Retail passion. Innovative solutions.

Retek® Security Manager™ 11.0.1

Operations Guide

Corporate Headquarters:

Retek Inc.
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403
USA

888.61.RETEK (toll free US)
Switchboard:
+1 612 587 5000

Fax:
+1 612 587 5100

European Headquarters:

Retek
110 Wigmore Street
London
W1U 3RW
United Kingdom

Switchboard:
+44 (0)20 7563 4600

Sales Enquiries:
+44 (0)20 7563 46 46

Fax:
+44 (0)20 7563 46 10

Customer Support

Customer Support hours

Customer Support is available 7x24x365 via email, phone, and Web access.

Depending on the Support option chosen by a particular client (Standard, Plus, or Premium), the times that certain services are delivered may be restricted. Severity 1 (Critical) issues are addressed on a 7x24 basis and receive continuous attention until resolved, for all clients on active maintenance. Retek customers on active maintenance agreements may contact a global Customer Support representative in accordance with contract terms in one of the following ways.

Contact Method	Contact Information
E-mail	support@retek.com
Internet (ROCS)	rocs.retek.com Retek's secure client Web site to update and view issues
Phone	+1 612 587 5800

Toll free alternatives are also available in various regions of the world:

Australia	+1 800 555 923 (AU-Telstra) or +1 800 000 562 (AU-Optus)
France	0800 90 91 66
Hong Kong	800 96 4262
Korea	00 308 13 1342
United Kingdom	0800 917 2863
United States	+1 800 61 RETEK or 800 617 3835

Mail

Retek Customer Support
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403

When contacting Customer Support, please provide:

- Product version and program/module name.
- Functional and technical description of the problem (include business impact).
- Detailed step-by-step instructions to recreate.
- Exact error message received.
- Screen shots of each step you take.

Contents

Chapter 1 – Introduction

Welcome to the Retek Security Manager (RSM) Operations Guide. The guide is designed so that you can view and understand the application's 'behind-the-scenes' processing, including: its technical architecture, key system administration tools and configuration settings, and its integration dataflow across the enterprise.

Overview

RSM is an application that provides a retailer's Retek applications with a centralized method of authenticating and authorizing system users.

RSM leverages a Light Directory Access Protocol (LDAP)-compliant directory service to authenticate valid users. RSM provides centralized administration screens for system administrators to create application, functional and data level permissions. RSM facilitates a centralized assignment of user security within the retailer's Retek enterprise.

Who this guide is written for

Anyone with an interest in developing a deeper understanding of the underlying processes and architecture supporting RSM functionality will find valuable information in this guide. There are two audiences in general for whom this guide is written:

- System analysts and system operations personnel:
  - Who are looking for information about RSM's processes internally or in relation to the systems across the enterprise.
  - Who operate RSM regularly.
- Integrators and implementation staff with overall responsibility for implementing RSM.

Where you can find more information

This guide does not show you how to use the front-end of RSM. Rather, the focus here is on how data is managed, how it flows, and how it is processed.

If you wish to find further information, see the following applicable Retek documents:

- RSM front-end documentation (for example, the RSM User Guide)
- RSM Installation Guide
- RSM Data Model
- RSM Javadoc

Chapter 2 – Backend system administration and configuration

Note: In this chapter, some examples refer to data within other Retek applications. These examples are included for illustration purposes only.

Supported Retek products

This version of RSM is compatible with the following Retek products:

- Retek Price Management (RPM) 11.x
- Retek Navigator 11.x
- Retek Merchandising System (RMS) 11.x suite (including Retek Sales Audit [ReSA]and Retek Trade Management [RTM]) 11.x
- Active Retail Intelligence (ARI) 11.x
- Retek Inventory Optimization (RIO) 11.x
- Allocations 11.x
- Retek Invoice Matching (ReIM) 11.x
- Retek Data Warehouse (RDW) 11.x

Supported environments

For information about requirements for RSM's client, application server, and database server, see the RSM Installation Guide. Note that Active Directory supplies additional administration tools that can be used in conjunction with RSM.

Bootstrap user

This specific username is entered into the RSM database when it undergoes initial loading. This username is hardcoded into the loading script. This user is assigned a permission so that he or she can initially enter RSM and set up the administration content of the application.

Procedure to change the name of the bootstrap user

After installation, the retailer should enter the database and change the name of this user in the table USER_ROLE to one that matches a user already entered into the retailer's LDAP-compliant directory server. This user should then be able to log into the system with permissions. A retailer can establish this user to be system administrator, who can establish the administration content of the application.

Logging and log4j.xml

Jakarta commons logging

The API that RSM components work with is built using Jakarta's Commons Logging package. Commons logging provides ‘an ultra-thin bridge between different logging libraries’, enabling the RSM application to remain reasonably 'pluggable' with respect to different logger implementations. Objects in RSM that require logging functionality maintain a handle to a Log object, which adapts logging requests to the (runtime configurable) logging provider.

In RSM, Log4j is the library under commons logging.

Additional information about Jakarta Commons Logging can be found at the following websites:

- http://jakarta.apache.org/commons/logging/
- http://jakarta.apache.org/commons/logging/api/index.html

Log4j.xml

The logging mechanism that is used for RSM is log4j.xml, which is the same as the server’s flat text log file. This logging mechanism reveals errors and other significant events that occur during the system’s runtime processing. In most cases, business exceptions and system exceptions ‘rise’ to the user interface. If an exception is displayed, it is logged. Log4j.xml is an open source product.

Application server logging occurs in RSM that should also be configured and monitored. See applicable application server documentation for more information.

Additional information about log4j can be found at the following website:

- http://jakarta.apache.org/log4j/docs/index.html

Logging levels

The level setting established in log4j.xml instructs the system to log that level of error and errors above that level. The logging levels are the following:

- Fatal
- Error
- Warning
- Info
- Debug

Note: In a production environment, the logging setting should be set to Error or Warn, so that system performance is not adversely impacted.

The level is established in the log4j.xml file.

For example:

```
<!-- ====================== -->
  <!-- Setup the loggers       -->
  <!-- ====================== -->

  <logger name="com.retek">
    <level value="TRACE"/>
  </logger>
```

Output file

RSM's default logging output file is system.out, the standard application server output file. The system allows another output file to be specified.

For example:

```
<param name="Target" value="System.out"/>
```

Exception handling

The two primary types of exceptions within the RSM system are the following:

- System exceptions
  For example, server connection and/or database issues are system exceptions.
- Business exceptions
  For example, a user tries to create a role without a description. Most exceptions that arise in the system are business exceptions.

Configurable files with content owned by other Retek applications

The files below share the following characteristics:

- Their use depends on the RSM functionality that they are leveraging.
- Their content is dependent upon the other Retek application's functional design, not on RSM.
- Under most circumstances, they should not have to be changed after installation.
- The prefix 'app' in the file's name refers to the other Retek application

Content_model_app.xml

A content model defines in an XML document the task groups and tasks that may be displayed in the task pad of a Retek GUI application window. RSM stores its own content model (content_model_rsm.xml) as well as the content models of any other Retek applications that use RSM to retrieve secure content. The content models must be stored under the RSM application ear in directory retek/conf. For example, if an implementation includes Retek Price Management (RPM) and RSM, the RPM content model (content_model_rpm.xml) is copied to the retek/conf directory under the RSM ear. Under most circumstances, this file should not have to be changed.

app_data_definition_app.xml

The application data definition file is used to define the application data hierarchy for the data level permissions. If RSM is being used to administer data level permissions for another Retek application, the file below must exist and be configured correctly. This file must be stored under the RSM application ear in directory retek/conf. Under most circumstances, this file should not have to be changed.

Here is an example of the RPM data definition (app_data_definition_rpm.xml) file:

```
<application id="app.rpm" resource_bundle="retek/messages_rpm" description_key="rpmDescription">
    <hierarchy_type id="merchandiseHierarchy" description_key="merchandiseHierarchyDescription" >
        <hierarchy_node id="departmentId" description_key="departmentDescription">
            <hierarchy_node id="classHierId" description_key="classHierDescription">
             <hierarchy_node id="subclassId" description_key="subclassDescription"/>
            </hierarchy_node>
        </hierarchy_node>
    </hierarchy_type>
    <hierarchy_type id="locationHierarchy" description_key="locationHierarchyDescription" >
      <hierarchy_node id="zoneGroupId" description_key="zoneGroupDescription">
          <hierarchy_node id="zoneId" description_key="zoneDescription"/>
      </hierarchy_node>
    </hierarchy_type>
</application>
```

messages_app.xml

The messages file is used to map data hierarchy nodes to their displayable values. If RSM is being used to administer data level permissions for other Retek applications, the file below must exist and be configured correctly. This file must be stored under the RSM application ear in directory retek/conf. Under most circumstances, this file should not have to be changed.

Here is an example of the messages_rpm.xml file:

```
rpmDescription=RPM
merchandiseHierarchyDescription=Merchandise Hierarchy
departmentDescription=Department
classHierDescription=Class Hierarchy
subclassDescription=Subclass
locationHierarchyDescription=Location Hierarchy
zoneGroupDescription=Zone Group
zoneDescription=Zone
```

services_app.xml

The services XML file defines the packages of the interface and implementation classes required by RSM. If RSM is being used to administer data level permissions for other Retek applications, the file below must exist and be configured correctly. This file contains information that RSM needs to call out to the applicable application to retrieve data-level information. This file must be stored under the RSM application ear in directory retek/conf. Under most circumstances, this file should not have to be changed.

Here is an example of the services file (services_rpm.xml) for RPM:

```
<services-config>
   <customizations>
      <interface package="com.retek.rsm.external.service" app="app.rpm">
         <impl  package="com.retek.rsm.external.service" />
      </interface>
   </customizations>
</services-config>
```

jndi_providers_app.xml

The jndi providers file contains iiop information for the Retek application that RSM calls. If RSM is being used to administer data level permissions for other Retek applications, the file below must exist and be configured correctly. This file contains information that RSM needs to call out to the applicable application to retrieve data-level information. This file must be stored under the RSM application ear in directory retek/conf. Under most circumstances, this file should not have to be changed. jndi_providers_app.xml is updated during installation to point to the address of the external application's server.

Here is an example of the jndi providers (jndi_providers_rpm.xml) file for RPM:

```
<ejb_context_overrides>
   <provider app="app.rpm" url="iiop://host:port"
factory="com.ibm.websphere.naming.WsnInitialContextFactory">
   </provider>
</ejb_context_overrides>
```

Security.properties

For LDAP authentication

These values are used for the configuration of the authentication process as it is run through LDAP. Once an LDAP schema is established, a retailer enters applicable LDAP properties to point to that schema.

ldap.initialcontextfactory

This internal Java-specific setting should not change from its initial value.

For example:

```
ldap.initialcontextfactory=com.sun.jndi.ldap.LdapCtxFactory
```

ldap.authenticationprovider.url

This value represents the authentication provider's URL. In a production environment, this setting would contain the retailer's address for its directory server.

For example:

```
ldap.authenticationprovider.url=ldap://64.238.67.60:379/
```

ldap.user.basedn

The values in this entry must correspond to entries in the LDAP server. DN stands for distinguished name. The top level of the LDAP directory tree is the base, referred to as the 'base DN'. This value represents the user base DN property.

For example:

```
ldap.user.basedn=ou=RSM,dc=rsmad,dc=local
```

ldap.authenticationmode

This value represents the authentication mode property. LDAP uses various ways to authenticate against a directory server, and the method of authentication can be set up. For almost all environments integrated with RSM, the value should be `simple`.

For example:

```
ldap.authenticationmode=simple
```

ldap.securityprotocol

Note: This setting is currently not used by RSM.

This value represents RSM's encryption protocol. SSL stands for secure socket layer (SSL). SSL is a protocol developed for private transmissions. SSL works by using a private key to encrypt data that's transferred over the SSL connection.

For user search

These settings provide the 'behind the scenes' login information for the system to connect to the directory server. For example, when an RSM user wishes to search on the directory server for a user, the RSM system must have a username and password to log in to the directory server to enable the search to occur. The filter property value represents the directory server-specific way of filtering user information by attribute (when the directory server is finding users and then limiting the results). Because various directory servers use different attributes to represent a username, this value must be updated if the retailer were to change directory servers.

For example:

```
ldap.usersearch.user=cn=Administrator,cn=users,dc=rcomad,dc=local
ldap.usersearch.password=PaSsW0rD
ldap.user.filter=(&(objectCategory=person)(objectClass=user) %v)
```

LoginModule configuration information

These settings configure the system to point to the applicable user repository (such as a directory server, database, and so on). The login module class value determines the class that is responsible for accessing the user repository for authentication.

For example:

```
loginmodule.class=com.retek.rsm.domain.security.dao.LdapLoginModule
```

For mapping LDAP to directory schema

The table below contains directory server-specific attributes concerning user information.

Various directory servers use different attributes to represent user information. If a retailer were to change directory servers, these values must be configured to reflect the new directory server.

Element	Definition
ldap.firstname.attrname	LDAP first name attribute name property
ldap.lastname.attrname	LDAP last name attribute name property
ldap.username.attrname	LDAP username attribute name property

User signature information

To facilitate single sign on functionality, a user signature is passed among a retailer's RSM-integrated applications. The following steps below describe how the user signature is created and used:

1 When a user first logs on to Retek Navigator (the application that contains the Retek's enterprise login screen), Retek Navigator sends RSM user and password data that requires authentication.

2 RSM calls the retailer's LDAP compliant directory service to authenticate the username and password data. Once a user is authenticated, RSM creates an encrypted a user signature, which is returned to Retek Navigator.

3 When a user uses Retek Navigator to launch other RSM-integrated applications, Retek Navigator passes the user signature to those applications. The application being launched (for example, ReIM) accepts the user signature and calls RSM to determine whether the user signature is valid. If the validation step is successful, the user accesses the application without having to go through that application's (for example, ReIM) login screen.

user.signature.cipher.algorithm

RSM uses an algorithm to generate a user signature. A retailer may change this algorithm and configure this property value to reflect the different algorithm being used.

For example:

```
user.signature.cipher.algorithm=HmacSHA1
```

user.signature.secretkey

To generate user signatures, the algorithm needs a secret key. Retek recommends that the retailer updates this value on a regular basis. A retailer can change this secret key if a compromise in security has occurred.

For example:

```
user.signature.secretkey=gjgh6382nEDmxMLc3DSkhYP0ah347495
```

user.signature.salt

The system uses the salt value to avoid dictionary attacks. Salt adds characters to what is being created (in this case, a user signature). Because of the salt value, for example, the encrypted value might have 100 digits rather than 10 digits. Breaking the encryption thus becomes more difficult.

For example:

```
user.signature.salt=¥!asdfghlll©ñ¤?#¥³1966
```

User authentication information

user.max.allowable.authentication.failures

This value represents the maximum number of times that a user can fail authentication before the user's account is locked.

For example:

```
user.max.allowable.authentication.failures=5
```

user.max.time.lock.useraccount

This value represents the maximum number of hours a user's account remains locked. If the account is locked and over the maximum time value, the next time that user logs onto the system, the lock releases.

For example:

```
user.max.time.lock.useraccount=30
```

Unlocking a user's account

There are two ways through which a user's account is 'unlocked', through a manual procedure or because when the maximum timeout value has been reached.

Manual procedure

1 Access the USER_LOGIN_INFO table.

2 Delete the row based on the user's name. The user's account is unlocked.

Encryption information

These settings are related to the system's method of password encryption. When enterprise users log in, data is transmitted 'over the wire'. The RSM system addresses security risks surrounding this transmission by utilizing an asymmetric encryption system. In this cryptographic system, public keys are used to encrypt messages and only corresponding private keys can be used to decrypt them. Values include the third party Java Cryptography Extension (JCE) provider abbreviation and the third party JCE provider class.

For example:

```
encryption.publickey.algorithm=rsa

encryption.jce.provider=BC

encryption.jce.provider.class=org.bouncycastle.jce.provider.BouncyCa
stleProvider

encryption.privatekey.modulus=11884130515889319239100533614587870066
77777777777777777777000000000000000000000000000044444444444444444444
44444444444444444444444444488888888888811111111155555555555555222222
22222222222222222888888888888888888888888888888888888888888888888800
26789098655154249396141542724701542165131223413701563973036211296 41

encryption.privatekey.exponent=11884130515889319239100533614587870066
77777777777777777777000000000000000000000000000044444444444444444444
44444444444444444444444444488888888888811111111155555555555555222222
22222222222222222888888888888888888888888888888888888888888888888880
02678909865515424939614154272470154216513122341370156397303621129641

encryption.publickey.modulus=11884130515889319239100533614587870066 7
77777777777777777770000000000000000000000000000444444444444444444444
44444444444444444444444444488888888888811111111155555555555555222222
22222222222222222888888888888888888888888888888888888888888888888002
67890986551542493961415427247015421651312234137015639730362112964 1

encryption.publickey.exponent=99999
```

Hibernate configuration

Note: This section describes hibernate configuration. For a general description of Hibernate, see the section, ‘Data access layer’ in “Chapter 3 – Technical architecture”.

Hibernate is designed to operate in many different environments, there are a number of configuration settings and properties that control the behaviour of Hibernate at runtime.

Some are optional and have default values. In RSM, these configuration setting and properties reside in a full configuration file named `hibernate.cfg.xml`. The configuration file is expected to be in the root of your `CLASSPATH`..

Property declaration in `hibernate.cfg.xml`

The settings described below are located in the `hibernate.cfg.xml`.

connection.datasource

On the application server, the system administrator creates the datasource (for example, DB2, Oracle, and so on) and gives it a name (Datasource.JNDI.name). The RSM system refers to that name.

For example:

```
<property name="connection.datasource">jdbc/RsmDataSource</property>
```

show_sql

When this setting is ‘true’, SQL statements are shown. This setting might be used during development, build time, performance tuning, debugging, and so on. The executed SQL statements can show against which table(s) queries are pointed.

For example:

```
<property name="show_sql">true</property>
```

use_outer_join

This setting relates to SQL queries. This setting enables outer join fetching (values can be true or false).

For example:

```
<property name="use_outer_join">false</property>
```

Jdbc.batch_size

A nonzero value in this setting enables the use of JDBC2 batch updates by Hibernate. Note that the value must be greater than 1.

For example:

```
<property name="jdbc.batch_size">10</property>
```

dialect

Note: This setting changes depending upon what database is being used with the system (Oracle, DB2, and so on).

This setting instructs the system as to how to construct SQL queries (for example, the SQL queries for Oracle are different than those for DB2).

For example:

```
  <property
name="dialect">net.sf.hibernate.dialect.Oracle9Dialect</property>
```

Hibernate logging

There are two aspects to Hibernate logging, its own logging mechanism and SQL output logging.

1 Hibernate's internal logging setting is established in log4j.xml. The commons-logging service directs output to log4j. To use log4j, the log4j.properties file must be in the classpath. An example properties file is distributed with Hibernate. The class to be logged and the logging level can be specified.

   For example:

```
!-- ======================================================== -->
  <!-- Hibernate trace at this level to log SQL parameters    -->
  <!-- ======================================================== -->


  <logger name="net.sf.hibernate.engine.QueryParameters">
    <level value="TRACE"/>
  </logger>
```

2 SQL statement logging is outputted to system.out.

Note: Because of the volume of SQL logging that occurs, Retek recommends that SQL not be logged in production.

   For example:

```
Hibernate SQL statements

[3/3/04 14:13:22:733 GMT-06:00] 6b64ab5b SystemOut     O Hibernate:
select hierarchie0_.HIER_TYP_ID as HIER_TYP1___,
hierarchie0_.HIER_GRP_ID as HIER_GRP2___,

 hierarchie0_.HIER_TYP_ID as HIER_TYP1_0_, hierarchie0_.HIER_GRP_ID
as HIER_GRP2_0_, hierarchie0_.ROOT as ROOT0_ from HIER_TYP
hierarchie0_ where hierarchie0_

.HIER_GRP_ID=?
```

Internationalization and localization

The technical infrastructure of RSM supports languages other than English. RSM has been subject to the modifications associated with ‘internationalization’, also known as I18N. (The I18N name stems from the fact that eighteen letters exist between the first ‘i’ and the last ‘n’ in the word ‘internationalization.’) Internationalization is the process of preparing software in order to ensure that it can efficiently handle multiple languages. In other words, the software is created so that it can be released into international markets.

Localization, also known as L10N, is the process of adapting software that has been internationalized so that it can be released into a local market with its own language. (The L10N name stems from the fact that ten letters separate the letter ‘l’ from the letter ‘n’ in the word ‘localization’.) Software is only internationalized once. However, software must undergo the localization process for every new language or location into which it is released.

This section describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Resources_xx.properties

An application that can run in various languages must be transformed into somewhat of a ‘generic’ product. That is, the features of the application that could be specific to just one language or locale (such as text, date formatting, and so on) must not be hard-coded into the software. Instead, locale-specific information is intentionally placed in files external to the application.

The majority of the locale specific functionality in RSM resides in two spots, the Resource_xx.properties files located in the rsm_client.jar and a description table located in the RSM database.

The content of the Resource_xx.properties file is interface related, as distinct from executable code. The text in these files is translated so that the interface functions in local settings. The retailer populates the Resources_xx.properties file, where the retailer’s applicable country equals xx. A retailer must populate and/or edit this files with a text editor.

Note: To prevent the application from expending resources translating English into English, the _en versions of these files do not contain examples of the text that a retailer would translate. To identify the text to be translated, a retailer should refer to the _fr versions of these files.

On the server side, one description table exists that contains localized information. Table NAMED_PERMISSION_DSC contains displayable fields used in administering workflow permissions. A retailer must populate and/or edit the rows in this table. Updates are required to the following columns:

- Language: The language to be translated to (for example, fr, en, and so on)
- Country: The country of the locale (for example, FR, US, and so on)
- Label: A short description of the named permission
- Dsc: A long description of the named permission

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated include the following, among others:

- Graphical user interface (GUI)
- Online help
- Some print documentation
- Error messages

Chapter 3 – Technical architecture

This chapter describes the overall software architecture for RSM. The chapter provides a high-level discussion of the general structure of the system, including the various layers of Java code. From the content, integrators can learn both about the pieces of the system and how they interact.

A description of RSM-related Java terms and standards is provided for your reference at the end of this chapter.

Overview

RSM's architecture is built upon a layered model. That is, layers of the application communicate with one another through an established hierarchy and are only able to communicate with neighboring layers. Any given layer need not be concerned with the internal functional tasks of any other layer.

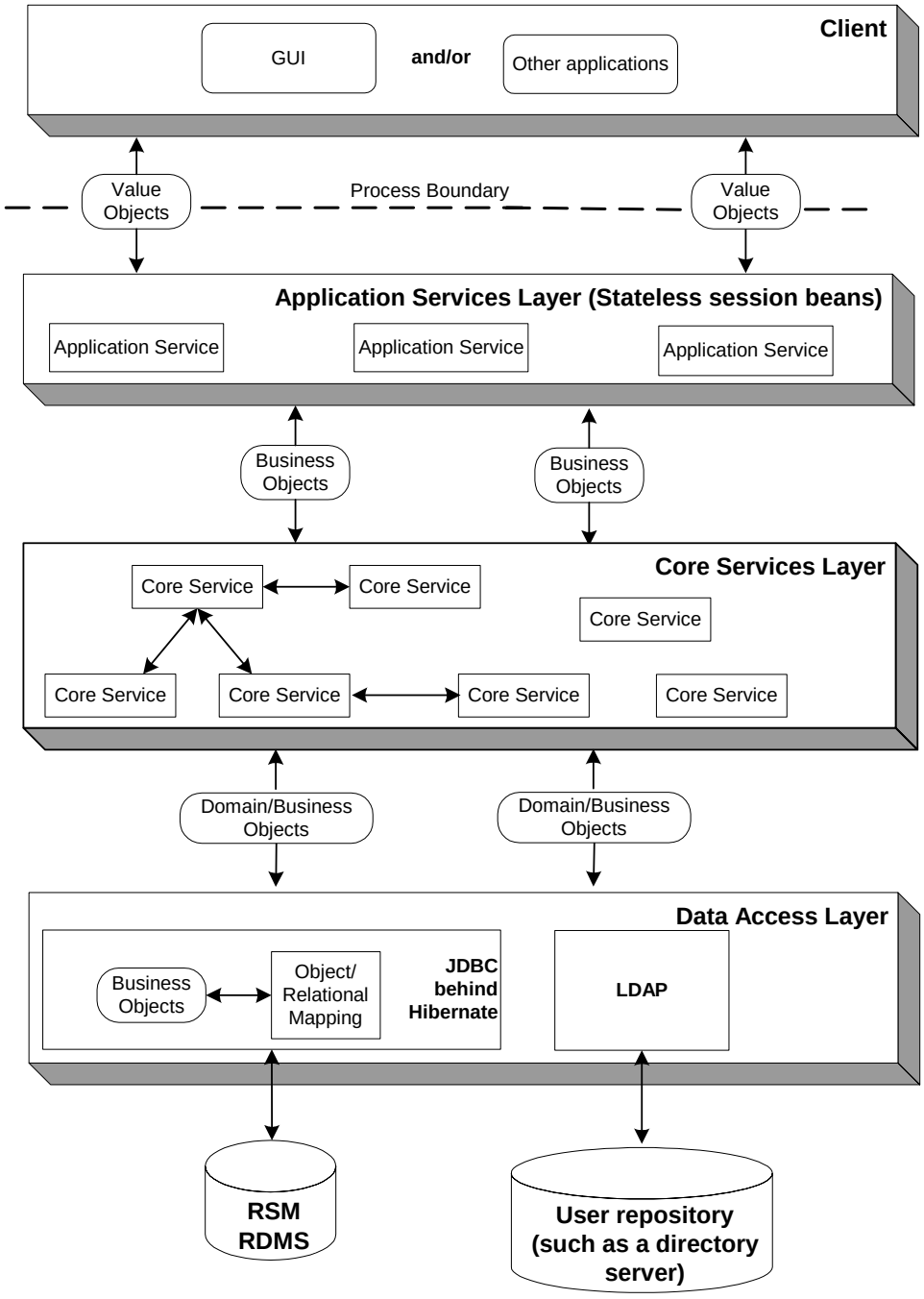
Conceptually, RSM's J2EE architecture is built upon 4-layers and implements what is defined as a service-oriented architecture. Such an architecture is essentially a collection of services that pass data, perform business processing, coordinate system activities, and render data into abstract objects. Defined in the abstract, a service is a function that is well-defined, self-contained, and does not depend on the context or state of other services within the system.

The application's layered Java architecture has the following advantages, among others:

- The separation of presentation, business logic, and data makes the software cleaner, more maintainable, and easier to modify.
- Java applications have enhanced portability which means the application is not 'locked' into a single platform. Upgrades are easier to implement, and hardware is easier to change.
- Logic is implemented using Java objects within a core services layer that is designed around proven architecture concepts.
- The DAO pattern provides database and data model independence.
- The RSM application has been designed so that the implementation of the interaction between the application and the user repository is generic (see the section 'Third party directory server' later in this chapter). Retailers can utilize a user repository that best fits their business needs.

The layered model

The following diagram, together with the explanations that follow, offer a high-level conceptual view of RSM's service-oriented architecture. The diagram highlights the separation of layers as well as their responsibilities within the overall architecture. Key areas of the diagram are described in more detail in the sections that follow.



RSM's technical architecture

GUI

The GUI is responsible for presenting data to the security administrator and for receiving data directly from the security administrator through the 'front end'. The application's front end was developed using a Java Swing framework, which is a toolkit for creating rich presentation in Java applications.

Application services layer (stateless session beans)

The application services layer has two primary purposes. First, this layer is interfaced for GUI maintenance related to security administration screens. Secondly, RSM is designed for integration not only with other Retek applications but with other external retailer systems. These applications use the application services layer to interact as clients to RSM, an enterprise security application. For example, an application such as RPM integrates with RSM through the application services layer. The application services layer becomes an integration point. The application services layer allows for the future possibility of different retailer interfaces.

Application services are designed to provide specific services and specific data requirements to a particular client. What application services a client calls depends upon its needs and the data formats it has. Application services are concerned with somewhat narrow processes. Not surprisingly, the names of application services correspond to client-related processes. For example, an application service might have a call to 'find all named permissions for a user'.

The application services layer of RSM's architecture implements the enterprise Java bean (EJB) type called stateless session beans (SSB). An SSB is a type of EJB that provides stateless service to a client. For example, a stateless session bean could be designed for the GUI. The application services reside on the server side of the process boundary (also known as the remote call boundary).

The application-specific services layer provides an interface between a particular client and the adjacent core services layer. To solve a business problem, application services call one or more core services. (Note that application services could also call other application services. For example, one application service has a large granularity and needs another one to perform minor grain transformations, and so on.)

An important way that application services accept incoming data from a client is via value objects. A value object is a data holder in a highly flat form; value objects facilitate improved system performance. For example, from the GUI, the value object data only has to be what is needed by an applicable screen or set of screens. The application services layer's primary function is to facilitate the conversion of value objects to business objects and business objects to value objects which are required by the adjacent layers. The value objects accepted from and returned to the application services layer are nothing more than data-centric classes which encapsulate closely related items. Value objects are used to provide a quick and lightweight method to transfer flat data items. The value objects passed between the application services layer and the application services layer contain very little, if any, data processing logic and in the context of the RSM are used solely to transfer data.

The application services depend upon both core services and business objects, translating back and forth between input from the client and business objects in the core services layer. The application services call the applicable core service at the applicable time.

Core services layer

This layer consists of a collection of separate and distinct services that encapsulate the RSM application's core business logic. Core services are 'core' in the sense that they work with the domain and business object model, and they contain the domain and business object rules for the application. Unlike application services, core services make no presumptions about how they might be used. In other words, core services contain generic views of business functionality as opposed to a narrow application service process.

Residing very close to the core services, business objects represent business problems. Business objects contain behaviors. For example, they perform validation and guard themselves from being used improperly.

Sometimes core services drive processes with business objects, but more often, core services are responsible for finding the business objects and sending them back to the data access layer. The core services layer is thus responsible for managing object persistence by interacting with the data access objects residing in the supporting data access layer.

To summarize, the core service layer consists of a collection of Java classes that implement an application's business related logic via one or more high-level methods. The core services represent all logical tasks that can be performed on an application's business objects.

To extend an example mentioned earlier, an application service might have a call to 'find all named permissions for a user'. The application service would instruct the core service to find the permissions, make the business object fetch the n number of named permissions that are applicable, and then send the data back to the GUI.

Data access layer

The data access layer renders the job of persistence 'abstract' (not tied to a specific type of database such as Oracle, DB2, and so on).

Note: See the RSM Installation Guide for the database(s) that RSM is certified against.

Database independence is achieved because database code does not permeate the actual database that the system uses. This layer strictly contains the mechanism to persist and retrieve application data (business objects) into and from the relational database. No business logic resides in this layer.

Datasources

The system is designed to include two datasources, an RSM RDMS and a user repository.

RSM RDMS and Hibernate

RSM uses Hibernate, an object/relational persistence and query service for Java. This object-relational framework provides the ability to map business objects residing in the core services layer to relational tables contained within the data store.

Conceptually, Hibernate encompasses most of the data access layer. Hibernate interacts with core services by passing/accepting business objects to/from the core services layer. Internally, Hibernate manages the conversion of RSM's business object to relational data elements required by the supporting relational database management system (RDMS).

For information about Hibernate-related configuration, see "Chapter 2 – Backend system administration and configuration."

User repository (such as a third party directory server)

To facilitate the authentication of users, RSM is integrated with a 3rd party directory service application. Core services interact using Light Directory Access Protocol (LDAP) which allows RSM to 'talk' with the 3rd party directory service. The LDAP standard defines a network protocol for accessing information in a directory.

Though RSM is configurable to use any LDAP-compliant directory server, the system is certified to work with Microsoft's Active Directory® (AD) and OpenLDAP®.

Active Directory is an LDAP-compliant directory server that stores enterprise user information. Microsoft's website describes Active Directory as having a "single-logon capability and a central repository for information for your entire infrastructure, vastly simplifying user and computer management and providing superior access to networked resources." OpenLDAP is an open source implementation of the **L**ightweight **D**irectory **A**ccess **P**rotocol. RSM queries LDAP for user information. No implementation specific enhancements are utilized.

RSM uses LDAP for two purposes:

- As the master repository of user information
- As a third-party authentication service

In the second case, RSM authenticates users by binding to the LDAP Directory Server as the user who is attempting to log in to RSM. The user's password is never stored in RSM; it is passed along when RSM tries to connect to the Directory Server. If the connection to the directory server succeeds, the user is considered authenticated in RSM.

If RSM cannot connect to a directory server; the user is not able to log in.

 **Note:** RSM never writes data to the LDAP Directory Server.

For additional information about Active Directory, see the following website:

- http://www.microsoft.com/

For additional information about OpenLDAP, see the following website:

- http://www.openldap.org/

A word about encrypting passwords

RSM uses an asymmetric cryptographic algorithm. The user passwords are encrypted from the client machine to the application server (and vice versa) using a public key, private key encryption algorithm. The Java cryptography extension (JCE) provider implementation is configurable and can be exchanged with another provider that supports public key encryption algorithms. See "Chapter 2 – Backend system administration and configuration".

Between the RSM application server and the user repository (such as the directory server), the data is not encrypted. RSM assumes that the retailer secures that piece of the network.

For more information about JCE libraries from Sun, see the following:

- http://java.sun.com/products/jce/

RSM architectural Java terms and standards

RSM is deployed using the J2EE-related technologies, coding standards, and design patterns defined in this section.

Data access object (DAO)

This design pattern isolates data access and persistence logic. The rest of the component can thus ignore the persistence details (the database type or version, for example).

Java Cryptography Extension (JCE)

A set of packages that provides a framework for encryption, key generation and key agreement, and algorithms.

Java Development Kit (JDK), version 1.4.1

Standard Java development tools from Sun Microsystems.

Enterprise Java Beans (EJB)

EJB technology is from Sun. See http://java.sun.com/products/ejb/. EJB refers to a specification for a server-side component model. RSM uses only stateless, session EJBs, which are stateless and clusterable, and which offer a remotely accessible entry point to an application server.

Enterprise Java Beans (EJB) container

An EJB container is the physical context in which EJBs exist. A container is a physical entity responsible for managing transactions, connection pooling, clustering, and so on. This container manages the execution of enterprise beans for J2EE applications.

J2EE server

The runtime portion of a J2EE product. A J2EE server provides EJB and Web containers.

The Java 2 Enterprise Edition (J2EE)

The Java standard infrastructure for developing and deploying multi-tier applications. Implementations of J2EE provide enterprise-level infrastructure tools that enable such important features as database access, client-server connectivity, distributed transaction management, and security.

Remote interface

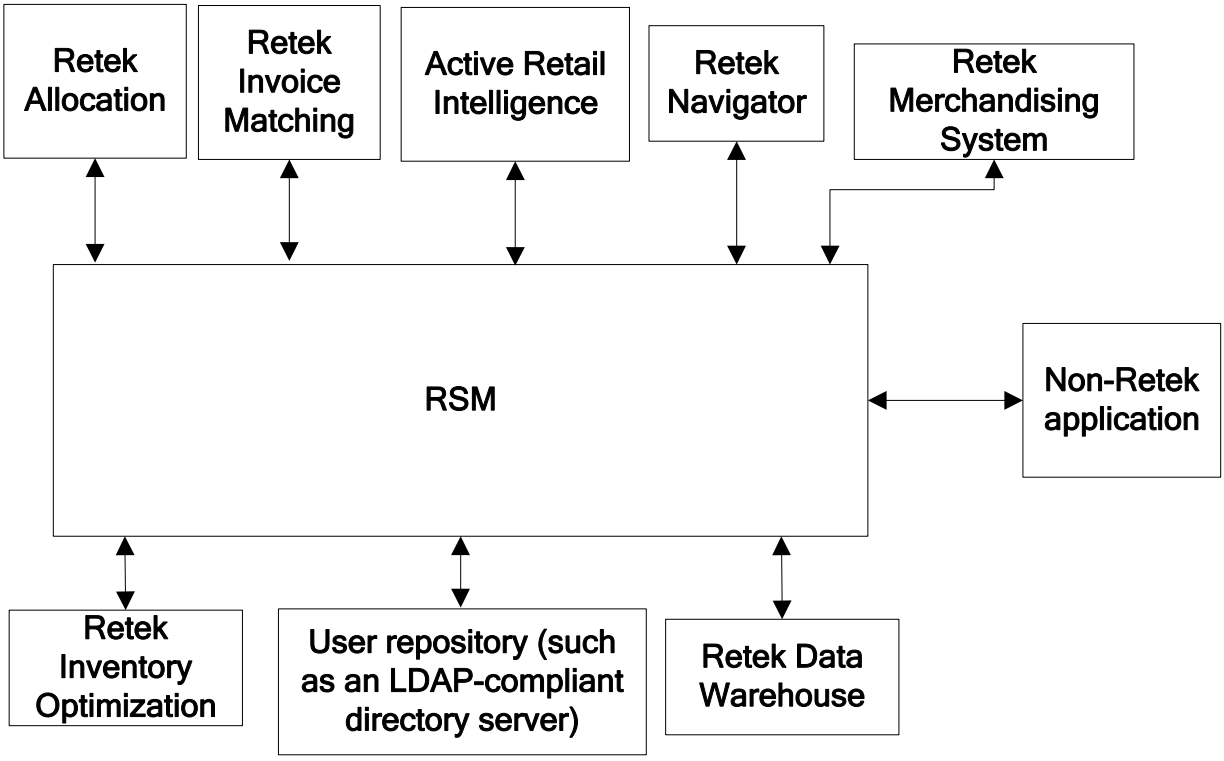
The client side interface to a service. This interface defines the server-side methods available in the client tier.

Chapter 4 – Integration interface dataflows

This chapter provides a functional overview of how RSM integrates with other Retek systems.

Overview

The diagram below illustrates the various Retek products and other systems that RSM interfaces with as well as the overall dataflow among the products. The accompanying explanations are written from a system-to-system perspective, illustrating the movement of data. Note that the data described in this chapter is not comprehensive but is intended to provide a high-level overview.



RSM dataflow across the enterprise

From application(s) to RSM

- User and password data or user signature that requires authentication
  When users enter their name and password into an application or when an application is called to send a user signature, the data is sent to RSM for authentication purposes.
- Named permissions (data seeding)
  Any application (Retek or non-Retek) that is utilizing RSM populates RSM tables with named permissions through a data seeding script.
- Data for data level permissions

  Applications may choose to administer their data level permissions through RSM. To do so, they must implement an RSM interface and return application specific data to RSM when requested.

From the LDAP-compliant directory server to RSM

Note: RSM never writes to the LDAP-compliant directory server.

- User data
  The user repository (such as an LDAP-compliant directory service) is the holder of user data for authentication purposes. RSM calls the user repository (such as an LDAP-compliant directory service) to retrieve necessary user attribute information and to store user names in RSM for the purpose of mapping roles to users and for displaying user names on the screen.
- Authentication verification
  RSM calls the LDAP compliant directory service to authenticate username and password data. Once a user is authenticated, RSM creates an encrypted user signature (unique to that user).

From RSM to another application

Note: Some Retek applications do not allow pre-authenticated users to be passed in. RSM provides for the storage and retrieval of username and password data for those applications.

- Authentication
  RSM analyzes the associated user signature and determines whether the user is authenticated without requiring a separate call back to the third party LDAP compliant directory service. Because of the existence of the user signature, other user signature-enabled applications 'know' that a user can be passed in.
- Authorization
  RSM returns information so that other applications can authorize users to engage in certain business functionality.
- Hierarchy Level / Data Level Permissions
  For applications that administer data level permissions through RSM, RSM calls out to the application services to get application specific data.

Chapter 5 – Functional design

Named permissions

One of RSM's primary purposes is to establish who has access to what business functionality. To facilitate this processing, any application (Retek or non-Retek) that is utilizing RSM populates RSM tables with named permissions. These are pieces of business functionality around which the Retek application has security. For example, if RPM has 'promotions' functionality surrounded by security, RPM creates a 'promotions' named permission. Named permissions data is sent to the RSM database during installation.

An RSM user could never change a named permission because the applicable outside application must respond to it. That is, once a user logs into an application (Retek or non-Retek), the application accesses RSM to request all the named permissions for this user. Within RSM, a user has a collection of roles, and roles have a collection of named permissions. For example, when the RPM user logs in, RSM provides the named permissions. RPM, in turn, asks 'does this user have 'promotion' capability'? If the user does *not* have the capability, RPM does not display that functionality for the user.

Actions and named permissions

When each RSM-integrated application populates the RSM database with named permissions (during installation), the application specifies potential actions that are possible against a named permission. Named permissions contain flags that determine specific actions (shown below) that can be taken in the system. For example, RPM might have a named permission script for Promotions that specifies the following for the actions:

- Edit:'`true`'
- View: '`true`'
- Approve: '`false`'
- Submit: '`false`'
- Emergency: '`false`'

The result of RPM's script would be that users in the RPM system could only be assigned 'view', 'edit' or no action with respect to Promotions functionality.

Type of action	Description
none	Users associated with the role have access to the permission but no actions.
Edit	Users associated with the role are allowed to create, update, and save any changes to a workflow.
View	Users associated with the role are allowed to see to all secured information in a workflow, but not make any changes to the data in the workflow.
Approve	Users associated with the role are allowed to change the status of a workflow to Approved
Submit	Users associated with the role are allowed to change the status of a workflow from Worksheet to Submitted.
Emergency	Users associated with the role are granted special access that goes beyond normal day-to-day access to functionality. They can thus bypass normal delays in processing.

Content models and named permissions

A content model defines in an xml document the task groups and tasks that are displayed in the task pad of a Retek GUI application window. By defining a content model and assigning named permissions to the content model attributes, applications can login to RSM and retrieve secure content in return. For example, an administrator can configure an application's content model to restrict certain tasks that are visible and/or editable by specific users. This is done by configuring named permissions in conjunction with content model tasks.

A Retek application's content model must be deployed with the RSM server. Check the Retek application's documentation before modifying an application's named permission settings.

Hierarchy (data level) permissions

RSM administers hierarchy (data level) permissions. To facilitate this functionality, any Retek application utilizing RSM for data level permissions populates RSM tables with its hierarchy types (that is, merchandise and location). Applications either provide the details of these types up front with SQL scripts or dynamically by implementing an RSM interface and exposing it to the RSM service. RSM does not understand application specific data (for example, RSM does not know the difference between departments and locations). To RSM the data is a tag (for example, department) and a specific value (for example, 1000). This information is passed back to calling applications, and it is the applications responsibility to apply the data level permissions appropriately.

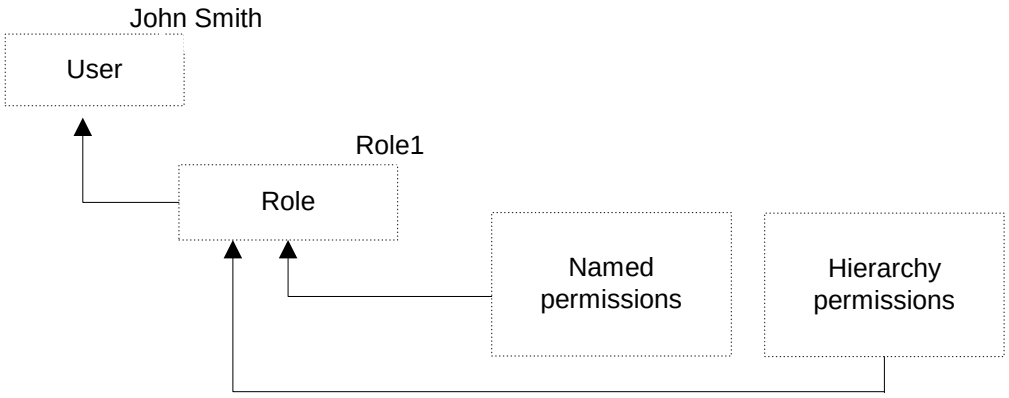
For example, when an RPM user logs in, RSM provides the hierarchy permissions for the user. RPM, in turn, asks 'does this user have access to 'Department 1000'? If the user does *not* have access, RPM does not display data from this department to the user. Like named permissions, within RSM a user has a collection of roles, and roles have a collection of hierarchy permissions.

Roles and users

RSM allows for the creation of security roles. Roles consist of a unique identifier, an arbitrary name, and any number of permissions. Roles are created by the retailer and are used as a mechanism for administering its security requirements.

As the diagram below illustrates, roles are used as a mechanism for grouping any number of permissions. The role then is assigned to various users.

The security administrator assigns permissions to roles. To continue the earlier example, the security administrator could only assign a role with 'view' or 'edit' promotions functionality. Suppose that the security administrator decided to assign a role with 'view' (a '`true`' flag behind the scenes) but not edit (a '`false`' flag behind the scenes), the security administrator could then assign a user, John Smith, to that role. John Smith could only view Promotions functionality.



The relationship among permissions, roles, and users

Password mapping

RSM contains functionality for administering Retek users and passwords. These users and passwords are used in conjunction with Retek Navigator to achieve sign-on across applications. For example, an enterprise user may have copies of their RMS (Oracle) user and password persisted in the RSM database. When users access RMS via Retek Navigator, they are prompted to log in again. The RSM user interface provides functionality to create, update or delete usernames and passwords. These users and passwords are not read from or persisted to any LDAP Directory Server or other enterprise data store.

Note: RSM is not the database of record for these values, and there currently is not automated data synchronization across applications. If either the user name or the password changes outside of RSM, either one must be manually updated using the RSM user interface.