accesses, or it may be based on administrative reasons. One reason for running a native adapter in its own Java Virtual Machine could be to better manage the resource usage of the virtual machine.

2

Installing and Configuring the Adapter

This chapter describes all of the installation and configuration procedures for the JDBC adapter.

See the Certified Software Platform Matrix (http://www.ptc.com/cs/doc/index.htm) for information on supported systems and platforms. You can view the matrix from:

http://www.ptc.com/appserver/cs/doc/refdoc.jsp

This URL directs you to the PTC Online Support Web page for reference documents. For your document search criteria, select **Windchill Info*Engine** from the **Product** drop-down list. Then select the certified software platform matrix for this release from the returned document list.

Topic **Page**

Installation Overview

Regardless of whether you are performing a new installation or upgrading an existing installation, PTC suggests you review the following summary before proceeding. Installing or upgrading the Info*Engine JDBC adapter takes several steps and requires strong knowledge of the current JDBC installation at your site.

1. Prepare yourself and your site for the installation or upgrade.

   Carefully read the sections detailing the information required before you can begin installing or upgrading all adapter software. The installation or upgrade may require special access permissions.

2. Mount the Windchill Info*Engine JDBC Adapter CD-ROM or if you have downloaded the software, navigate to the directory that contains your adapter software.

   If the adapter software was distributed on a CD-ROM, be sure you know the appropriate mount point at your site for a CD-ROM installation.

3. Install the adapter.

   You can install the adapter into any directory.

4. Configure the adapter and resolve error messages, if any, that appear during startup.

5. Test the installation of the adapter and resolve error messages, if any, which appear.

Before You Begin an Adapter Installation

Several items must be obtained or considered before beginning the installation and configuration of the JDBC adapter. Read these items carefully before beginning the installation process.

1. Ensure that the Java 2 Software Development Kit version 1.4 (SDK) has been installed and is running on the adapter host. If Info*Engine has been installed on this host, then the SDK has already been installed.

   If you have not downloaded and installed the SDK for the machine running the adapter host, you can obtain free or evaluation copies from the following website:

   http://www.javasoft.com/products

2. Ensure that the JDBC class libraries and the drivers appropriate for your platform, operating system, and databases have been installed.

   JDBC class libraries and drivers are available from the vendors of your databases, not from PTC. Many database vendors offer downloadable drivers via the World Wide Web. The Sun Microsystems website contains a list of JDBC drivers that are publicly available for downloading from various vendors:

   http://industry.java.sun.com/products/jdbc/drivers

   The Info*Engine JDBC adapter and Info*Engine do not work properly without the appropriate database drivers. JDBC drivers that support the JDBC 3.0 API can be used with the JDBC adapter. To ensure optimum performance between the adapter and the databases you will access, PTC highly recommends using type 4 drivers, which convert JDBC calls into a network protocol used by a database directly.

   Type 2 drivers will work well with the Info*Engine JDBC adapter, but performance will be less than the optimum performance provided by type 4 drivers. Type 3 drivers also work with the JDBC adapter, but they usually require additional software and the performance will also not be optimum.

   The type 1 driver, known as the JDBC-ODBC bridge, may be used with the JDBC Adapter; however, performance will be less than optimum and additional software from your database vendors may be required. The type 1 driver is part of many versions of the JDK and JRE software and may already exist at your site. It is important to remember that the type 1 driver does not allow direct access to databases on its own. An ODBC client must be used between the JDBC-ODBC bridge and your databases. Many databases today come with the ODBC client as part of their installation or take advantage of an ODBC client native to the operating system on which they run. In either case, you must configure your ODBC client to communicate with your database.

3. Ensure that the Info*Engine components are installed and functioning correctly.

   The Info*Engine installation procedure includes requirements for installing a compatible Web server and servlet engine. The JDBC adapter works with the same Web servers and servlet engines as Info*Engine. For information on installing and configuring Info*Engine, see the *Info*Engine Installation and Configuration Guide*.

Installing and Configuring the JDBC Adapter

The installation and configuration process for the JDBC adapter is outlined in the following sections. The process differs depending on whether the adapter is to be used in-process with Info*Engine, out-of-process with Info*Engine on the same host or out-of-process with Info*Engine on a different host. To complete the installation and configuration, you must create a JDBC Adapter LDAP entry after you complete the steps to install and configure the JDBC Adapter as described in the section titled Creating the JDBC Adapter LDAP Entry.

The InstallAnywhere framework is used by the JDBC Adapter installation program. For basic information about using InstallAnywhere see the *Windchill Info*Engine Installation and Configuration Guide*.

Installation and Configuration for Info*Engine on the Same Host

The installation and configuration process for the JDBC adapter to run in-process with Info*Engine (or out-of-process with Info*Engine on the same host) can be broken down into the following steps. These are applicable to Windows, but the procedure is nearly the same even for the other supported Operating Systems.

Note: You can click **Previous** and **Next** throughout the installation process if you want to revise the installation selections.

1. Start the installer as follows:

   – On a Windows system, if you are installing the adapter from a CD-ROM and have autorun enabled, the installer starts automatically.

     If the installer does not start within 30 seconds or you have downloaded the software, double-click the setup.vbs file that resides at the top level on the CD or downloaded directory.

   – On a UNIX system, execute the setup file that resides at the top level on the CD or downloaded directory.

   The **Before You Begin** panel opens.

2. Click **Next**. On the **Select Installation Type** panel, you will be asked to select one of three installation types; these options are listed in the table below:

Installation Type	Description
Typical	Selecting this results in the most common application features getting installed.
Configuration	Selecting this configures an existing JDBC adapter installation.
Custom	Selecting this enables one to customize the features being installed.

   Select **Typical.** This option will install default settings, and is recommended for most users.

3. Click **Next**. On the **Select Directory** panel, specify the installation directory for the JDBC adapter. If the specified directory does not already exist, a panel appears asking if you want to create the new directory. Click **Yes** to proceed. You can also click **Browse** to find an existing directory.

4. Click **Next**. The **Specify Option** panel opens.

   Select one of the following options:

   – **In-process with Info*Engine**

     If you choose this option, continue on to the next step.

   – **Out-of-process (with Info*Engine on the same host)**

     If you choose this option, continue on to the next step.

   – **Out-of-process (with Info*Engine on a different host)**

     If you choose this option, see Installation and Configuration for Info*Engine on a Different Host.

5. Click **Next**. On the **Specify Directory** panel, specify the installation directory for Info*Engine. You can also click **Browse** to find the directory.

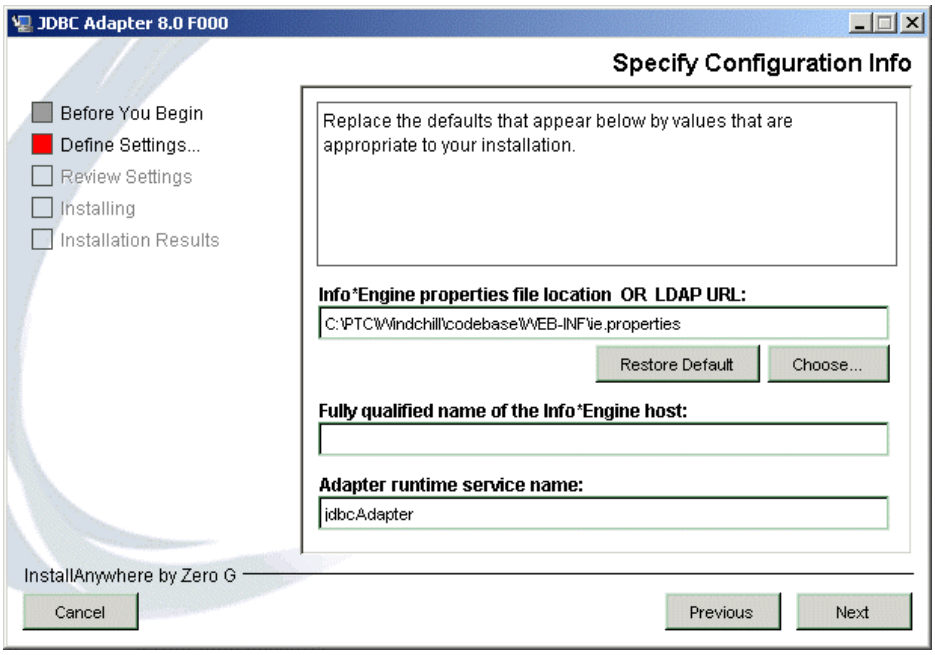
6. Click **Next**. The next panel is determined by your previous selection:

   – If you selected **Out-of-process (with Info*Engine on the same host)** in the previous step, then, the **Specify Database Type** panel opens. Specify whether the adapter will connect to an Oracle or a non-Oracle database.

   – If you selected **In-prcoess with Info*Engine** in the previous step, then you are not prompted for the database type; skip the next step and go to Step 8.

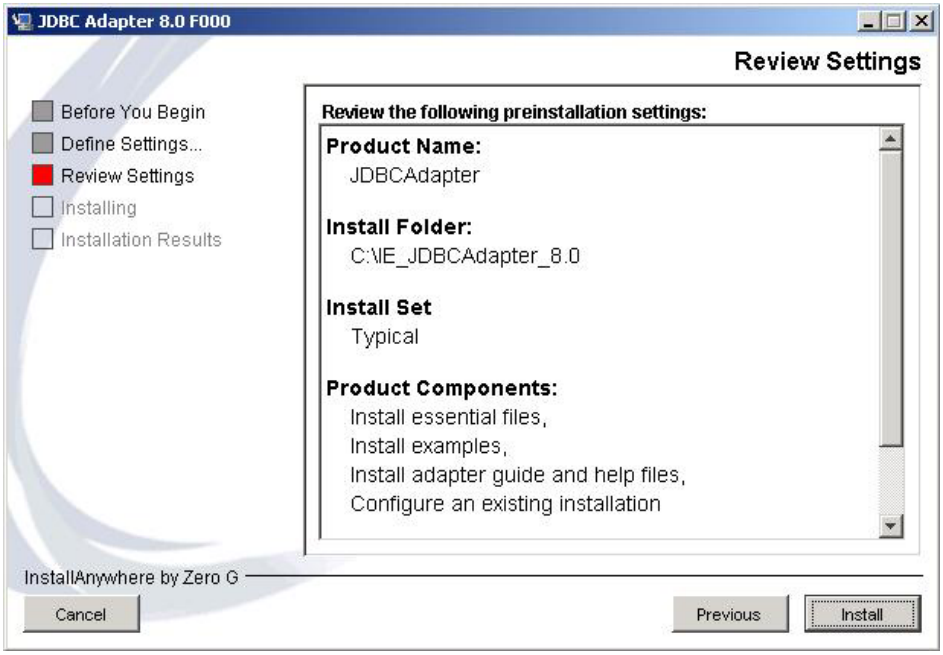
7. Click **Next**. If you chose Oracle from the previous panel for the database type, specify the installation directory for Oracle.

8. Click **Next**. The **Specify Configuration Info** panel opens. The default values appearing in this panel may be replaced by those that are appropriate to your installation.



9. Click **Next**. If you chose to install the adapter to run out-of-process, the **Select Shortcut Location** panel opens. Select a location for the shortcut to the JDBC Adapter.

10. Click **Next**. On the **Review Settings** page, review the pre-installation settings information that appears. If you need to make a change, click **Previous**.



11. Click **Install**.

12. When the installation completes successfully, the **Installation Complete** panel displays the directory where the JDBC Adapter was installed.

    Note: The installation log files are located in the <*installation_directory*>\installer\logs directory. The log files for the installation are named:

    - JDBCAdapter_InstallLog.xml
    - JDBCAdapter_PtcInstall.log

Note: When the adapter connects to a database other than Oracle, complete the installation by carrying out the following manual steps:

- When installing the adapter to run in-process, copy the JDBC driver provided by the database vendor to the <*ie_dir*>\codebase\WEB-INF\lib directory.
- When installing the adapter to run out-of-process, copy the JDBC driver provided by the database vendor to the adapter installation location and include this location in the CLASSPATH entry in the adapter start file.

Installation and Configuration for Info*Engine on a Different Host

The installation and configuration process for the JDBC adapter to run as an out of process one with Info*Engine running on a different host can be broken down into the following steps. These are applicable to Windows, but the procedure is nearly the same even for the other supported Operating Systems.

Note: You can click **Previous** and **Next** throughout the installation process if you want to revise the installation selections.

1. On a Windows system, if you are installing the adapter from a CD-ROM and have autorun enabled, the installer starts automatically.

   If the installer does not start within 30 seconds or you have downloaded the software, double-click the setup.vbs file that resides at the top level on the CD or downloaded directory.

   On a UNIX system, execute the setup file that resides at the top level on the CD or downloaded directory.

   The **Before You Begin** panel opens.

2. Click **Next**. On the **Select Installation Type** panel, you will be asked to select one of three installation types; these options are listed in the table below:

Installation Type	Description
Typical	Selecting this results in the most common application features getting installed.
Configuration	Selecting this configures an existing JDBC adapter installation.
Custom	Selecting this enables one to customize the features being installed.

   Select **Typical.** This option will install default settings, and is recommended for most users.

3. Click **Next**. On the **Select Directory** panel, specify the installation directory for the JDBC Adapter. You can also click **Browse** to find the directory.

4. Click **Next**. The **Specify Option** panel opens.

   Select one of the following options:

   – **In-process with Info*Engine**

     If you choose this option, see Installation and Configuration for Info*Engine on the Same Host.

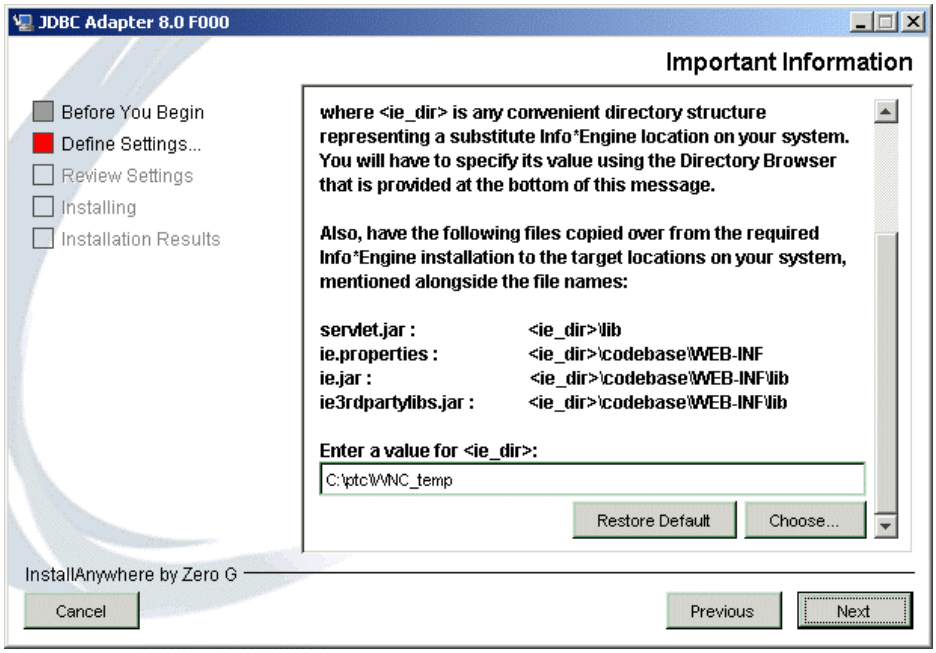
   – **Out-of-process (with Info*Engine on the same host)**

     If you choose this option, see Installation and Configuration for Info*Engine on the Same Host.

   – **Out-of-process (with Info*Engine on a different host)**

     If you choose this option, continue on to the next step.

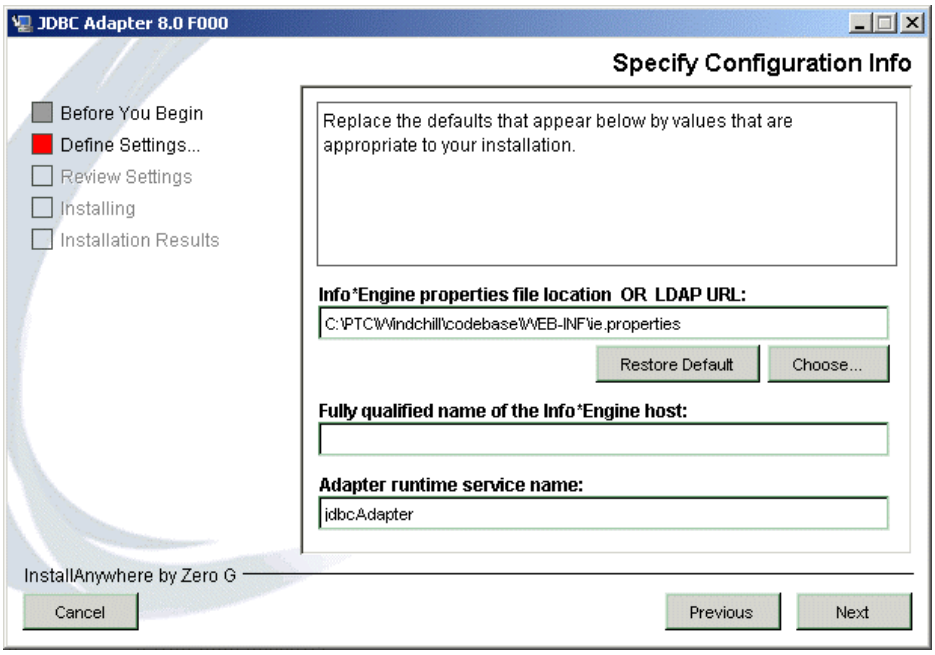
5. Click **Next**. The **Important Information** panel opens. Follow all the instructions in the panel and enter a value for <*ie_dir*> in the field that appears at the bottom of the panel.



6. Click **Next**. On the **Specify Database Type** panel, specify whether the adapter will connect to an Oracle or a non-Oracle database.

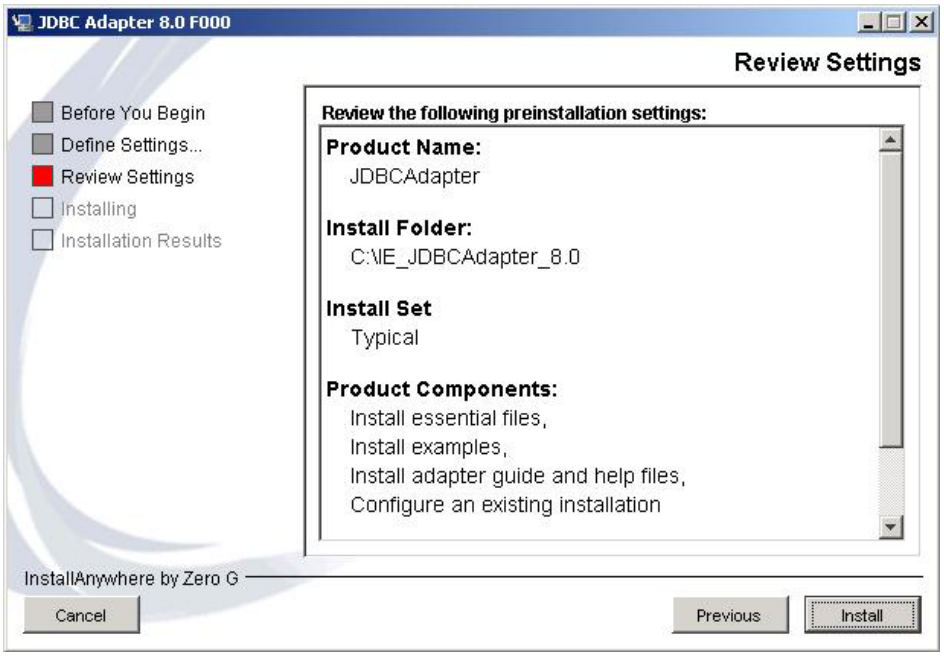
7. Click **Next**. If you chose Oracle from the previous panel for the database type, specify the installation directory for Oracle.

8. Click **Next**. The **Specify Configuration Info** panel opens. The default values appearing in this panel may be replaced by those that are appropriate to your installation.



9. Click **Next**. On the **Select Shortcut Location** panel, select a location for the shortcut to the JDBC Adapter.

10. Click **Next**. On the **Review Settings** page, review the pre-installation settings information that appears. If you need to make a change, click **Previous**.



11. Click **Install**.

12. When the installation completes successfully, the **Installation Complete** panel displays the directory where the JDBC Adapter was installed.

    Note: The installation log files are located in the <*installation_directory*>\installer\logs directory. The log files for the installation are named:

    - JDBCAdapter_InstallLog.xml
    - JDBCAdapter_PtcInstall.log

Note: When the adapter connects to a database other than Oracle, complete the installation by carrying out the following manual steps:

- When installing the adapter to run in-process, copy the JDBC driver provided by the database vendor to the <*ie_dir*>\codebase\WEB-INF\lib directory.
- When installing the adapter to run out-of-process, copy the JDBC driver provided by the database vendor to the adapter installation location and include this location in the CLASSPATH entry in the adapter start file.

Installing the JDBC Adapter Form and Help

The JDBC adapter maintains its properties as attributes in Info*Engine adapter LDAP entries. To define the adapter name and other adapter attributes, and to establish adapter properties, you must create a JDBC adapter LDAP entry using the Info*Engine Property Administrator.

Use the Info*Engine Property Administrator to add or modify existing Info*Engine adapter service LDAP entries. Running the adapter installer results in the file jdbc.ldif getting installed under the specified installation location, which when imported into the Property Administrator provides the form you use to create the JDBC adapter LDAP entry.

For general information about using the Info*Engine Property Administrator, see the *Info*Engine Installation and Configuration Guide* and the online help provided in the administrator.

When installing the adapter to run out-of-process with Info*Engine on a different host, the file jdbc.ldif has to be imported and the help files copied over manually as described below (you can skip this section, if you are installing the adapter to run in-process or out-of-process with Info*Engine on the same host).

To import the JDBC adapter form and add the form help, do the following:

1. Log into the Info*Engine host as any user.
2. Copy the jdbcAdapter.jar file from the adapter installation directory which you created in the previous section to the codebase\WEB-INF\lib directory where Info*Engine is installed.

   The Info*Engine Property Administrator needs the resource bundle that is in the jdbcAdapter.jar file in order to display the correct text on the JDBC adapter form.
3. Add the JDBC property and form help files to the Info*Engine Property Administrator help directories.

   a. Copy the jdbcadaptersproperties_w.html file from the <*jdbc_dir*>\codebase\infoengine\jsp\admin\help\en_US\Adapters directory to the following Info*Engine directory:

      <*ie_dir*>\codebase\infoengine\jsp\admin\help\en_US\Adapters

      where <*jdbc_dir*> is the adapter installation location and <*ie_dir*> is the directory where Info*Engine is installed.

b. Copy the jdbc subdirectory and all files in this subdirectory from the <*jdbc_dir*>\codebase\infoengine\jsp\admin\help\en_US\Adapters directory to the Info*Engine directory:

   <*ie_dir*>\codebase\infoengine\jsp\admin\help\en_US\Adapters

   where <*jdbc_dir*> is the adapter installation location and <*ie_dir*> is the directory where Info*Engine is installed.

4. From the Info*Engine Property Administrator, click the **[Import/Export]** link.
5. Click the **Overwrite leaf entries if they exist** check box.
6. Browse to the jdbc.ldif file that resides under the adapter installation location.
7. Click **OK** to import the form.

This overwrites any existing JDBC form and returns you to the main page.

Verification

Performing a Typical installation of the JDBC adapter on the same host where Info*Engine is installed causes certain example JSP and XML task files to be installed on your system. The JSP files get installed under <*ie_dir*>\codebase\infoengine\jsp\examples\JDBCAdapter\examples while the XML task files get installed under <*ie_dir*>\tasks\infoengine\examples\JDBCAdapter\example-tasks, where <*ie_dir*> is the Info*Engine installation directory. Performing a JDBC adapter installation on a host different from the Info*Engine host places the examples in the JDBC installation directory; you must manually copy the examples to an appropriate location under the Info*Engine installation directory as seen in the paths above.

These examples allow you to test the adapter installation.

1. Edit the INSTANCE parameter in each example to the name of the adapter instance you are testing. For further information on the INSTANCE parameter see the section titled Naming the Adapter in Webject INSTANCE Parameters.
2. Edit the parameter values in each example. Specify valid values for your site.
3. Run the examples to confirm that your adapter installation is operating correctly.

Creating the JDBC Adapter LDAP Entry

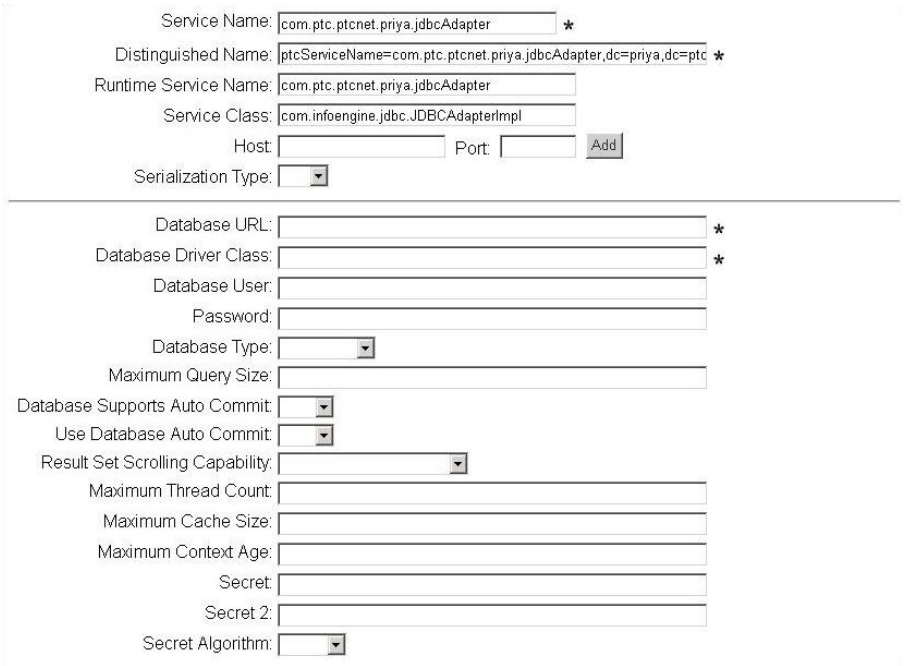
As stated earlier in this guide, the JDBC adapter properties are maintained as attributes in Info*Engine adapter LDAP entries.

Use the Info*Engine Property Administrator to add or modify existing Info*Engine adapter service LDAP entries.

For general information about using the Info*Engine Property Administrator, see the *Info*Engine Installation and Configuration Guide* and the online help provided in the Property Administrator.

Use the following procedure to create a JDBC Adapter LDAP entry:

1. To create a new JDBC adapter service LDAP entry, select **JDBC Adapter** from the **Create Entry** list on the Info*Engine Property Administrator main page. A form that includes the following displays:



2. Choose values for the required fields. The required fields are marked with an asterisk (*). Click on the field heading to display information about the JDBC properties on the form. For additional information see the section titled JDBC Adapter Properties later in this chapter.

   All forms include the following set of common fields that are located at the top of the form:

   Service Name
   Distinguished Name
   Runtime Service Name
   Service Class
   Host
   Port
   Serialization Type

When the form displays, Property Administrator populates the **Service Name**, **Distinguished Name**, and **Runtime Service Name** fields with suggested names. These names are based on information provided when you logged in to the administrator and also information that is stored in the form. You can change these names to match the criteria set up for your site LDAP entries. For additional information about these fields, view the online help for your form or talk to your Info*Engine administrator. Additional information about setting up the criteria your site should use for creating LDAP entries can be found in the *Info*Engine Installation and Configuration Guide*.

Service Class contains the service class name used for the adapter. If you are using the adapter as an in-process adapter, leave the default name. If the adapter will only be used out of process, you must delete the name in the **Service Class** field and add the host and port used to access the adapter in the **Host** and **Port** fields.

The **Serialization Type** field allows you to change the type of data serialization Info*Engine uses when passing data to an out-of-process JDBC adapter. By default, Info*Engine components use Java serialization when passing data between components. Java serialization preserves data type information so that the data can be easily manipulated from within an Info*Engine custom application, Java Server Page, or task. To pass only XML, you would change the type to **XML**.

For help on **Additional Properties, Co-resident Services,** and **Additional Services** options click the **Help** link on the main Property Administrator page.

3. Click **Create Adapter** to complete.

Using the Adapter In Process

If you are using the adapter as an in-process adapter, leave the default service class name in the **Service Class** field.

Using the Adapter Out of Process

If you are using the adapter only out of process, you must delete the name in the **Service Class** field and add the host and port used to access the adapter in the **Host** and **Port** fields.

Sample Start File Contents

Performing a Typical installation to run the adapter out-of-process with Info*Engine results in a start file getting installed under the adapter installation location. This file can be used for starting the JDBC adapter as an out-of-process adapter.

Sample Windows Start File Contents

The following lines document the start file for the Windows Operating System - note that this is only a sample and the formatting may not match that of the actual file that gets installed on your system.

```
@echo off
REM Start up  script for JDBC Adapter

REM Replace the variables in the following lines with the appropriate values:
REM <javaHome>    is the location of the installed Java SDK.
REM <webAppHome>  is the location of the Info*Engine installation directory.
REM <jdbcJarHome> is the location of the jdbcAdapter.jar file.
REM <oracleHome>  is the location of the Oracle Installation.

set JAVA_HOME=<javaHome>
set WEB_APP_HOME=<webAppHome>
set JDBC_HOME=<jdbcJarHome>
set ORACLE_HOME=<oracleHome>

REM This classpath includes the JDBC driver classes.
REM This is required for the Query-Objects, Query-Attributes and Delete-Objects
REM webjects.
REM If different on your system, change the path to the classes12.zip file in the
REM next line.
set CLASSPATH=%ORACLE_HOME%\jdbc\lib\classes12.zip

REM This classpath is required to run JDBC Adapter.
set CLASSPATH=%CLASSPATH%;%JDBC_HOME%\jdbcAdapter.jar

REM This classpath is required so that all necessary Info*Engine .jar files are
REM loaded.
set CLASSPATH=%CLASSPATH%;%WEB_APP_HOME%\codebase\web-inf\lib\wc3rdpartylibs.jar
set CLASSPATH=%CLASSPATH%;%WEB_APP_HOME%\codebase\web-inf\lib\ie.jar
set CLASSPATH=%CLASSPATH%;%WEB_APP_HOME%\codebase\web-inf\lib\ie3rdpartylibs.jar
set CLASSPATH=%CLASSPATH%;%WEB_APP_HOME%\lib\servlet.jar
set CLASSPATH=%CLASSPATH%;%JAVA_HOME%\lib\tools.jar

REM If LDAP adapter entries are not used, then comment out the lines that follow
REM the comments.
REM If LDAP adapter entries are used, then replace the following variables with the
REM appropriate values:

REM <propUrl> is the location of the ie.properties file, which is in the
REM      codebase/WEB-INF directory where Info*Engine is installed. This file
REM      contains a reference to the LDAP branch that contains the Info*Engine and
REM      adapter properties, and contains LDAP validation information. If you do
REM      not have access to this file, replace this variable with an LDAP URL that
REM      provides the same information that is in the file. For additional
REM      information about this file, see the Info*Engine Installation and
REM      Configuration Guide.
REM <domain> is the host domain where Info*Engine is installed.

set IEPROPFILE=<propUrl>
set IENAMINGSERVICENAME=<domain>.namingService

echo JDBC Adapter
REM The following line starts the JDBC adapter as a standalone process
%JAVA_HOME%\bin\java.exe -cp "%CLASSPATH%" -DpropFile="%IEPROPFILE%" -
DserviceName=jdbcAdapter -DnamingServiceName="%IENAMINGSERVICENAME%"
com.infoengine.jdbc.JDBCAdapter
pause
```

Sample UNIX Start File Contents

The following lines document the start file for the UNIX Operating System - note that this is only a sample and the formatting may not match that of the actual file that gets installed on your system.

```
#!/bin/csh
# Start up  script for JDBC Adapter

# Replace the variables in the following lines with the appropriate values:
# <javaHome>    is the location of the installed Java SDK.
# <webAppHome>  is the location of the Info*Engine installation directory.
# <jdbcJarHome> is the location of the jdbcAdapter.jar file.
# <oracleHome>  is the location of the Oracle Installation.

set JAVA_HOME=<javaHome>
set WEB_APP_HOME=<webAppHome>
set JDBC_HOME=<jdbcJarHome>
set ORACLE_HOME=<oracleHome>

# This classpath includes the JDBC driver classes.
# This is required for the Query-Objects, Query-Attributes and Delete-Objects
# webjects.
# If different on your system, change the path to the classes12.zip file in the
# next line.
set CLASSPATH={$ORACLE_HOME}/jdbc/lib/classes12.zip

# This classpath required to run the JDBC Adapter.
set CLASSPATH={$CLASSPATH}:{$JDBC_HOME}/jdbcAdapter.jar

# This classpath is required so that all necessary Info*Engine .jar files are
# loaded.
set CLASSPATH={$CLASSPATH}:{$WEB_APP_HOME}/codebase/WEB-INF/lib/wc3rdpartylibs.jar
set CLASSPATH={$CLASSPATH}:{$WEB_APP_HOME}/codebase/WEB-INF/lib/ie.jar
set CLASSPATH={$CLASSPATH}:{$WEB_APP_HOME}/codebase/WEB-INF/lib/ie3rdpartylibs.jar
set CLASSPATH={$CLASSPATH}:{$WEB_APP_HOME}/lib/servlet.jar
set CLASSPATH={$CLASSPATH}:{$JAVA_HOME}/lib/tools.jar

# If LDAP adapter entries are not used, then comment out the lines that follow the
#    comments.
#    If LDAP adapter entries are used, then replace the following variables with
#    the appropriate values.
# <propUrl> is the location of the ie.properties file, which is in the
#    codebase/WEB-INF directory where Info*Engine is installed.
#    This file contains a reference to the LDAP branch that contains
#    the Info*Engine and adapter properties, and contains LDAP validation
#    information.
#    If you do not have access to this file, replace this variable with an LDAP
#    URL that provides the same information that is in the file.
#    For additionalinformation about this file,
#    see the Info*Engine Installation and Configuration Guide.
# <domain> is the host domain where Info*Engine is installed.

set IEPROPFILE=<propUrl>
set IENAMINGSERVICENAME=<domain>.namingService

echo JDBC Adapter
echo Using Classpath...
echo $CLASSPATH

# The following line starts the JDBC adapter as a standalone process:
$JAVA_HOME/bin/java -classpath {$CLASSPATH} -DpropFile={"$IEPROPFILE"} -
DserviceName=jdbcAdapter -DnamingServiceName={$IENAMINGSERVICENAME}
com.infoengine.jdbc.JDBCAdapter
```

ie.properties Location and Contents

The start files use the ie.properties file that the Info*Engine installer generates as the value for the -DpropFile parameter on the Java start command.

The ie.properties file is located in the codebase/WEB-INF directory where Info*Engine is installed and contains a reference to the LDAP branch that contains the Info*Engine properties. This reference also contains validation information to ensure that the services can access the LDAP directory.

The installer generates the contents of the ie.properties file based on the values that were entered when Info*Engine was installed. For more information about this file, see the *Info*Engine Installation and Configuration Guide*.

Example Naming Service Launch Property

The Naming Service enables you to automatically start out-of-process adapters that are on the same host as the Naming Service when the Naming Service starts.

To launch a component when the Naming Service starts, a Naming Service Launch property must contain the Java startup command for the component or must name a script file containing the startup command. You can set Naming Service Launch properties through the Info*Engine Property Administrator by editing the **Launch** fields in the Naming Service LDAP entry.

Note: When you install the Naming Service, Naming Service Launch properties are defined for both the Info*Engine server and the E-Mail Broker (if you configure it) and they contain the file paths to the files that start the server and the E-Mail Broker.

To set up a JDBC Launch property for an out-of-process JDBC adapter, add an additional **Launch** field in the Naming Service LDAP entry. For example, assume that your start file is named startJDBC.bat and is located in the /bin/infoengine directory where Info*Engine is installed. Then the **Launch** property that you enter is similar to the following:

```
cmd.exe /C start "JDBC Adapter" /MIN
C:/ptc/Windchill/bin/infoengine/startJDBC.bat
```

Naming the Adapter in Webject INSTANCE Parameters

You define the adapter name to use in the INSTANCE parameter when you configure the adapter through the Info*Engine Property Administrator.

An adapter name can be one of the following forms:

- A simple name, which is defined in the **Service Name** field on the Info*Engine Property Administrator service form. Simple names are stored in the ptcServiceName attribute of the adapter LDAP entry. To use a simple name, the adapter LDAP entry must reside within the Naming Service search path. For example, assume that

"com.myCompany.myHost.jdbcAdapter" is the ptcServiceName attribute of the adapter LDAP entry in the Naming Service search path. Then, the following INSTANCE parameter can be used:

```
<ie:param name="INSTANCE"
data="com.myCompany.myHost.jdbcAdapter"/>
```

- A fully-qualified distinguished name. When configuring the adapter, you specify the distinguished name on the Info*Engine Property Administrator form. This name consists of the ptcServiceName attribute and the other attributes that define the location of the LDAP entry.

  For example if the "com.myCompany.myHost.jdbcAdapter" entry is located on "host1" at "dc=IeProps,dc=myHost,dc=myCompany, dc=com,ou=Applications,o=myCompany", then the distinguished name is used in the INSTANCE parameter in the following form:

  ```
  <ie:param name="INSTANCE" data="ldap://host1/
  ptcServiceName=com.myCompany.myHost.jdbcAdapter,
  dc=IeProps,dc=myHost,dc=myCompany,dc=com,
  ou=Applications,o=myCompany"/>
  ```

- A domain-based reference name. This name is just another way of identifying the distinguished name when the LDAP directory that has the Info*Engine entries is constructed using dc=com as a root-level entry and other dc attributes for subtree entries.

  The format of a Info*Engine domain-based reference name is:

  ptcServiceName@*dc_attributes*

  In this format, *ptcServiceName* is the value of the ptcServiceName attribute and *dc_attributes* are the dc attributes that make up the domain location of the LDAP entry, where each attribute is separated from the next attribute using a period.

  Note: The domain-based reference name can only be used when the LDAP directory that has the Info*Engine entries is constructed using dc=com as a root-level directory or when the Naming Service .serviceDomainBase property is set to include those attributes beyond the domain that are used in the distinguished name of the entry.

  For example, if the ptcServiceName attribute value is "com.myCompany.myHost.jdbcAdapter" and the entry is located in the "dc=myHost,dc=myCompany,dc=com,ou=Applications, o=myCompany" branch, then the following domain-based reference name could only be used in the INSTANCE parameter if the .serviceDomainBase property is set to "ou=Applications,o=myCompany":

  ```
  <ie:param name="INSTANCE" data="com.myCompany.myHost.
  jdbcAdapter@myHost.myCompany.com"/>
  ```

  Use the Info*Engine Property Administrator to set the .serviceDomainBase property. For more information, see the property help in the Property Administrator.

JDBC Adapter Properties

The following JDBC adapter properties can be set for a JDBC adapter using the Info*Engine Property Administrator JDBC adapter form. The properties are listed alphabetically by the JDBC adapter form label, followed by the property name in parenthesis:

Database Driver Class (drivers)

Specifies the name of the JDBC driver class. This is specific to the database that the adapter will connect to. For example, oracle.jdbc.driver.OracleDriver is the driver class for the Oracle Thin Driver. For more information consult your driver manual.

Database Supports Auto Commit (autoCommitSupported)

Specifies whether or not the underlying database supports the auto-commit facility, by taking a value **true** or **false** respectively. See the note provided at the end of the section for more information.

Database Type (databaseType)

Specifies the type of database that the adapter connects to. If it is Oracle, select "Oracle 8i" or "Oracle 9i" depending on what version of Oracle is in use. If it is other than Oracle, select "Non Oracle".

Database URL (url)

Specifies the JDBC URL used to establish the connection. For example, the URL for an Oracle Thin Driver could be in the following format:

jdbc:oracle:thin:@hostname:1521:databaseName

Database User (dbuser)

Specifies the default user used when making a connection to the database.

Maximum Cache Size (maxContextCacheSize)

Specifies the maximum number of connections to cache. The default is ten (10). Connection pooling is always active.

Maximum Context Age (maxContextAge)

Specifies the maximum time, in seconds, a connection will stay active if not used. The default is sixty (60) seconds.

Maximum Query Size (maxQuerySize)

Specifies the maximum number of rows returned from a database query. The default is 2000. Anything beyond the maximum is silently dropped by the JDBC drivers.

Maximum Thread Count (socketAccess.maxThreadCount)

Specifies the maximum number of concurrent threads used by the JDBC adapter. This value defaults to 5. Setting this value higher allows more concurrent connections, but also requires more resources (memory and CPU cycles).

Password (passwd)

Specifies the default password to use when making the connection to the database.

Result Set Scrolling Capability (resultSetScrollingCapability)

Specifies the scrolling capability for result sets that are generated by way of executing SQL queries. A value of "Default" provides for default behaviour while a value of "TYPE_FORWARD_ONLY" allows the cursor to move in the forward direction only, thereby rendering the result sets unscrollable. The default is "Default".

Set this property to "TYPE_FORWARD_ONLY" only when the underlying database provides for scrollable result sets by default and you wish to make them unscrollable for some reason. For most databases, such a setting is redundant since the default behaviour itself provides for result sets that are unscrollable.

Secret (secret.text)

Specifies the secret used to sign and validate requests.

Secret 2 (secret.text2)

Specifies a string used to sign and validate requests to a task processor or adapter. The secret.text2 property generates a more comprehensive request signature than the secret.text property.

Secret Algorithm (secret.algorithm)

Specifies the algorithm used to encrypt secrets. Valid values for this property are SHA-1 and MD5.

Use Database Auto Commit (autoCommit)

Specifies whether or not the auto-commit facility provided by the underlying database is to be used, by taking a value **true** or **false** respectively.

If autoCommitSupported is true, setting autoCommit to true/false would mean that the auto-commit facility would be used/otherwise respectively. If on the other hand, autoCommitSupported is false (meaning that the underlying database does not support the auto-commit facility), the value in autoCommit would be disregarded but the changes caused by the execution of SQL statements would be explicitly committed.

Note: With Oracle as the database type, if the user sets autoCommitSupported to an invalid value (say false), the property would be set to true internally by the software. However, for other database types, it is the user's responsibility to provide valid and consistent values for autoCommitSupported and autoCommit through the Property Administrator.

JDBC Adapter Logging Capabilities

The logging capabilities available through the JDBC adapter are dependent on whether you are running the adapter in process or out of process.

General information about JDBC adapter logging can be found in the following sections. For additional information about logging, see the Property Administrator help and the *Windchill Info*Engine Administration and Implementation Guide*.

In-process Adapter Logging Capabilities

When you are running the JDBC adapter in-process, whether a log file of a given type is created is determined by the logging options associated with the servlet or server. The servlet log files are used when the adapter webjects are in a JSP and the server log files are used when the adapter webjects are in a task. For example, if in the **Logging** section of the servlet LDAP entry form the **Debug** and **Information** fields are set to **true**, and all other logging fields are **false**, only the following files are created:

<*service_name*>_debug.log
<*service_name*>_info.log

where <*service_name*> is the name of the service corresponding to the servlet.

The default location for the log files is <*ie_dir*>/logs (where <*ie_dir*> is the directory where Info*Engine is installed); however, the location can be changed to any other existing location using the Property Administrator. With the above settings, the debug and information messages that are specific to the adapter when it is called from a JSP are written to the corresponding files along with debug and information messages for the servlet.

The logging options associated with the adapter have no effect on the logging behavior when the adapter is running in process. For example, if **Debug** is set to **true** on the Property LDAP entry form for the servlet (or the server) but is set to **false** on the Property LDAP entry form for the adapter, the debug messages of the adapter are still written to the debug log file.

Out-of-process Adapter Logging Capabilities

When running the JDBC adapter out of process, whether a log file of a given type is created is determined by the logging options associated with the adapter. These options can be specified on the Property LDAP form for the adapter. For example, if you set **Debug** and **Information** to **true**, and all other options to **false**, only the following files are created:

<*adapter_service_name*>_debug.log
<*adapter_service_name*>_info.log

where <*adapter_service_name*> is the runtime service name of the out-of-process adapter. The default location for these files is the directory from which you started the adapter, but can be changed to any other existing location using the Property Administrator. With the above settings, the debug and information messages that are specific to the adapter are written to the corresponding files.

3

The Webject Library

This chapter describes the webjects that are available through the JDBC adapter. Each webject description includes the syntax, parameters, and in most cases, an example.

For further examples of the use of JDBC adapter webjects, see the Custom Applications chapter of the *Info*Engine User's Guide*.

Topic **Page**

Webject Library Overview

The action and query webjects valid for the JDBC adapter are detailed in the following sections.

Because display and group webjects are available for all adapters, they are described in the *Info*Engine User's Guide*.

In tables at the beginning of the parameter descriptions, the parameters are categorized as being either Required, Select, or Optional:

- A parameter is listed in the Required column when it is always required.
- A parameter is listed in the Select column when there is a relationship between the specified parameter and another parameter. For example, the SORTED and SORTBY parameters of the Query-Attributes webject are Select because if you specify the SORTED parameter, you must also specify the SORTBY parameter.

  A parameter is also listed in the Select column when its specification depends on the context in which the webject is being executed. For example, in the Put-Blob-Stream webject, the FILENAME parameter must be specified if no file was uploaded through the browser.
- A parameter is listed in the Optional column when it is always optional and when it is not related to another parameter.

Processing BLOBs

It is possible to control how BLOBs are processed by the adapter webjects. The BLOB_COUNT parameter can be included on any JDBC adapter webject. This parameter controls how BLOBs are consumed by specifying how many BLOBs should be delivered to the adapter webject. You can specify a value of 0 when no BLOBs should be delivered to the webject. If you omit the BLOB_COUNT parameter, all remaining BLOBs are delivered to the webject.

Running the Webject Examples

Performing a Typical installation of the JDBC adapter on the same host where Info*Engine is installed causes certain example JSP and XML task files to be installed on your system. The JSP files get installed under <*ie_dir*>\codebase\infoengine\jsp\examples\JDBCAdapter\examples while the XML task files get installed under <*ie_dir*>\tasks\infoengine\examples\JDBCAdapter\example-tasks, where <*ie_dir*> is the Info*Engine installation directory. Performing a JDBC adapter installation on a host different from the Info*Engine host places the examples in the JDBC installation directory; you must manually copy the examples to an appropriate location under the Info*Engine installation directory as seen in the paths above.

To run the examples on your system, complete the following procedure:

1. Build the demo database by running the BuildDemoDb.jsp file. Use the following URL:

   http://*hostname*/Windchill/infoengine/jsp/examples/JDBCAdapter/examples/BuildDemoDb.jsp

   This file builds and populates the demo database which the examples run against.
2. Replace the values of the INSTANCE, DBUSER and PASSWD webject parameters in the JSP files with values appropriate to your installation.
3. Run the JSP files in the examples directory by using the following URL:

   http://*hostname*/Windchill/infoengine/jsp/examples/JDBCAdapter/examples/*JDBC.jsp*

   where *hostname* is the name of the system where Info*Engine is installed, and *JDBC.jsp* is the name of the JSP example file to be run.

Batch-Execute-Procedure

Description

Executes an SQL stored procedure for multiple sets of input parameter values.

Syntax

```
<ie:webject name="Batch-Execute-Procedure" type="ACT">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="FIELD" data="input_data"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbuser_passwd"/>
  <ie:param name="SQL" data="procedure_call"/>
</ie:webject>
```

Parameters

Required	Select	Optional
FIELD		BLOB_COUNT
INSTANCE		CONNECTION_ATTEMPTS
SQL		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

FIELD

Specifies type and value information for a given IN argument to the stored procedure. The value specified for this webject parameter will typically be of the form:

<name>.<type>=<value>

where <name>, <type> and <value> are the name, the SQL type and the value of the IN argument respectively. For ARRAY and STRUCT type arguments however, there will be an SQL type name in addition to the type itself (see the note provided at the end of the following page).

For a stored procedure that takes 'n' IN arguments, there will be 'n' such FIELD entries defining an input set and there would typically be two or more such input sets. Thus, if the stored procedure is to be executed for 'm' sets of input parameter values as a batch, there will be (m x n) FIELD entries in the webject.

This parameter is required.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

SQL

Specifies the signature of the call to the stored procedure. The value specified for this webject parameter may typically be of the form:

<procedure_name>(in1, in2, in3, in4)

where <procedure_name> is the name of the stored procedure and in1, in2, in3 and in4 are the IN arguments to the procedure.

This parameter is required.

Note:

1. The webject provides support for ARRAY and STRUCT type IN arguments only when the underlying database type is Oracle. Executing the webject for ARRAY or STRUCT type IN arguments for database types other than Oracle would result in an exception getting thrown.

2. For ARRAY and STRUCT type IN arguments, the value specified for the webject parameter FIELD will be of the form:

   <name>.<type>.<type_name>=<value>

   where <name> is the name of the IN argument, <type> is either ARRAY or STRUCT, <type_name> is the SQL type name and <value> is a comma separated set of either ARRAY elements or STRUCT attribute values.

3. The case associated with any of the specified SQL types is not important, but that associated with any of the SQL type names is (relevant for ARRAY and STRUCT types).

4. Values specified for VARCHAR type IN arguments are to be enclosed within single quotes.

5. Values specified for DATE type IN arguments must be of the form 'dd-<month_name>-yyyy' where <month_name> is either the first three characters of the given month's name or its full name. The value may be optionally enclosed within single quotes.

   You can also specify either of the following for the DATE type IN argument if you are connecting to an Oracle database:

   – sysdate, which uses the system date as defined on the computer in use

   – An empty string ('')

     An empty string passes the date as is to the stored procedure being executed. For example, when passing this to a procedure that inserts table rows, a NULL value is inserted into the DATE type column (assuming the table allows it).

6. Values specified for TIME type IN arguments must be of the form 'hh:mm:ss'.

7. A null value may be specified for a given IN argument by specifying NULL as its value.

8. The value specified for the webject parameter SQL must be the signature of a stored procedure that does not have OUT or INOUT arguments and that returns a simple update count. If any of these conditions are violated, an exception will be thrown upon executing the webject.

Example

The following example documents the BatchExecuteProcedure.jsp file, which is located in the *<ie_dir>*/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory. It illustrates the Batch-Execute-Procedure webject for standard SQL types.

```
<%@page language="java" session="false" errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core" prefix="ie"%>
<html>
<head><title>Batch-Execute-Procedure Webject</title>
<BASE>
</head>
<body bgcolor="#AABBCC">
<h1>Batch-Execute-Procedure webject: </h1>
<h3>Executes an SQL stored procedure for multiple input sets as a batch</h3>

  <ie:webject name="Batch-Execute-Procedure" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}" default="jdbcAdapter"/>
    <ie:param name="DBUSER"    data="dbuser_name"/>
    <ie:param name="PASSWD"    data="dbuser_passwd"/>


    <ie:param name="FIELD"     data="INP_EMPNO.INTEGER=8000"/>
    <ie:param name="FIELD"     data="INP_JOB.VARCHAR='SR. CLERK'"/>
    <ie:param name="FIELD"     data="INP_MGR.INTEGER=7566"/>
    <ie:param name="FIELD"     data="INP_HD.DATE='14-Jan-2002'"/>
    <ie:param name="FIELD"     data="INP_SAL.INTEGER=1000"/>
    <ie:param name="FIELD"     data="INP_DN.INTEGER=20"/>


    <ie:param name="FIELD"     data="INP_EMPNO.INTEGER=8001"/>
    <ie:param name="FIELD"     data="INP_JOB.VARCHAR='SR. CLERK'"/>
    <ie:param name="FIELD"     data="INP_MGR.INTEGER=7566"/>
    <ie:param name="FIELD"     data="INP_HD.DATE='24-Jun-2002'"/>
    <ie:param name="FIELD"     data="INP_SAL.INTEGER=950"/>
    <ie:param name="FIELD"     data="INP_DN.INTEGER=20"/>

    <ie:param name="FIELD"     data="INP_EMPNO.INTEGER=8002"/>
    <ie:param name="FIELD"     data="INP_JOB.VARCHAR='SR. CLERK'"/>
    <ie:param name="FIELD"     data="INP_MGR.INTEGER=7566"/>
    <ie:param name="FIELD"     data="INP_HD.DATE='30-Dec-2002'"/>
    <ie:param name="FIELD"     data="INP_SAL.INTEGER=900"/>
    <ie:param name="FIELD"     data="INP_DN.INTEGER=20"/>


    <ie:param name="SQL"       data="UPDATE_EMP_RECORD(INP_EMPNO, INP_JOB, INP_MGR,
INP_HD, INP_SAL, INP_DN)"/>
  </ie:webject>

</body>
</html>
```

Clearly, the above example would execute the SQL stored procedure called UPDATE_EMP_RECORD for 3 different sets of input parameter values. The 3 input sets have been shown as separate blocks in the above example for clarity. Each set consists of 6 FIELD entries, which provide type and value information for the 6 IN arguments to the stored procedure.

Besides standard types, the webject provides support for ARRAY and STRUCT type IN arguments. The following example documents the BatchExecuteProcArray.jsp file, which is located in the *<ie_dir>*/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory. It illustrates batch execution of a stored procedure that takes ARRAY type IN arguments.

```
<%@page language="java" session="false" errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core" prefix="ie"%>
<html>
<head><title> Batch-Execute-Procedure Webject </title>
<BASE>
</head>
<body bgcolor="#AABBCC">
<h1> Batch-Execute-Procedure webject: </h1>
<h3>Executes an SQL stored procedure for multiple input sets as a batch.</h3>
<h3>Illustrates batch execution for ARRAY type parameters.</h3>

  <ie:webject name="Batch-Execute-Procedure" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}" default="jdbcAdapter"/>
    <ie:param name="DBUSER"    data="dbuser_name"/>
    <ie:param name="PASSWD"    data="dbuser_passwd"/>


    <ie:param name="FIELD"     data="INP_NUM.INTEGER=1"/>
    <ie:param name="FIELD"     data="INP_ARRAY1.ARRAY.INT_ARRAY=(10, 20, 30)"/>
    <ie:param name="FIELD"     data="INP_ARRAY2.ARRAY.REAL_ARRAY=(12.3, 23.4,
34.5)"/>
    <ie:param name="FIELD"
data="INP_ARRAY3.ARRAY.STRING_ARRAY=('A','SAMPLE','STRING')"/>
    <ie:param name="FIELD"
data="INP_ARRAY4.ARRAY.ADDRESS_ARRAY=(ADDRESS('SCIOTO',101),ADDRESS('ST.
MARY',254))"/>

    <ie:param name="FIELD"     data="INP_NUM.INTEGER=2"/>
    <ie:param name="FIELD"     data="INP_ARRAY1.ARRAY.INT_ARRAY=(40, 50, 60)"/>
    <ie:param name="FIELD"     data="INP_ARRAY2.ARRAY.REAL_ARRAY=(43.4, 56.7,-
67.8)"/>
    <ie:param name="FIELD"
data="INP_ARRAY3.ARRAY.STRING_ARRAY=('ANOTHER','SAMPLE','STRING')"/>
    <ie:param name="FIELD"
data="INP_ARRAY4.ARRAY.ADDRESS_ARRAY=(ADDRESS('PATRICK',124),ADDRESS('ST.
JOHN',22))"/>


    <ie:param name="FIELD"     data="INP_NUM.INTEGER=3"/>
    <ie:param name="FIELD"     data="INP_ARRAY1.ARRAY.INT_ARRAY=(-70, 80, -90)"/>
    <ie:param name="FIELD"     data="INP_ARRAY2.ARRAY.REAL_ARRAY=(-72.3, 83.4,-
94.5)"/>
    <ie:param name="FIELD"
data="INP_ARRAY3.ARRAY.STRING_ARRAY=('YET','ANOTHER','SAMPLE','STRING')"/>
    <ie:param name="FIELD"
data="INP_ARRAY4.ARRAY.ADDRESS_ARRAY=(ADDRESS('PARKER',6),ADDRESS('BISHAN',11))"/>


    <ie:param name="SQL"       data="UPDATE_VARRAY_EXAMPLE2(INP_NUM, INP_ARRAY1,
INP_ARRAY2, INP_ARRAY3, INP_ARRAY4)"/>
  </ie:webject>

</body>
</html>
```

And the following example documents the BatchExecuteProcStruct.jsp file, which is located in the *<ie_dir>*/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory. It illustrates batch execution of a stored procedure that takes a STRUCT type IN argument.

```
<%@page language="java" session="false" errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core" prefix="ie"%>
<html>
<head><title>Batch-Execute-Procedure Webject</title>
<BASE>
</head>
<body bgcolor="#AABBCC">
<h1>Batch-Execute-Procedure webject: </h1>
<h3>Executes an SQL stored procedure for multiple input sets as a batch.</h3>

<h3>Illustrates batch execution for STRUCT type parameters.</h3>

  <ie:webject name="Batch-Execute-Procedure" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}" default="jdbcAdapter"/>
    <ie:param name="DBUSER"    data="dbuser_name"/>
    <ie:param name="PASSWD"    data="dbuser_passwd"/>
    <ie:param name="FIELD"     data="INP_EMPNO.INTEGER=101"/>
    <ie:param name="FIELD"     data="INP_EMPID.STRUCT.PERSON=('GREG',ADDRESS('VAN
NESS',345))"/>
    <ie:param name="FIELD"     data="INP_EMPNO.INTEGER=102"/>
    <ie:param name="FIELD"
data="INP_EMPID.STRUCT.PERSON=('PATTERSON',ADDRESS('GEARY',412))"/>
    <ie:param name="FIELD"     data="INP_EMPNO.INTEGER=103"/>
    <ie:param name="FIELD"
data="INP_EMPID.STRUCT.PERSON=('MATHEWS',ADDRESS('BURNTSTONES',169))"/>
    <ie:param name="SQL"       data="UPDATE_PEOPLE_RECORD(INP_EMPNO, INP_EMPID)"/>
  </ie:webject>

</body>
</html>
```

Create-Object

Description

Creates database objects by adding records to a table.

Syntax

```
<ie:webject name="Create-Object" type="ACT">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="class"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="FIELD" data="field"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbpassword"/>
</ie:webject>
```

Parameters

Required	Select	Optional
CLASS		BLOB_COUNT
FIELD		CONNECTION_ATTEMPTS
INSTANCE		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Specifies the name of the table in which you are creating records. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

FIELD

Specifies the attribute values to be added. The value for this parameter is specified in the following manner:

name='*value*'

where *name* is the object attribute name, and *value* is the value to store in the attribute. The *value* portion of the parameter value must be enclosed in single quotes. Multiple values can be specified for this parameter. This parameter is required.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

Example

The following example documents the CreateObject.jsp file, which is located in the *<ie_dir>*/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, a new employee record is created in the EMP table, including the employee number, name, job title, manager, date of hire, salary, and department. The database is then queried and the newly created record is displayed using the Query-Objects and Display-Table webjects respectively.

Note: To run this example on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

```
<%@page language="java" session="false"
                                        errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                  prefix="ie"%>

<html>
<head><title>Create-Object Webject</title>
<BASE>
</head>
<body>

<h1>Create-Object webject: </h1>
<h3>Creates database objects by adding records to a table</h3>

<ie:webject name="Create-Object" type="ACT">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                    default="jdbcAdapter"/>
  <ie:param name="DBUSER"    data="dbuser_name"/>
  <ie:param name="PASSWD"    data="dbuser_passwd"/>
  <ie:param name="CLASS"     data="EMP"/>
  <ie:param name="FIELD"     data="EMPNO='1110'"/>
  <ie:param name="FIELD"     data="ENAME='Herman'"/>
  <ie:param name="FIELD"     data="JOB='ENGINEER'"/>
  <ie:param name="FIELD"     data="MGR='7990'"/>
  <ie:param name="FIELD"     data="HIREDATE='23-MAY-1999'"/>
  <ie:param name="FIELD"     data="SAL='2200'"/>
  <ie:param name="FIELD"     data="COMM=''"/>
  <ie:param name="FIELD"     data="DEPTNO='40'"/>
  <ie:param name="GROUP_OUT" data="createObject"/>
</ie:webject>

<ie:webject name="Query-Objects" type="OBJ">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                    default="jdbcAdapter"/>
  <ie:param name="CLASS"     data="EMP"/>
  <ie:param name="WHERE"     data="()"/>
  <ie:param name="GROUP_OUT" data="emp"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP"/>

</body>
</html>
```

Delete-Objects

Description

Deletes database objects by deleting records from a table.

Syntax

```
<ie:webject name="Delete-Objects" type="ACT">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="class"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="WHERE" data="where_clause"/>
</ie:webject>
```

Parameters

Required	Select	Optional
CLASS		BLOB_COUNT
INSTANCE		CONNECTION_ATTEMPTS
WHERE		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Specifies the name of the table from which you wish to delete objects. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

WHERE

Specifies search criteria for the database objects to delete. The value for this parameter is specified as an SQL formatted where clause. This parameter is required.

Example

The following example documents the DeleteObjects.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the database record including the ENAME attribute value of Herman is deleted from the EMP table. The database is then queried and the updated EMP table is displayed using the Query-Objects and Display-Table webjects respectively.

Note: To run this example on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

```
<%@page language="java" session="false"
                                        errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                  prefix="ie"%>

<html>
<head><title>Delete-Objects Webject</title>
<BASE>
</head>
<body>

<h1>Delete-Objects webject: </h1>
<h3>Delete database objects by removing records from a
Table</h3>

<ie:webject name="Delete-Objects" type="ACT">
  <ie:param name="DBUSER"    data="dbuser_name"/>
  <ie:param name="PASSWD"    data="dbuser_passwd"/>
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                    default="jdbcAdapter"/>
  <ie:param name="CLASS"     data="${@FORM[]table[]}"
                                          default="EMP"/>
    <ie:param name="WHERE"     data="${@FORM[]where[]}"
                                 default="ENAME='Herman'"/>
  <ie:param name="GROUP_OUT" data="DeleteObject"/>
</ie:webject>

<ie:webject name="Query-Objects" type="OBJ">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                    default="jdbcAdapter"/>
  <ie:param name="DBUSER"    data="dbuser_name"/>
  <ie:param name="PASSWD"    data="dbuser_passwd"/>
  <ie:param name="CLASS"     data="${@FORM[]table[]}"
                                          default="EMP"/>
  <ie:param name="WHERE"     data="()"/>
  <ie:param name="GROUP_OUT" data="employees"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP"/>

</body>
</html>
```

Describe-Attributes

Description

Describes the attribute (or column) definition detail of the specified database table.

Syntax

```
<ie:webject name="Describe-Attributes" type="OBJ">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="class"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="GROUP_OUT" data="group"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbpassword"/>
</ie:webject>
```

Parameters

Required	Select	Optional
CLASS		BLOB_COUNT
INSTANCE		CONNECTION_ATTEMPTS
		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Specifies the name of the table whose attribute definition detail is being described. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple Instance parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

GROUP_OUT

Identifies the group returned by the webject. Use this name as the GROUP_IN parameter value of a display webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

Example

The following example documents the DescribeAttributes.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the column definition detail is requested for the EMP table. The definition detail is then displayed using the Display-Table webject.

Note: To run this example on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

```
<%@page language="java" session="false"
                                        errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                  prefix="ie"%>

<html>
<head><title>Describe-Attributes Webject</title>
<BASE>
</head>
<body>

<h1>Describe-Attributes webject: </h1>
<h3>Describe the attribute or column definition details</h3>

<ie:webject name="Describe-Attributes" type="OBJ">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                    default="jdbcAdapter"/>
  <ie:param name="DBUSER"    data="dbuser_name"/>
  <ie:param name="PASSWD"    data="dbuser_passwd"/>
  <ie:param name="CLASS"     data="${@FORM[]table[]}"
                                           default="EMP"/>
  <ie:param name="GROUP_OUT" data="DescribeAttribute"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP"/>

</body>
</html>
```

Do-SQL

Description

Executes one or more general SQL statements either individually or as a batch. Do-SQL is a general-purpose webject. It can be used both as a query and an action webject.

Syntax

```
<ie:webject name="Do-SQL" type="ACT">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="MAX_QUERY_SIZE" data="max_query_size"/>
  <ie:param name="MODE" data="mode_of_execution"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="SQL" data="an_SQL_statement"/>
</ie:webject>
```

Parameters

Required	Select	Optional
INSTANCE		BLOB_COUNT
SQL		CONNECTION_ATTEMPTS
		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		MAX_QUERY_SIZE
		MODE
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

MAX_QUERY_SIZE

Specifies maximum number of objects returned from a database query. The default for this parameter is 2000. If there are more than the maximum number of objects, the JDBC driver silently drops the extra objects. MAX_QUERY_SIZE is unused when MODE is set for batch execution. This parameter is optional.

MODE

Specifies the mode of execution and has relevance only when executing multiple SQL commands. To run in batch mode, set this parameter to "BATCH". If the parameter is not provided or if it is set to any value other than "BATCH", the SQL commands (provided as values for the SQL webject parameter) will get executed individually rather than as a batch. This parameter is optional.

Note: The case associated with the value provided for the webject parameter MODE is unimportant. Thus, the value can be "BATCH", "Batch", "batch" etc.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

SQL

Specifies the SQL statement to be executed. There can be multiple entries of this webject parameter. When MODE is set for batch execution, this parameter can only take an SQL update command (i.e., a command that returns a simple update count) as its value. If this condition is violated, an exception would be thrown upon executing the webject. However, there is no such restriction for the normal mode of execution. This parameter is required.

Example

The following example documents the DoSql.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory. It illustrates the Do-Sql webject for the normal mode of execution.

In this example, the record including the ENAME attribute of ADAMS is retrieved from the EMP table. The record is then displayed using the Display-Table webject.

```
<%@page language="java" session="false"
                                          errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                    prefix="ie"%>

<html>
<head><title>Do-SQL Webject</title>
<BASE>
</head>
<body>

<h1>Do-SQL webject: </h1>
<h3>Performs a general SQL statement</h3>
```

```
<ie:webject name="Do-SQL" type="ACT">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                        default="jdbcAdapter"/>
  <ie:param name="DBUSER"      data="dbuser_name"/>
  <ie:param name="PASSWD"      data="dbuser_passwd"/>
  <ie:param name="SQL"         data="SELECT * FROM EMP WHERE
                                        (ENAME='ADAMS')"/>
    <ie:param name="GROUP_OUT" data="DoSql"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP"/>

</body>
</html>
```

The following example documents the BatchUpdate.jsp file, which illustrates the Do-Sql webject for the batch mode of execution. Running this example inserts 3 additional rows to the table EMP by executing the SQL update commands (provided as values for the SQL webject parameter) as a batch.

```
<%@page language="java" session="false"
errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>

<html>
<head><title>Do-SQL Webject</title>
<BASE>
</head>
<body bgcolor="#AABBCC">

<h1>Do-SQL webject: </h1>
<h3>Executes two or more SQL update commands as a batch</h3>

<ie:webject name="Do-SQL" type="ACT">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                        default="jdbcAdapter"/>
  <ie:param name="DBUSER"      data="dbuser_name"/>
  <ie:param name="PASSWD"      data="dbuser_passwd"/>

     <ie:param name="SQL" data="INSERT INTO EMP VALUES
(8000,'John','CLERK',7902,'11-JAN-1999',800,NULL,20)"/>

         <ie:param name="SQL" data="INSERT INTO EMP VALUES
(8001,'Jim','CLERK',7902,'21-JUN-1999',850,NULL,20)"/>

         <ie:param name="SQL" data="INSERT INTO EMP VALUES
(8002,'Jack','CLERK',7902,'27-DEC-1999',900,NULL,20)"/>



    <ie:param name="MODE"      data="BATCH"/>

 <ie:param name="GROUP_OUT" data="BatchUpdate"/>
</ie:webject>

</body>
</html>
```

Note: To run the above examples on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

Execute-Procedure

Description

Executes a stored procedure or function and returns any OUT or return arguments. This is the only JDBC adapter webject that can return more than one group.

This webject requires a procedure or function that has been previously stored in a database which supports stored procedures. Refer to your database documentation for a complete description of how to use stored procedures and functions with your database.

Note: The webject provides support for multiple IN arguments, multiple OUT arguments, as well as multiple IN OUT arguments. Besides, it provides support for STRUCT and ARRAY type arguments.

Note: The webject is recommended for use with ARRAY and STRUCT type arguments only when the underlying database type is Oracle.

Syntax

```
<ie:webject name="Execute-Procedure" type="ACT">
  <ie:param name="ATTRIBUTE" data="name.datatype"/>
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="FIELD" data="data"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="RETURN" data="datatype"/>
  <ie:param name="SQL" data="procedure_call"/>
</ie:webject>
```

Parameters

Required	Select	Optional
INSTANCE	ATTRIBUTE	BLOB_COUNT
SQL	FIELD	CONNECTION_ATTEMPTS
	RETURN	CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		PASSWD

ATTRIBUTE

Specifies an OUT argument to the stored procedure. The value for this parameter will typically be of the form:

name.datatype

where *name* and *datatype* are the name and SQL type of the OUT argument respectively. The name is used in the value specified for the SQL parameter. The following is a list of possible data types:

ARRAY	DOUBLE	REAL
BIGINT	FLOAT	SMALLINT
BINARY	INTEGER	STRUCT
BIT	LONGVARBINARY	TIME
CHAR	LONGVARCHAR	TIMESTAMP
CURSOR	NULL	TINYINT
DATE	NUMERIC	VARBINARY
DECIMAL	OTHER	VARCHAR

If there are multiple OUT arguments to the stored procedure, each OUT argument results in an output group that contains the value of the argument obtained by executing the procedure. The name of the resulting group is the same as the value specified for the ATTRIBUTE parameter. For example, consider the following ATTRIBUTE parameter specifications:

```
<ie:param name="ATTRIBUTE" data="sal.NUMBER"/>
<ie:param name="ATTRIBUTE" data="deptno.NUMBER"/>
```

Executing the procedure results in two output groups named sal.NUMBER and deptno.NUMBER being created. These group names can then be specified as GROUP_IN parameter values in a display webject.

To pass an IN OUT argument to the stored function or procedure using this webject, the argument name must be specified in both the ATTRIBUTE and FIELD parameters.

The ATTRIBUTE parameter can be multi-valued. The number of values specified will depend on the number of IN OUT and OUT arguments passed to the stored procedure.

Note: For ARRAY and STRUCT type arguments, the value specified for this parameter will be of the form:

<*name*>.<*datatype*>.<*type_name*>

where <*name*> is the name of the argument, <*datatype*> is either ARRAY or STRUCT and <*type_name*> is the SQL type name associated with the ARRAY or STRUCT type argument.

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For

example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

FIELD

Specifies an IN argument to the stored procedure. The value for this parameter is typically of the form:

name.type=value

where *name*, *type,* and *value* are the name, SQL type and value of the IN argument respectively.

To pass an IN OUT argument to the stored function or procedure using this webject, the argument name must be specified in both the ATTRIBUTE and FIELD parameters.

The FIELD parameter can be multi-valued. The number of values specified will depend on the number of IN and IN OUT arguments passed to the procedure.

Note: For ARRAY and STRUCT type IN arguments, the value specified for this parameter will be of the form:

<*name*>.<*type*>.<*type_name*>=<*value*>

where <*name*> is the name of the IN argument, <*type*> is either ARRAY or STRUCT, <*type_name*> is the SQL type name associated with the ARRAY or

STRUCT type argument and <*value*> is a comma separated set of either ARRAY elements or STRUCT attribute values.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

RETURN

Specifies the data type of the value returned by the stored function. The following is a list of possible data types:

ARRAY	DOUBLE	REAL
BIGINT	FLOAT	SMALLINT
BINARY	INTEGER	TIME
BIT	LONGVARBINARY	TIMESTAMP
CHAR	LONGVARCHAR	TINYINT
CURSOR	NULL	VARBINARY
DATE	NUMERIC	VARCHAR
DECIMAL	OTHER	

This parameter is required only if the webject is executing a stored function. If this parameter is specified when the webject is executing a stored procedure, then an error is returned.

SQL

The SQL parameter describes the signature of the call. The names defined in the FIELD and ATTRIBUTE parameters are used here. The value for this parameter is typically of the form:

proc(*in*, *in_out*, *out*)

where *proc* is the name of the stored procedure or stored function and *in*, *in_out* and *out* are the IN, IN OUT and OUT arguments respectively of the stored procedure.

If there are no IN OUT arguments to the stored procedure, the number of FIELD and ATTRIBUTE parameter values should equal the number of IN and OUT arguments (respectively) specified in the SQL parameter. For example, consider the following webject parameter specifications:

```
<ie:param name=FIELD data="in1.varchar='first in'"/>
<ie:param name=FIELD data="in2.varchar='second in'"/>
<ie:param name=ATTRIBUTE data="out1.varchar"/>
<ie:param name=SQL="proc(in1, in2, out1)"/>
```

The FIELD and ATTRIBUTE parameters describe the data, and the SQL parameter describes how the information is passed to the procedure or function. By keeping the information and the signature separate, the webject accepts any combination of arguments.

Note: If the stored procedure takes IN OUT arguments, there will be a FIELD and an ATTRIBUTE parameter for each IN OUT argument in addition to those specified for the IN and OUT arguments as described previously.

This parameter is required.

Examples

The stored functions and procedures executed in the following examples are created when the demo database is built. For more information see the section titled Running the Webject Examples earlier in this chapter.

Executing a Stored Procedure

The following example documents the ExecuteProcedure.jsp file, which is located in the *<ie_dir>*/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the stored procedure TESTPROC is executed. The results of both the OUT arguments of the stored procedure are then displayed using Display-Table webjects.

Note: To run this example on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

```
<%@page language="java" session="false"
                                        errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                  prefix="ie"%>

<html>
<head><title>Execute Procedure Webject</title>
<BASE>
</head>
<body>

<h1>Execute-Procedure webject: </h1>
<h3>Execute a SQL stored procedure</h3>

<ie:webject name="Execute-Procedure" type="ACT">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                    default="jdbcAdapter"/>
  <ie:param name="DBUSER"    data="dbuser_name"/>
  <ie:param name="PASSWD"    data="dbuser_passwd"/>
  <ie:param name="FIELD"     data="IN_ARG.varchar='in_param'"/>
  <ie:param name="FIELD"
data="INOUT_ARG.varchar='inout_param'"/>
  <ie:param name="ATTRIBUTE" data="INOUT_ARG.varchar"/>
  <ie:param name="ATTRIBUTE" data="OUT_ARG.varchar"/>
  <ie:param name="SQL"
                    data="TESTPROC(IN_ARG,OUT_ARG,INOUT_ARG)"/>
</ie:webject>

</hr>

<ie:webject name="Display-Table" type="DSP">
  <ie:param name="GROUP_IN"   data="INOUT_ARG.varchar"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP">
  <ie:param name="GROUP_IN"   data="OUT_ARG.varchar"/>
</ie:webject>

</body>
</html>
```

Executing a Stored Procedure that takes a STRUCT type IN argument

The following example documents the Executeinsert_emprec.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the stored procedure insert_emp_rec is executed. This procedure takes two IN arguments, and the values specified for these (using the FIELD parameters) are used to insert a row into the EMP_LIST table. Note that this table has a STRUCT type column with the SQL type name EMP_REC.

Note: To run this example on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

```
%@page language="java" session="false" errorPage="IEError.jsp"%
%@taglib uri="http://www.ptc.com/infoengine/taglib/core" prefix="ie"%


<html>
<head><title>Execute Procedure Webject</title>
<BASE>
</head>
<body>
<h1>Execute-Procedure webject: </h1>
<h3>Execute a SQL stored function</h3>
  <ie:webject name="Execute-Procedure" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}" default="jdbcAdapter"/>
    <ie:param name="DBUSER"    data="db_user"/>
    <ie:param name="PASSWD"    data="db_passwd"/>
    <ie:param name="FIELD"     data="in1.INTEGER=667"/>
    <ie:param name="FIELD"     data="in2.STRUCT.EMP_REC=(George M
Willis,ARRAY.STRINGARRAY=('196 Florence','Arden Hills'),150000.50,23-Aug-1999)"/>
    <ie:param name="SQL"       data="insert_emp_rec(in1,in2)"/>
  </ie:webject>
</body>
</html>
```

Executing a Stored Procedure that takes a STRUCT type OUT argument

The following example documents the Executeget_emprec.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the stored procedure get_emp_rec is executed. This procedure takes an INTEGER type IN argument and a STRUCT type OUT argument. Executing the procedure results in the retrieval of the row that was inserted into the EMP_LIST table by the previous example. The STRUCT type value that is retrieved is stored in an output group, which is then displayed using a Display-Table webject.

```
%@page language="java" session="false" errorPage="IEError.jsp"%
%@taglib uri="http://www.ptc.com/infoengine/taglib/core" prefix="ie"%


<html>
<head><title>Execute Procedure Webject</title>
<BASE>
</head>
<body>
```

```
<h1>Execute-Procedure webject: </h1>
<h3>Execute a SQL stored function</h3>
  <ie:webject name="Execute-Procedure" type="ACT">
    <ie:param name="INSTANCE"   data="${@FORM[]instance[]}" default="jdbcAdapter"/>
    <ie:param name="DBUSER"     data="db_user"/>
    <ie:param name="PASSWD"     data="db_passwd"/>
    <ie:param name="FIELD"      data="in1.INTEGER=667" />
    <ie:param name="ATTRIBUTE"  data="in2.STRUCT.EMP_REC" />
    <ie:param name="SQL"        data="get_emp_rec(in1,in2)"/>
  </ie:webject>

  <ie:webject name="Display-Table" type="DSP">
  <ie:param name="GROUP_IN"       data="in2.STRUCT.EMP_REC"/>
  </ie:webject>

</body>
</html>
```

Executing a Stored Function with Cursor Data Type

The following example documents the ExecuteReturnCursor.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the stored function named return_allemployees is executed. The result of the OUT argument of the stored function has a data type of CURSOR, and is then displayed using the Display-Table webject.

Note: To run this example on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

```
<%@page language="java" session="false"
                                          errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                  prefix="ie"%>

<html>
<head><title>Execute Procedure Webject</title>
<BASE>
</head>
<body>

<h1>Execute-Procedure webject: </h1>
<h3>Execute a SQL stored function</h3>

<ie:webject name="Execute-Procedure" type="ACT">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                     default="jdbcAdapter"/>
  <ie:param name="DBUSER"    data="dbuser_name"/>
  <ie:param name="PASSWD"    data="dbuser_passwd"/>
  <ie:param name="RETURN"    data="CURSOR"/>
  <ie:param name="SQL"       data="return_allemployees"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP"/>

</body>
</html>
```

Executing a Stored Function with Array Data Type

The following example documents the ExecuteReturnVarray.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the stored function named return_varray is executed. The OUT argument of the stored function has a data type of NUMARRAY, and is then displayed using the Display-Table webject. The NUMARRAY data type was created in the database when the BuildDemo.Db.jsp file was run.

Note: To run this example on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

```
<%@page language="java" session="false"
                                        errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                    prefix="ie"%>

<html>
<head><title>Execute Procedure Webject</title>
<BASE>
</head>
<body>

<h1>Execute-Procedure webject: </h1>
<h3>Execute a SQL stored function</h3>

<ie:webject name="Execute-Procedure" type="ACT">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                  default="jdbcAdapter"/>
  <ie:param name="DBUSER"    data="dbuser_name"/>
  <ie:param name="PASSWD"    data="dbuser_passwd"/>
  <ie:param name="RETURN"    data="ARRAY.NUMARRAY"/>
  <ie:param name="FIELD"     data="i.varchar='decimal'"/>
  <ie:param name="SQL"       data="return_varray(i)"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP"/>

</body>
</html>
```

Prepared-Batch-Update

Description

Executes a parameterized SQL update command for multiple sets of input parameter values.

Syntax

```
<ie:webject name="Prepared-Batch-Update" type="ACT">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="DELIMITER" data="delimiter_used"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbuser_passwd"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="PARAMTYPES"
data="delimiter_separated_parameter_types_in_order"/>
  <ie:param name="PARAMVALUES"
data="delimiter_separated_parameter_values_in_order"/>
  <ie:param name="SQL" data="a_parameterized_SQL_statement"/>
</ie:webject>
```

Parameters

Required	Select	Optional
INSTANCE	DELIMITER	BLOB_COUNT
PARAMTYPES		CONNECTION_ATTEMPTS
PARAMVALUES		CONNECTION_ATTEMPT_INTERVAL
SQL		DBUSER
		GROUP_OUT
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

DELIMITER

Specifies the delimiter that is used to separate any two consecutive entries in the string specified as value for the webject parameter PARAMTYPES or PARAMVALUES. This webject parameter is a Select parameter considering that it is required only when the SQL statement being executed takes two or more SQL parameters.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

PARAMTYPES

Specifies the types of SQL parameters in the order they occur in the command. The following is a list of all possible types that may be specified:

ARRAY, BIGINT, BIT, DATE, DOUBLE, FLOAT, REAL, INTEGER, NULL, NUMERIC, DECIMAL, SMALLINT, STRUCT, TIME, TINYINT, VARCHAR

The value for the webject parameter PARAMTYPES is specified as a string consisting of any of the above said types, with any two consecutively occurring types separated by a delimiter.

PARAMVALUES

Specifies the values of SQL parameters in the order they occur in the command. The value for the webject parameter PARAMVALUES is specified as a string consisting of values for the SQL parameters, with any two consecutively occurring values separated by a delimiter.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

SQL

Specifies the parameterized SQL update command. There can be only one entry of this webject parameter. This is in sharp contrast to the Do-SQL webject, which can have multiple entries of the SQL webject parameter.

Note:

1. The webject provides support for ARRAY and STRUCT type SQL parameters only when the underlying database type is Oracle. Executing the webject for ARRAY or STRUCT type SQL parameters for database types other than Oracle would result in an exception getting thrown.
2. The case associated with any of the standard SQL types specified as value for the webject parameter PARAMTYPES is not important. However, when dealing with ARRAY or STRUCT type parameters, the SQL type name must be specified in exactly the same way as it was created.
3. Values specified for VARCHAR, DATE and TIME type SQL parameters follow the same rules as those described for the Batch-Execute-Procedure webject.
4. A null value may be specified for a given parameter (provided the associated column allows it) by specifying NULL for the value.
5. The value provided for the SQL webject parameter must be a parameterized SQL command that returns a simple update count. If this condition is violated, an exception would be thrown upon executing the webject.

Example

The following example documents the PreparedBatchUpdate.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory. It illustrates the Prepared-Batch-Update webject for standard SQL types.

```
<%@page language="java" session="false"
errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>
<html>
<head><title>Prepared-Batch-Update Webject</title>
<BASE>
</head>
<body bgcolor="#AABBCC">
<h1>Prepared-Batch-Update webject: </h1>
<h3>Executes a parameterized SQL update command for multiple
input sets as a batch</h3>

  <ie:webject name="Prepared-Batch-Update" type="ACT">
    <ie:param name="INSTANCE"    data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="DBUSER"      data="dbuser_name"/>
    <ie:param name="PASSWD"      data="dbuser_passwd"/>
    <ie:param name="SQL"         data="INSERT INTO EMP VALUES
(?, ?, ?, ?, ?, ?, ?, ?)"/>
    <ie:param name="DELIMITER"   data="^"/>
    <ie:param name="PARAMTYPES"
data="INTEGER^VARCHAR^VARCHAR^INTEGER^DATE^INTEGER^INTEGER^INTE
GER"/>
    <ie:param name="PARAMVALUES"
data="8000^'John'^'CLERK'^7902^'11-Jan-1999'^900^NULL^20"/>
    <ie:param name="PARAMVALUES"
data="8001^'Jim'^'CLERK'^7902^'21-Jun-1999'^850^NULL^20"/>
    <ie:param name="PARAMVALUES"
data="8002^'Jack'^'CLERK'^7902^'27-Dec-1999'^800^NULL^20"/>
    <ie:param name="GROUP_OUT"   data="PreparedBatchUpdate"/>
  </ie:webject>

</body>
</html>
```

The webject provides support for ARRAY and STRUCT type SQL parameters in addition to the standard types. The following example documents the PreparedBatchUpdArray.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory. It illustrates batch execution for ARRAY type SQL parameters.

```
<%@page language="java" session="false"
errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>
<html>
<head><title>Prepared-Batch-Update Webject</title>
<BASE>
</head>
<body bgcolor="#AABBCC">
<h1>Prepared-Batch-Update webject: </h1>
<h3>Executes a parameterized SQL update command for multiple
input sets as a batch.</h3>
<h3>Illustrates batch execution for ARRAY type parameters.</h3>

  <ie:webject name="Prepared-Batch-Update" type="ACT">
    <ie:param name="INSTANCE"    data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="DBUSER"      data="dbuser_name"/>
    <ie:param name="PASSWD"      data="dbuser_passwd"/>
    <ie:param name="SQL"         data="INSERT INTO
VARRAY_EXAMPLE2 VALUES (?, ?, ?, ?, ?)"/>
    <ie:param name="DELIMITER"   data="^"/>
```

```
    <ie:param name="PARAMTYPES"
data="INTEGER^ARRAY:INT_ARRAY^ARRAY:REAL_ARRAY^ARRAY:STRING_ARR
AY^ARRAY:ADDRESS_ARRAY"/>
    <ie:param name="PARAMVALUES" data="1^INT_ARRAY(1, 2,
3)^REAL_ARRAY(1.23, 2.34,
3.45)^STRING_ARRAY('a','sample','string')^ADDRESS_ARRAY(ADDRESS
('Scioto',101),ADDRESS('St. Mary',254))"/>
    <ie:param name="PARAMVALUES" data="2^INT_ARRAY(4, 5,
6)^REAL_ARRAY(4.34, 5.67, -
6.78)^STRING_ARRAY('another','sample','string')^ADDRESS_ARRAY(A
DDRESS('Patrick',124),ADDRESS('St. John',22))"/>
    <ie:param name="PARAMVALUES" data="3^INT_ARRAY(-7, 8, -
9)^REAL_ARRAY(-7.23, 8.34, -
9.45)^STRING_ARRAY('yet','another','sample','string')^ADDRESS_A
RRAY(ADDRESS('Parker',6),ADDRESS('Bishan',11))"/>
    <ie:param name="GROUP_OUT"   data="PreparedBatchUpdArray"/>
  </ie:webject>

</body>
</html>
```

And the following example documents the PreparedBatchUpdStruct.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory. It illustrates batch execution for STRUCT type SQL parameters.

```
<%@page language="java" session="false"
errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>
<html>
<head><title>Prepared-Batch-Update Webject</title>
<BASE>
</head>
<body bgcolor="#AABBCC">
<h1>Prepared-Batch-Update webject: </h1>
<h3>Executes a parameterized SQL update command for multiple
input sets as a batch.</h3>
<h3>Illustrates batch execution for STRUCT type
parameters.</h3>

  <ie:webject name="Prepared-Batch-Update" type="ACT">
    <ie:param name="INSTANCE"    data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="DBUSER"      data="dbuser_name"/>
    <ie:param name="PASSWD"      data="dbuser_passwd"/>
    <ie:param name="SQL"         data="INSERT INTO PEOPLE
VALUES (?, ?)"/>
    <ie:param name="DELIMITER"   data="^"/>
    <ie:param name="PARAMTYPES"  data="INTEGER^STRUCT:PERSON"/>
    <ie:param name="PARAMVALUES" data="101^PERSON('Greg',
ADDRESS('Van Ness', 345))"/>
    <ie:param name="PARAMVALUES" data="102^PERSON('Patterson',
ADDRESS('Geary', 412))"/>
    <ie:param name="PARAMVALUES" data="103^PERSON('Mathews',
ADDRESS('Burntstones', 169))"/>
    <ie:param name="GROUP_OUT"
data="PreparedBatchUpdStruct"/>
  </ie:webject>

</body>
</html>
```

Put-Blob-Stream

Description

Stores BLOB data in an object attribute, or table column.

Definition

BLOB (noun): An acronym for Binary Large OBject. Any random large block of bits that needs to be stored in a database, such as a picture or sound file. A BLOB is an object stored in a database that cannot be interpreted within the database itself.

Syntax

```
<ie:webject name="Put-Blob-Stream" type="ACT">
  <ie:param name="ATTRIBUTE" data="attribute_with_blob_data"/>
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="tablename"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="FILENAME" data="local_file_name"/>
  <ie:param name="GROUP_OUT" data="group"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="WHERE" data="where_clause"/>
</ie:webject>
```

Parameters

Required	Select	Optional
ATTRIBUTE	FILENAME	BLOB_COUNT
CLASS		CONNECTION_ATTEMPTS
INSTANCE		CONNECTION_ATTEMPT_INTERVAL
WHERE		DBUSER
		GROUP_OUT
		PASSWD

ATTRIBUTE

Specifies the attribute or column in the database table in which to store the binary data. This parameter is required.

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Identifies the table name in which to store the binary data. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

FILENAME

Specifies the fully qualified path name of the file with binary data which is to be stored. For additional information on uploading files, see the *Info*Engine User's* Guide. If a file is not uploaded through the browser, then this parameter must be specified.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

WHERE

Specifies search criteria for the database object in which to store the binary data. The value for this parameter is specified as an SQL formatted where clause. This parameter is required.

Example

The following example documents the PutBlobStream.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the UploadBlob.jsp file, located in the same examples directory, provides the input form necessary for providing the appropriate parameter values for this example and should be run to execute PutBlobStream.jsp.

The **JDBC Adapter Instance** field of the input form specifies the INSTANCE parameter value appropriate to your installation. The **Name** field of the input form specifies the name of the location where the file containing BLOB data is to be put. The **File** field of the input form specifies the full path name of the file which contains the BLOB data. The **MIME Type** field enables the user to input a value for the MIME type – if the value input by the user for this field is uploaded, it can be fetched from the database for the given name while downloading the blob content. When the form is submitted, PutBlobStream.jsp is called. The data specified on the form provides the parameter values for the webjects.

In the PutBlobStream.jsp file that is shown below, note that values are inserted into the table BLOBTEST for the columns FILENAME and MIMETYPE. These would typically be the values input by the user for the fields **File** and **MIME Type** while executing UploadBlob.jsp.

The modified file is as follows:

```
<%@page language="java" session="false"
errorPage="IEError.jsp"%>

<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>

<html>
<head><title>Put-Blob-Stream Webject</title>
<BASE>
</head>
<body>
<h1>Put-Blob-Stream Webject</h1>
<h3>Stores BLOB data in an object attribute, or table
column</h3>

<ie:unit>
<ie:webject name="Do-Sql" type="ACT">
   <ie:param name="INSTANCE"  data="${FORM[]instance[]}"
default="jdbcAdapter" />
   <ie:param name="CLASS"     data="BLOBTEST"/>
   <ie:param name="SQL"       data="INSERT INTO BLOBTEST
VALUES('${FORM[]filename[]}',EMPTY_BLOB(),'${FORM[]file[]}',${F
ORM[]mimetype[]})"/>
   <ie:param name="GROUP_OUT" data="temp" />
</ie:webject>
<ie:failure>
<!-- No failure processing -->
</ie:failure>
</ie:unit>

<ie:webject name="Put-Blob-Stream" type="ACT">
   <ie:param name="INSTANCE"  data="${FORM[]instance[]}"
default="jdbcAdapter"/>
   <ie:param name="ATTRIBUTE" data="FILECONTENT"/>
   <ie:param name="CLASS"     data="BLOBTEST"/>
   <ie:param name="WHERE"
data="NAME='${FORM[]filename[]}'"/>
   <ie:param name="FILENAME"  data="${FORM[]file[]}"/>
   <ie:param name="GROUP_OUT" data="PutBlob"/>
</ie:webject>

<%
  String file = request.getParameter ("file");
%>

<b><i><%=file %></i> Uploaded to Database</b>
</body>
</html>
```

Put-Bulk-Stream

Description

Saves an uploaded file to the file system local to the adapter. For additional information on uploading files, see the *Info*Engine User's* Guide.

Note: Put-Bulk-Stream cannot be run as a standalone task. It must be run through a JSP page or an HTML template.

Syntax

```
<ie:webject name="Put-Bulk-Stream" type="ACT">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="FILENAME" data="filename"/>
  <ie:param name="GROUP_OUT" data="group_name/>
  <ie:param name="INSTANCE" data="instance_name"/>
  <ie:param name="PASSWD" data="dbpassword"/>
</ie:webject>
```

Parameters

Required	Select	Optional
FILENAME		BLOB_COUNT
INSTANCE		CONNECTION_ATTEMPTS
		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

FILENAME

Specifies the fully qualified path name of the file to be saved. This parameter is required.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

Example

The following example documents the PutBulkStream.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

This example uses the JSP useBean directive in order to display the image. To understand this example you must be familiar with the JSP useBean directive and its use. For further information about the API methods called within the useBean directive, see the API documentation that is in the following location:

<*ie_dir*>/codebase/infoengine/docs/apidocs

where <*ie_dir*> is the location where Info*Engine is installed.

In this example, the UploadBulk.jsp file, located in the same examples directory, provides the input form necessary for providing the appropriate parameter values for this example and should be run to execute PutBulkStream.jsp.

The **JDBC Adapter Instance** field of the input form should specify the INSTANCE parameter value appropriate to your installation. The **Name** field of the input form specifies the name of the file where the bulk data is to be put. When the form is submitted, PutBulkStream.jsp is called. The data specified on the form provides the parameter values for the webjects.

```
<%@page language="java" session="false"
                                        errorPage="IEError.jsp"%>
<%@ taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                    prefix="ie" %>
<html>
<head>
<title> BULK Upload</title>
</head>
<BODY>
<form method="POST" action="PutBulkStream.jsp"
                                   enctype="multipart/form-data">
<TABLE>
 <tr>
  <td align=right>
     <B>JDBC Adapter Instance: </B>
  </td>
  <td>
     <INPUT name="instance" type="text" size=50>
  </td>
 </tr>
 <tr>
  <td align=right>
     <B>Name: </B>
  </td>
```

```
 <td>
    <INPUT name="filename" type="text" size=50>
 </td>
</tr>
<tr>
 <td align=right>
    <B>File: </B>
 </td>
 <td>
    <INPUT name="file" type="file" size=50>
 </td>
</tr>
<tr>
 <td align=left>
    <INPUT name="submit" type="submit"  value="Submit"
                                                    id=button>
 </td>
</tr>
</table>
</form>
</body>
</html>
```

In the following PutBulkStream.jsp, the Put-Bulk-Stream webject puts the bulk data in the file specified in the **Name** field of the input form.

```
<%@page language="java" session="true" errorPage="IEError.jsp"
import="com.infoengine.object.factory.*,com.infoengine.object.*"%>

<html>
<head><title>Put-Bulk-Stream Webject</title>
<BASE>
</head>
<body>
<h1>Put-Bulk-Stream Webject:</h1>
<h3> Save and Upload a file to Adapter's local file system</h3>

<% String message_to_display = "";
   boolean upload_failed = false; %>

<jsp:useBean id="ie" class="com.infoengine.jsp.InfoEngine" scope="session">
          <%
          ie.setServletRequest (request);
          ie.setEnableExceptions (true);
          %>
        </jsp:useBean>
       <%
       IeMultipartInputStream is = new IeMultipartInputStream (request);
        String filename = is.getParameter ("filename");
        String instance = is.getParameter ("instance");
       %>
        <jsp:useBean id="put" class="com.infoengine.SAK.ObjectWebject">
          <%
          put.setService (ie);
          put.setName ("Put-Bulk-Stream");
          put.addParam ("INSTANCE", instance);
          put.addParam ("FILENAME", filename);
          put.addParam ("GROUP_OUT", "GroupOut");
          put.setInputStream (is);

          try
            {
             put.invoke ();
             message_to_display = "Uploaded bulk data to " + filename;
```

```
            }
          catch (Exception e)
            {
             message_to_display = e.getMessage ();
             upload_failed = true;
            }
          %>
        </jsp:useBean>


<b> <%= message_to_display%> </b>

<% if (upload_failed == true)
    { %>
     <h3>Upload failed.</h3>
 <% } %>

</body>
</html>
```

Put-Clob-Stream

Description

Stores character data in an object attribute or table column.

Definition

CLOB (noun): An acronym for Character Large OBject. A CLOB column stores single-byte fixed-width character objects, such as text documents. A CLOB is an object stored in a database that cannot be interpreted within the database itself.

Syntax

```
<ie:webject name="Put-Clob-Stream" type="ACT">
  <ie:param name="ATTRIBUTE" data="attribute_with_clob_data"/>
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="tablename"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="FILENAME" data="local_file_name"/>
  <ie:param name="GROUP_OUT" data="group"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="WHERE" data="where_clause"/>
</ie:webject>
```

Parameters

Required	Select	Optional
ATTRIBUTE	FILENAME	BLOB_COUNT
CLASS		CONNECTION_ATTEMPTS
INSTANCE		CONNECTION_ATTEMPT_INTERVAL
WHERE		DBUSER
		GROUP_OUT
		PASSWD

ATTRIBUTE

Specifies the attribute or column in the database table in which to store the character data. This parameter is required.

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Identifies the table name in which to store the character data. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

FILENAME

Specifies a fully qualified path name of the file in which the character data is to be stored. For additional information on uploading files, see the *Info*Engine User's* Guide. If a file is not uploaded through the browser, then this parameter must be specified.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

WHERE

Specifies search criteria for the database table row in which to store the character data. The value for this parameter is specified as an SQL formatted where clause. This parameter is required.

Example

The following example documents the PutClobStream.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the UploadClob.jsp file, located in the same examples directory, provides the input form necessary for providing the appropriate parameter values for this example and should be run to execute PutClobStream.jsp.

The **JDBC Adapter Instance** field of the input form should specify the INSTANCE parameter value appropriate to your installation. The **Name** field of the input form specifies the name of the location where the file containing CLOB data is to be put. The **File** field of the input form specifies the full path name of the file which contains the CLOB data. The **MIME Type** field enables the user to input a value for the MIME type – if the value input by the user for this field is uploaded, it can be fetched from the database for the given name while downloading the clob content. When the form is submitted, PutClobStream.jsp is called. The data specified on the form provides the parameter values for the webjects.

```
<%@page language="java" session="false"
errorPage="IEError.jsp"%>

<%@ taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie" %>

<html>
<head>
<title> CLOB Upload</title>
</head>
<BODY>
<H2> Upload File</H2>
<form method="POST" action="PutClobStream.jsp">
<TABLE>
 <tr>
  <td align=right>
     <B>JDBC Adapter Instance: </B>
  </td>
  <td>
     <INPUT name="instance" type="text" size=50>
  </td>
 </tr>
 <tr>
  <td align=right>
     <B>Name: </B>
  </td>
  <td>
     <INPUT name="filename" type="text" size=50>
  </td>
 </tr>
 <tr>
  <td align=right>
     <B>File: </B>
  </td>
  <td>
     <INPUT name="file" type="text" size=50>
  </td>
</tr>
 <tr>
```

```
  <td align=right>
    <b>MIME Type:</b>
  </td>
  <td>
    <input name="mimetype" type="text" size=50>
  </td>
 </tr>
<tr>
  <td>
  </td>
  <td align=left>
     <INPUT name="submit" type="submit"  value="Submit"
id=button>
  </td>
 </tr>
</table>
</form>
</body>
</html>
```

In the following PutClobStream.jsp, the Do-Sql webject creates the location in the CLOBTEST table where the file containing CLOB data is to be placed, using the data from the **Name** field of the input form to identify the location. Note that values are inserted into the table for the columns FILENAME and MIMETYPE as well, these being the values input by the user for the fields **File** and **MIME Type** while executing UploadClob.jsp. The Put-Clob-Stream webject then puts the file specified in the **File** field of the input form into the FILECONTENT column of the CLOBTEST table, in the location specified in the **Name** field.

```
<%@page language="java" session="false"
errorPage="IEError.jsp"%>

<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>

<html>
<head><title>Put-Clob-Stream Webject</title>
<BASE>
</head>
<body>
<h1>Put-Clob-Stream Webject:</h1>
<ie:unit>
<ie:webject name="Do-Sql" type="ACT">
   <ie:param name="INSTANCE"  data="${FORM[]instance[]}"
default="jdbcAdapter" />
   <ie:param name="CLASS"     data="CLOBTEST"/>
   <ie:param name="SQL"       data="INSERT INTO CLOBTEST
VALUES('${FORM[]filename[]}',EMPTY_CLOB(),'${FORM[]file[]}',${F
ORM[]mimetype[]})"/>
   <ie:param name="GROUP_OUT" data="temp" />
</ie:webject>
<ie:failure>
<!-- No failure processing -->
</ie:failure>
</ie:unit>
<ie:webject name="Put-Clob-Stream" type="ACT">
   <ie:param name="INSTANCE"  data="${FORM[]instance[]}"
default="jdbcAdapter"/>
   <ie:param name="ATTRIBUTE" data="FILECONTENT"/>
   <ie:param name="CLASS"     data="CLOBTEST"/>
   <ie:param name="WHERE"
data="NAME='${FORM[]filename[]}'"/>
```

```
   <ie:param name="FILENAME"  data="${FORM[]file[]}"/>
   <ie:param name="GROUP_OUT" data="PutClob"/>
</ie:webject>
<%
  String file = request.getParameter ("file");
%>
<b><i><%=file %></i> Uploaded to Database</b>
</body>
</html>
```

Query-Attributes

Description

Returns all unique records that match the specified query criteria. The records come from the specified table. If duplicate values exist, then the duplicate value is returned only once.

Syntax

```
<ie:webject name="Query-Attributes" type="OBJ">
  <ie:param name="ATTRIBUTE" data="attribute"/>
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="class"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="MAX_QUERY_SIZE" data="max_query_size"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="SORTBY" data="sortby"/>
  <ie:param name="SORTED" data="[ASC | DESC]"/>
  <ie:param name="WHERE" data="where_clause"/>
</ie:webject>
```

Parameters

Required	Select	Optional
CLASS	SORTBY	ATTRIBUTE
INSTANCE	SORTED	BLOB_COUNT
WHERE		CONNECTION_ATTEMPTS
		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		MAX_QUERY_SIZE
		PASSWD

ATTRIBUTE

Specifies which attributes from the queried object to return. The default for this parameter is to return all attributes. Multiple values can be specified for this parameter. This parameter is optional.

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Specifies which SQL-related table you want to query. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

MAX_QUERY_SIZE

Specifies maximum number of objects returned from a database query. The default for this parameter is 2000. If there are more than the maximum number of objects, the JDBC driver silently drops the extra objects. This parameter is optional.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

SORTBY

Defines the attribute names to be used as the sort parameters. Multiple values can be specified for this parameter. Attributes are sorted in the order in which each SORTBY parameter appears in the webject. If no values are specified for this parameter, no sorting will occur. If SORTED is specified, then one SORTBY parameter should be specified for each occurrence of the SORTED parameter.

SORTED

Describes how to sort the attribute names specified in the SORTBY parameter. Valid values are ASC for ascending and DESC for descending. For each occurrence of this parameter, one SORTBY parameter should also be specified. The default value for this parameter is ASC.

WHERE

Specifies search criteria for the database objects to return. The value for this parameter is specified as an SQL formatted where clause. This parameter is required.

Example

The following example documents the QueryAttributes.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the database is queried for unique values of the DEPTNO attribute of the EMP table. The results are sorted in descending order, and are displayed using the Display-Table webject.

Note: To run this example on your own system, you need to replace the value of the INSTANCE parameter with a value appropriate to your installation.

```
<%@page language="java" session="false"
                                        errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                    prefix="ie"%>

<html>
<head><title>Query-Attributes Webject</title>
<BASE>
</head>
<body>

<h1>Query-Attributes webject:</h1>
<h3> Based on specified query criteria, returns a set of
distinct records from a table</h3>

<ie:webject name="Query-Attributes" type="OBJ">
  <ie:param name="INSTANCE"    data="${@FORM[]instance[]}"
                                  default="jdbcAdapter"/>
  <ie:param name="ATTRIBUTE"   data="${@FORM[]column[]}"
                                         default="DEPTNO"/>
  <ie:param name="SORTBY"      data="DEPTNO"/>
  <ie:param name="SORTED"      data="DESC"/>
  <ie:param name="CLASS"       data="${@FORM[]table[]}"
                                           default="EMP"/>
  <ie:param name="WHERE"       data="()"/>
  <ie:param name="GROUP_OUT"   data="QueryAttribute"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP"/>

</body>
</html>
```

Query-Objects

Description

Returns all records that match the specified query criteria.

Syntax

```
<ie:webject name="Query-Objects" type="OBJ">
  <ie:param name="ATTRIBUTE" data="attribute"/>
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="class"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="MAX_QUERY_SIZE" data="max_query_size"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="SORTBY" data="attribute"/>
  <ie:param name="SORTED" data="[ASC | DESC]"/>
  <ie:param name="WHERE" data="where_clause"/>
</ie:webject>
```

Parameters

Required	Select	Optional
CLASS	SORTBY	ATTRIBUTE
INSTANCE	SORTED	BLOB_COUNT
WHERE		CONNECTION_ATTEMPTS
		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		MAX_QUERY_SIZE
		PASSWD

ATTRIBUTE

Specifies the name of the attributes to be returned. The default value for this parameter is to return all attributes. Multiple values can be specified for this parameter. This parameter is optional.

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Specifies which table to query. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

MAX_QUERY_SIZE

Specifies maximum number of objects returned from a database query. The default for this parameter is 2000. If there are more than the maximum number of objects, the JDBC driver silently drops the extra objects. This parameter is optional.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

SORTBY

Defines the attribute names to be used as the sort parameters. Multiple values can be specified for this parameter. Attributes are sorted in the order in which each SORTBY parameter appears in the webject. If no values are specified for this parameter, no sorting will occur. If SORTED is specified, then one SORTBY parameter should be specified for each occurrence of the SORTED parameter.

SORTED

Describes how to sort the attribute names specified in the SORTBY parameter. Valid values are ASC for ascending and DESC for descending. For each occurrence of this parameter, one SORTBY parameter should also be specified. The default value for this parameter is ASC.

WHERE

Specifies search criteria for the database objects to return. The value for this parameter is specified as an SQL formatted where clause. This parameter is required.

Example

The following example documents the QueryObjects.jsp file, which is located in the <ie_dir>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the database is queried for all values of the DEPTNO attribute of the EMP table. The results are sorted in descending order, and are displayed using the Display-Table webject.

Note: To run this example on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

```
<%@page language="java" session="false"
                                          errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                    prefix="ie"%>

<html>
<head><title>Query-Objects Webject</title>
<BASE>
</head>
<body>
<h1>Query-Objects webject: </h1>
<h3>Query specified table in a database</h3>

<ie:webject name="Query-Objects" type="OBJ">
  <ie:param name="INSTANCE"    data="${@FORM[]instance[]}"
                                    default="jdbcAdapter"/>
  <ie:param name="ATTRIBUTE"   data="${@FORM[]column[]}"
                                   default="DEPTNO"/>
  <ie:param name="SORTBY"      data="DEPTNO"/>
  <ie:param name="SORTED"      data="DESC"/>
  <ie:param name="CLASS"       data="${@FORM[]table[]}"
                                     default="EMP"/>
  <ie:param name="WHERE"       data="()"/>
  <ie:param name="GROUP_OUT"   data="QueryObject"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP"/>

</body>
</html>
```

Send-Blob-Stream

Description

Sends any BLOB information from an object as a stream to the browser or other application which calls the webject.

Definition

BLOB (noun): An acronym for Binary Large OBject. Any random large block of bits that needs to be stored in a database, such as a picture or sound file. A BLOB is an object stored in a database that cannot be interpreted within the database itself

Syntax

```
<ie:webject name="Send-Blob-Stream" type="ACT">
  <ie:param name="ATTRIBUTE" data="attribute_with_blob_data"/>
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="tablename"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="FILENAME" data="file_name"/>
  <ie:param name="GROUP_OUT" data="group"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="MIMETYPE" data="mimetype"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="WHERE" data="where_clause"/>
</ie:webject>
```

Parameters

Required	Select	Optional
ATTRIBUTE		BLOB_COUNT
CLASS		CONNECTION_ATTEMPTS
FILENAME		CONNECTION_ATTEMPT_INTERVAL
INSTANCE		DBUSER
MIMETYPE		GROUP_OUT
WHERE		PASSWD

ATTRIBUTE

Specifies which attribute or column in the database to return as binary data. This parameter is required.

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Specifies the name of the table containing the BLOB data. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

FILENAME

Specifies the name of the file to return. The value specified for this attribute is used to write the Content-Disposition HTTP Header. When the BLOB information is saved as a file, the FILENAME value is shown as the default file name.

If placed in single quotes, the string typed here will be used as the fully qualified name for the file data. If no quotes are used, the string represents the attribute which contains the file path. This parameter is required.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

MIMETYPE

Specifies the name of the attribute that has a mime type. If placed in single quotes, the string typed here is then used as the mime type for the BLOB data, otherwise this value represents a column in the table. This parameter is required.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

WHERE

Specifies search criteria for the database object containing BLOB data to return. The value for this parameter is specified as an SQL formatted where clause. This parameter is required.

Example

The following example documents the SendBlobStream.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

This example uses the JSP useBean directive in order to display the image. To understand this example you must be familiar with the JSP useBean directive and its use. For further information about the API methods called within the useBean directive, see the API documentation that is in the following location:

<*ie_dir*>/codebase/infoengine/docs/apidocs

where <*ie_dir*> is the location where Info*Engine is installed.

In this example, the DownloadBlob.jsp file, located in the same examples directory, provides the input form necessary for providing the appropriate

parameter values for this example and should be run to execute SendBlobStream.jsp.

The **JDBC Adapter Instance** field of the input form should specify the INSTANCE parameter value appropriate to your installation. The **Name** field specifies the name of the location where the file containing BLOB data is located. The **File Name** field specifies the full path name of the file which contains the BLOB data. The **Mime Type** field specifies the mime type of the file. When the form is submitted, SendBlobStream.jsp is called. The data specified on the form provides the parameter values for the webjects.

```
<%@page language="java" errorPage="IEError.jsp"
import="java.util.*,com.infoengine.object.factory.*,com.infoeng
ine.SAK.*,com.infoengine.object.*" %>

<%@ taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>

<ie:getService varName="vdb"/>
<jsp:useBean id="qa" class="com.infoengine.SAK.ActionWebject">
<%      out.clearBuffer();
    qa.setService(vdb);
         qa.setName("Send-Blob-Stream");
   boolean user_input_file_name = true;

         boolean user_input_mime_type = false;
    String file_name = request.getParameter("filename");

    if(file_name != null){
       file_name = file_name.trim();
    if(!(file_name.startsWith("'") &&
file_name.endsWith("'"))){user_input_file_name = false;}

    }else{ file_name ="";user_input_file_name = false;}

String mtype = request.getParameter("mimetype");
if(mtype != null){

         mtype = mtype.trim();

         if(mtype.startsWith("'") && mtype.endsWith("'"))

          user_input_mime_type = true;

         }else    mtype = "";
         StringBuffer name=new StringBuffer("NAME='");
         name.append(request.getParameter("name"));
         name.append("'");
    if(user_input_file_name == false || user_input_mime_type ==
false)
     {
      /* No value was input by the user for File Name and/or
MIME type; the following webject fetches these from the
database and stores in group "temp" */

       %><ie:webject name="Query-Objects" type="OBJ">
              <ie:param name="INSTANCE"
data="${@FORM[]instance[]}" default="jdbcAdapter"/>
              <ie:param name="CLASS"     data="BLOBTEST"/>
              <%if(user_input_mime_type == false){%>
    <ie:param name="ATTRIBUTE"
data="${@FORM[]mimetype[]}"/><%}%>
<%if(user_input_file_name == false){%>
<ie:param name="ATTRIBUTE"
data="${@FORM[]filename[]}"/><%}%>
```

```
            <ie:param name="WHERE"
data="<%=name.toString()%>"/>
            <ie:param name="GROUP_OUT" data="temp"/>
            </ie:webject><%

     /* Fetch the MIME type and/or file name from the group
"temp" */
     IeGroup ie_group = (vdb.getCollection()).getGroup("temp");
     Group group = new Group(ie_group);
     Enumeration elements = group.getElements();
     Element elem = null;
     Att att1 = null, att2 = null;
     if(elements.hasMoreElements())
      {
       elem = (Element)elements.nextElement();
       att1 = elem.getAtt(mtype);
       if(user_input_mime_type == false && att1 != null)
        {
         mtype = (att1.getValue()).toString();
         mtype = "'" + mtype + "'";
        }
       att2 = elem.getAtt(file_name);
       if(user_input_file_name == false && att2 != null)
        {
         file_name = (att2.getValue()).toString();
        }
      }
    }
        StringBuffer fname=new StringBuffer("filename='");
        fname.append(file_name);
        fname.append("'");
qa.addParam("INSTANCE",request.getParameter("instance"));
        qa.addParam("ATTRIBUTE","FILECONTENT");
        qa.addParam("DBUSER","gbabu");
        qa.addParam("PASSWD","gbabu");
        qa.addParam("CLASS","BLOBTEST");
        qa.addParam("WHERE",name.toString());
        qa.addParam("MIMETYPE", mtype);
        qa.addParam("FILENAME", file_name);
        qa.addParam("GROUP_OUT","SendBlob");
        qa.setOutputStream(response.getWriter());
        response.addHeader("Content-
Disposition",fname.toString());
        /* Make sure the single quotes in mtype do not get
passed over to
        setContentType() */
if(mtype != "" && mtype != null)
        response.setContentType(mtype.substring(1,
mtype.length()-1));
        qa.invoke();        response.getWriter().close();
%>
</jsp:useBean>
```

Send-Bulk-Stream

Description

Retrieves a file from the file system local to the adapter and streams it back to the browser.

Syntax

```
<ie:webject name="Send-Bulk-Stream" type="ACT">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="tablename"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="FILENAME" data="file_name"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="INSTANCE" data="instance_name"/>
  <ie:param name="MIMETYPE" data="mimetype"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="WHERE" data="where_clause"/>
</ie:webject>
```

Parameters

Required	Select	Optional
FILENAME	CLASS	BLOB_COUNT
INSTANCE	WHERE	CONNECTION_ATTEMPTS
MIMETYPE		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Identifies the table name object containing the bulk attribute being returned. If the value specified for FILENAME is not placed in single quotes, then this parameter must be specified.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

FILENAME

Specifies the file to return. If placed in single quotes, the string typed here will be used as the fully qualified name for the file data. If no quotes are used, the string represents the attribute which contains the file path. If the database needs to be queried, then CLASS and WHERE parameters become required. This parameter is required.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

MIMETYPE

Specifies the mime type of the file. If placed in single quotes, the string typed here will be used as the mime type for the bulk data, otherwise this value represents a column in the table. This parameter is required.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

WHERE

Specifies search criteria for the database object containing bulk data to return. The value for this parameter is specified as an SQL formatted where clause. If the value specified for FILENAME is not placed in single quotes, then this parameter must be specified.

Example

The following example documents the SendBulkStream.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

This example uses the JSP useBean directive in order to display the image. To understand this example you must be familiar with the JSP useBean directive and its use. For further information about the API methods called within the useBean directive, see the API documentation that is in the following location:

<*ie_dir*>/codebase/infoengine/docs/apidocs

where <*ie_dir*> is the location where Info*Engine is installed.

In this example, the DownloadBulk.jsp file, located in the same examples directory, provides the input form necessary for providing the appropriate parameter values for this example and should be run to execute SendBulkStream.jsp.

The **JDBC Adapter Instance** field of the input form should specify the INSTANCE parameter value appropriate to your installation. The **File Name** field specifies the full path name of the file which contains the bulk data. The **Mime Type** field specifies the mime type of the file. When the form is submitted, SendBulkStream.jsp is called. The data specified on the form provides the parameter values for the webjects.

```
<%@page language="java" session="false"
errorPage="IEError.jsp"%>

<%@ taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie" %>

<html>
<head>
<title> Bulk Download</title>
</head>
<BODY>
<H2> Download File</H2>
<form method="POST" action="SendBulkStream.jsp" >
<TABLE>
 <tr>
  <td align=right>
     <B>JDBC Adapter Instance: </B>
  </td>
  <td>
     <INPUT name="instance" type="text" size=50>
  </td>
 </tr>
 <tr>
  <td align=right>
     <B>File Name: </B>
</td>
<td>
     <INPUT name="filename" type="text" size=50>
  </td>
 </tr>
```

```
<tr>
  <td align=right>
     <B>MIME Type: </B>
  </td>
  <td>
     <INPUT name="mimetype" type="text" size=50>
  </td>
 </tr>
 <tr>
  <td align="right">
    <b>Name:</b>
  </td>
  <td>
    <input name="name" type="text" size="50">
  </td>
 </tr>
 <tr>
  <td>
  </td>
  <td align=left>
     <INPUT name="submit" type="submit"  value="Retrieve"
id=button>
  </td>
 </tr>
</table>
</form>
</body>
</html>
```

The file SendBulkStream.jsp that gets executed upon submitting the above form is as follows:

```
<%@page language="java" errorPage="IEError.jsp"
import="java.util.*,com.infoengine.object.factory.*,com.infoeng
ine.SAK.*,com.infoengine.object.*" %>

<%@ taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>

<ie:getService varName="vdb"/>
<jsp:useBean id="qa" class="com.infoengine.SAK.ActionWebject">
<%      out.clearBuffer();
   qa.setService(vdb);
   qa.setName("Send-Bulk-Stream");
boolean user_input_file_name = true;
boolean user_input_mime_type = false;

String file_name =
request.getParameter("filename");if(file_name != null &&
!file_name.equals("")){
file_name = file_name.trim();

if(!(file_name.startsWith("'") && file_name.endsWith("'"))){
user_input_file_name = false;                                }
}else{
   file_name ="FILENAME";              user_input_file_name =
false;                             }
```

```
   /* Fetch the MIME type from the request, so it could be used
for the response stream */

String mime_type = request.getParameter("mimetype");
if(mime_type != null && !mime_type.equals("")){
mime_type = mime_type.trim();

if(mime_type.startsWith("'") && mime_type.endsWith("'"))
       user_input_mime_type = true;
}else
mime_type = "MIMETYPE";
   qa.addParam("INSTANCE",request.getParameter("instance"));
   qa.addParam("DBUSER","dbuser_name");
   qa.addParam("PASSWD","dbuser_passwd");
   if(user_input_mime_type == false || user_input_file_name ==
false)
    {
     /* No value was input by the user for MIME Type and/or
File Name */
     String name = request.getParameter("name");
     name = "NAME='" + name + "'";
     /* The following webject fetches the values for MIME type
and file name  from the database and stores them in group "temp"
*/
     %><ie:webject name="Query-Objects" type="OBJ">
       <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
       <ie:param name="CLASS"     data="BULKFILE"/>
                              <%if(user_input_file_name ==
false){%>

          <ie:param name="ATTRIBUTE" data="<%=file_name%>"/>
       <%}%>
       <%if(user_input_mime_type == false){%>
          <ie:param name="ATTRIBUTE" data="<%=mime_type%>"/>
<%}%>
       <ie:param name="WHERE"     data="<%=name%>"/>
       <ie:param name="GROUP_OUT" data="temp"/>
</ie:webject><%
     /* Fetch the MIME type and/or file name from the group
"temp" */
     IeGroup ie_group = (vdb.getCollection()).getGroup("temp");
     Group group = new Group(ie_group);
     Enumeration elements = group.getElements();
     Element elem = null;
     Att att1 = null, att2 = null;
     if(elements.hasMoreElements())
      {
       elem = (Element)elements.nextElement();
       att1 = elem.getAtt(mime_type);
       if(user_input_mime_type == false && att1 != null)
        {
         mime_type = (att1.getValue()).toString();
         mime_type = "'" + mime_type + "'";
        }
       att2 = elem.getAtt(file_name);
       if(user_input_file_name == false && att2 != null)
        {
         file_name = (att2.getValue()).toString();
         file_name = "'" + file_name + "'";
        }
      }
    }
```

```
    qa.addParam("MIMETYPE", mime_type);
    qa.addParam("FILENAME", file_name);
    qa.addParam("GROUP_OUT","SendBlob");
    qa.setOutputStream(response.getWriter());
    /* Make sure the single quotes in mime_type do not get
passed to setContentType() */
   if(!mime_type.equals("") && mime_type != null)
    response.setContentType(mime_type.substring(1,
mime_type.length()-1));
    qa.invoke();
    response.getWriter().close();
%>
</jsp:useBean>
```

Send-Clob-Stream

Description

Sends any CLOB information from a database object as a stream.

Definition

CLOB (noun): An acronym for Character Large OBject. A CLOB column stores single-byte fixed-width character objects, such as text documents. A CLOB is an object stored in a database that cannot be interpreted within the database itself.

Syntax

```
<ie:webject name="Send-Clob-Stream" type="ACT">
  <ie:param name="ATTRIBUTE" data="attribute_with_clob_data"/>
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="tablename"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="GROUP_OUT" data="group"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="MIMETYPE" data="mimetype"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="WHERE" data="where_clause"/>
</ie:webject>
```

Parameters

Required	Select	Optional
ATTRIBUTE		BLOB_COUNT
CLASS		CONNECTION_ATTEMPTS
INSTANCE		CONNECTION_ATTEMPT_INTERVAL
MIMETYPE		DBUSER
WHERE		GROUP_OUT
		PASSWD

ATTRIBUTE

Specifies the attribute or column in the database table from which to return the character data. This parameter is required.

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Identifies the name of the object containing the CLOB attribute being returned. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

MIMETYPE

Describes the name of the attribute that has a mime type. If placed in single quotes, the string typed here will be used as the mime type for the CLOB data, otherwise this value represents a column in the table. This parameter is required.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

WHERE

Specifies search criteria for the database object containing CLOB data to return. The value for this parameter is specified as an SQL formatted where clause. This parameter is required.

Example

The following example documents the SendClobStream.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

This example uses the JSP useBean directive in order to display the image. To understand this example you must be familiar with the JSP useBean directive and its use. For further information about the API methods called within the useBean directive, see the API documentation that is in the following location:

<*ie_dir*>/codebase/infoengine/docs/apidocs

where <*ie_dir*> is the location where Info*Engine is installed.

In this example, the DownloadClob.jsp file, located in the same examples directory, provides the input form necessary for providing the appropriate parameter values for this example and should be run to execute SendClobStream.jsp.

The **JDBC Adapter Instance** field of the input form specifies the INSTANCE parameter value appropriate to your installation. The **Name** field specifies the attribute containing CLOB data to be returned. The **File Name** field specifies the name of the object containing the CLOB attribute being returned. The **Mime Type** field specifies the name of the attribute that has a mime type. See the MIMETYPE parameter description for further details. When the form is submitted, SendClobStream.jsp is called. The data specified on the form provides the parameter values for the webjects.

```
<%@page language="java" session="false"
                                          errorPage="IEError.jsp"%>
<%@ taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                    prefix="ie" %>
<html>
<head>
<title> CLOB Download</title>
</head>
<BODY>
<H2> Download File</H2>
<form method="POST" action="SendClobStream.jsp" >
<TABLE>
 <tr>
  <td align=right>
     <B>JDBC Adapter Instance: </B>
  </td>
  <td>
     <INPUT name="instance" type="text" size=50>
  </td>
 </tr>
 <tr>
  <td align=right>
     <B>Name: </B>
  </td>
  <td>
     <INPUT name="name" type="text" size=50>
  </td>
 </tr>
 <tr>
  <td align=right>
     <B>File Name : </B>
  </td>
  <td>
     <INPUT name="filename" type="text" size=50>
  </td>
 </tr>
 <tr>
  <td align=right>
     <B>Mime Type: </B>
  </td>
  <td>
     <INPUT name="mimetype" type="text" size=50>
  </td>
 </tr>
 <tr>
  <td align=left>
     <INPUT name="submit" type="submit"  value="Retrieve"
                                                      id=button>
  </td>
 </tr>
</table>
</form>
</body>
</html>
```

The following SendClobStream.jsp file streams the specified plain.txt file from the FILECONTENT column of the CLOBTEST table to the calling application.

Note: To run this example on your own system, you need to replace the values of the DBUSER and PASSWD parameters with values appropriate to your installation.

The file SendClobStream.jsp that gets executed upon submitting the above form is shown below:

```
<%@page language="java" errorPage="IEError.jsp"
import="java.util.*,com.infoengine.object.factory.*,com.infoeng
ine.SAK.*,com.infoengine.object.*" %>

<%@ taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>

<ie:getService varName="vdb"/>
<jsp:useBean id="qa" class="com.infoengine.SAK.ActionWebject">
<%      qa.setService(vdb);
        qa.setName("Send-Clob-Stream");
   boolean user_input_file_name = true;
   boolean user_input_mime_type = false;
   String file_name = request.getParameter("filename");
   if(file_name != null){
      file_name = file_name.trim();
   if(!(file_name.startsWith("'")&& file_name.endsWith("'"))){
       user_input_file_name = false;
   }
   }else{
      file_name ="";
      user_input_file_name = false;
   }
   String mtype = request.getParameter("mimetype");
   if(mtype != null){
          mtype = mtype.trim();
          if(mtype.startsWith("'") && mtype.endsWith("'"))
               user_input_mime_type = true;
   }else
      mtype = "";
        StringBuffer name=new StringBuffer("NAME='");
        name.append(request.getParameter("name"));
        name.append("'");
   if(user_input_file_name == false || user_input_mime_type ==
false)
    {
     /* No value was input by the user for File Name and/or
MIME type; the following webject fetches these from the
database and stores in group "temp" */
      %><ie:webject name="Query-Objects" type="OBJ">
             <ie:param name="INSTANCE"
data="${@FORM[]instance[]}" default="jdbcAdapter"/>
             <ie:param name="CLASS"     data="CLOBTEST"/>
             <%if(user_input_mime_type == false){%>
   <ie:param name="ATTRIBUTE" data="${@FORM[]mimetype[]}"/>
             <%}%>

             <%if(user_input_file_name == false){%>
   <ie:param name="ATTRIBUTE" data="${@FORM[]filename[]}"/>
             <%}%>
             <ie:param name="WHERE"
data="<%=name.toString()%>"/>
```

```
                <ie:param name="GROUP_OUT" data="temp"/>
              </ie:webject><%

     /* Fetch the MIME type and/or file name from the group
"temp" */
 IeGroup ie_group = (vdb.getCollection()).getGroup("temp");
     Group group = new Group(ie_group);
     Enumeration elements = group.getElements();
     Element elem = null;
     Att att1 = null, att2 = null;
     if(elements.hasMoreElements())
      {
       elem = (Element)elements.nextElement();
       att1 = elem.getAtt(mtype);
       if(user_input_mime_type == false && att1 != null)
        {
         mtype = (att1.getValue()).toString();
         mtype = "'" + mtype + "'";
        }
       att2 = elem.getAtt(file_name);
       if(user_input_file_name == false && att2 != null)
        {
         file_name = (att2.getValue()).toString();
        }
      }
    }
        StringBuffer fname=new StringBuffer("filename='");
        fname.append(file_name);
        fname.append("'");
qa.addParam("INSTANCE",request.getParameter("instance"));
        qa.addParam("ATTRIBUTE","FILECONTENT");
        qa.addParam("DBUSER","dbuser_name");
        qa.addParam("PASSWD","dbuser_passwd");
        qa.addParam("CLASS","CLOBTEST");
        qa.addParam("WHERE",name.toString());
        qa.addParam("MIMETYPE",mtype);
        qa.addParam("FILENAME",file_name);
        qa.addParam("GROUP_OUT","SendClob");
        qa.setOutputStream(response.getWriter());
        response.addHeader("Content-
Disposition",fname.toString());
        /* Make sure the single quotes in mtype are not passed
to setContentType() */
   if(!mtype.equals("") && mtype != null)
        response.setContentType(mtype.substring(1,
mtype.length()-1));
        qa.invoke();
        response.getWriter().close();
%>
</jsp:useBean>
```

Transaction

Description

This webject may be used to start or end a database transaction, create a savepoint, perform a database commit, or rollback the changes either wholly or to a given savepoint within the transaction. Any of the above said actions may be achieved through this webject, by simply providing the appropriate webject parameters.

Syntax

```
<ie:webject name="Transaction" type="ACT">
  <ie:param name="INSTANCE"   data="instance"/>
  <ie:param name="DBUSER"     data="dbuser_name"/>
  <ie:param name="PASSWD"     data="dbuser_passwd"/>
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="ACTION"     data="type_of_action"/>
  <ie:param name="SAVEPNTNAME" data="savepoint_name"/>
  <ie:param name="SESSION_ID" data="session_id_string"/>
  <ie:param name="GROUP_OUT"  data="group_out"/>
</ie:webject>
```

NOTE: The parameter SAVEPNTNAME can be used only when the ACTION parameter takes a value of "SAVEPOINT" or "ROLLBACK".

Parameters

Required	Select	Optional
INSTANCE	SESSION_ID	BLOB_COUNT
ACTION	SAVEPNTNAME	CONNECTION_ATTEMPTS
	GROUP_OUT	CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		PASSWD

The following sections describe the possible actions for the Transaction webject.

START

This is to begin a database transaction. This would be the first action to occur in a task composed of a series of webjects participating in a given transaction.

INSTANCE, ACTION and GROUP_OUT are the required webject parameters while the remaining ones are optional. These have been described below:

ACTION

Specifies the type of action associated with the given transaction. In order to begin a database transaction, this must be set to "START". Specifying multiple ACTION webject parameters within a given <ie:webject> element is an error.

GROUP_OUT

Specifies the name of the group returned by the webject. A START transaction webject stores the session ID in this group, which in turn would be accessed by the subsequently executed webjects that participate in the same transaction. It is therefore a required webject parameter.

SESSION_ID

This webject parameter is optional. If the user does not provide this parameter, a session ID will be generated internally and stored in the output group. However, if the user does provide this parameter, the specified value will simply get stored in the output group.

Note:

1. The case associated with the value provided for the webject parameter ACTION is unimportant. Thus, the value can be "START", "start", etc.
2. The webject will throw an exception if the value provided by the user for the webject parameter SESSION_ID is an ID that is already being used by some other concurrently executing webject.
3. Executing the webject for this action would result in a connection instance being checked out from an existing pool of connections, which in turn would be used for the given transaction.

COMMIT

This is to perform a database commit. Executing this webject would cause any previously made changes to be saved permanently to the database.

INSTANCE, ACTION and SESSION_ID are the required webject parameters while the remaining ones are optional. These have been described below:

ACTION

Specifies the type of action associated with the given transaction. In order to perform a database commit, this must be set to "COMMIT". Specifying multiple ACTION webject parameters within a given <ie:webject> element is an error.

SESSION_ID

The webject uses this to fetch the database connection corresponding to the given transaction and it is on this connection that a commit is performed. It is therefore a required webject parameter.

The user may specify a value for this webject parameter as shown below:

<ie:param name="SESSION_ID" data="${<value>[]session_id[]}"/>

The above syntax makes use of dynamic parameter value substitution, where <value> is the value provided by the user for the webject parameter

GROUP_OUT in the START transaction webject that was used to start the given transaction.

Note:

1. The case associated with the value provided for the webject parameter ACTION is unimportant. Thus, the value can be "COMMIT", "commit", etc.
2. Executing the webject for this action would result in the database connection corresponding to the given transaction to be released back into a pool of connections. However, the session ID associated with the transaction would still be intact and may be referenced by any of the subsequently executed webjects that are participating in the same transaction.

SAVEPOINT

This is to set a savepoint within a given transaction. Once a savepoint has been created, the transaction can be rolled back to that savepoint without affecting any of the changes that occurred prior to its creation.

INSTANCE, ACTION, SAVEPNTNAME and SESSION_ID are the required webject parameters while the remaining ones are optional. These have been described below:

ACTION

Specifies the type of action associated with the given transaction. In order to create a savepoint within the transaction, this must be set to "SAVEPOINT". Specifying multiple ACTION webject parameters within a given <ie:webject> element is an error.

SAVEPNTNAME

Specifies the name of the savepoint instance to be created. This is a required webject parameter.

SESSION_ID

The webject uses this to identify the given transaction and it is within this transaction that the given savepoint is created. It is therefore a required webject parameter.

The user may specify a value for this webject parameter as shown below:

<ie:param name="SESSION_ID" data="${<value>[]session_id[]}"/>

where <value> is the value provided by the user for the webject parameter GROUP_OUT in the START transaction webject that was used to start the given transaction.

Note:

1. The case associated with the value provided for the webject parameter ACTION is unimportant. Thus, the value can be "SAVEPOINT", "savepoint" etc.

2. Savepoint creation is currently supported only for Oracle 9i. For any other database type, an exception would be thrown upon executing the webject for this action.

3. There are certain restrictions with regard to the value that can be provided for the webject parameter SAVEPNTNAME. Thus, the first character must be a letter and any of the following characters must either be a letter or a digit or any of the characters '#', '$' or '_'. An exception would be thrown upon violating any of these restrictions.

4. If the value provided for the webject parameter SAVEPNTNAME is the name of a previously created savepoint instance that exists within the same transaction, the currently created instance will replace the old instance.

ROLLBACK

This is to rollback previously made changes either wholly, or to a given savepoint within the transaction.

INSTANCE, ACTION and SESSION_ID are the required webject parameters while the remaining ones are optional. These have been described below:

ACTION

Specifies the type of action associated with the given transaction. In order to rollback any changes, this must be set to "ROLLBACK". Specifying multiple ACTION webject parameters within a given <ie:webject> element is an error.

SAVEPNTNAME

Specifies the name of the savepoint to rollback to. This is an optional parameter, since the user may choose to rollback either all of the changes made during the given transaction (by not providing this parameter at all) or only those changes made after the given savepoint was set within the transaction.

SESSION_ID

The webject uses this to fetch the database connection corresponding to the given transaction and it is on this connection that a rollback is performed. It is therefore a required webject parameter.

The user may specify a value for this webject parameter as shown below:

<ie:param name="SESSION_ID" data="${<value>[]session_id[]}"/>

where <value> is the value provided by the user for the webject parameter GROUP_OUT in the START transaction webject that was used to start the given transaction.

Note:

1. The case associated with the value provided for the webject parameter ACTION is unimportant. Thus, the value can be "ROLLBACK", "rollback", etc.

2. Rollback to a savepoint is currently supported only for Oracle 9i. For any other database type, an exception would be thrown upon executing the webject for this action.

3. The value that can be provided for the webject parameter SAVEPNTNAME has the same restrictions here as it does for the SAVEPOINT action. Besides, it must correspond to an existing savepoint instance within the given transaction.

END

This is to end a given transaction, so that previously made changes are saved permanently to the database.

INSTANCE, ACTION and SESSION_ID are the required webject parameters while the remaining ones are optional. These have been described below:

ACTION

Specifies the type of action associated with the given transaction. In order to end the transaction, this must be set to "END". Specifying multiple ACTION webject parameters within a given <ie:webject> element is an error.

SESSION_ID

The webject uses this to identify the given transaction. Once the transaction has been identified, a database commit is performed, thereby ending the transaction. It is therefore a required webject parameter.

The user may specify a value for this webject parameter as shown below:

<ie:param name="SESSION_ID" data="${<value>[]session_id[]}"/>

where <value> is the value provided by the user for the webject parameter GROUP_OUT in the START transaction webject that was used to start the given transaction.

The preceding sections describe only those webject parameters that are specific to various actions. Besides these, there are certain webject parameters that are common to all actions and these have been described in the following sections:

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the Info*Engine User's Guide.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

Example

The following example documents the Transaction.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory. It illustrates how the Transaction webject may be used in a task to effectively manage a database transaction.

```
<%@page language="java" session="false"
errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
prefix="ie"%>

<html>
<head><title>Transaction Management</title>
<BASE>
</head>
<body bgcolor="#AABBCC">
<h1>Transaction Management: </h1>
<h3>Illustrates the various transaction webjects.</h3>

<ie:unit>

  <ie:webject name="Transaction" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="DBUSER"    data="dbuser_name"/>
    <ie:param name="PASSWD"    data="dbuser_passwd"/>
    <ie:param name="ACTION"    data="START"/>
    <ie:param name="GROUP_OUT" data="session"/>
  </ie:webject>

  <ie:webject name="Do-SQL" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
    <ie:param name="SQL"       data="INSERT INTO EMP VALUES
(8000,'John','CLERK',7902,'11-Jan-1999',900,NULL,20)"/>
    <ie:param name="SQL"       data="INSERT INTO EMP VALUES
(8001,'Jim','CLERK',7902,'21-Jun-1999',850,NULL,20)"/>
    <ie:param name="SQL"       data="INSERT INTO EMP VALUES
(8002,'Jack','CLERK',7902,'27-Dec-1999',800,NULL,20)"/>
    <ie:param name="MODE" data="batch"/>
    <ie:param name="GROUP_OUT" data="DoSql"/>
  </ie:webject>

  <ie:webject name="Prepared-Batch-Update" type="ACT">
    <ie:param name="INSTANCE"    data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
    <ie:param name="SQL"         data="UPDATE EMP SET SAL = ?,
HIREDATE = ? WHERE EMPNO = ?"/>
    <ie:param name="DELIMITER"   data="^"/>
    <ie:param name="PARAMTYPES"  data="INTEGER^DATE^INTEGER"/>
    <ie:param name="PARAMVALUES" data="800^'12-Jan-
1999'^8000"/>
    <ie:param name="PARAMVALUES" data="750^'22-Jun-
1999'^8001"/>
    <ie:param name="PARAMVALUES" data="700^'28-Dec-
1999'^8002"/>
    <ie:param name="GROUP_OUT"   data="PreparedBatchUpdate"/>
  </ie:webject>
```

```
  <ie:webject name="Transaction" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
    <ie:param name="ACTION"    data="COMMIT"/>
  </ie:webject>


  <ie:webject name="Transaction" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
    <ie:param name="ACTION"    data="SAVEPOINT"/>
    <ie:param name="SAVEPNTNAME" data="svpnt#1"/>
  </ie:webject>


  <ie:webject name="Do-SQL" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
    <ie:param name="SQL"       data="INSERT INTO EMP VALUES
(8003,'Jerry','CLERK',7902,'20-DEC-1999',800,NULL,20)"/>
    <ie:param name="GROUP_OUT" data="DoSql"/>
  </ie:webject>


  <ie:webject name="Create-Object" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
    <ie:param name="CLASS"     data="EMP"/>
    <ie:param name="FIELD"     data="EMPNO='1110'"/>
    <ie:param name="FIELD"     data="ENAME='Herman'"/>
    <ie:param name="FIELD"     data="JOB='ENGINEER'"/>
    <ie:param name="FIELD"     data="MGR='7990'"/>
    <ie:param name="FIELD"     data="HIREDATE='23-MAY-1999'"/>
    <ie:param name="FIELD"     data="SAL='2200'"/>
    <ie:param name="FIELD"     data="COMM=''"/>
    <ie:param name="FIELD"     data="DEPTNO='40'"/>
    <ie:param name="GROUP_OUT" data="CreateObject"/>
  </ie:webject>


  <ie:webject name="Transaction" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
    <ie:param name="ACTION"    data="SAVEPOINT"/>
    <ie:param name="SAVEPNTNAME" data="svpnt#1"/>
  </ie:webject>


  <ie:webject name="Delete-Objects" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
    <ie:param name="CLASS"     data="${@FORM[]table[]}"
```

```
default="EMP"/>
    <ie:param name="WHERE"     data="${@FORM[]where[]}"
default="ENAME='Herman'"/>
    <ie:param name="GROUP_OUT" data="DeleteObject"/>
  </ie:webject>


  <ie:webject name="Transaction" type="ACT">
    <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
    <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
    <ie:param name="ACTION"    data="ROLLBACK"/>
    <ie:param name="SAVEPNTNAME" data="svpnt#1"/>
  </ie:webject>


  <ie:success>
    <ie:webject name="Create-Group" type="GRP">
      <ie:param name="ELEMENT" data="SUCCESS=success"/>
      <ie:param name="DELIMITER" data=":"/>
      <ie:param name="GROUP_OUT" data="success"/>
    </ie:webject>
    <ie:webject name="Transaction" type="ACT">
     <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
     <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
     <ie:param name="ACTION"    data="END"/>
    </ie:webject>
  </ie:success>
  <ie:failure>
    <ie:webject name="Create-Group" type="GRP">
      <ie:param name="ELEMENT" data="FAILURE=failure"/>
      <ie:param name="DELIMITER" data=":"/>
      <ie:param name="GROUP_OUT" data="failure"/>
    </ie:webject>
    <ie:webject name="Transaction" type="ACT">
      <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
default="jdbcAdapter"/>
      <ie:param name="SESSION_ID"
data="${session[]session_id[]}"/>
      <ie:param name="ACTION"    data="ROLLBACK"/>
    </ie:webject>
    <ie:webject name="Throw-Exception" type="MGT"/>
  </ie:failure>
</ie:unit>
</body>
</html>
```

The example modifies the table EMP as described below:

A transaction is first started using the START Transaction webject. Some three arbitrary rows are then inserted into the table using the Do-Sql webject, which are then updated using the Prepared-Batch-Update webject. The changes made are saved permanently into the database through the COMMIT transaction webject. A savepoint called "svpnt#1" is then set through the SAVEPOINT transaction webject.

Two more rows are inserted using the Do-Sql and Create-Object webjects and a savepoint called "svpnt#1" is set once again, so that it replaces the previously set savepoint. The row that was inserted using the Create-Object webject is then deleted using the Delete-Objects webject. A rollback is then

performed to the savepoint called "svpnt#1" through the ROLLBACK transaction webject. This undoes the deletion caused by the Delete-Objects webject.

Note the use of the success and failure blocks within a unit in the example, to provide for success and failure processing. If all of the above described actions went through fine, control would enter the success block; here, an output group called "success" is created and the transaction is ended using the END transaction webject. This saves the previously made changes permanently into the database. On the other hand, if any of the actions had failed, control would directly enter the failure block; here, an output group called "failure" is created and the previously made changes are then rolled back wholly. This way, it is ensured that the database is not left in an inconsistent state. Also, note the use of a number of other JDBC Adapter webjects in the example - in order for these to participate in a transaction, it is only required to use the SESSION_ID webject parameter in each of these. This ensures that all such webjects get executed as part of the same transaction. If the SESSION_ID parameter is not provided for any of these other webjects, they would get executed the old way, rather than as part of the given transaction.

Update-Objects

Description

Updates column values in a table.

Syntax

```
<ie:webject name="Update-Objects" type="ACT">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CLASS" data="class"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="FIELD" data="attributes"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbpassword"/>
  <ie:param name="WHERE" data="where_clause"/>
</ie:webject>
```

Parameters

Required	Select	Optional
CLASS		BLOB_COUNT
FIELD		CONNECTION_ATTEMPTS
INSTANCE		CONNECTION_ATTEMPT_INTERVAL
WHERE		DBUSER
		GROUP_OUT
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CLASS

Specifies the name of the table to update. This parameter is required.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

FIELD

Specifies the attribute or table column to be updated. The value for this parameter is specified in the following manner:

name='*value*'

where *name* is the attribute name, and *value* is the value to store in the attribute. The *name* portion of the parameter value must exactly match the attribute or column name. Multiple values can be specified for this parameter. This parameter is required.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

WHERE

Specifies search criteria for the database table row to update. The value for this parameter is specified as an SQL formatted where clause. This parameter is required.

Example

The following example documents the UpdateObjects.jsp file, which is located in the <*ie_dir*>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the SAL field of the database record in the EMP table for the employee named ALLEN is updated. The database is then queried and the updated record is displayed using the Query-Objects and Display-Table webjects respectively.

Note: To run this example on your own system, you need to replace the values of the INSTANCE parameter with values appropriate to your installation.

```
<%@page language="java" session="false"
                                        errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                    prefix="ie"%>
<html>
<head><title>Update-Objects Webject</title>
<BASE>
</head>
<body>
<h1>Update-Objects Webject: </h1>
<h3>Updates column values in a table</h3>

<ie:webject name="Update-Objects" type="ACT">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                      default="jdbcAdapter"/>
  <ie:param name="CLASS"     data="EMP"/>
  <ie:param name="WHERE"     data="${@FORM[]where[]}"
                                    default="ENAME='ALLEN'"/>
  <ie:param name="FIELD"     data="sal='3700'"/>
  <ie:param name="GROUP_OUT" data="update"/>
</ie:webject>

<ie:webject name="Query-Objects" type="OBJ">
  <ie:param name="INSTANCE"  data="${@FORM[]instance[]}"
                                      default="jdbcAdapter"/>
  <ie:param name="CLASS"     data="EMP"/>
  <ie:param name="WHERE"     data="()"/>
  <ie:param name="GROUP_OUT" data="emp"/>
</ie:webject>

<ie:webject name="Display-Table" type="DSP"/>

</body>
</html>
```

Validate-User

Description

Validates the identity of the SQL-type database user.

Syntax

```
<ie:webject name="Validate-User" type="ACT">
  <ie:param name="BLOB_COUNT" data="number_of_BLOBs"/>
  <ie:param name="CONNECTION_ATTEMPT_INTERVAL" data="interval"/>
  <ie:param name="CONNECTION_ATTEMPTS" data="attempts"/>
  <ie:param name="DBUSER" data="dbuser_name"/>
  <ie:param name="GROUP_OUT" data="group_out"/>
  <ie:param name="INSTANCE" data="instance"/>
  <ie:param name="PASSWD" data="dbpassword"/>
</ie:webject>
```

Parameters

Required	Select	Optional
INSTANCE		BLOB_COUNT
		CONNECTION_ATTEMPTS
		CONNECTION_ATTEMPT_INTERVAL
		DBUSER
		GROUP_OUT
		PASSWD

BLOB_COUNT

Specifies how many BLOBs to deliver to the webject. Specifying a value of 0 results in no BLOBs being delivered. Specifying a value of more than 0 results in up to that specified number of BLOBs being delivered. For example, if this parameter is specified with a value of five (5), then no more than five BLOBs will be delivered to the webject.

The default behavior for this parameter is that all remaining BLOBs are delivered to the webject. This parameter is optional.

CONNECTION_ATTEMPTS

Defines the maximum number of times to attempt establishing a connection to an adapter before returning an error. The default value is 1. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPTS defines the maximum number of times to iterate through the list of adapter instances.

CONNECTION_ATTEMPT_INTERVAL

Defines the amount of time, in seconds, to delay between connection attempts. The default value is 60 seconds. This parameter is optional.

If multiple INSTANCE parameter values are specified, the value of CONNECTION_ATTEMPT_INTERVAL defines the number of seconds to wait between the attempts to iterate through the entire list of adapter instances.

DBUSER

Specifies the name to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

GROUP_OUT

Identifies the group returned by the webject. This parameter is optional.

INSTANCE

Specifies the name of the adapter that executes the webject. Adapter names are defined when the adapter is configured for use in your Info*Engine environment. This parameter is required.

In order to provide the ability to connect to other adapters if a specific adapter is not available, this parameter can be multi-valued. Info*Engine attempts to connect to the adapters in the order given. If the first adapter specified is not available, the next adapter listed is tried, and so on, until a connection is made. If a connection cannot be established with any listed adapter, an error is returned.

In conjunction with this parameter, you can include two other parameters, CONNECTION_ATTEMPTS and CONNECTION_ATTEMPT_INTERVAL.

PASSWD

Specifies the password to use when logging in to the data repository. If this parameter is specified in this webject, the webject value takes precedence over any value specified in the credentials mapping settings or in the adapter LDAP entry. If this parameter is not specified here, it must be specified in the credentials mapping settings or in the adapter LDAP entry. For more information about credentials mapping, see the *Info*Engine User's Guide*.

Example

The following example documents the ValidateUser.jsp file, which is located in the <ie_dir>/codebase/infoengine/jsp/examples/JDBCAdapter/examples directory.

In this example, the database user is validated, and the output of the Validate-User webject is displayed using the Display-Xml webject.

Note: To run this example on your own system, you need to replace the values of the INSTANCE, DBUSER and PASSWD parameters with values appropriate to your installation.

```
<%@page language="java" session="false"
                                        errorPage="IEError.jsp"%>
<%@taglib uri="http://www.ptc.com/infoengine/taglib/core"
                                                    prefix="ie"%>

<html>
<head><title>Validate-User Webject</title>
<BASE>
</head>
<body>

<h1>Validate-User Webject: </h1>
<h3>Validate the identity of the database user</h3>

<ie:webject name="Validate-User" type="ACT">
  <ie:param name="INSTANCE"      data="${@FORM[]instance[]}"
                                        default="jdbcAdapter"/>
  <ie:param name="DBUSER"        data="dbuser_name"/>
  <ie:param name="PASSWD"        data="dbuser_passwd"/>
  <ie:param name="GROUP_OUT"     data="emp"/>
</ie:webject>

<ie:webject name="Display-Xml" type="DSP"/>

</body>
</html>
```