

The AFRL JBI Platform Services

Version 1.1

10 October 2003

AFRL/IFSE

JBI Program Office

1.	DESCRIPTION.....	1
1.1	OUTLINE OF THIS DOCUMENT	1
2.	BACKGROUND.....	1
3.	VERSION 1.1 CONTENTS.....	1
3.1	JBI COMMON API (VERSION 1.0).....	2
3.2	J2EE APPLICATION SERVER (JBOSS 3.2.1)	3
3.3	INFORMATION OBJECT REPOSITORY (IOR) (VERSION 1.1)	3
3.4	METADATA REPOSITORY (MDR) (VERSION 1.1).....	3
3.5	SECURITY INFRASTRUCTURE (VERSION 1.1).....	3
3.6	INFORMATION MANAGEMENT (IM) TOOLS (VERSION 1.1)	4
4.	AFRL JBI VERSION 1.1 ARCHITECTURE	4
4.1	JBI COMMON API (CAPI) (VERSION 1.0)	5
4.1.1	<i>Information Object Specification for AFRL JBI</i>	5
4.1.2	<i>Publish and Subscribe</i>	5
4.1.3	<i>Query</i>	6
4.1.4	<i>MDR Interface</i>	6
4.1.5	<i>Authentication</i>	6
4.1.6	<i>Access Control</i>	7
4.2	J2EE APPLICATION SERVER	8
4.2.1	<i>Facades/Interfaces/Beans</i>	8
4.2.2	<i>Connection Pooling</i>	8
4.2.3	<i>Security Domains</i>	8
4.2.4	<i>Security Proxies</i>	9
4.2.5	<i>J2EE Access Control</i>	9
4.2.6	<i>Application Server Access Control for JMS</i>	9
4.3	INFORMATION OBJECT REPOSITORY (IOR).....	9
4.3.1	<i>Oracle Implementation</i>	Error! Bookmark not defined.
4.3.2	<i>Archive</i>	9
4.3.3	<i>Query</i>	10
4.4	METADATA REPOSITORY (MDR)	10
4.4.1	<i>Information Object Schemas</i>	10
4.4.2	<i>Interaction with JMS Topic Manager</i>	11
4.4.3	<i>Oracle-Specific Implementation Features</i>	Error! Bookmark not defined.
4.5	SECURITY INFRASTRUCTURE	11
4.5.1	<i>Security Database for IM Staff and Client Applications</i>	11
4.5.2	<i>Authentication</i>	12
4.5.3	<i>Access Control</i>	12
4.6	INFORMATION MANAGEMENT (IM) TOOLS.....	13
4.6.1	<i>J2EE Application Server Console</i>	13
4.6.2	<i>IM Staff Web Services</i>	13
5.	DISTRIBUTION AND DISCLAIMER STATEMENT	13
6.	INSTALLATION INSTRUCTIONS	14
6.1	INSTALLING THE JBI COMMON API VERSION 1.0	14
6.1.1	<i>Download the JBI Common API Version 1.0 Specification and JavaDocs</i>	14
6.2	WEB SERVICES	14
6.2.1	<i>Downloading the Apache Web Server</i>	Error! Bookmark not defined.
6.2.2	<i>Configuring the Apache Web Server</i>	Error! Bookmark not defined.
6.3	DATABASES AND REPOSITORIES.....	14
6.3.1	<i>Using Oracle as a Repository</i>	14
6.3.2	<i>Using MySQL as a Repository</i>	16
6.4	J2EE APPLICATION SERVER	17

6.4.1	Download the JBoss 3.2.1 J2EE Application Server	17
6.4.2	Installing the JBoss 3.2.1 Application Server with the Tomcat/Catalina 4.1.24 Servlet Engine	18
6.4.3	Configuring the JBoss Application Server to run AFRL JBI Components.....	18
6.4.4	Configuring JBoss for your databases.....	18
6.4.5	Launching the JBoss Application Server.....	19
7.	RUNNING THE WEB SERVICE EXAMPLES	21
7.1	SECURITY ADMINISTRATION WEB TOOL: WALK-THROUGH	23
7.1.1	Running the Security Application.....	23
7.2	METADATA REPOSITORY ADMINISTRATION WEB TOOL: WALK-THROUGH.....	25
7.3	INFORMATION OBJECT REPOSITORY ADMINISTRATION WEB TOOL: WALK-THROUGH	28
8.	APPENDIX A - INSTALLATION REQUIREMENTS	1
8.1	SERVER SIDE REQUIREMENTS.....	1
8.2	CLIENT SIDE REQUIREMENTS	1
	• JAVA JDK 1.4.1 OR HIGHER.....	1
9.	APPENDIX B - JBOSS CONFIGURATION FILES	2
9.1	JBOSS_ROOT/SERVER/JBI_ORACLE/CONF/LOGIN-CONFIG.XML	
	JBOSS_ROOT/SERVER/JBI_MYSQL/CONF/LOGIN-CONFIG.XML	2
9.2	JBOSS_ROOT/SERVER/JBI_ORACLE/CONF/LOG4J.XML	
	JBOSS_ROOT/SERVER/JBI_MYSQL/CONF/LOG4J.XML	2
9.3	JBOSS_ROOT/SERVER/JBI_ORACLE/CONF/SERVER.KEYSTORE	
	JBOSS_ROOT/SERVER/JBI_MYSQL/CONF/SERVER.KEYSTORE.....	3
9.4	JBOSS_ROOT/SERVER/JBI_ORACLE/DEPLOY/MDRIOR-ORACLE-DS.XML	
	JBOSS_ROOT/SERVER/JBI_MYSQL/DEPLOY/IOR-MYSQL-DS.XML	3
9.5	JBOSS_ROOT/SERVER/JBI_ORACLE/DEPLOY/MDRIOR-ORACLE-SERVICE.XML	
	JBOSS_ROOT/SERVER/JBI_MYSQL/DEPLOY/MDR-MYSQL-SERVICE.XML.....	3
9.6	JBOSS_ROOT/SERVER/JBI_ORACLE/DEPLOY/CMP-ORACLE-DS.XML	
	JBOSS_ROOT/SERVER/JBI_MYSQL/DEPLOY/CMP-MYSQL-DS.XML.....	3
9.7	JBOSS_ROOT/SERVER/JBI_ORACLE/DEPLOY/JMS-ORACLE-DS.XML	
	JBOSS_ROOT/SERVER/JBI_MYSQL/DEPLOY/JMS-MYSQL-DS.XML.....	3
9.8	JBOSS_ROOT/SERVER/JBI_ORACLE/DEPLOY/SECURITY-ORACLE-DS.XML	
	JBOSS_ROOT/SERVER/JBI_MYSQL/DEPLOY/SECURITY-MYSQL-DS.XML.....	3
9.9	JBOSS_ROOT/SERVER/JBI_ORACLE/JBOSS-SERVICE.XML	
	JBOSS_ROOT/SERVER/JBI_MYSQL/JBOSS-SERVICE.XML	4
9.10	JBOSS_ROOT/SERVER/JBI_ORACLE/JBOSS-SSL-RMI-SERVICE.XML	
	JBOSS_ROOT/SERVER/JBI_MYSQL/JBOSS-SSL-RMI-SERVICE.XML	4
10.	APPENDIX C - ORACLE 9I 9.2.X INSTALLATION NOTES	1
10.1	RUN THE 9I INSTALLER FOR THE ORACLE 9I SERVER	1
10.1.1	Click Install/De-install Products.....	1
10.1.2	Click next on the Welcome screen (Figure C-1)	1
10.1.3	File Locations (Figure C-2)	2
10.1.4	Available Products (Figure C-3)	3
10.1.5	Installation Types (Figure B-4).....	4
10.1.6	Database Configuration (Figure C-5).....	5
10.1.7	Oracle Services for Microsoft Transaction Server (Figure C-6).....	6
10.1.8	Oracle Services for Microsoft Transaction Server.....	7
10.1.9	Database Identification (Figure C-8).....	8
10.1.10	Database File Location (Figure C-9).....	9
10.1.11	Database Character Set (Figure C-10)	10
10.1.12	Summary (Figure C-11)	11
10.1.13	Database Configuration Assistant (Figure C-12).....	12
10.1.14	Click OK	12
10.1.15	End of Installation (Figure C-13)	13

10.2	INSTALLATION OF ORACLE PATCH.....	13
11.	APPENDIX D – XPATH TO SQL-92 CONVERSION.....	1
12.	APPENDIX E - TROUBLESHOOTING	3
12.1	JBOSS TROUBLE SHOOTING.....	3
12.2	WEB BASED INFORMATION MANAGEMENT TOOLS	3
12.3	REPOSITORY MIGRATION.....	6
13.	APPENDIX F - MIGRATING FROM PLATFORM SERVICES 1.0.1	7
	IF YOU HAVE A PLATFORM SERVICES 1.0.1, YOU CAN MIGRATE YOUR CURRENT SCHEMAS AND USERS/ROLES TO THE PLATFORM SERVICES 1.1 REPOSITORY USING THE INCLUDED MIGRATION SCRIPT.	7
	NOTE 1: THE JBOSS APPLICATION SERVER MUST BE RUNNING BEFORE USING THE MIGRATION SCRIPT. PLEASE SEE SECTION 6.4 FOR DETAILS ON STARTING JBOSS.	7
13.1	MIGRATING FROM ORACLE XMLTYPE REPOSITORY (PLATFORM SERVICES 1.0.1) TO ORACLE RELATIONAL.....	7
13.2	MIGRATING FROM ORACLE XMLTYPE REPOSITORY (PLATFORM SERVICES 1.0.1) TO MYSQL.....	8

1. Description

This document describes the software contents of the AFRL JBI Platform Services Distribution, delivered by the Joint Battlespace Infosphere's (JBI) in-house group. This document provides the step-by-step instructions to extract the necessary information from the CD, download open source and other reference material, and properly configure the software comprising the AFRL JBI Platform Services. This document does not describe how to acquire the necessary commercial off the shelf (COTS) products, but references proper configuration of the product(s) once procured.

1.1 Outline of This Document

The next section will briefly describe JBI in-house platform development activities. Section 3 describes the contents of this delivery as list of capabilities with some background discussion. Section 4 contains a detailed discussion of Version 1.1 features from an architectural perspective. Section 5 contains distribution and disclaimer notes regarding the delivered software. Section 6 includes high-level installation instructions that reference configuration scripts on the CD, whose purpose is to explain the basic flow of the installation process. Section 7 lists AFRL technical support points of contact. Section 8 contains simple instructions on using the Information Management Staff tools and gaining access to the administrative applications. The Appendices provide amplifying information on the content of the main document, as well as troubleshooting notes and procedures.

2. Background

Wherever possible, and as much as is achievable in any given implementation, the AFRL JBI platform service functionality resembles the JBI capabilities identified in the Scientific Advisory Board's JBI Report. Care has been taken to isolate the software interfaces and components from their underlying technologies so that others could be rapidly integrated and tested. The overall goal is to divorce ourselves from a particular implementation, technology, or vendor's tool; and, while a noble goal, project schedules and other harsh realities force implementations that aren't always to everyone's expectations. However, we believe the platform service architecture that has evolved over the duration of the project to date is a solid foundation on which to plug in new technologies if required, and will support incremental enhancements to make the architecture more survivable, robust, and secure as new concepts and implementations take hold.

This CD contains the platform service components that have been designed and implemented by the AFRL JBI group, exclusively. This group has also developed sample applications for using the Common API and client applications that interact with the platform. These applications, along with the client side infrastructure, are delivered on this CD.

3. Version 1.1 Contents

The Version 1.1 consists of documentation and prototype AFRL JBI platform service software in binary format. Source code is provided for the Application Server components, database stored

procedures, and Java Server Pages (JSP) deployed at the web tier on an as needed basis. Contact the AFRL JBI in house group to inquire about source code distribution.

The platform service components provided on the CD consist of: the JBI Common API Version 1.0, a J2EE application server – JBoss Version 3.2.1, a Metadata Repository (MDR) Version 1.1, an Information Object Repository (IOR) Version 1.1, a security infrastructure Version 1.1, and Information Management (IM) Staff Tools Version 1.1. Each of these components is described in the next several sub-sections.

3.1 JBI Common API (Version 1.0)

The JBI Common API (CAPI) is a programming language-agnostic specification that defines the interface between JBI clients and the platform. Its development is controlled by the Infospherics.org working group that meets periodically and maintains technical forums to facilitate API evolution. The current version of the specification, Version 1.0, defines how to create Information Objects, connect to the JBI platform for authentication, and initiate publish, subscribe or query sequences (*sessions*) subject to access control. Future versions will address issues like quality of service (QoS), and distributed collaboration at the API level. The specification download contains UML diagrams and Java language mappings documented using JavaDoc (CAPI Version 1.0).

The CD also contains the JBI Version 1.0 Information Object Metadata Standard, which has not changed since the V0.18 platform services release. This standard defines a core set of base metadata that every information object published using the JBI platform adheres to. All V1.0 clients delivered in conjunction with the Platform Services adhere to this standard.

The Platform Services the CD contains a browser link to the Infospherics.org download page. There, you will find the latest specification in binary format and discussion threads addressing the latest technical issues. The working group will continue to evolve the API specification over time; however, the included JBI Common API will be a baseline-controlled *snapshot* of the Common API. The baseline includes the Java language bindings for their Version 1.0 release.

Once you review the Java documents describing the mapping of the specification to the Java language, you'll note that the following capabilities and underlying technology implementations have been developed:

- Client-side Publish and Subscribe using the Java Message Service (JMS) with Xpath support for subscription specification over the published metadata.
- XPath support for Query specification, including wild cards.
- Interface to Metadata Repository (MDR) for reading and writing of information object schema.
- IOR interface for archiving information objects.
- Security infrastructure interface with SSL-enable communications with the platform.
- Support for retrieving partial Query results.
- Implements the Infospherics.org Common API Version 1.0 Specification and Java Language Mapping Version 1.0.

- Supports the JBI Version 1.0 Information Object Metadata Standard (also referred to as the Information Object *Model*).

3.2 J2EE Application Server (JBoss 3.2.1)

A fully configured JBoss 3.2.1 Java 2 Enterprise Edition (J2EE) Application Server is contained on the CD. The J2EE container is configured out of the box to manage Session Beans that support the MDR, IOR and JAAS business rules.

- The application server manages three security domains during run-time: one security domain for managing information dissemination among clients, one for the web administrative components, and one for the J2EE management console.
- The JBoss JMS Server (JBossMQ) is used as the underlying JMS server to support publish and subscribe operations. Currently, JBossMQ server is managed within the J2EE container.

3.3 Information Object Repository (IOR) (Version 1.1)

Version 0.18 introduced the IOR as a permanent store for published information objects designated as *archived*, and has been upgraded in AFRL JBI Platform V1.1. Specifically, the XMLType used in 0.18 has been replaced with a relational implementation, increasing the performance in both archival (and thus publication) rate and query response. The IOR feature set includes:

- Low-level support for archiving, deleting and querying for information objects.
- Stores intermediate Query result sets for incremental retrieval using getNext.
- Indexing support for fine-tuning database storage and retrieval of information objects.

3.4 Metadata Repository (MDR) (Version 1.1)

This repository contains detailed metadata schema data on all information objects that will be exchanged among clients using the platform. The CD contains build scripts that populate the repository with a default set of XML Schema Definition (XSD) version 1.1 documents. This release features tools to migrate XSD files from AFRL JBI Platform V1.0.1. Please see section 6.3 (Databases and Repositories) and Appendix F (Migrating from Platform Services 1.0.1) for more information. MDR features include:

- Storing metadata schema in XSD format for all information objects that will be published.
- Implementing the common baseline of metadata schema.
- Rudimentary support for recognizing specific metadata fragments.

3.5 Security Infrastructure (Version 1.1)

The Security Database contains the JAAS-based principals, roles and role mappings that are used to control access to web services, core J2EE components and JMS servers. The JAAS

administrator manages this repository using only a browser to map the individual operational roles to system privileges. An administrator can create additional users and roles. Features of the security infrastructure include:

- A security repository that maintains the Information Management Staff and user/client application profiles.
- A set of J2EE components that comprise the bulk of the security infrastructure by providing JBI authentication, de-authentication and access control support to the Common API.

3.6 Information Management (IM) Tools (Version 1.1)

A Tomcat servlet and JSP engine is configured to run inside the JBoss Java virtual machine (JVM) and support Secure Sockets Layer (SSL) connections with browser clients. The provided JBoss packaged with the platform includes the integrated Tomcat module and only requires minor modifications described in Section 6. For now, the JBoss/Tomcat bundle will provide the necessary web services, which includes: administration of the MDR and Java Authentication and Authorization Services (JAAS) principals and roles.

- Administrative support for the MDR, J2EE console, and Security Database.
- Automated management of JMS Topics during the creation and deletion of metadata schemas.
- Administrative support to delete ranges of archived information objects.

4. AFRL JBI Version 1.1 Architecture

Figure 1.0 illustrates the Version 1.1 architecture for the JBI platform. Each of the sub-sections below describes the implementation of the architecture in greater detail than Section 3.

Current Prototype Architecture

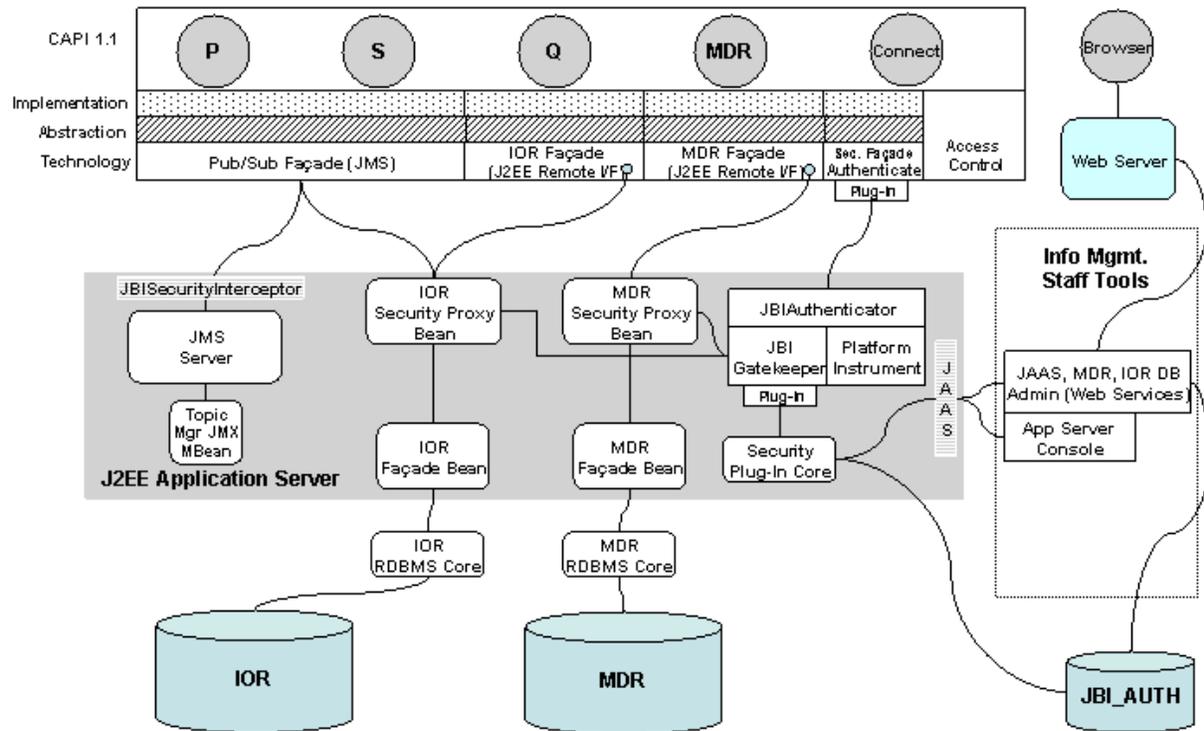


Figure 1 – AFRL JBI Version 1.1 Architecture

4.1 JBI Common API (CAPI) (Version 1.0)

The existence of the Common API is thanks to the work done by the Infospherics.org group – a collection of rogue scientists and engineers plucked from government, industry and academia. AFRL’s contribution to Infospherics.org includes Steering Committee participation, design and architecture recommendations, programming language mappings, documentation, and actual prototype implementations.

4.1.1 Information Object Specification for AFRL JBI

The information object (IO) is the unit of information management within the JBI. Information objects are created by publishers (clients) and disseminated to subscribers. The IO comprises two parts, the metadata and the payload. The payload is the information that is being distributed and the metadata is the “information about the information” that uniquely identifies the payload and allows for the matching of the information object to existing subscriptions and future queries.

4.1.2 Publish and Subscribe

JBI CAPI clients are essentially applications that publish and subscribe to information objects using JMS publish and subscribe mechanisms. Publishing clients assign values to metadata

specified in XML Schema before writing an information object to a publishing sequence. Likewise, subscribing clients specify certain values they are interested in by providing XPath expressions. In order for JMS technologies to be used at the implementation layer, conversion to and from XPath to SQL 92 is required.

4.1.2.1 XPath to SQL-92 Conversion

JMS message selectors are based on a subset of SQL-92 conditional expression syntax and are used in the WHERE clause of SQL SELECT statements. Since JBI information objects are published using metadata and client subscriptions/query criteria are specified using XPath expressions, the platform provides an XPath to SQL conversion utility in the client's address space that essentially flattens hierarchical metadata so they can be accurately evaluated as message selectors on the JMS server. For a complete description of the algorithmic conversion and pseudo code, reference Appendix D.

4.1.2.2 Archive

JBI V1.1 clients can independently determine whether information objects should be archived for query retrieval by other clients. If archiving is desired, another call will be made to the platform, this time passing the information object to an interface that will permanently store the object in the IOR.

4.1.3 Query

JBI CAPI clients that query for information objects are not JMS clients - they use a different method to retrieve information objects. The retrieval path requires that clients access J2EE components running within the Application Server, and not the JMS server directly. Querying clients submit XPath expressions against the IOR, and the underlying implementation converts these expressions to an SQL query which is applied to the repository.

Upon first execution of the Query feature, the clients will receive a fixed array of information objects in a *result set*. If the number of information objects returned is less than the maximum size of the array, the client has received all it asked for. If it received exactly the maximum number of information objects, chances are, there are more information objects waiting for the client. A *getNext* feature allows the client to retrieve all of the information objects in an incremental fashion over time.

4.1.4 MDR Interface

Version 1.0 of the Common API provides access to the MDR, including the ability to retrieve all defined information objects and add new ones. Current support at the CAPI level only allows for the definition of information objects based on type, version and XSD for the metadata. The Information Management Staff toolkit provides additional administrative support to manipulate the MDR contents, including basic validation of the metadata.

4.1.5 Authentication

Authentication is implemented in this release of the platform services for all client operations, including authentication on JBI connection and de-authentication on JBI disconnection. This

feature is modular enough to support the implementation of multiple technologies via *plug-in* capabilities, including the possibility of the following implementations:

- *Dummy Authentication*. This feature could essentially allow any client to authenticate upon connection to the platform. This implementation is not included on the CD but could be implemented very easily.
- *Java Authentication and Authorization Services (JAAS)*. Many J2EE Application Servers support JAAS, and many low-level classes and methods developed by AFRL can be reused because they are build on top of JAAS capabilities. There is an enhanced JAAS version on the CD that initially uses a JAAS Subject for authentication, but later uses a JBI gatekeeper implementation to control access.
- *Secure Remote Password (SRP) Protocol*. This protocol is a public key exchange handshake developed by Thomas Wu of Stanford. It is suitable for negotiating secure connections using a user-supplied password, while eliminating the security problems traditionally associated with passwords. This method maintains verifier stores in the Application Server that can be used to quickly and efficiently manage dynamic profiles. Unfortunately, this version of the software does not include an SRP implementation – the development team needed additional development time and testing. Instead, the spirit of encrypting information via SRP was kept alive in the current implementation that encrypts information flow using SSL.

Note: The implementation that was ultimately chosen is a beefed up JAAS implementation using SSL encryption, a JBI Security Interceptor for JMS access control, Entity Bean caching and a modified database login module to manage user profiles in the JBI_AUTH database. A follow-up white paper will be provided upon request that describes this architecture in more detail.

The authentication and de-authentication implementations are very pluggable, isolating the implementation layers from the Common API. Any attempts to replace the security infrastructure should begin at this layer, particularly before designing an alternate access control layer.

4.1.6 Access Control

Once users (client applications) are authenticated, access to all J2EE resources are controlled by components that reside within the J2EE Application Server, including the JMS server. The security infrastructure takes advantage of standard J2EE declarative security that controls access to EJB components using JAAS underpinnings, but adds quite a few unique features, such as:

- Internal controls that maintain connection identifiers to prevent rogue applications from *spoofing* or posing as legitimate JBI clients without properly authenticating themselves.
- A security *triple* that controls access to all information objects based on the operation performed on the object and the client application's operational role. For a more detailed description of this feature and a clarification of the term *role*, see Section 8.1.

- Dynamic role creation and revocation by IM Staff during run-time as clients are connected to the JBI and dynamically creating and deleting publisher, subscriber and query sequences.
- A security interceptor that uniquely controls access to the JMS server that is based on the JBoss security interceptor model.
- The use of SSL-enabled connections between Common API clients and the platform to protect the transmission of sensitive credentials.

This implementation of access control is partly tied to the JBoss platform and would have to be removed *en masse* and replaced with an entirely new one if it doesn't meet particular security needs. However, there are certainly components or lessons learned that could be reused during the design and development of an alternate security infrastructure.

4.2 J2EE Application Server

Version 1.1 uses the JBoss 3.2.1 J2EE Application Server. Below are some notable things to know about our current implementation of J2EE components.

4.2.1 *Facades/Interfaces/Beans*

J2EE client interfaces are normally implemented using remote calls to the Application Server using JNDI or RMI/IIOP using SSL for encryption. However, our implementation provides another level of abstraction, (i.e., *interface*) for hiding the underlying implementation details from CAPI clients. While not explicitly required in alternate configurations, the facades/interfaces implemented as EJB Session Beans allow different approaches to be swapped in with minimal impact.

4.2.2 *Connection Pooling*

With pooling, connections to the datasource itself are not created and destroyed at the whim of the calling code. Instead datasource connections are opened and kept on reserve in the pool. Calling code asks the connection pool for a connection when needed and places the connection back in the pool when finished. Connection Pooling allows more efficient access to datasources by allowing the administrator to configure the number of connections allowed, connection time out intervals, and number of connections to keep in the pool during the lifetime of the application. These strengths, however, come at the cost of flexibility at the JDBC level as the implementation strives to remain independent of specific vendor tools and technologies.

4.2.3 *Security Domains*

Access control to platform services will be implemented solely using one J2EE Application Server vendor's solution. Although based on standard J2EE role-based authentication and access control, this particular vendor allows various plug-ins to be utilized and security domains created to control access to resources.

There are two security domains that are created on server startup in Version 1.1 and supported by a database login module. Information Management staff profiles are managed within a domain that accesses information in a Security Database, while user/client profiles will be managed

exclusively within a separate domain supported by the same Security Database. The two different domains are required because the IM staff's roles and privileges are modeled after a more traditional roles-based architecture, whereas, JBI users/clients use a security triple of (*role, JBI operation, information object type/version*) to control access to system services.

Actually, a third security domain is configured and maintained, but is used exclusively by the J2EE Java Management Extension (JMX) console.

4.2.4 Security Proxies

For the MDR and IOR façade beans security proxies have been implemented which house all security related logic. Role checks, runtime role granting and generation are performed here. Upon successful authorization the proxies forward valid requests (query, archive, etc) to their respective façade bean implementation. This layer of abstraction ensures that alternate façade implementations (i.e. Oracle vs MySQL) at the server level can be expected to have the same security logic performed without needing to duplicate the code in each replacement façade.

4.2.5 J2EE Access Control

Once a Security Domain is defined, individual EJB components are deployed to the J2EE Application Server within a domain. J2EE role-based access can be defined in the standard deployment descriptors, but, in addition, can be subject to profiles defined in the plug-ins deployed in that domain. Thus, JBoss security interceptors (both standard and JBI-developed) are invoked during run-time that can prevent unauthorized access to platform components.

4.2.6 Application Server Access Control for JMS

Access to the JMS Server, JBossMQ, is controlled using an AFRL-specific security interceptor within the same security domain as the Façade beans access.

4.3 Information Object Repository (IOR)

The IOR is the “persistence” of JBI. Clients are able to archive information objects into the IOR for querying at a later time. It consists of a client side façade, server side security proxy, server side façade (Stateless Session Bean) and core classes implementing raw technologies.

The core of the IOR is implemented using standard relational database strategies. The 1.1 IOR implementation is offered in two flavors: MySQL Relational (requiring MySQL 4.0.x) and Oracle Relational (Requiring Oracle 9.2.x.x). Performance is comparable between technologies and query performance is greatly improved over the 1.0.1 IOR implementation. From a usage and functionality standpoint there have been no significant changes to the implementation from 1.0.1 -> 1.1.

4.3.1 Archive

The archive feature of the IOR is implemented by a set of relational tables generated at runtime when a new schema (InfoObjectDescriptor) is registered with the Metadata Repository. At registration the xml schema is evaluated and “flattened” into a relational table structure which is then used to persist the metadata. The payload itself is stored as a BLOB.

It is worth noting that due to the new relational table structures, archive performance has been greatly improved in 1.1.

4.3.2 Query

Query is implemented by converting the provided XPath expression into an equivalent SQL query over the flattened relational structure that contains the persistent Information Objects (please see Appendix D). Query performance has been exponentially improved in 1.1.

The following table describes the supported XPath constructs (through the common API) during query operation.

XPath Construct	Xpath Description
"/"	Denotes the root of the tree in an XPath expression. For example, /PO refers to the child of the root node whose name is "PO".
"/"	Also used as a path separator to identify the children node of any given node. For example, /PO/PNAME identifies the purchase order name element, a child of the root element.
"//"	Used to identify all descendants of the current node. For example, PO//ZIP matches any zip code element under the "PO" element.
"*"	Used as a wildcard to match any child node. For example, /PO/*/STREET matches any street element that is a grandchild of the "PO" element.
[]	Used to denote predicate expressions. XPath supports a rich list of binary operators such as OR, AND, and NOT. For example, /PO[PONO=20 and PNAME="PO_2"]/SHIPADDR select out the shipping address element of all purchase orders whose purchase order number is 20 and whose purchase order name is "PO_2". [] is also used for denoting an index into a list. For example, /PO/PONO[2] identifies the second purchase order number element under the "PO" root element.

In addition, functions used to retrieve data supports the following: string and numerical equality; range predicates; XPath functions, spaces, namespaces, value case sensitivity, entity handling, parent-ancestor and sibling axes, attribute searching under wildcards, and uses XML schema or DTD information.

IOR Administration does include a delete function that will *clean out* unwanted information objects. This function is available from the IM staff web services and allows a user to specify information object type, version, and number of newest or oldest objects to be deleted.

4.4 Metadata Repository (MDR)

The Metadata Repository handles the storage and retrieval of schemas. It consists of a client side façade, server side security proxy, server side façade (Stateless Session Bean) and core classes implementing raw technologies (Oracle and MySQL).

4.4.1 Information Object Schemas

Since the information object types will be using the same schema, the object type, which is a metadata element, implicitly defines the information being transferred and will be the major

criteria to match publishers and subscribers. The MDR stores schemas as XSD that gets stored directly in database tables.

4.4.2 Interaction with JMS Topic Manager

Every time a new information object and its corresponding metadata schema are entered into the MDR, a JMS topic will be created behind the scenes to support the transport of these message types within the JMS server. Currently, there is a JMX Mbean that controls the creation and deletion of topics that can be accessed via the J2EE JMX console.

4.5 Security Infrastructure

4.5.1 Security Database for IM Staff and Client Applications

The configuration of the security database follows the scheme designed by JBoss that utilizes a Database Server Login Module for authentication and authorization. However, we have modified the database schema to include three tables instead of the two as defined by JBoss and JAAS. The security database is named JBI_AUTH and contains three tables named Principals, Roles, and Principal_Role_Map. These entities may be constructed using the createJBI_AUTH.sql script file contained in this distribution. The Security Database is implemented using an Oracle 9i or MySQL RDBMS and stores encrypted passwords using an MD5 hashing algorithm.

The authentication and access control features are discussed in Sections 4.1.5 and 4.1.6, respectively, from a Common API perspective. This section will describe these features from a component perspective in an architectural framework. Figure 2 is a representation of the underlying infrastructure that is discussed in the next two sub-sections.

Current Prototype Architecture

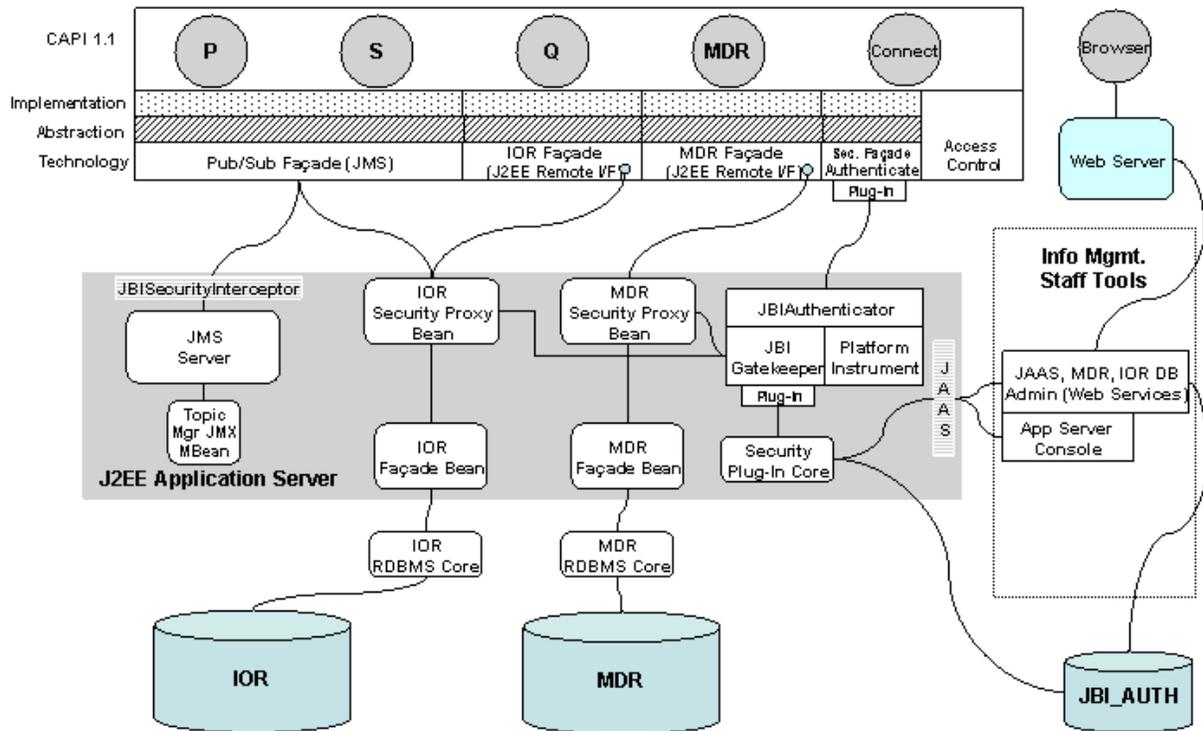


Figure 2: Detailed View of the JBI Security Infrastructure

4.5.2 Authentication

Authentication is essentially implemented using a combination of JAAS, SSL encryption and EJB session bean technologies at the platform to provide a plug-in capability at the CAPI layer. Once a user is authenticated, a JAAS subject is created, its values gleaned and stored as a list of roles in the JBI Gatekeeper Entity Bean. This bean persists client connection identifiers, user names, and their roles and provides an *isCallerInRole* method that controls access to a particular component within the platform. The Platform Instrument bean returns the list of currently authenticated users connected to an instance of the platform and is used to quickly obtain relevant information on the clients currently performing operations on the platform. This instrumentation feature set will be expanded in future in-house releases and will be made available to the IM Staff toolset.

4.5.3 Access Control

Access control procedures are slightly different for normal EJB access versus JMS server interaction. The latter has its own unique JBI Security Interceptor that is similar to the JBoss Security Interceptor that is used as is within the Application Server. Access control is essentially implemented as a check within the JBI Gatekeeper to ensure a user is in a particular role before trying to access a platform service component. For example, a CAPI *publish* operation is

ultimately mapped to a JMS publish operation on a topic session and a CAPI *query* operation will make a call to an IOR Façade Bean method. The check to make sure the client is in role governs access to these methods/services during run-time operation.

4.6 Information Management (IM) Tools

Several Information Management tools are provided to allow for easier handling of metadata and information objects. These tools provide basic functionality needed to prepare and maintain the Metadata Repository and Information Object repository for general use.

4.6.1 J2EE Application Server Console

JBoss releases already have a pre-installed JMX console that can be used to manage Application Server resources. We have taken this application and deployed it within its own Security Domain, requiring authentication and authorization before accessing its user interface. The user name and password are the same for accessing standard administration screens.

4.6.2 IM Staff Web Services

The Security administrative functions allow for the creation and removal of users, as well as the altering of user passwords and roles. The current roles apply to both Information Management staff profiles, as well as Common API Client Profiles.

For a detailed walk through of the Security Administration Tool See Section 8.1 of this document.

The MDR administrative functions will block insertion of ill-formed XSD documents into the MDR, but will not validate XSDs at this time. It will also allow an administrator to update, delete, and view the information object schemas.

For a detailed walk through of the MDR Administration Tool See Section 8.2 of this document.

The IOR administrative functions allow a staff member to remove information objects by specifying information object type, version and the first/last number of archived information objects. This feature can be useful for removing obsolete information objects and allowing for more optimal query performance on a smaller set of archived records.

For a detailed walk through of the IOR Administration Tool See Section 8.3 of this document.

5. Distribution and Disclaimer Statement

The AFRL JBI Platform Services Version 1.1 software is provided "as is". The Government is not liable or responsible for any loss that may occur due to use of the AFRL JBI Platform Services Version 1.1 software, or its respective clients.

The Government is not responsible for updating or maintaining this software release.

6. Installation Instructions

The installation process consists of downloading, configuring, acquiring and executing various binary distributions, README files, open source projects and COTS packages. This section describes the various sub-components that are required to be in place in the near-term, and is broken down into sections that refer to those components: the JBI Common API, Web Services, Databases and Repositories and the J2EE Application Server.

The platform service components should reside on at least two workstations when installed. One workstation should be running a single Oracle 9i or MySQL instance and the other should contain the Common API core classes, a Web Server and the J2EE Application Server.

6.1 Installing the JBI Common API Version 1.0

The server side portion of the Common API gets installed during the normal installation of the platform services. Each JBI client (client machine) should take care of installing the client side libraries to synchronize the two. The /components/capi/links directory contains a link to download an evolving JBI Common API specification, or you can just point your browser as mentioned in the next section.

6.1.1 Download the JBI Common API Version 1.0 Specification and JavaDocs

Point your browser to: <http://www.infospherics.org/api/> and click on the JBI Common API link.

6.2 Web Services

By installing JBoss/Tomcat and deploying the proper archive file, the web services will be configured properly for this release.

6.3 Databases and Repositories

The latest release of Platform Services now supports a MySQL database as the underlying repository. In addition, the Oracle repository structure has changed to provide much better performance.

If you need to migrate your schemas/users from version 1.0.1, please refer to Appendix F after completing this section.

6.3.1 Using Oracle as a Repository

This section describes the installation and configuration of an Oracle-based JBI repository.

6.3.1.1 Environment Setup

6.3.1.1.1 Oracle Service Name setup

You must set the TWO_TASK or LOCAL environment variable to the service name that was created during the Oracle 9iR2 installation. If you are unsure of the service name, please see the tnsnames.ora file located in the <ORACLE_HOME>/network/admin directory.

```
Windows example (from a command prompt)
set LOCAL=<service name>
```

Unix (Bourne shell) example:
TWO_TASK=<service name>
Export TWO_TASK

Unix (C shell) example:
setenv TWO_TASK <service name>

It is important to note that these variables are only valid over the lifetime of the session. To make them valid over future sessions, add the LOCAL environment variable under system variables (Windows) or add the TWO_TASK environment variable to the *.cshrc* file (Unix).

6.3.1.2 Installing the AFRL JBI Oracle Build and Populate Scripts

IMPORTANT! IF YOU ARE UPGRADING AN EXISTING PLATFORM SERVER WHICH ALREADY HAS AN ORACLE REPOSITORY, DO NOT COMPLETE THE REST OF THIS SECTION. PLEASE GO TO APPENDIX F. PROCEEDING WITH THIS SECTION WILL DESTROY EXISTING USERS/ROLES.

The script **create_jbi_repositories1.1.sql** is responsible for creating the JBI repositories as well as data needed to initially standup Platform Services version 1.1. During script execution, a log file **create_jbi_repositories1.1.log** is created, which contains the script output. The log file is located in the same directory as the script file. After the script has completed, the log should be reviewed for errors. It is important to note that the following errors are likely to occur on a clean install and should be ignored:

ORA-00942: table or view does not exist

ORA-02289: sequence does not exist

ORA-04080: trigger does not exist

The previous release of Platform Services required several database scripts to setup the Repository. This Version 1.1 requires a single script for Repository installation.

Open a command prompt or terminal window and navigate to the */server-side/sql-scripts/* directory on the CD.

For Unix:

Extract **sql-scripts-v1.1-nix.tgz** to a temporary directory. In the temporary directory cd to the *sqlScripts/oracle* directory. **create_jbi_repositories1.1.sql** should reside in this directory.

For Windows:

Extract **sql-scripts-v1.1-win.zip** to a temporary directory. In the temporary directory cd to the *sqlScripts/oracle* directory. **create_jbi_repositories1.1.sql** should reside in this directory.

Issue the following command:

```
>sqlplus system/<system password> @ create_jbi_repositories1.1.sql
```

Where <system password> is the Oracle system password assigned by your DBA

The script will ask you to assign the following passwords:

- JBI Auth password (JBI_AUTH)
- JBI Repository password (JBI_REPOSITORY)
- JMS password (JMS)
- CMP password (CMP)

For a standard JBI configuration, use the passwords indicated in the parentheses. *If you choose to use passwords other than the default, you MUST configure the JBoss Application Server configuration files appropriately. See Appendix B for details.*

6.3.2 Using MySQL as a Repository

This section describes the configuration and setup of an MySQL 4.0.x (<http://www.mysql.com>) instance that will contain the Metadata Repository, Information Object Repository and the Security Database. This section provides information for running the scripts that build the MDR, IOR, SECURITY, CMP, and JMS databases. The scripts are responsible for creating the schemas, tables, and populating the table with basic information. It is important to note that re-running the scripts will DELETE ALL of data and revert the database to the original defaults.

Please note that SQL scripts can also be executed in the control center that is freely available from www.mysql.com

6.3.2.1 Starting and Configuring an MySQL Server Instance

Reference the Manual documentation (MYSQL-HOME/Docs) for information on configuring and MySQL Server Instance.

```
MYSQL-HOME/bin/mysqld -O lower_case_table_names=0 max_allowed_packet=16M
```

This starts MySQL database with consistent table names and sets the memory parameter that controls the maximum allowed packet size in megabytes.

6.3.3 Installing the AFRL JBI MySQL Build and Populate Scripts

6.3.3.1 Populating MDR, IOR, and Security databases

Open a command prompt or terminal window and navigate to the `/server-side/sql-scripts/` directory on the CD.

For Unix:

Extract **sql-scripts-v1.1-nix.tgz** to a temporary directory. In the temporary directory cd to the *sqlScripts/mysql* directory. **create_jbi_repositories1.1.sql** should reside in this directory.

For Windows:

Extract **sql-scripts-v1.1-win.zip** to a temporary directory. In the temporary directory cd to the *sqlScripts/mysql* directory. **create_jbi_repositories1.1.sql** should reside in this directory.

Open a command prompt or terminal window and go to the installed MySQL directory and issue the following command:

```
shell> mysql -u root
```

Now to import MySQL scripts, issue the following command:

```
mysql>  
source DIR_YOU_EXTRACTED_TO/sqlScripts/mysql/create_jbi_repositories1.1.sql
```

This should create and populate databases with appropriate data. Please note that you can use the control center GUI instead of the command line. The control center is a very intuitive tool, and definitely a very useful tool for the management.

Please note that additional schemas must be inserted through the Information Management tools (via Web), since the insertion will cause the automatic creation of proper tables (based on the supplied schema) in the IOR.

Note: The MySQL tutorial (www.mysql.com) provides solutions for most of the problems encountered during installation and operation.

6.4 J2EE Application Server

The JBoss Application Server will be the J2EE container we use to conduct much of the *heavy lifting* on the platform, including security infrastructure, messaging, service naming and JMS topic management.

6.4.1 Download the JBoss 3.2.1 J2EE Application Server

Before installation make sure the JAVA_HOME environment variable exists and references a valid 1.4.1 or higher JDK. Make sure the JAVA_OPTS environment variable exists as well. Initially it should be set to -mx512m, which essentially sets the default heap size for java to be 512Megs of ram. This helps to prevent out of memory errors during heavy usage of JMS message traffic, queries and archives.

6.4.2 *Installing the JBoss 3.2.1 Application Server with the Tomcat/Catalina 4.1.24 Servlet Engine*

Windows Users:

Using your favorite zip utility unzip
`PLATFORM_CD_ROOT/server-side/jbi-server-v1.1-win.zip`
to the desired location.

Unix/Linux Users:

Using GNU tar extract
`PLATFORM_CD_ROOT/server-side/jbi-server-v1.1-nix.tgz`
to the desired location.

Ex. **tar -zxvf `jbi-server-v1.1-nix.tgz` /opt**

6.4.3 *Configuring the JBoss Application Server to run AFRL JBI Components*

JBoss 3.2.1 with embedded Tomcat 4.1.24 is used as the J2EE container and application server. The out of the box deployment configuration is a modified default deployment, high level differences include:

- Standard http disabled
- SSL enabled (https)
- Secured jmx-console
- MySQL & Oracle based CMP for entity beans

6.4.4 *Configuring JBoss for your databases*

cd to the `YOUR_JBOSS_ROOT/server/jbi/deploy` directory.

Note: JBoss monitors the deploy directory for changes and attempts to deploy any files it finds in this location. Be sure the only files in this directory are the ones required for deployment. Placing temporary or duplicate files in this directory can cause problems during startup and runtime.

6.4.4.1 For Oracle Users The files:

`mdrior-oracle-service.xml`
`security-oracle-service.xml`

contain the information pertaining to the databases you setup in a prior section.

To change the specifics for connecting to a particular database open the corresponding xml and search for the following text:

```
jdbc:oracle:thin:@YOUR-SERVER-IP-HERE:1521:YOUR-SID-HERE
```

Replace `YOUR-SERVER-IP-HERE` with the ip address of your Oracle database.
Replace `YOUR-SID-HERE` with the SID of your Oracle database.

6.4.4.2 For MySQL Users The Files:

```
mdr-mysql-service.xml  
ior-mysql-service.xml  
security-mysql-service.xml  
cmp-mysql-service.xml  
jms-mysql-service.xml
```

contain the information pertaining to the databases you setup in a prior section.

To change the specifics for connecting to a particular database open the corresponding xml and search for the following text:

```
jdbc:mysql://localhost:3306/CMP
```

If your MySQL database server is not running on the same machine as your JBoss server replace `localhost` with the ip address of your MySQL Server.

6.4.5 *Launching the JBoss Application Server*

To launch the JBoss application server using the configuration we have created cd to the `JBOSS_ROOT/bin` directory and enter the following at the command line:

For Windows:

```
run -c jbi_mysql  
or  
run -c jbi_oracle
```

Depending upon which database you to used for your installation.

For *nix:

```
sh ./run.sh -c jbi_mysql  
or  
sh ./run.sh -c jbi_oracle
```

You should see deployment messages scrolling on the JBoss 3.2.1 application console. Examine the messages for any erroneous activity. Warnings can be disregarded as long as they are not accompanied by errors. The application server is online and ready to serve requests when a line similar to the following is output to the console or appears in the `server.log` file:

```
12:13:32,046 INFO [Server] JBoss (MX MicroKernel) [3.2.1 Date:200211021607] Started in  
0m:16s:953ms
```

For a listing of normal errors/warnings see Appendix E: Troubleshooting

You are now ready to set up and test the example client applications. For instructions regarding this please read the ClientInfrastructureV1.1 document located in the client-side directory of this CD.

7. Running the Web Service Examples

Recommended Browsers:

- Netscape Navigator 6.0 or higher
- Internet Explorer 6.0 or higher

Older browsers such as Netscape 4.7 may work, but will have problems when trying to display xml content (such as MDR Schemas).

To access the deployed web services in the JBoss/Tomcat bundle, launch your browser of choice and in the Address bar point it to:
<https://hostname:8443/RepositoryTools>

Note: The address listed above IS case sensitive, and the text Hostname should be replaced with the fully qualified hostname of the computer on which you installed the JBI Platform Server Components.

You should be presented with a page identical to the image on the right.



For a default installation a “Super Admin” user account can be used to initially login and configure the platform. The username is **jbiAdmin** and the password is **moniker**.

Note: The user jbiAdmin cannot be deleted and cannot have the Super Admin roles revoked. You can however change the password of jbiAdmin at anytime and it is highly recommended that you do so after your initial setup is complete. For details on administrating users see section 8.1

Once logged in you will be presented with the following screen:

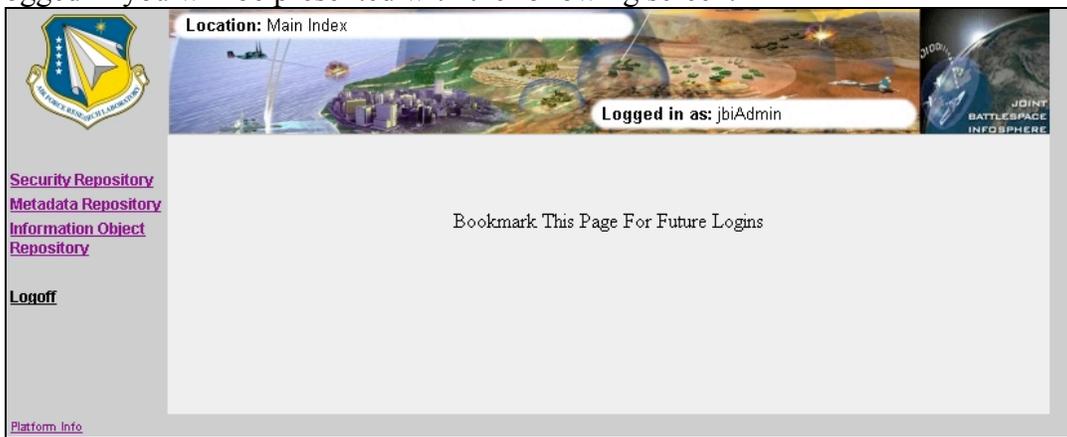
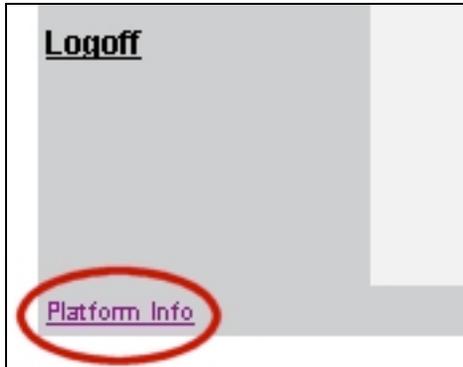


Figure 4: Repository Tools Main Index

Notice the “**Logoff**” link. This link is present in every page of the Repository Tools and should be used when you intend to leave the application to browse outside web pages or use another web based application.

The “**Platform Info**” link is also present at the bottom left of every page of the Repository Tools. This link provides quick access to valuable information about the installed JBI Platform Server side components.



This page is very useful for troubleshooting configuration problems or gathering information for support requests.

JBI Platform Information	
JBI Platform Server Components Version:	1.1
InfoObjectRepository Technology:	MySQL Relational
MetadataRepository Technology:	MySQL Relational
Supported Java SDK Versions:	1.4.1 or higher
Java SDK Version Detected:	1.4.2
Java Home Detected:	c:\Progra~1\j2sdk_nb\j2sdk1.4.2\jre
Application Server:	JBoss 3.2.1/Tomcat 4.1.24
Server Hardware Architecture:	x86
Server OS:	Windows XP
<input type="button" value="Thanks!"/>	

Figure 5: Repository Tools About Dialog

Now that we are familiar with the basics lets explore the other Administrative Tools accessible from the Main Index.

7.1 Security Administration Web Tool: Walk-through



Figure 6: Repository Tools User Administrator

This tool provides an easy to use front end to the Security Repository. Using this tool an administrator can easily add, remove, or update existing users. In addition an administrator can also assign roles to users of the JBI Platform and its web based services. JBI Platform accounts are defined by a set of Users and their roles

7.1.1 Running the Security Application

From the main menu of the web services home page, click in the Security Repository to either list and update existing users or add new ones. Example user accounts and roles are supplied as part of the default installation. If the creation of additional users or role modifications is necessary, bring up the User Update screen as illustrated below.

User accounts are currently comprised of two credentials: user name and password. Roles can be mapped one at a time for each participating user. You can assign multiple roles by CTRL mouse clicking in your browser window, giving users/clients publish, subscribe, query, archive or MDR access for each information object defined in the scenario. If you don't see a particular role, you may have to either add a new schema or package in the MDR.

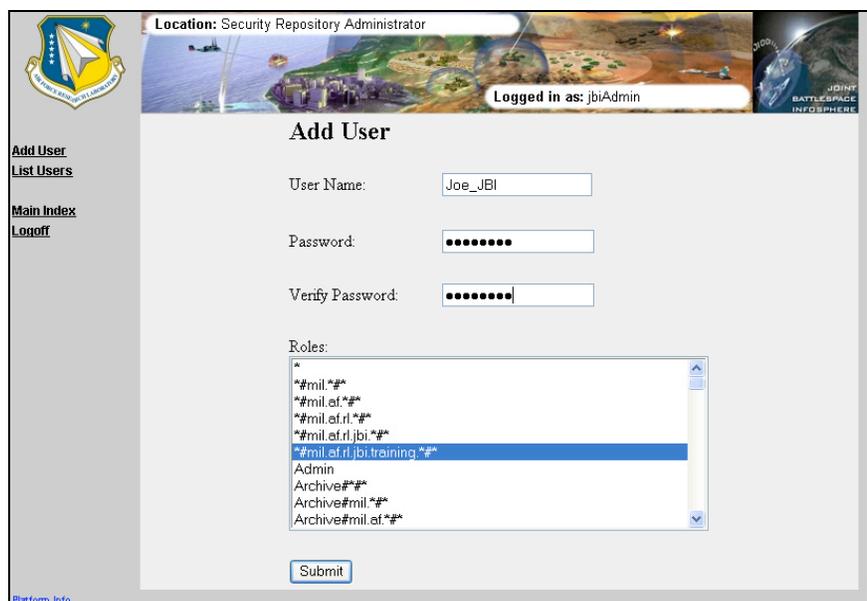


Figure 4: Updating a User Account

It is important to note that adding or revoking access based on user roles has an instantaneous effect on publish, query and MDR actions. Subscriptions may take up to a half a minute or so for the changes to propagate out to the client applications. When all roles are configured properly, the following screen gives a listing of all users in the scenario and their roles at a glance.

User Name	Roles	Update Roles	Delete User
jbiAdmin	*	Update	
jbossAdmin	JBossAdmin	Update	Delete
Joe_Pubber	*#mil.af.rl.jbi.training.*#*	Update	Delete
	IORUser		
	MDRUser		
vincelec	*	Update	Delete

Figure 5: User Accounts Listing

7.1.2 Using wildcards

When assigning roles to users. Explicit roles are no longer needed. For example if you assign your types in the MDR to packages (for more on packages see the next section) you can wildcard over any part of the package. For example if your MDR had 3 types named like the following:

mil.af.rl.jbi.training.at0 1.0

mil.af.rl.jbi.training.basic 1.0

mil.af.rl.jbi.training.xmlxpath 1.0

You could give a user publish privileges to all three by assigning the role:

Publish#mil.af.rl.jbi.training.*#1.0

If you wanted a user to be able to perform any action on all three (pub,sub,query,archive,read,write) you could assign the role:

***#mil.af.rl.jbi.training.*#1.0**

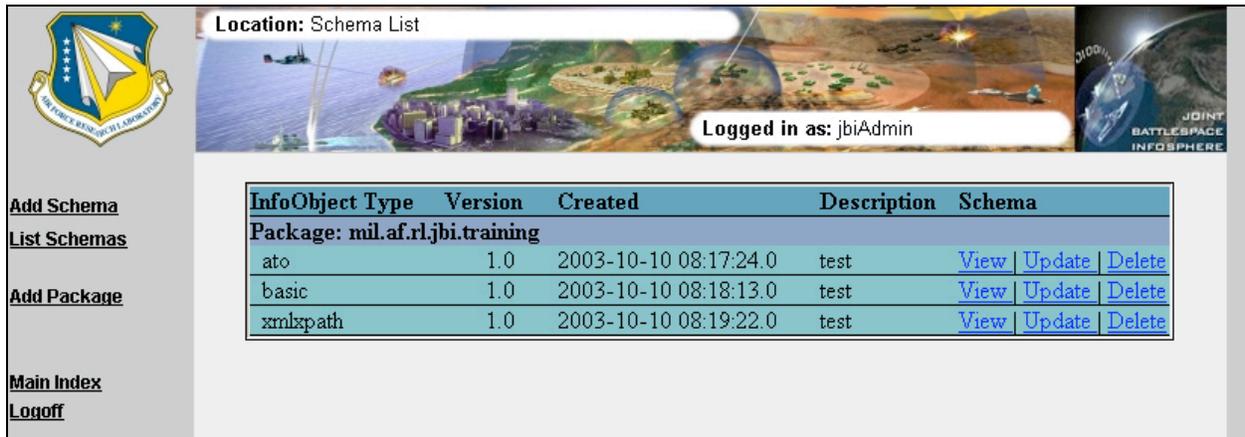
As you can see the usefulness of this is determined mainly by how well you define your packages and how strongly you enforce the naming of your types. See the next section 7.2 for suggestions on how to best use packaging.

7.2 Metadata Repository Administration Web Tool: Walk-through

This tool provides an easy to use front end to the Metadata Repository. Using this tool an administrator can easily add, remove, update, or browse existing schemas.

Note: Users of this tool must have the Admin, MDRAdmin, or MDRUser and MDRRead#SomeType#SomeVersion role. For details on how to assign roles to a user see section 8.1

Upon logging in click the “Metadata Repository” link. The user will be presented with the following screen:



The screenshot shows the Metadata Repository Administration Web Tool interface. At the top left is a logo for the Joint Battlespace Infosphere. The main header area displays "Location: Schema List" and "Logged in as: jbiAdmin". Below the header is a table listing schemas. The table has columns for InfoObject Type, Version, Created, Description, and Schema. The schemas listed are: ato (Version 1.0, Created 2003-10-10 08:17:24.0, Description test), basic (Version 1.0, Created 2003-10-10 08:18:13.0, Description test), and xmlxpath (Version 1.0, Created 2003-10-10 08:19:22.0, Description test). Each row has links for View, Update, and Delete. On the left side of the interface, there are navigation links: Add Schema, List Schemas, Add Package, Main Index, and Logoff.

InfoObject Type	Version	Created	Description	Schema
Package: mil.af.rl.jbi.training				
ato	1.0	2003-10-10 08:17:24.0	test	View Update Delete
basic	1.0	2003-10-10 08:18:13.0	test	View Update Delete
xmlxpath	1.0	2003-10-10 08:19:22.0	test	View Update Delete

Figure 6: Metadata Schema Listing

Here all Schemas currently registered with the Metadata Repository which the user has MDRRead permission for are displayed. To view the content of a Schema (the XSD) Simply Click the “View” link.

To refresh the list click the “List Schemas” link.

To remove a schema permanently from the Metadata Repository click the “Delete” link and choose yes when asked to confirm.

Note: Removing Metadata Repository entries does more than de-register the schema. All Publish/Subscribe/Query/Archive roles associated with the schema are removed from the Security Repository and revoked from all users that were previously assigned them. In addition schemas cannot be “undeleted”.

To add a new Schema click the “Add Schema” link. A screen similar to the following will be displayed:



Figure 7: Metadata Repository Adding a New Schema

Type - The name of the Information Object Type the schema will represent. This field is required. A valid schema name must contain the full package and type name and cannot contain any of the following characters:

% & , ; ? * " ' ~ + = | /

Version - The version of the Information Object Type the schema will represent. This field is required. Version numbers must be entered in the format:

MajorNumber.MinorNumber

Description – A human readable plain text description regarding the Information Object Type the new schema represents. This field is optional.

Schema – The xsd file that describes the Information Object Type. This field is required.

After satisfying all required fields click “submit” If any errors have been detected in your entry (Corrupt Schema file, Invalid Type name, invalid version #, etc.) You will be returned to this screen with the errors displayed. Upon successful registration of the new Type with the Metadata Repository the user will be returned to the “Schema List” screen.

Behind the scenes a new JMS Topic will be created and Publish/Subscribe/Archive/Query roles will be created in the Security Repository for the new Type and Version. Once the relevant client users have been assigned these roles (See section 8.1) they will be able to Publish, Subscribe, Query, and Archive for the newly added Type and Version.

Note: Internet Explorer users may want to visit the troubleshooting section of this document(Appendix E) to work around potential problems with this particular piece of functionality.

7.2.1 Putting types in Packages.

The 1.1 incarnation of the MDR borrows a wonderful concept from java. Types can now belong to packages. The package structure is very simple using periods as a delimiter. By creating types in a package hierarchy the security portion of the Web Tools can really shine.

For example, imagine your JBI has two communities of interest one community is a training community. They will be learning the JBI creating example types, deleting types, publishing example metadata... and as is the case with most newcomers creating havoc as they do so.

Your second community of interest is not novices they are creating real no kidding types with carefully crafted metadata. In the world of 1.0.1 these worlds collided on a regular basis. In 1.1 we can partition these two communities from each other using packages. So instead of having an MDR that looks like this:

MyAbusedTrainingType 1.0
ATO 1.0
MyATO 1.0
WhoseATOIsThis 1.0

Your MDR might look like this:

mil.af.rl.jbi.training.MyAbusedTrainingType 1.0
mil.af.rl.jbi.training.MyATO 1.0
mil.af.rl.jbi.training.WhoseATOIsThis 1.0

mil.af.rl.jbi.projectx.ATO 1.0

Notice now the training folks are creating their types under a different package... In the security tool we can enforce that the training folks can only create types under the **mil.af.rl.jbi.training** package by assigning those users

MDRRead#mil.af.rl.jbi.training.*#*
MDRWrite#mil.af.rl.jbi.training.*#*

Even better if these were the only Read/Write roles our training folks had. They would not even be able to see **mil.af.rl.jbi.projectx.ATO 1.0** in their MDR list!

Packages can be created in one of two ways.

- 1) An admin with sufficient permission can create a type with the new package as part of the types name. The new roles related to the package structure will be created along with all roles in the package hierarchy.
- 2) An admin can click the “Add Package” link and enter the name of the package they wish to create. After adding the package nothing will seem to have changed in the MetadataRepository, but in the security tool the new roles related to the package structure will be created along with all roles in the package hierarchy. This method is useful for when you know what package you want your types to belong to but not necessarily what the types will be. Using the roles created you can give your developers access over the newly created package and they will only be able to create types within that package leaving the rest of the JBI players unscathed.

7.3 Information Object Repository Administration Web Tool: Walk-through

This tool provides an easy to use front end to the Information Object Repository. Using this tool an administrator can easily view Information Object Repository statistics, and remove expired Information Objects. The IORWrite#type#version role is required for users to be able to remove objects from the repository. Users can only remove types for which they have the IORWrite#type#version role.

*Note: Users of this tool must have the *, Admin, IORAdmin, or IORUser and IORRead#SomeType#SomeVersion role. For details on how to assign roles to a user see section 8.1*

8. Appendix A - Installation Requirements

8.1 Server Side Requirements

- Java JDK 1.4.1 or higher
- Oracle9i Release 2 (9.2.0.4)
 - OR
- MySQL 4.0.16

**See the vendor's software documentation for the exact requirements of each of the above mentioned items.

8.2 Client Side Requirements

- **Java JDK 1.4.1 or higher**

Note: JDK 1.4.0 is NOT supported in this release of the software. Also, to run the client application sample programs, you'll need access to several JBoss 3.2.1 client-side libraries (client infrastructure installer is included in the client-side directory on the CD).

9. Appendix B - JBoss Configuration Files

The JBoss Application Server included as part of the 1.1 JBI Platform is custom configured to include such services as JDBC data sources and SSL connections for accessing web services. JDBC allows the server to communicate with databases over a network or on a local machine, while SSL encrypts those communications to ensure their confidentiality. This Appendix describes the changes and additions that have been made to the standard JBoss distribution.

9.1 **JBOSS_ROOT/server/jbi_oracle/conf/login-config.xml** **JBOSS_ROOT/server/jbi_mysql/conf/login-config.xml**

This file contains the configuration for application policies (Security Domains) and login modules.

9.2 **JBOSS_ROOT/server/jbi_oracle/conf/log4j.xml** **JBOSS_ROOT/server/jbi_mysql/conf/log4j.xml**

This file contains the configuration for the built in logging capabilities of jboss and deployed applications. The threshold is configurable and can be any of the following levels:

DEBUG – Everything. No logging is omitted.

JBI_DEBUG – This and all the levels listed below are logged. The above are omitted.

INFO – This and all the levels listed below are logged. The above are omitted.

JBI_INFO – This and all the levels listed below are logged. The above are omitted.

WARN – This and all the levels listed below are logged. The above are omitted.

JBI_WARN – This and all the levels listed below are logged. The above are omitted.

ERROR – This and all the levels listed below are logged. The above are omitted.

JBI_ERROR – This and all the levels listed below are logged. The above are omitted.

FATAL – This and all the levels listed below are logged. The above are omitted.

JBI_FATAL – Only JBI_FATAL output is logged. All other levels are omitted.

OFF – No output is logged.

The console configuration sets the threshold for the output seen when running jboss at the command line. (If running jboss from a shell script on a production server this logging level can be set to OFF for better performance).

The file configuration sets the threshold for the output written to the “JBOSS_ROOT/server/jbi_oracle/log/server.log” file or the “JBOSS_ROOT/server/jbi_mysql/log/server.log” file. Depending on the database technology you are using for your repositories.

Note: More logging means less performance. If you need speed and are running a production JBI lower logging levels are recommended. If you experience problems with the platform that require trouble shooting. It is then that the logging level should be increased.

9.3 JBOSS_ROOT/server/jbi_oracle/conf/server.keystore JBOSS_ROOT/server/jbi_mysql/conf/server.keystore

A generic self signed secure certificate (KeyStore) is provided to allow for the implementation of SSL. It can be replaced with a valid signed certificate provided that the tomcat jboss-service.xml is updated to reference the appropriate KeyStore file with the correct password.

9.4 JBOSS_ROOT/server/jbi_oracle/deploy/mdrrior-oracle-ds.xml JBOSS_ROOT/server/jbi_mysql/deploy/ior-mysql-ds.xml

This descriptor configures a pool of InformationObject Repository database connections within the JBoss application server. Connection specifics such as database location, username, and password are specified here. Connection timeouts, max pool size, and minimum pool size can also be specified.

9.5 JBOSS_ROOT/server/jbi_oracle/deploy/mdrrior-oracle-service.xml JBOSS_ROOT/server/jbi_mysql/deploy/mdr-mysql-service.xml

This descriptor configures a pool of Metadata Repository database connections within the JBoss application server. Connection specifics such as database location, username, and password are specified here. Connection timeouts, max pool size, and minimum pool size can also be specified.

9.6 JBOSS_ROOT/server/jbi_oracle/deploy/cmp-oracle-ds.xml JBOSS_ROOT/server/jbi_mysql/deploy/cmp-mysql-ds.xml

This descriptor configures a pool of database connections within the JBoss application server for use by the CMP engine. All entity beans are persisted in the CMP database. Connection specifics such as database location, username, and password are specified here. Connection timeouts, max pool size, and minimum pool size can also be specified.

9.7 JBOSS_ROOT/server/jbi_oracle/deploy/jms-oracle-ds.xml JBOSS_ROOT/server/jbi_mysql/deploy/jms-mysql-ds.xml

This descriptor configures a pool of database connections within the JBoss application server for use by the JMS engine. JMS messages that are queued for delivery beyond the memory caches upper limit are stored temporarily in the JMS database. Connection specifics such as database location, username, and password are specified here. Connection timeouts, max pool size, and minimum pool size can also be specified.

9.8 JBOSS_ROOT/server/jbi_oracle/deploy/security-oracle-ds.xml JBOSS_ROOT/server/jbi_mysql/deploy/security-mysql-ds.xml

This descriptor configures a pool of Security Repository database connections within the JBoss application server. Connection specifics such as database location, username, and password are specified here. Connection timeouts, max pool size, and minimum pool size can also be specified.

9.9 JBOSS_ROOT/server/jbi_oracle/jbossweb-tomcat.sar/META-INF/jboss-service.xml
JBOSS_ROOT/server/jbi_mysql/jbossweb-tomcat.sar/META-INF/jboss-service.xml

This descriptor configures which connectors are enabled for the embedded Tomcat server. By default only the http connector is enabled. Our updated descriptor disables this and instead enables the https connector. The connector is configured with the path and password to the server.keystore secure certificate file. This file also specifies which directory server logs are written to.

9.10 JBOSS_ROOT/server/jbi_oracle/deploy/jboss-ssl-rmi-service.xml
JBOSS_ROOT/server/jbi_mysql/deploy/jboss-ssl-rmi-service.xml

This descriptor configures ssl encryption for EJB interactions. For an out of the box configuration same keystore used for https is used here. This can also be pointed to a real keystore.

10. Appendix C - Oracle 9.2.x Installation Notes

The following is a sample installation of Oracle 9.2.0.1. Please consult your Oracle documentation for more detailed installation instructions. Although the base installation (9.2.0.1) is the minimal functional version for JBI, *updating to the latest patch is highly recommended*. At the time of this printing, the latest patch upgrades Oracle to version **9.2.0.4**.

Note that the figures in this Appendix depict a Windows XP installation; however, the same screens would appear during a Solaris installation as well.

10.1 Run the 9i Installer for the Oracle 9i Server

10.1.1 Click *Install/De-install Products*

10.1.2 Click *next* on the Welcome screen (Figure C-1)

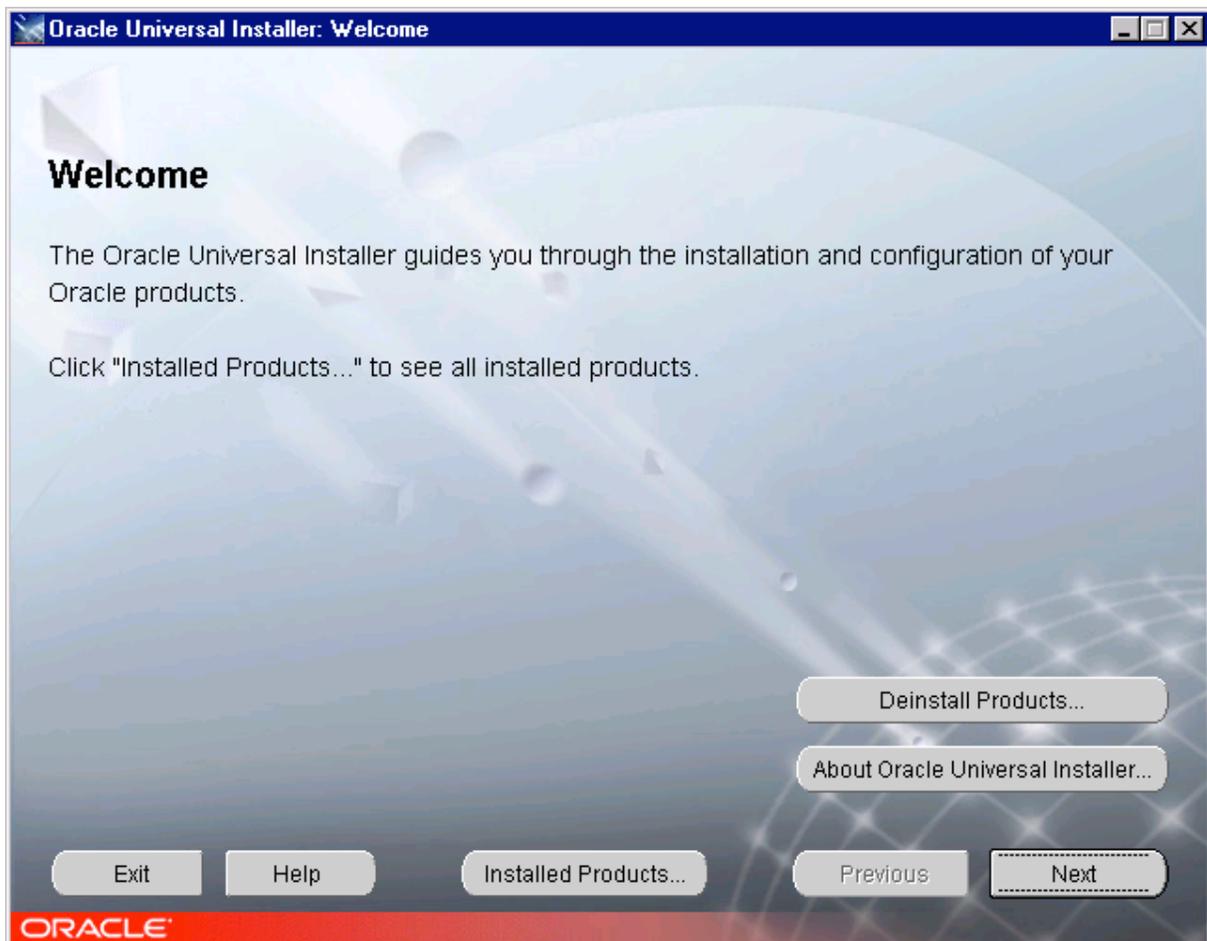


Figure C-1: Welcome screen

10.1.3 File Locations (Figure C-2)

Destination:

Name: **OraHome92**

Path: <pathname>\ora92,

where <pathname> is where you want to install Oracle

Click **next**

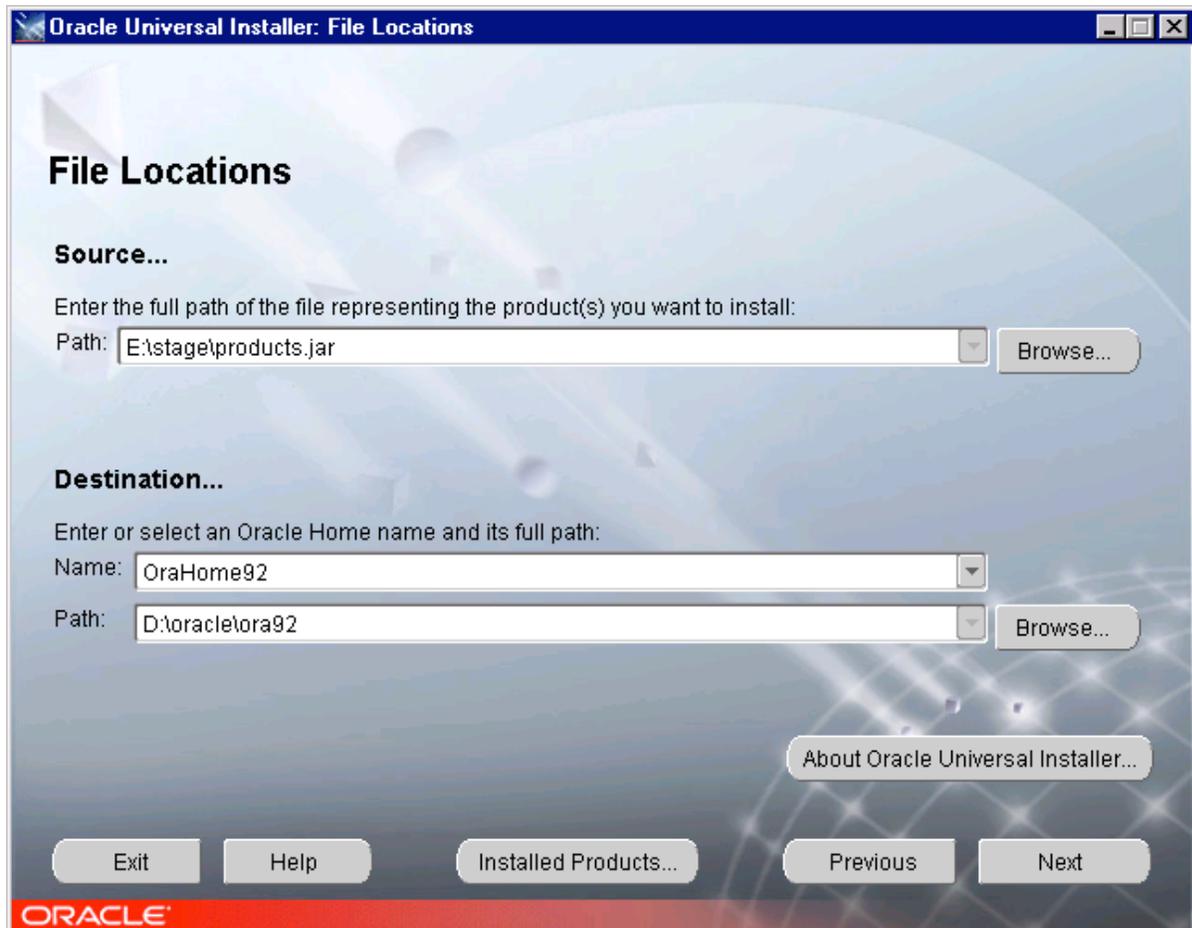


Figure C-2: File Locations

10.1.4 Available Products (Figure C-3)

Select **Oracle9i Database 9.2.0.1.0** (*NOTE*: version may vary)

Click **next**

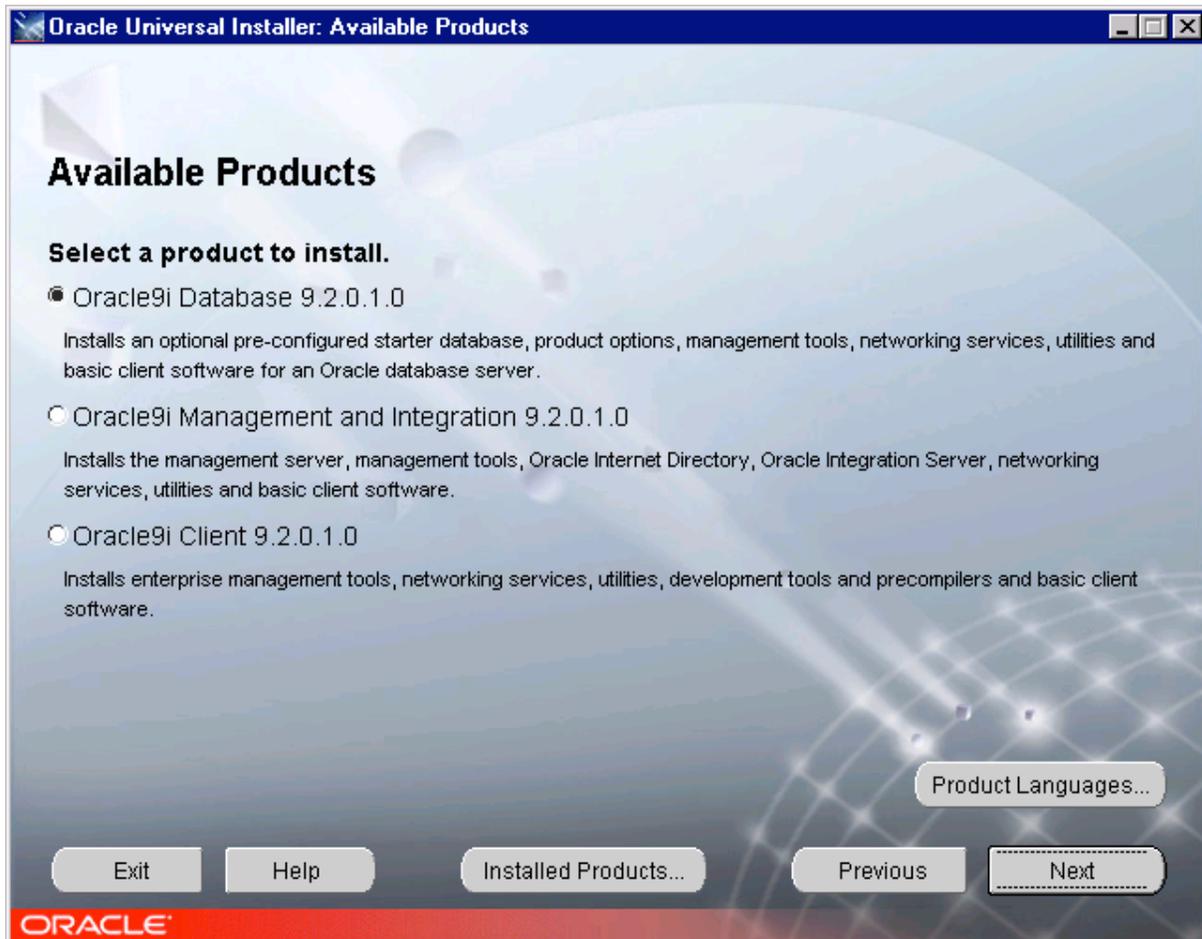


Figure C-3: Available Products

10.1.5 Installation Types (Figure B-4)

Select **Enterprise Edition**

Click **next**

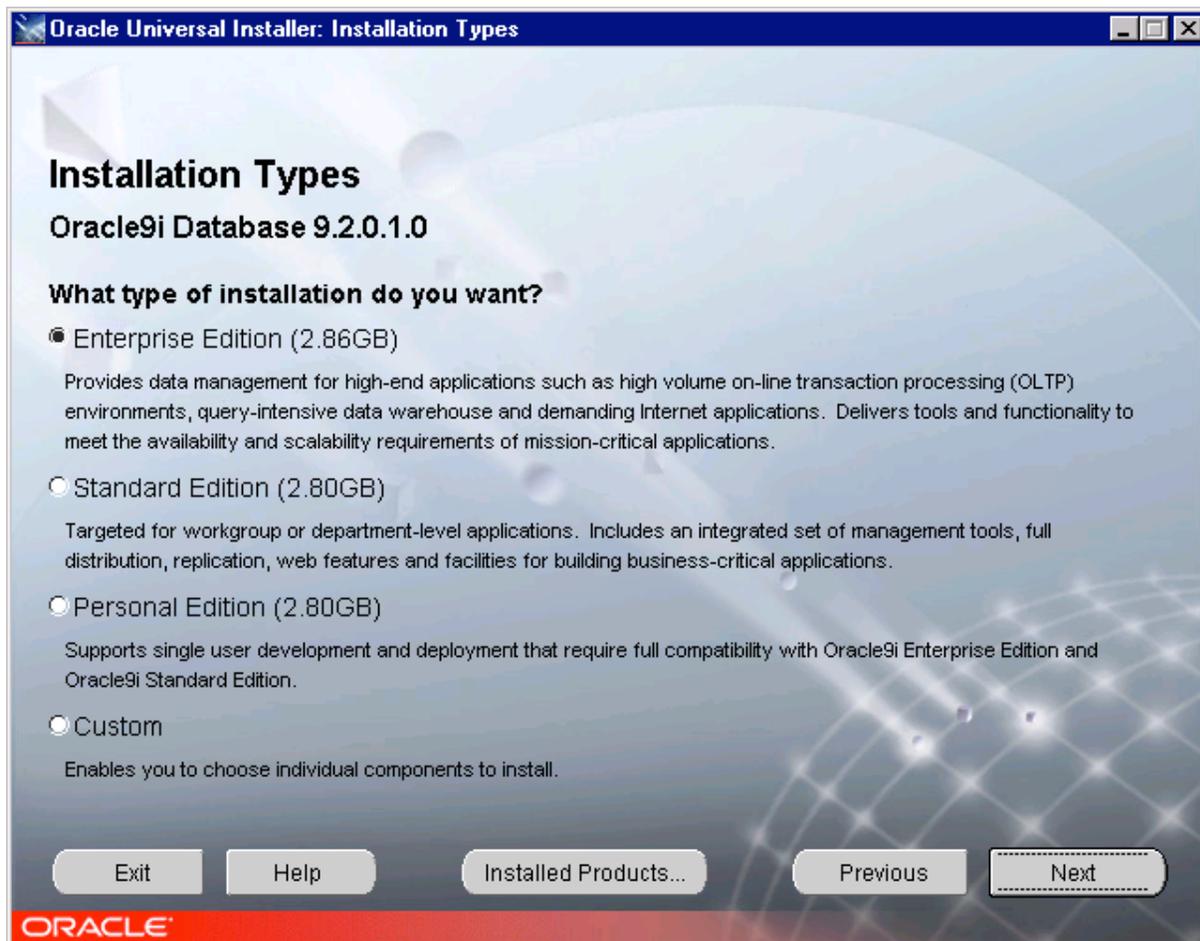


Figure C-4: Installation Types

10.1.6 Database Configuration (Figure C-5)

Select **General Purpose**

Click **next**

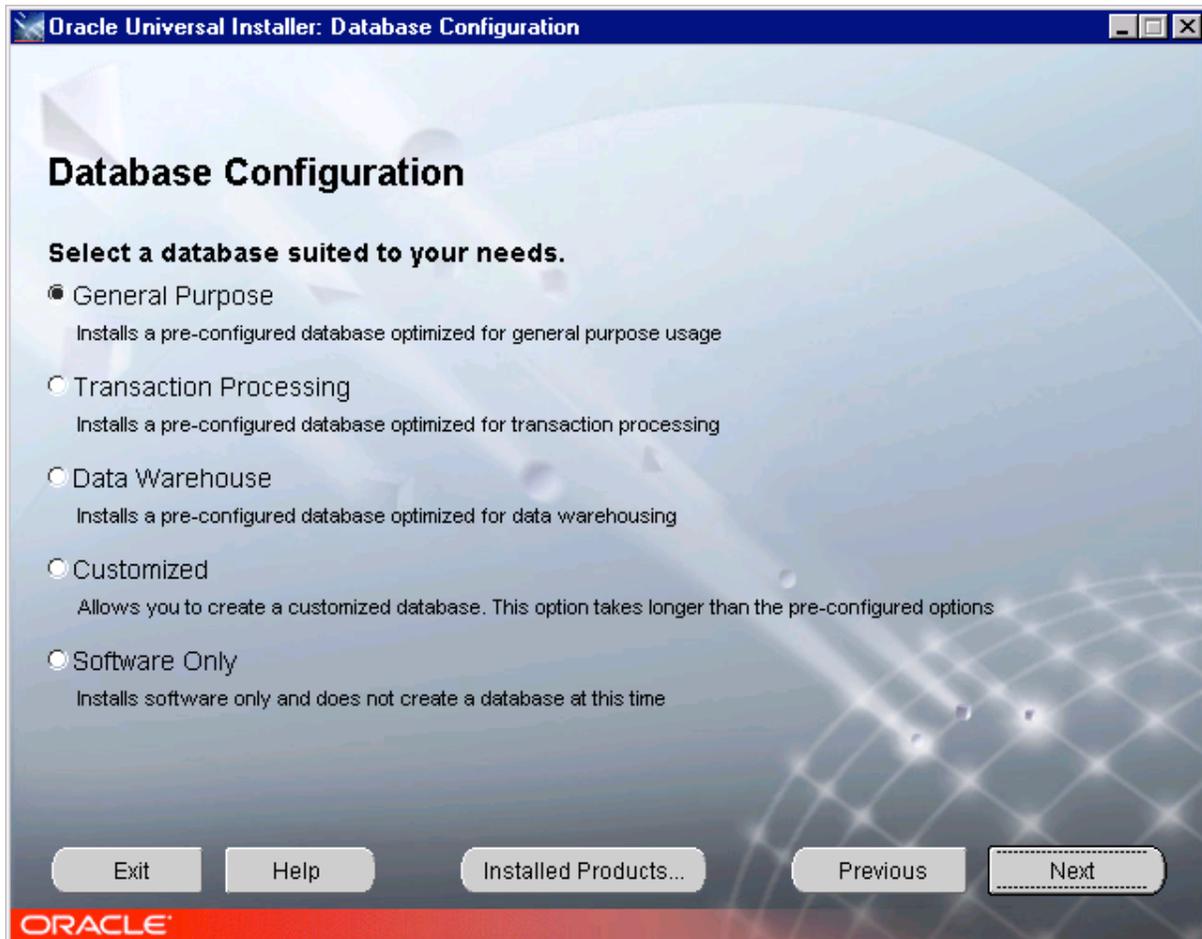


Figure C-5: Database Configuration

10.1.7 Oracle Services for Microsoft Transaction Server (Figure C-6)

Click **next**

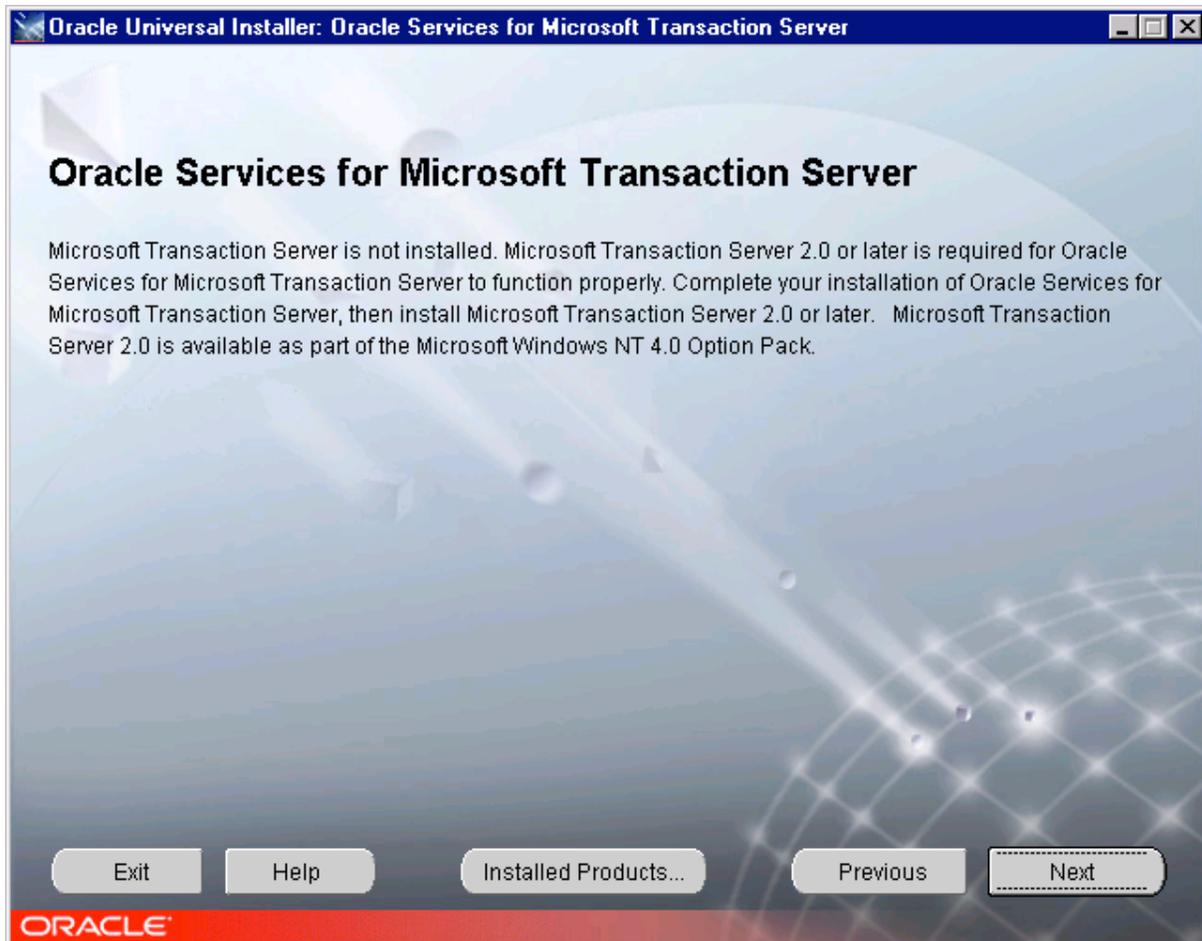


Figure C-6: Oracle Services for Microsoft Transaction Server

10.1.8 Oracle Services for Microsoft Transaction Server

Oracle MTS Recovery Service Configuration (Figure C-7)

Port number: **2030** (default)

Click **next**

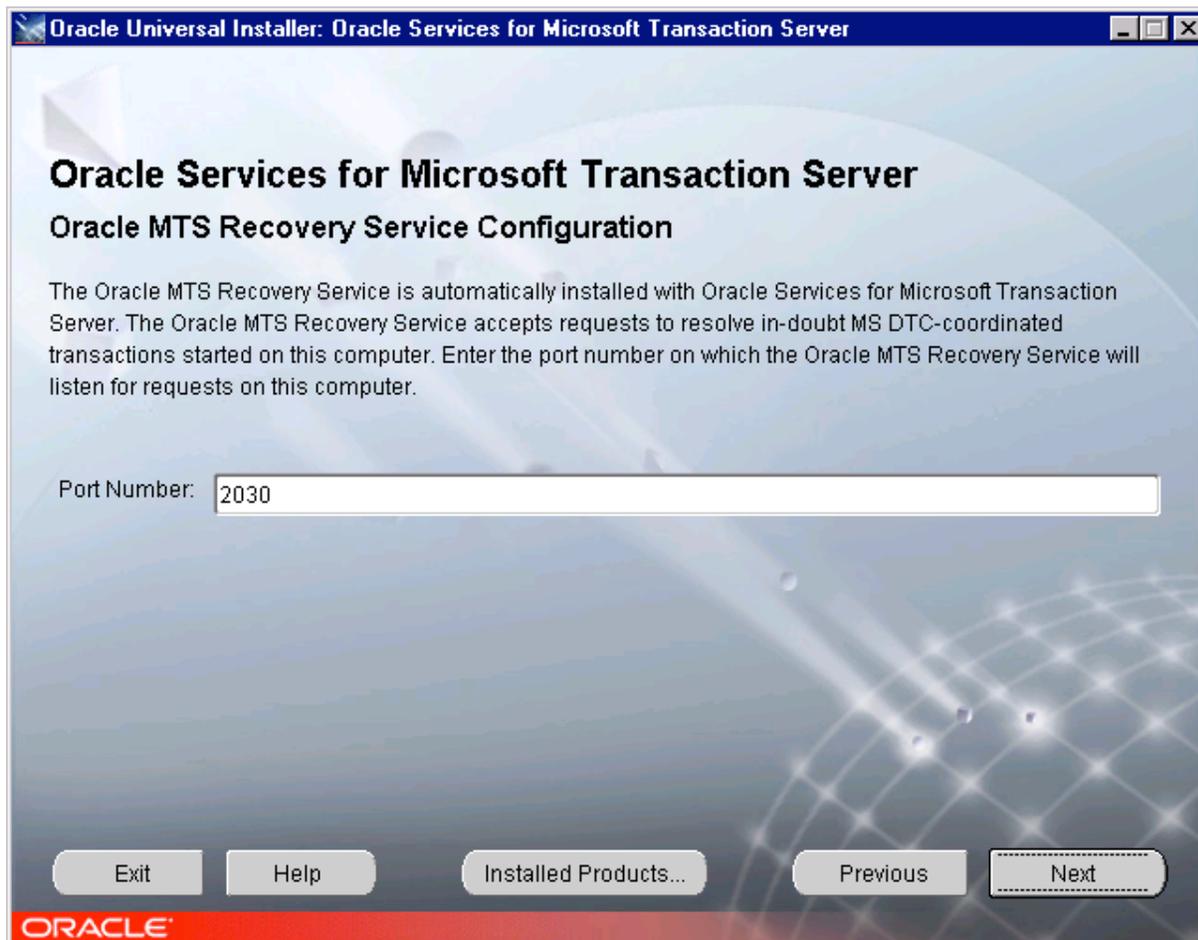


Figure C-7: Oracle MTS Recovery Service Configuration

10.1.9 Database Identification (Figure C-8)

Global Database Name: **oasis.dise.rl.af.mil**

Substitute your DNS sub-domain name for ".dise.rl.af.mil"

Click **next**

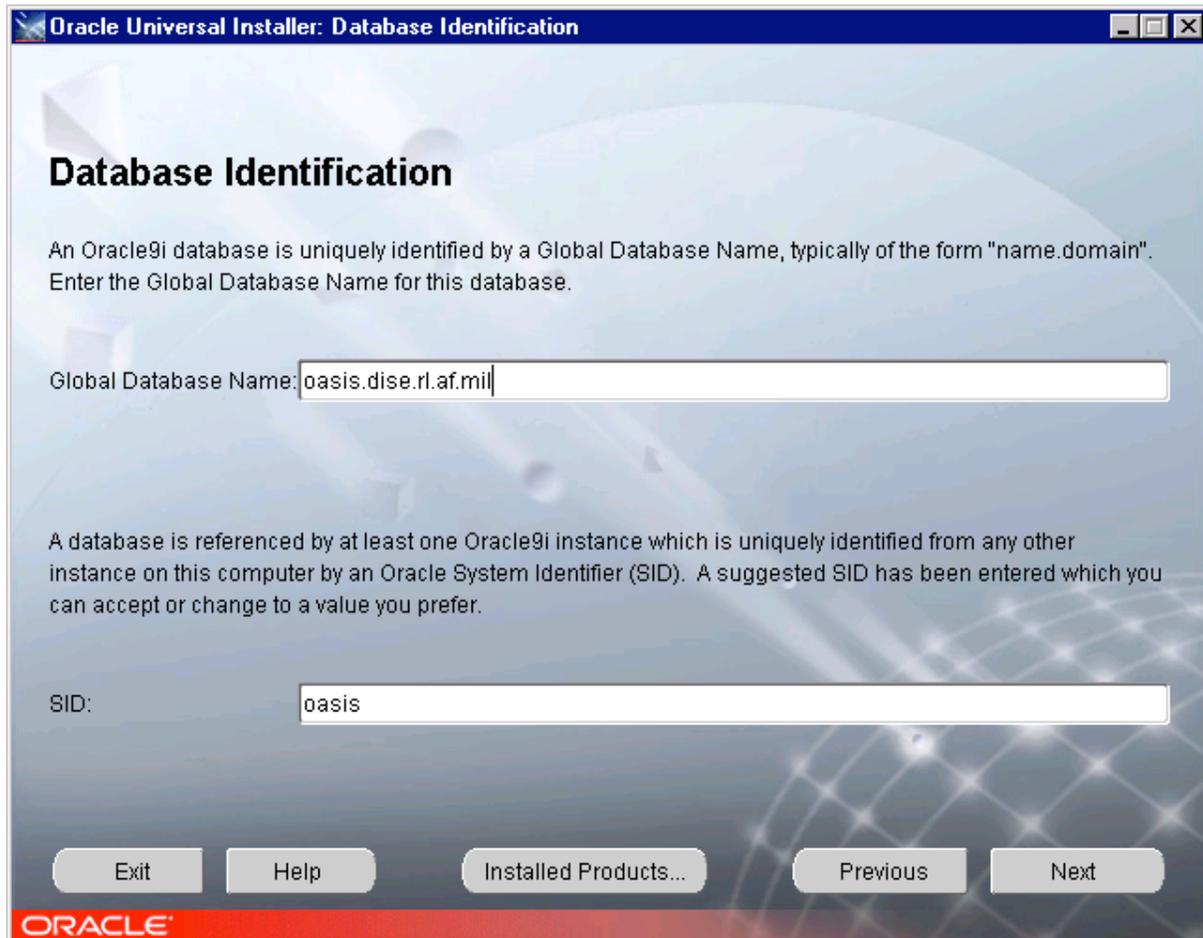


Figure C-8: Database Identification

10.1.10 Database File Location (Figure C-9)

Choose Directory to store database Files
Click **next**

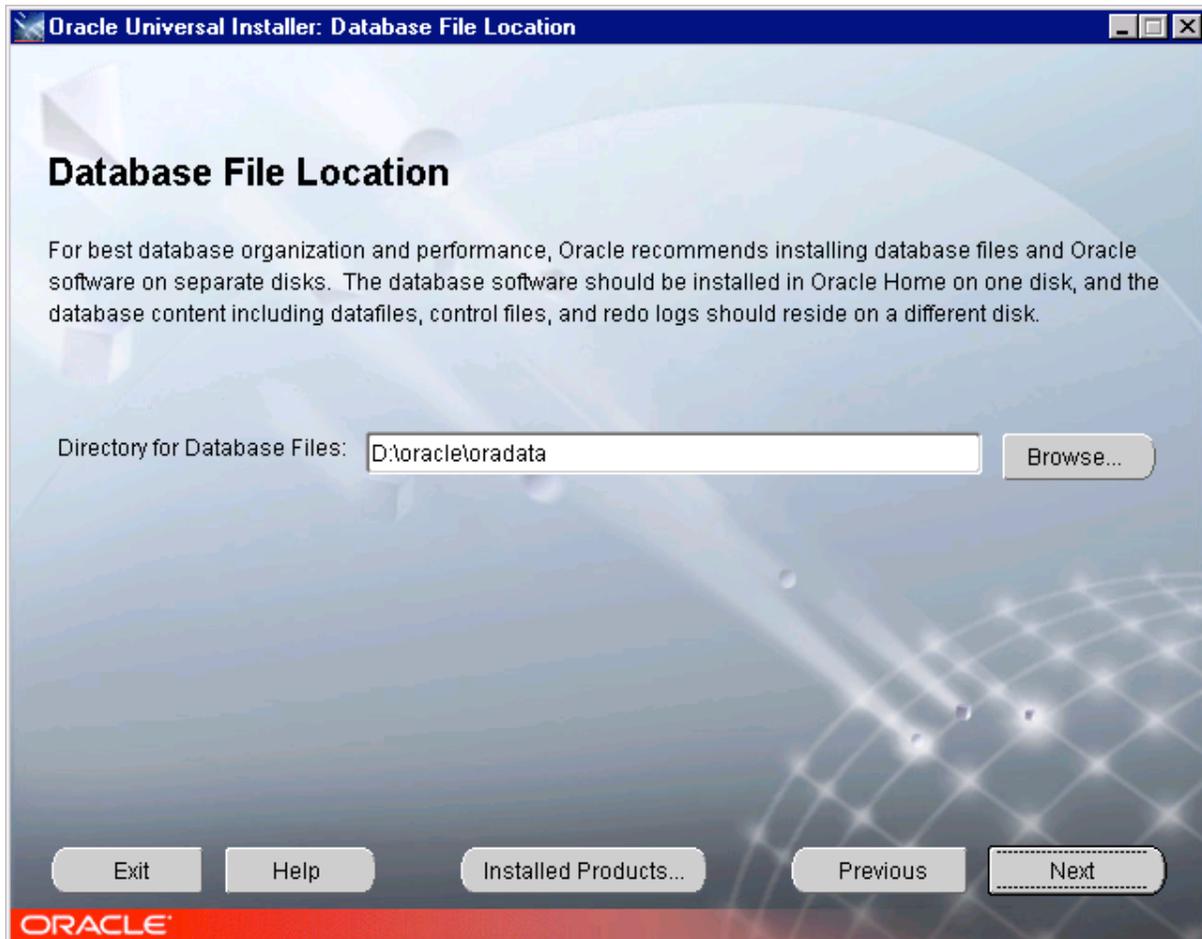


Figure C-9: Database File Location

10.1.11 Database Character Set (Figure C-10)

Choose **Use the default character set**
Click **next**

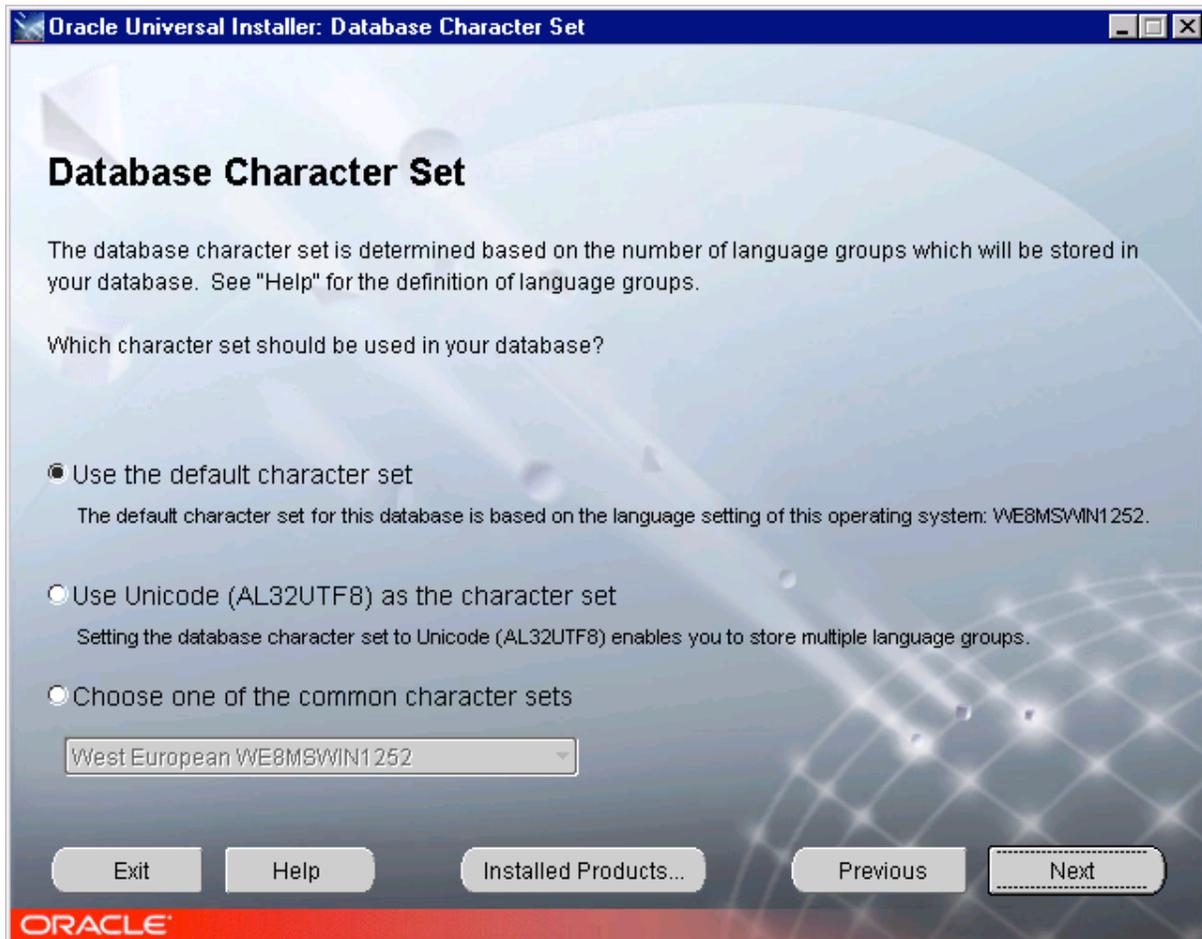


Figure C-10: Database Character Set

10.1.12 Summary (Figure C-11)

Click **Install**

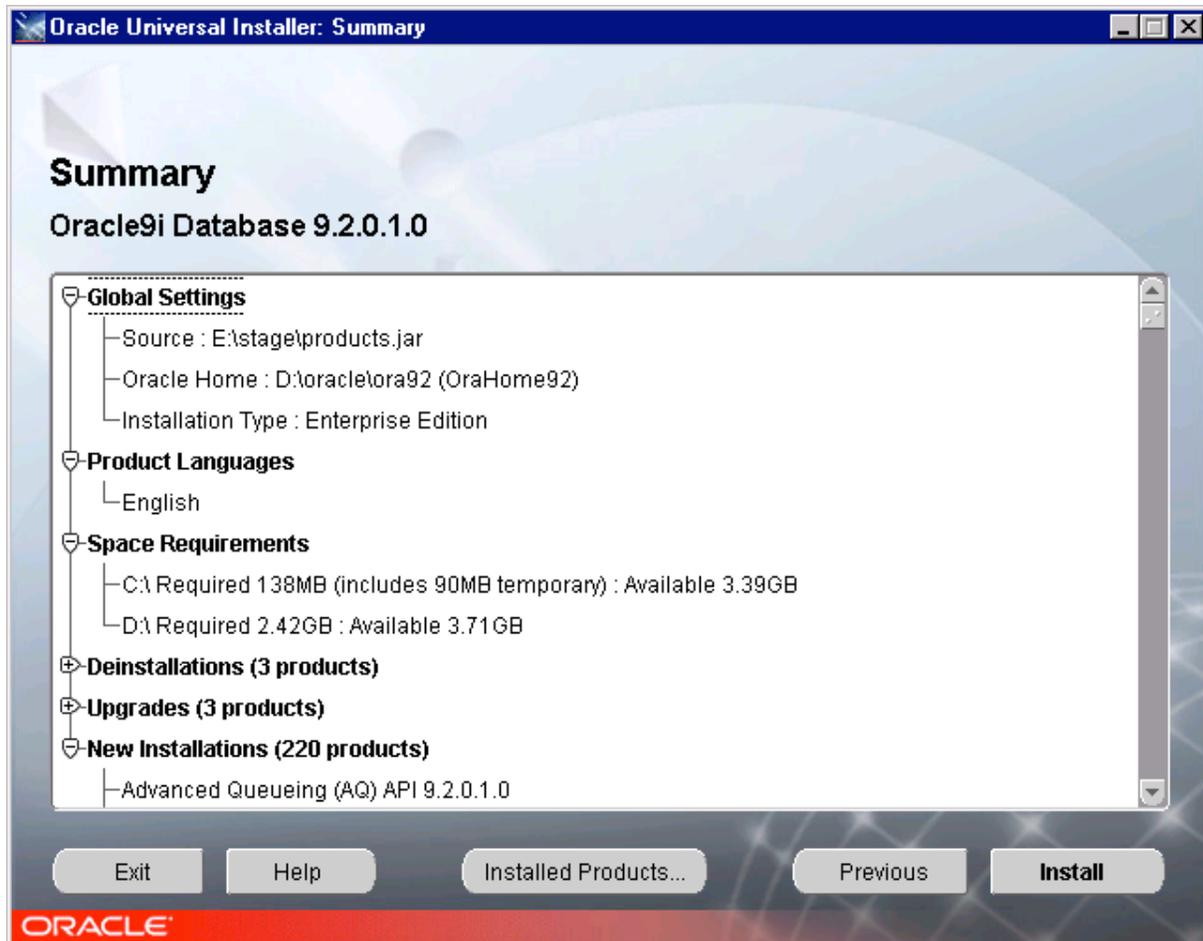


Figure C-11: Summary

10.1.13 Database Configuration Assistant (Figure C-12)

Specify passwords for **SYS** and **SYSTEM**

10.1.14 Click *OK*

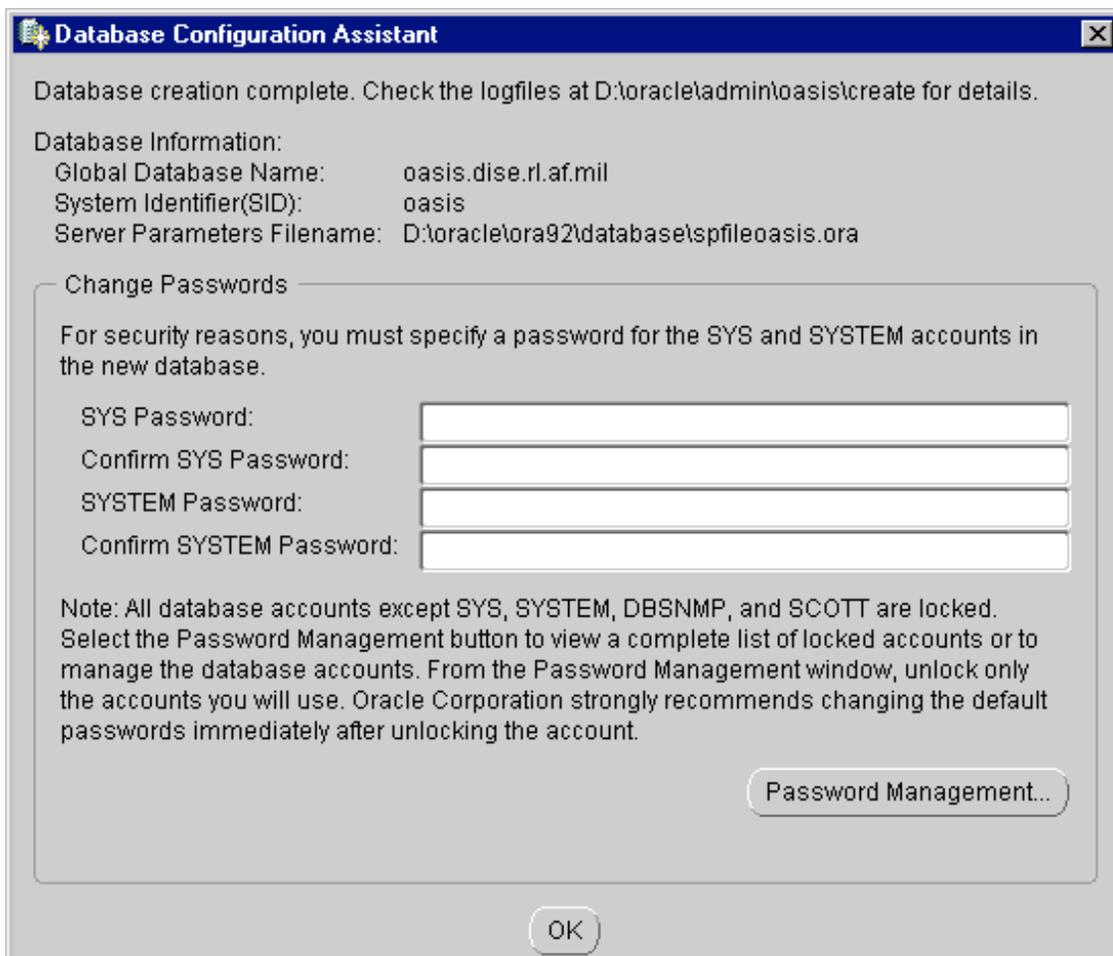


Figure C-12: Database Configuration Assistant

10.1.15 End of Installation (Figure C-13)

Click **exit**

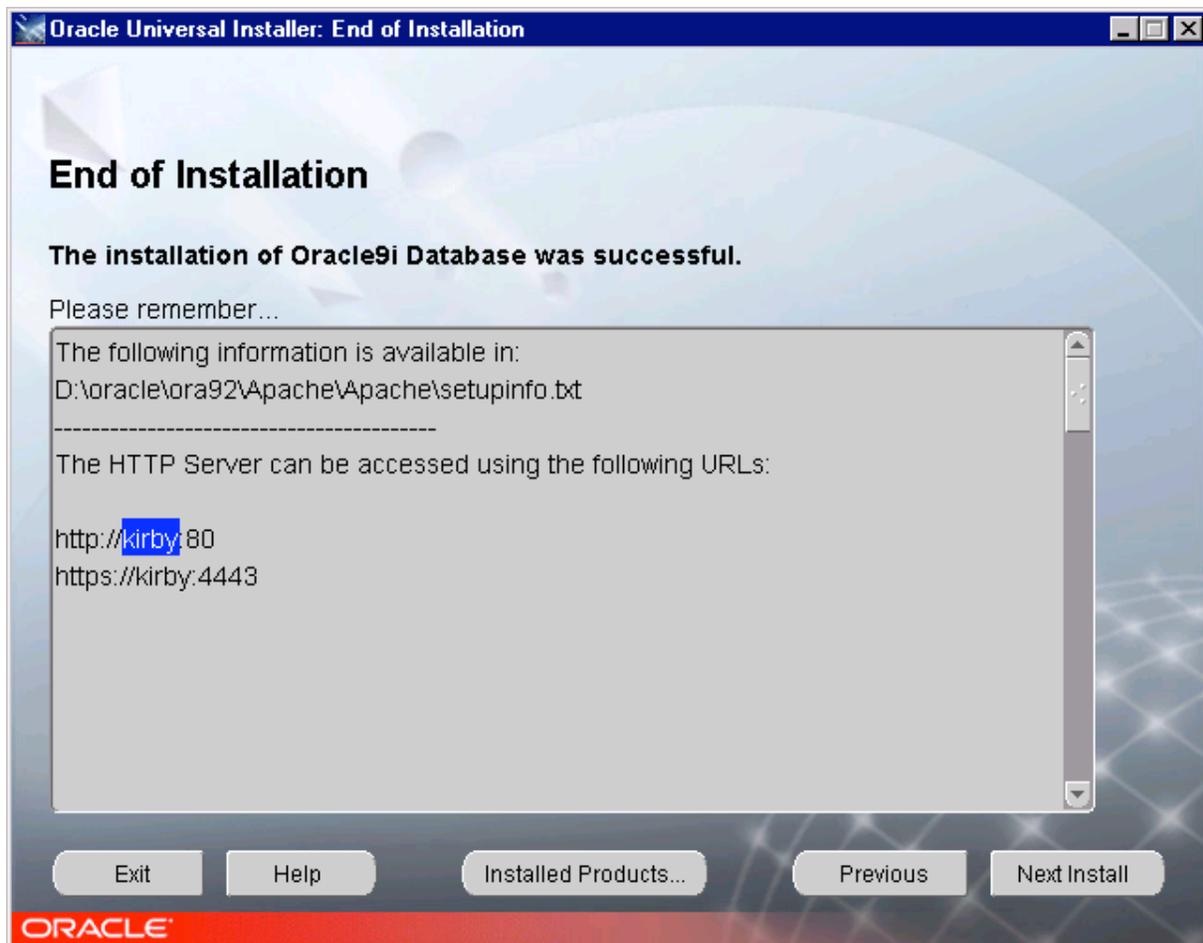


Figure C-13: End of Installation

10.2 Installation of Oracle Patch

11. Appendix D – XPath to SQL-92 Conversion

Note: this package (*mil.af.rl.jbi.util.parsers.XPathToSQL*) makes heavy use of ‘*dom4j_full.jar*’ jar file. This is an excellent open source tool for dealing with xml. The functionality of this package includes DOM navigation, XPath support, and XSLT transformations.

This conversion tool uses XPath as underneath engine. The converter allows us to map tree structures in to simple flat table.

The Version 1.1 JBI platform services make use of this technique to setup headers inside every JMS message for an easy predicate evaluation. This technique can also be used to represent JBI Metadata as a simple table in relational database. This approach has a limitation dealing with repeating elements of the same type. Thus it is necessary to have unique paths in the metadata.

Consider this XML example:

```
<base>
  <name>AFRL-RRS</name>
  <location>Rome, NY</ location >
  <status>Closed</status>
  <emp>
    <military>100</ military >
    <civilian>1000</civilian>
  </emp>
  <rsv>
    <military>300</ military >
    <civilian>10</civilian>
  </rsv>
</base>
```

Let us take look at all unique paths (they start at root, and end at the leaf node):

```
/ base / name = AFRL-RRS
/ base /location = Rome, NY
/ base /status = Closed
/ base / emp /military = 100
/ base / emp /civilian = 1000
/ base / rsv /military = 300
/ base /rsv /civilian = 10
```

This table would get converted it to this:

```
ior841300496= AFRL-RRS
ior325218159= Rome, NY
ior_1496446096= Closed
ior_25613332=100
ior830893588=1000
ior_14568976=300
ior_34712001=10
```

Actual algorithm for generating new element path (Java notation):

```
[1] String newPath = "ior" + unique_path.hashCode().
[2] newPath = newPath.replace('-', '_');
```

This algorithm uses `java.lang.String.hashCode()` method for generating a unique numerical value based on the value of the string. The hash code for a `java.lang.String` object is computed as:

$$s[0]*31^{(n-1)} + s[1]*31^{(n-2)} + \dots + s[n-1]$$

Using integer arithmetic, where $s[i]$ is the i^{th} character of the string, n is the length of the string, and $^{\wedge}$ indicates exponentiation. (The hash value of the empty string is zero.)

In practice it is not a good idea to have variable names as numerical values. In JMS you can not have variable names as numerical values. Databases have problems with column names as numerical values, and cannot contain negative signs in the column name. Appending the generated hash to "ior" string we effectively take care of this little problem. Some returned hash values are negative, and this creates a problem since databases do not like that. Replacing the negative sign with an underscore solves this problem (2nd line in algorithm).

Now let us take a look at this XPath to SQL 92 converter:

```
/base/name = AFRL-RRS
```

Generates and executes an SQL string:

```
SELECT InforObject_x FROM table_x WHERE ior841300496= AFRL-RRS"
```

Now let's try using a relative path as a query:

```
//military > 1000
```

Returns an SQL string:

```
(ior_25613332>1000) OR (ior_14568976 >1000)
```

Let's try a combination of two:

```
(//military > 1000) AND (//civilian < 50)
```

This would create an SQL string:

```
((ior_25613332>1000) OR (ior_14568976 >1000) )
AND (ior830893588<50) OR (ior_34712001<50) )
```

12. Appendix E - Troubleshooting

12.1 JBoss Trouble Shooting

Problem: JBoss keeps showing java.net.SocketException: Connection reset by peer in the server logs

Solution: This occurs each time a JBI client disconnects. Note the status of the message is [WARN]. These messages can generally be ignored.

Problem: JBoss keeps getting OutOfMemoryErrors during heavy usage of query and archive.

Solution: Increase the default heap size for Java. This can be done by setting the environment variable JAVA_OPTS to a higher number. The recommended initial setting is -mx512m (this tells Java to allow up to 512 MB of ram to be used before throwing an OutOfMemoryError)

12.2 Web Based Information Management Tools

Problem: Cannot load the web based management tools JBoss throws the following exception: org.apache.jasper.JasperException: Unable to compile class for JSP
An error occurred at line: -1 in the jsp file: null

Solution: *The JAVA_HOME environment variable has not been set correctly. An environment variable named JAVA_HOME must exist and be set to the explicit path of a working java sdk 1.4.1 or higher installation.*

Problem: Clicking the “view” link in the Metadata Repository tool just shows pops up a blank page. Where is the schema?

Solution: This most commonly occurs with older versions of Netscape Navigator. From the menu at the top of the blank page, click “View” then “Page Source” to display the XML document with all of its tags and elements properly rendered.

Problem: I am using IE and the Metadata Repository keeps giving me “Page Cannot Be Displayed” errors when I add a new Schema.

Solution: This problem is an Internet Explorer exclusive and appears to be a bug in the way the Internet Explorer web browser uploads xml content. (It seems to have been introduced by one of the plethora of IE related security updates.) To avoid seeing this error open an Internet Explorer browser window. Click “Tools” -> “Internet Options”. In the new window that pops up click the “Advanced” tab. Now uncheck the box that reads “Show friendly HTTP error messages”. Click “Apply”. Then click “OK” to close the window.

12.3 Known issues with XML Schemas.

Problem: I am using special namespaces in my Schemas and

- the migration tool fails to import it.

Or

- The mdr refuses to accept it.

Solution: Currently the MySQL and Oracle relational implementations do not support namespaces other than the namespace “xsd”.

Problem: I am not inlining the definition of my complex types and

- the migration tool fails to import it.

Or

- The mdr refuses to accept it.

EX.

```
<xsd:complexType name="SpecificObjectData">
  <xsd:sequence>
    <xsd:element name="InfoObjectPOC" type="InfoObjectPOC">
      <xsd:element name="Classification" type="xsd:string"/>
      <xsd:element name="OperationalCode" type="xsd:string"/>
      <xsd:element name="Theater" type="xsd:string"/>
      <xsd:element name="PayloadClass" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
<xsd:complexType name="InfoObjectPOC">
  <xsd:sequence>
    <xsd:element name="Email" type="xsd:string"/>
    <xsd:element name="Phone" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Solution: Currently the MySQL and Oracle relational implementations do not support non-inlined complex type definitions... If you have a schema with a definition analogous to the above a suitable workaround is to inline the definition of the complex type like in the example below.

```
<xsd:complexType name="SpecificObjectData">
  <xsd:sequence>
    <xsd:element name="InfoObjectPOC">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="Email" type="xsd:string"/>
          <xsd:element name="Phone" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="Classification" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:element name="OperationalCode" type="xsd:string"/>
  <xsd:element name="Theater" type="xsd:string"/>
  <xsd:element name="PayloadClass" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
```

13. Repository Migration

When migrating the repositories the following exception may occur:

```
Exception in thread "main" java.lang.NoClassDefFoundError: java/sql/Savepoint
  at oracle.jdbc.driver.OracleDriver.getConnectionInstance(OracleDriver.java:521)
  at oracle.jdbc.driver.OracleDriver.connect(OracleDriver.java:325)
  at java.sql.DriverManager.getConnection(Unknown Source)
  at java.sql.DriverManager.getConnection(Unknown Source)
  ....
```

This error occurs when the incorrect version of Java is in the PATH environment variable. This will likely occur when you install Oracle after Java; the path to Oracle's Java (1.3.1 or 1.1.8) will be added to the PATH variable before the Java 1.4 path. The fix is to move the Java 1.4 path in front of the Oracle Java path.

13.1 Appendix F - Migrating from Platform Services 1.0.1

If you have a Platform Services 1.0.1, you can migrate your current schemas and users/roles to the Platform Services 1.1 Repository using the included migration scripts.

NOTE 1: The JBoss Application Server must be running before using the migration script. Please see section 6.4 for details on starting JBoss.

NOTE 2: after extracting the migration archive (as described below) make sure the client.properties file for the migration tool points to the ip of your JBoss server. The default value is localhost.

In addition the Oracle OCI client drivers must be properly installed on the machine you are using to run the export utility.

OCI Library setup

This is a type II driver and therefore requires external library support. The library must be accessible to the user running the JBoss Application Server as well as the user running the migration scripts (if applicable). This setup is platform dependent.

For Windows it is necessary to include the path where the library ocijdbc9.dll resides. Under system variables (MyComputer/Properties) add the following to the PATH variable:

[ORACLE_HOME]\bin

where [ORACLE_HOME] is the directory of the Oracle installation.

For Unix/Linux, the library path must contain the path to the libocijdbc9.so library. Edit the .cshrc file for the users referenced above, by adding [ORACLE_HOME]/lib to the LD_LIBRARY_PATH environment variable, where [ORACLE_HOME] is the directory of the Oracle installation.

13.2 Migrating from Oracle XMLType Repository (Platform Services 1.0.1) to Oracle Relational

The migration utility is located in the *PLATFORM_CD_ROOT/server-side/utilities*. Simply unzip migration_utility1.0.1-1.1.zip to the desired directory then follow the instructions below:

The environment variable ORACLE_HOME must be set to the root directory of your Oracle installation prior to running this utility. You must run the Repository Migration Utility with the following parameters:

Windows:

```
runOracleRepositoryUpgrade.bat <system password> <Oracle tnsnames entry>
```

Unix/Linux:

```
runOracleRepositoryUpgrade.csh <system password> <Oracle tnsnames entry>
```

The script requires the Oracle SYSTEM password and tnsnames entry. Please see you DBA if you do not know the system password, and refer to 6.3.1.1.1 for information on the Oracle tnsnames entry.

After successful completion of the utility, your schemas and user/roles have been migrated to Platform Services 1.1.

13.3 Migrating from Oracle XMLType Repository (Platform Services 1.0.1) to MySQL

There are several steps to necessary for successful migration from Oracle XMLType to MySQL. The first step is to export the legacy data from the Oracle database. This export creates an intermediate script that is run by the Platform Migration script, located in the *PLATFORM_CD_ROOT/server-side/utilities* directory. Finally, after verifying that JBoss is running, import the legacy data into MySQL.

Step 1: Export the legacy data

Important: the Oracle database with the Repository data must be running for a successful export!

Windows:

```
runOracleToMySQLRepositoryMigration.bat export
```

Unix/Linux:

```
runOracleToMySQLRepositoryMigration.csh export
```

The utility prompts you for the Oracle SYSTEM password. Please see you DBA if you do not know the system password.

After successful completion of the utility, your schemas and user/roles have been preserved in the file *MySQL_script.sql* in the utilities directory.

Step 2: Start JBoss with MySQL as the repository:

```
run -c jbi_mysql
```

(refer to Chapter 6 for installation/configuration instructions)

Step 4: Import the legacy data

Windows:

```
runOracleToMySQLRepositoryMigration.bat import
```

Unix/Linux:

```
runOracleToMySQLRepositoryMigration.csh import
```

The utility prompts you for the MySQL root password and the database name, which were set during the installation of MySQL. Please see you DBA if you are unsure of these values.

After successful completion of the utility, your schemas and user/roles have been migrated to MySQL.

If you witnessed any errors during the final stages of the migration similar to the following:

```
Importing Type/Version=mil.af.rl.my.wonderful.type/1.0 ...failed!  
org.infospherics.commonAPI.impl.exception.PlatformFailureException:  
Exception within updateInfoObjectDescriptor in the MetadataRepository  
class interacting with MDR: EJBException;; nested exception is:  
...
```

Please see the TroubleShooting appendix