edocs®

BLM Reference Guide

V4.0
Document ID: TSGN-05-4.0-01
Date Published: 9.10.03

Preface

In This Section

Using this Manual

Welcome to the BLM Reference Guide.

This manual is a reference guide to the BLM and its components.

Before You Get Started

You should be familiar with the following:

- Your application architecture
- Programming Java and Java Server pages
- Designing or working with databases
- eXtended Markup Language (XML)

Who Should Read this Manual

This manual is a reference guide for the Business Logic Manager (BLM). This guide is for anybody who needs information about the objects in the BLM and how they are organized.

- Administrators

  You will find information about configuring the authentication of users and applications in the BLM. There is also some information on configuring access to BLM objects. You should also find the BLM Configuration File Reference useful when you work on deployment of your solution.

- Developers

  This guide contains all the reference information you need when working with the objects in the BLM. This guide covers some of the basic configuration as it relates to your solution. In order to help you easily understand how the BLM is organized, this guide also contains diagrams of the different packages. You can also learn about how the BLM API and its object types. You may also want to consult the list of configuration files for information about these files.

- Project Architect

  The most useful part of this document is the organization of BLM objects. You use this guide to learn the content of the packages and how the objects are organized. You should also review the section on configuring the BLM because this covers security. This section not only covers limiting access to BLM objects, it also covers the authentication of users and applications. You also learn how to declare users or applications as being trusted, meaning that they are not required to login.

- Project Manager

  As this reference guide contains information about the BLM and its packages, APIs and security settings, you will find almost all of the information in this guide as useful when managing your project.

How this Manual is Organized

This manual contains the following chapters:

- **Overview of the BLM**

  This chapter covers the basics of the Business Logic Manager (BLM) and its role in Account Management solutions.

  It contains information about:

  - The BLM and its components
  - How the BLM works with other components

- **Configuring the BLM**

  This chapter covers the configuration of the BLM.

  It contains information about:

  - Configuring the access to BLM objects
  - Displaying rate plans
  - Configuring user authentication

- **BLM Object Reference**

  This chapter covers the contents of the BLM.

  It contains information about:

  - How the BLM is organized
  - BLM object types
  - Organization of the APIs
  - BLM API packages
  - BLM Interface packages

- **BLM Configuration File Reference**

  This appendix covers the BLM configuration files.

  It contains information about their:

  - File name

- Location
- Use

What Typographical Changes and Symbols Mean

This manual uses the following conventions:

TYPEFACE	MEANING	EXAMPLE
Italics	Manuals, topics or other important items	Refer to *Developing Connectors*.
Small Capitals	Software and Component names	Your application uses a database called the CID.
Fixed Width	File names, commands, paths, and on screen commands	Go to `//home/my file`

Finding the Information You Need

The product suite comes with comprehensive documentation set that covers all aspects of building Account Management solutions. You should always read the release bulletin for late-breaking information.

Getting Started

If you are new to the edocs Telco Solutions, you should start by reading *Introducing Telco Service Manager*. This manual contains an overview of the various components along with a list of the available features. It introduces various concepts and components you must be familiar with before moving on to more specific documentation. Once you have finished, you can read the manual that covers different aspects of working with the application. At the beginning of each manual, you will find an introductory chapter that covers concepts and tasks.

Designing Your Solution

While reading *Introducing Telco Service Manager*, you should think about how the different components can address your Account Management Solution's needs.

You can refer to *Developing Telco Service Manager* for information about extending the object model, application security, and other design issues. The *CID Reference Guide* also gives you the information about how the information in your solution is managed and stored.

Installing Your Telco Service Manager

You should start by reading the Release Bulletin. For detailed installation and configuring information, refer to *Installing Telco Service Manager*. This manual covers installing TSM on one or more computers. It also contains the information you need to configure the different components you install. You might also refer to *Developing Telco Service Manager* and *Developing Connectors for Telco Service Manager* as these manuals contain information on customizing applications and working with other software.

Building Account Management Solutions

If you are designing and programming *Telco Service Manager*, you have several different sources of information. If you are programming the user interface of the solution, you should read *Developing User Interfaces for Telco Service Manager*. You also refer to the BLM Specification for detailed information about programming the user interface. For configuring the various components, you refer to *Installing Telco Service Manager*and sections in other documents that deal with the component to configure.

If you are working with the business logic of your solution, you should read *Developing Telco Service Manager*. You can also refer to the *BLM Reference Guide* for more information about the design and structure of the BLM object model. For information about how this information is stored, you should refer to the *CID Reference Guide* along with the CID Reference documentation for your database. In order to develop your application, you most likely will need to install and run the Loopback Connector. This component mimics back-end applications for development purposes. For information about installing and running this component, refer to *Using the Loopback Connector with Telco Service Manager*.

Integrating Account Management Solutions

If you are involved in configuring your solution to work with Operation Support Software (OSS), you should read *Developing Connectors with Telco Service Manager*. This manual helps you understand the integration architecture and shows you how to build connectors to connect to today's market-leading OSS software. You can also read *Using the Loopback Connector with Telco Service Manager* for information about a connector built for development purposes. Other manuals you can refer to for information about configuring your application include *Introducing Telco Service Manager* and *Developing Telco Service Manager*.

Managing Telco Service Manager (TSM)

If you are responsible for managing TSM, you should read the *Installing Telco Service Manager* for information about configuring various components and information about working with different application servers. *Administrating Telco Service Manager* covers what you need to know about managing your solution at runtime. For information about OSS systems, you should read *Developing Connectors with Telco Service Manager*.

If You Need Help

Technical support is available to customers who have valid maintenance and support contracts with edocs. Technical support engineers can help you install, configure, and maintain your edocs application.

To reach the U.S. Service Center, located in Natick, MA (Monday through Friday 8:00am to 8:00pm EST):

- Telephone: 508.652.8400
- Toll Free: 877.336.3362
- E-support: support.edocs.com (This requires a one-time online registration)
- E-mail: support@edocs.com

When you report a problem, please be prepared to provide us the following information:

- What is your name and role in your organization?
- What is your company's name?
- What is your phone number and best times to call you?
- What is your e-mail address?
- In which edocs product did a problem occur?
- What is your Operating System version?
- What were you doing when the problem occurred?
- How did the system respond to the error?
- If the system generated a screen message, please send us that screen message.
- If the system wrote information to a log file, please send us that log file.

If the system crashed or hung, please tell us.

Contents

CHAPTER 1

Overview of the BLM

In This Section

About the BLM

A key part of the CSS Engine, the Business Logic Manager (BLM) enables Communications Service Providers to rapidly implement an e-business solution tailored precisely to their operational requirements and business processes.

The BLM contains a set of Java packages that correspond to business objects you can manage in your application. The BLM also comes with a set of APIs you use in your JSPs to access and manipulate these objects.

About the BLM and TSM

The CSS Engine holds the core application logic of your application, both transactional logic, allowing customers to view and to make changes to their contract and other relationship data; and analytical logic, allowing enterprise customers and other users to analyze contract, billing, usage and other relationship data from a number of different perspectives.

The object model for customer self-service comprises a comprehensive suite of Java (J2EE) packages, each addressing a specific aspect of service delivery for the customer. The CSS Engine includes multiple parallel models of the customer's business hierarchy (for enterprise customers); a flexible service catalog designed to meet the extended personalization needs of 3G wireless services; and a detailed representation of the customer's mobile contract or prepaid account, as well as trouble ticketing; electronic billing; and application personalization options for each user. The CSS Engine also provides a set of secure data analysis objects to retrieve and build cost, usage, transactional and other analysis reports; and a persistent service-centric shopping cart supporting the selection, configuration and purchase of complex mobile service contracts, together with advanced enterprise features such as order templates, bulk ordering, and dedicated corporate offers.

The CSS Engine includes:

- The customizable Business Logic Manager (BLM)
- Set of APIs to manage information in the CID
- The Data Access Layer (DAL) that manages the handling of data in the CID

The Data Access Layer (DAL) is responsible for passing data between the physical data repositories and process and the Business Logic Manager and (indirectly) the SmartLink Framework. Although data within the system can be regarded as being stored in a single location - the CID - the DAL allows a more flexible architecture.

The DAL is able to access any external data (for example: detailed billing) via standard SQL statements or by calling any specific Java API provided by the back end system. It manages the routing of data to and from the correct location, isolating the business logic from this complexity. If more servers are added, the complexity increases but the BLM is still protected by the DAL.

For more information about:

- The BLM APIs, refer to the *BLM API Reference Documentation*.

- Security and using LDAP or other methods of authentication, refer to *Managing Security* in *Developing Telco Service Manager*.

- Configuring and customizing the behavior of the BLM, refer to the *BLM Reference Guide*.

- Working with the DAL and accessing external data sources, refer to *Accessing External Data Sources* in *Developing Telco Service Manager*.

CHAPTER 2

Configuring the BLM

In This Section

About Configuring the BLM

A key part of the CSS Engine, the Business Logic Manager (BLM) enables Communications Service Providers to rapidly implement an e-business solution tailored precisely to their operational requirements and business processes.

The BLM contains a set of Java packages that correspond to business objects you can manage in your application. The BLM also comes with a set of APIs you use in your JSPs to access and manipulate these objects.

You can configure:

- Access to BLM objects
- Authentication modes
- Display of rate plans
- Trust modes

Managing Access to BLM Objects

When developing applications that have several users and organizations, TSM has to be able to limit access to certain features. For example, TSM may want to allow some users to change their billing address or rate plans. At the same time, you may also want some users to be able to view such information but not be allowed to change it.

You can easily manage the access to the BLM and the CID. By using the built-in security, you can assign access rights for any user of the application. This security includes user scopes and roles. The CID manages the users and different roles. You associate roles with scopes in the `security.xml` customization file to set the access rights to each object and its different actions (or features) in the BLM.

Managing access to the BLM objects and features involves:

- Creating User Roles in the CID
- Choosing the User Scopes that correspond to the roles
- Assigning right to the objects

When modifying change security configurations and settings, you need to restart your application server to take into account your changes.

Creating and Configuring User Roles

User roles are similar to job descriptions or titles. User roles describe the role of the user in everyday terms. By using user roles, you can adapt your authorization rules to match any organization or company. You can create as many roles you like and each user has one or more roles.

User roles are located in the `ROLE_DEF` table in the CID. After creating a user role, you can assign the role to a user login.

This `ROLE_DEF` table contains the following default roles:

ROLE_ID	ROLE NAME	DESCRIPTION
0	SYSTEM	Internal role used by system components which require access to all objects and features such as the Synchronizer agent.
3	GUEST_FIRSTLOGIN	Role for guest users when using the application for the first time to create their login
4	GUEST_NEWCUSTOMER	Role for guest users
2	TRUSTED	Internal role for trust mode
12	SUBSCRIBER	Role for service and contract subscribers

ROLE_ID	ROLE NAME	DESCRIPTION
13	DEALER	Role for point of sales dealers or other distributors to manage residential or small business customers
14	TELCO	Role for Communication Service Providers or other service providers to manage their users
15	SUPPLIER	Role for dealers to be managed like a user in an organization
16	TELCO BUSINESS ACCOUNT MANAGER	Role for Communication Service Providers or other service providers to manage specified large business customers
17	TELCO SENIOR BUSINESS ACCOUNT MANAGER	Role for Communication Service Providers or other service providers to manage all large business customers
18	CONTRACT CUSTOMER ADMINISTRATOR	Role for users that are responsible for managing a set of contracts
22	CUSTOMER ADMIN	Role for users that are responsible for managing a set of users
23	DEALER ADMIN	Role for users that are responsible for managing a set of dealers
24	TELCO ADMIN	Role for users that are responsible for managing a set of telcos
25	SUPPLIER ADMIN	Role for users that are responsible for manageing a set of suppliers

You also can assign a priority to different roles. When processing requests, the priority sets the order in which the messages are processed.

For more information, refer to *Specifying Processing Priority* in *Developing Connectors*.

To create a user role in the CID

1 In the `ROLE_DEF` table, add a new record then enter the following mandatory information:

- `ROLE_ID` This number is used to assign this role to users
- `ROLE_CODE` This name corresponds to the name used to define access rules in the BLM configuration file and the JSPF

2 If needed, enter a `ROLE_NAME`, `STRING_ID` and other optional information.

To assign a role to a login

As users can have more than one job function (for instance, a user may be both a dealer administrator and supplier administrator), logins can have more than one associated role.

To create the login, you use the `UserF.createUserLogin` method. This method has an attribute where you can pass an array of roles to assign to the login.

To change the roles associated with the login, use the `UserF.doModifyLogin` method.

When changing the roles, you must pass the entire list of roles to associate with this user:

- To add a role, you pass both the current roles and the new role.
- To remove a role, you pass the list of roles except the role you want to remove.

For more information about the `UserF` methods and their attributes, refer to the *BLM API Reference Documentation*.

Choosing User Scopes

User scopes define the general access rules to BLM objects. Not only do scopes determine access, they also let you to limit certain features of your applications to specific roles. Associated with a User Role, these scopes determine what objects can be accessed according to the user's member and organization level information.

The types of scopes include:

- Intra-organization scopes
  - All objects of the organization hierarchy
  - All objects related to the member's level or below
  - All objects owned by the member including associated sub-objects
  - All objects explicitly managed by the member (limited to contracts and members)
- Inter-organization scopes
  - All organizations
  - All organizations managed by the member's level
  - All organizations managed by the top level of the member's organization
  - All organizations managed by the member

You use these scopes in the BLM configuration file to associate a user role with a user scope. In this file you assign the role and scope for each object. When assigning access rights to different features, if the scope is an inter-organization scope, you can specify a list of organization types that have access to the object. If you do not specify the organization types, all organization types have access to the object.

The scopes include:

SCOPE	TYPE	DESCRIPTION
OrganizationScope	Intra	Users can access all objects within the organization to which they belong
SubHierarchyScope	Intra	Users can access all objects related to the member's level or below
MemberScope	Intra	All objects owned by the member including associated sub-objects
ExplicitScope	Intra	Users have access to members or contracts they explicitly manage. These objects must belong to the same organization as the user.
SystemScope	Intra Inter	Users have access to all objects
MemberManagedScope	Inter	Users can access objects within the organizations they manage
LevelManagedScope	Inter	Users can access all objects within organizations managed by their level
OrganizationManagedScope	Inter	Users can access all objects within organizations managed by their organization
ExternalOrganizationScope	Inter	Users can access all objects in other organizations they do not belong to

Assigning Rights to Objects

Assigning access rights to objects in the BLM involves setting the access rules for each BLM feature users can access. A BLM feature is an action that can be carried out using the objects of the BLM.

To authorize access to a BLM feature, you enter a list of authorized roles and their associated scopes and filters in the BLM `security.xml` file. This customization file centralizes all of the access rights to objects in the BLM. This file is located in `<home_dir>/classes/nmycfg/blm`.

Assigning rights to objects involves:

- Modifying access rights of an existing role:
  - Adding access rights to a BLM feature
  - Modifying the scopes associated with the access rights of a BLM feature
  - Remove access rights from a BLM feature

- Removing all access rights of an existing role
- Specifying access rights of a new role

When working with the `security.xml` file, we strongly recommend working with an XML editor that can display XML files in tabular form. When working with this file, you must first find the features that you want to change. By using a tabular display, you can easily sort features by their name, functional domain or impacted object. Not only is it easier to find the BLM feature this way, editing and modifying settings are much easier.

---

security.xml File Format

The contents of the `security.xml` customization file include:

- `<security>` element
- `<checkpoint>` elements
- `<ROLECODE>` elements

The tags and attributes of this file are case sensitive. When working with this file, be sure to respect the case of these elements.

This sample shows the structure:

```
<security >

    <checkpoint functionaldomain="domain name" object="object name" action="feature name" securitypath="object
name">

          <ROLECODE>RoleScope1 RoleScope2(orgType1code , orgType2code….)</ROLECODE>

          ...

    </checkpoint>

    <checkpoint>

    ...

    </checkpoint>

</security>
```

For detailed information about the structure of this file, refer to the HTML *security.xml Reference* documentation.

<security> Element

This element is the main element of the `security.xml` customization file.

This element has no attributes and contains one or more `<checkpoint>` elements.

<checkpoint> Element

This element corresponds to a specific BLM feature. This element corresponds to a BLM action that you can assign access rights.

The `<checkpoint>` tag has the following attributes:

- `functional domain`
- `object`
- `action`
- `securitypath`

This element contains one or more `<ROLECODE>` elements. These `<ROLECODE>` elements contain the information about which roles and scopes have the right to access the parent BLM action described in the `<checkpoint>` element.

The syntax of the element:

```
<checkpoint functionaldomain="domain name" object="object name"
action="feature name" securitypath="object name">
```

Do not remove a `<checkpoint>` element from this configuration file and never modify any of the `<checkpoint>` attributes.

functionaldomain Attribute

This attribute specifies the functional domain of the BLM feature.

VALUE	DESCRIPTION
Billing and Payment	Features related to billing accounts, invoices and payments
Contract management	Features for contract management from its creation to its modification (add service…)
Dedicated Offer management	Features for browsing of dedicated offers and managing their association with organizations
Hierarchy management	Features for creating, modifying, and browsing hierarchies (organizations, levels and members)
Notification	Features for requesting notification of an organization for synchronization with CID2CBU
Order Documentation	Features for ordering documentation for an organization or a member
Order Validation	Features for instantiating, updating, and approving/denying requests

VALUE	DESCRIPTION
Organization view management	Features for creating, modifying, and browsing organization views
Persistent shopping cart	Features for saving, updating, removing, and instantiating persistent shopping cart
Personal data management	Features for getting and setting personalization data of a member
Prepaid management	Features for reloading prepaid contracts
Security management	Features related to user access rights management from creating logins to modifying roles
Trouble ticketing	Features for creating, modifying and instantiating trouble tickets

object Attribute

This attribute specifies the object providing the BLM feature.

VALUE	DESCRIPTION
object name	the name of the object
Not applicable	the feature is not related to a specific object

action Attribute

This attribute specifies the name of the BLM feature.

VALUE	DESCRIPTION
feature name	the name of the feature

The name for instantiating an object always begins with `Get`.

securitypath Attribute

This attribute specifies the heirarchical objects used for security and is used to manage security for intragorganizational scopes.

For instance, a manager of a contract may not be declared as the responsible for payment. By not being the responsible for payment, the manager does not have the right to access the subinvoice of a managed contract. You can use this element to specify that accessing a contract subinvoice object is subject to the security of the Contract object.

This attribute helps you understand how security is handled for each feature.

The value of this attribute determines the list of scopes that are valid for the feature:

SCOPE/SECURITY PATH	ORGANIZATION	MEMBER	CONTRACT	NOT APPLICABLE
OrganizationScope	YES	YES	YES	NO
SubHierarchyScope	YES	YES	YES	NO
MemberScope	NO	YES	YES	NO
ExplicitScope	NO	YES	YES	NO
ExternalOrganizationScope	YES	YES	YES	NO
OrganizationManagedScope	YES	YES	YES	NO
LevelManagedScope	YES	YES	YES	NO
MemberManagedScope	YES	YES	YES	NO
SystemScope	YES	YES	YES	YES

If different objects apply security to a feature depending on the context, the less restrictive rules apply. For example, Add trouble ticket for an invoice depends on the payment responsible that could be a member or a level.

<ROLECODE> Element

This element specifies the scopes of the permitted roles of the feature.

The name of the element and tag correspond to the value of the `ROLE_CODE` specified in the `ROLE_DEF` table.

For example, there is a role for subscribers. In the `ROLE_DEF` table, the `ROLE_CODE` of this role is `SUBSCRIBER`. The corresponding `<ROLECODE>` element in the `security.xml` configuration file uses the `<SUBSCRIBER>` element.

The tag has no attributes and contains the following:

- One or more scope names separated by a blank space.
- If a scope accepts organization types (extra organization scopes), their organization type codes are specified in parentheses and separated by a comma.

The syntax of the element:

```
<ROLECODE>RoleScope1
RoleScope2(orgType1code,orgType2code…)</ROLECODE>
```

If a `<ROLECODE>` element is empty, access to the feature is denied.

Example of security.xml for

This example shows the settings for the modifying contract BLM feature. This `<checkpoint>` element has the following attributes:

- functional domain = Contract Management
- object = Contract
- action = Modify
- securitypath = Contract

There are several different `<ROLECODE>` elements declaring the security settings of this BLM action for each role.

The <security> root element start tag	`<security>`
	 `...`

The <checkpoint> element start tag and attributes	`<checkpoint functionaldomain="Contract Management" object="Contract" action="Modify" securitypath="Contract">`

The list of role elements along with the authorized scopes and organization types	`<SUBSCRIBER>MemberScope</SUBSCRIBER>` `<CUSTADMIN>SubHierarchyScope</CUSTADMIN>` `<CONTRACT_CUSTADMIN>MemberScope ExplicitScope</CONTRACT_CUSTADMIN>` `<DEALER>ExternalOrganizationScope(CONSUMER)</DEALER>` `<TELCO>ExternalOrganizationScope(CONSUMER)</TELCO>` `<TELCO_ACCT_MGR_SR>ExternalOrganizationScope(BUSINESS)</TELCO_ACCT_MGR_SR>` `<TELCO_ACCT_MGR>MemberManagedScope</TELCO_ACCT_MGR>`
The `<checkpoint>` element end tag	`</checkpoint>` ...
The `<security>` root element end tag	`</security>`

Modifying the Access Rights of Existing Roles

To add access rights to a BLM feature

1 Go to `<home_dir>/classes/nmycfg/blm`.

2 Open `security.xml`.

3 Find the `<checkpoint>` element that corresponds to the BLM feature.

4 Find the `<ROLECODE>` element corresponding to the role to modify.

5 In the `<ROLECODE>` element, add the scopes. Use the following syntax:

```
<ROLECODE>RoleScope1 RoleScope2(orgType1code,orgType2code…)
</ROLECODE>
```

6 Save your changes.

To modify the access rights of a BLM feature

1 Go to `<home_dir>/classes/nmycfg/blm`.

2 Open `security.xml`.

3 Find the `<checkpoint>` element that corresponds to the BLM feature.

4 Find the `<ROLECODE>` element corresponding to the role to modify.

5 In the `<ROLECODE>` element, modify the list of scopes. Use the following syntax:

```
<ROLECODE>RoleScope1 RoleScope2(orgType1code,orgType2code…)
</ROLECODE>
```

6 Save your changes.

To remove access rights from a BLM feature

1 Go to `<home_dir>/classes/nmycfg/blm`.

2 Open `security.xml`.

3. Find the `<checkpoint>` element that corresponds to the BLM feature.
4. Find the `<ROLECODE>` element corresponding to the role to modify.
5. Do one of the following:
   - To remove specific access rights, remove the scope from the list
   - To remove all of the access rights, remove the entire list of scopes
6. Save your changes.

Make sure you do not remove the feature that is declared in the `<checkpoint>` element. To remove access to a specific feature, edit the `<ROLECODE>` elements.

Removing All Access Rights from an Existing Role

To remove all access rights of a role

1. Go to `<home_dir>/classes/nmycfg/blm`.
2. Open `security.xml`.
3. For each `<ROLECODE>` element corresponding to the role, remove the entire list of scopes and organization types.
4. Save your changes.

Make sure you remove the contents for all of the `<ROLECODE>` elements in this file.

Specifying Access Rights of a New Role

To declare access rights of a new role

1. Go to `<home_dir>/classes/nmycfg/blm`.
2. Open `security.xml`.
3. Find the `<checkpoint>` element that corresponds to the BLM feature.
4. Add a `<ROLECODE>` element corresponding to the new role.
5. In the new `<ROLECODE>` element, enter the list of scopes. Use the following syntax:

```
<ROLECODE>RoleScope1 RoleScope2(orgType1code,orgType2code…)
</ROLECODE>
```

6. Repeat for each feature `<checkpoint>` element that corresponds to the BLM feature you want the new role to access.
7. Save your changes.

Make sure that you do not declare the same role twice for a given feature.

If you use the `security.xsd` to check the format of your `security.xml` customization file, you need to declare the new role. For more information about the security.xsd file, refer to *Validating the Format of the security.xsd File*.

Validating the Format of the security.xml File

To help you verify the format and contents of the `security.xml` customization file, you can use the `security.xsd` schema. This file is located in `<home_dir>/classes/nmycfg/blm`.

Your XML editor uses this schema to check the following:

- XML Syntax
- Validity of the scopes
- Unicity of the declared roles

The `security.xsd` file is only for checking the format and content of your `security.xml` customization file while developing TSM. It is not for production environments and should never be used at runtime.

To declare a new role in the security.xsd file

1. Go to `<home_dir>/classes/nmycfg/blm`.
2. Open `security.xsd`.
3. Find the `<xs:complexType name="ACL">` element.
4. Under the `<xs:all>` element, go to the end of the list of `<xs:element>` elements .
5. Declare your role by creating a new `<xs:element>`. Use the syntax:
   ```
   <xs:element name="<NEWROLECODE>" type="scopes" minOccurs="0"/>
   ```
6. Save your changes.

Example of a new role in the security.xsd file:

```
...
<xs:complexType name="ACL">
   <xs:all>
      <xs:element name="SUBSCRIBER" type="scopes" minOccurs="0"/>
      <xs:element name="CUSTADMIN" type="scopes" minOccurs="0"/>
      ...
      <xs:element name="<NEWROLECODE>" type="scopes" minOccurs="0"/>
   </xs:all>
...
```

Configuring the Display of Rate Plans

The BLM manages access to data in both the CID and external data sources which can lead to some issues when applying your business logic to the display of rate plan information.

When a customer changes rate plans, what rate plan information do you want to display? From the customer's point of view, they have signed up for the new rate plan and are not concerned with the data processing and validation of their change request. Does your business logic require approval of such changes before letting customers access information about their new rate plans? Or does it grant tacit approval of their request so they can see information about the new rate plan they just signed up for?

You can configure the BLM to display either the information for the existing rate plan or new rate plan.

Configuring the display of rate plans involves:

- Specifying the way to display rate plans in the `config.xml` configuration file

To change the display of rate plans

1. Go to `<home_dir>/classes/nmycfg/blm`.
2. Open `config.xml`.
3. Do one of the following:
   - To display the current rate plan until approved by the legacy system, change the `UseRequestedRatePlan` value attribute to `false`.
   - To display the new rate plan without approval of the legacy system, change the `UseRequestedRatePlan` value attribute to `true`.
4. Save your changes.

Configuring Authentication

You can use either the CID or LDAP authentication method to grant users access to your application. By default, the CID is used for authentication.

You use the `functionlist.xml` file to configure the authentication method to use and other settings such as the one-way hash function to use to encrypt passwords in the CID. This file is located in `<home_dir>/classes/nmycfg/dal`.

Using LDAP involves:

- Specifying the authentication method to use in the `functionlist.xml` file:
  - JNDI
  - Netscape Directory
- Activating the method to use in the `LDAP.xml` file
- Specifying the servers to use and their port numbers in the `LDAP.xml` file

Specifying Password Encryption

If you use the CID for authentication, you can choose which one-way hash function to use to encrypt passwords.

You can use one of the following:

- MD5
- SHA
- None

To specify the encryption method

1. Go to `<home_dir>/classes/nmycfg/dal`.
2. Open `functionlist.xml`.
3. Go to the `<encryption_scheme>` element.
4. Do one of the following:
   - To use MD5, change the element to:

     `<encryption_scheme>MD5</encryption_scheme>`
   - To use SHA, change the element to:

     `<encryption_scheme>SHA</encryption_scheme>`
   - To not encrypt passwords, change the element to:

     `<encryption_scheme></encryption_scheme>`

5 Save your changes.

Example of MD5 Encryption

```
<XML_CONFIGURATOR __IS_HASH__="true">
...
<default_session_id>0</default_session_id>

<function_list>com.netonomy.dal.api.DalFunctionList</function_list>

<authenticator>com.netonomy.dal.api.authenticator.CidAuthenticator</authenticator>

<encryption_scheme>MD5</encryption_scheme>
<route_session_id>0</route_session_id>
<use_route_table>false</use_route_table>
<use_function_routing>true</use_function_routing>
</XML_CONFIGURATOR>
```

Specifying the Authentication Method

You specify the authentication method in the `functionlist.xml` file.

To specify the authentication method

1 Go to `<home_dir>/classes/nmycfg/dal`.

2 Open `functionlist.xml`

3 Find the `<authenticator>` element.

4 Do one of the following:

- To use the CID, change the `<authenticator>` element to:

  `<authenticator>com.netonomy.dal.api.authenticator.CidAuthenticator</authenticator>`

- To use LDAP via JNDI, change the `<authenticator>` element to:

  `<authenticator>com.netonomy.dal.api.authenticator.ldap.JNDIAuthenticator</authenticator>`

- To use LDAP via Netscape, change the `<authenticator>` element to:

  `<authenticator>com.netonomy.dal.api.authenticator.ldap.NetscapeAuthenticator</authenticator>`

5 Save your changes.

Example of `functionlist.xml`

```
<XML_CONFIGURATOR __IS_HASH__="true">

<!--

#  CID Authentication:  <authenticator>com.netonomy.dal.api.authenticator.CidAuthenticator</authenticator>

#  LDAP Authentication:
<authenticator>com.netonomy.dal.api.authenticator.ldap.JNDIAuthenticator</authenticator>

#  LDAP Authentication:
<authenticator>com.netonomy.dal.api.authenticator.ldap.NetscapeAuthenticator</authenticator>

-->

<default_session_id>0</default_session_id>

<function_list>com.netonomy.dal.api.DalFunctionList</function_list>

<authenticator>com.netonomy.dal.api.authenticator.CidAuthenticator</authenticator>

<encryption_scheme>SHA</encryption_scheme>

<route_session_id>0</route_session_id>

<use_route_table>false</use_route_table>

<use_function_routing>true</use_function_routing>

</XML_CONFIGURATOR>
```

Specifying the Authentication Server

You specify the LDAP authentication servers and their ports in the `LDAP.xml`file.

In this file, you specify:

- The LDAP authenticator to use (JNDI or Netscape)
- The name and port of the read and write server
- Pooling information
- Login verification
- LDAP parameters

To specify the authenticator to use

1. Go to `<home_dir>/classes/nmycfg/dal`.
2. Open `LDAP.xml`.
3. Verify that the section that corresponds to your authenticator is not commented out.
4. Verify that all other authenticator sections are commented out.
5. Save you changes.

Example of LDAP.xml Configured for JNDI

Example of `LDAP.xml`configured for JNDI

```
<!-- Section to use for the standard JNDI LDAP driver (with
com.netonomy.dal.api.authenticator.ldap.JNDIAuthenticator) -->

<jndi_ldap_initial_ctx_factory>com.sun.jndi.ldap.LdapCtxFactory</jndi_ldap_initial_ctx_factory>

<jndi_ldap_provider_url>ldap://localhost:389</jndi_ldap_provider_url>

<jndi_ldap_security_authentication>simple</jndi_ldap_security_authentication>

<!-- Section to use for the Netscape LDAP JDK driver (with
com.netonomy.dal.api.authenticator.ldap.NetscapeAuthenticator) -->

<!--

<read_server_name>localhost:389</read_server_name>

<write_server_name>localhost:389</write_server_name>

<read_server_use_pool>true</read_server_use_pool>

<read_server_min_pool>10</read_server_min_pool>

<read_server_max_pool>50</read_server_max_pool>

<write_server_use_pool>true</write_server_use_pool>

<write_server_min_pool>10</write_server_min_pool>

<write_server_max_pool>10</write_server_max_pool>

    -->
```

To specify the authentication server

1 Go to `<home_dir>/classes/nmycfg/dal`.

2 Open `LDAP.xml`.

3 For the read server, enter the name and port number in the `<read_server_name>` element using the following syntax:

```
<read_server_name>Server_Name:Port_Number</read_server_name>
```

4 For the write server, enter the name and port number in the `<write_server_name>` element using the following the syntax:

```
<write_server_name>Server_Name:Port_Number</write_server_
name>
```

5 For each server, you can:

- Enable or disable the use of the connection pool (true/false) using the following element:

  ```
  <read_server_use_pool>true/false</read_server_use_pool>
  ```

- Enter the minimum or maximum pool values using the following elements:

  ```
  <read_server_min_pool>10</read_server_min_pool>
  <read_server_max_pool>50</read_server_max_pool>
  ```

- Enter the regular expression for the login in the `<schema_DN_regexp>` element using the following syntax:

```
<schema_DN_regexp>your_login_regexp</schema_DN_regexp>
```

6 Enter the following LDAP parameters:

- User Distinguished Name in the `<schema_DN_template>` element using the following syntax:

  ```
  <schema_DN_template>uid=$login,o=customers,o=company.com
  </schema_DN_template>
  ```

- Object class hierarchy relative to the user in the `<schema_objectclass>` element using the following syntax:

  ```
  <schema_objectclass>top, Person</schema_objectclass>
  ```

- Object attributes (role and password) in the `<schema_password>` and `<schema_roles>` elements using the following syntax:

  ```
  <schema_password>userpassword</schema_password>
  <schema_roles>roles</schema_roles>
  ```

7 DN and password of the LDAP user with administrative rights in the `<schema_DN_regexp>` element using the following syntax:

```
<schema_DN_regexp>your_login_regexp</schema_DN_regexp>
```

8 Save your changes.

Example of LDAP.xml

Example of ***`LDAP.xml`***

`#  Multiple server example:`
`#        <read_server_name>Server1:portA Server2:portB Server3:portC</read_server_name>`
`#  Schema DN template examples:`
`#        <schema_DN_template>cn=$login,o=customers,o=netonomy.com</schema_DN_template>`
`#        <schema_DN_template>uid=$login, o=netonomy.com</schema_DN_template>`
`#        <schema_DN_template>uid=$login, objectclass=$param[0], o=netonomy.com</schema_DN_template>`	

Port numbers for 'read' and 'write' servers	`<read_server_name>Server:389</read_server_name>`
	`<write_server_name>Server:389</write_server_name>`

Connection pool values:
For each server, you can enable or disable the use of the connection pool (true/false)
Enter the minimum or maximum pool values for each server
	`<read_server_use_pool>false</read_server_use_pool>`
	`<read_server_min_pool>10</read_server_min_pool>`
	`<read_server_max_pool>50</read_server_max_pool>`
	`<write_server_use_pool>false</write_server_use_pool>`
	`<write_server_min_pool>10</write_server_min_pool>`
	`<write_server_max_pool>10</write_server_max_pool>`

Login verification	`<schema_DN_regexp>\$(login|[0-9]+)</schema_DN_regexp>`

LDAP parameters
 	
User Distinguished Name
 	
Object class hierarchy relative to user
 	
Object attributes (role and password)
 	
The DN and password of the LDAP user with administrative rights
	`<schema_DN_template>uid=$login,o=customers,o=netonomy.com</schema_DN_template>`
	
	`<schema_objectclass>top, netonomyPerson</schema_objectclass>`
	
	`<schema_password>userpassword</schema_password>`
	
	`<schema_roles>roles</schema_roles>`
	
	`<user_netonomy_DN>uid=admin, ou=Administrators, ou=TopologyManagement, o=NetscapeRoot</user_netonomy_DN>`
	
	`<user_netonomy_Password>admin</user_netonomy_Password>`

Setting Authentication Modes

The BLM manages the authentication of users. This means that the BLM checks to see if the user account exists and if the password is valid. If TSM is part of a large Internet site or if users are using handsets, they may already be authenticated and you do not want them to have to reenter logon information. In this case, the BLM uses the trust mode to authenticate users.

You application channels can use the trust mode. For example, users of the wireless application channel are considered as being trusted users and are not required to enter passwords for authentication. On the other hand, users using the Internet log in and enter their user name and password.

To handle these situations, the BLM authentication modes are:

- **Normal**

  In the normal authentication mode, the JSP creates a session then uses the standard authentication API using the user name and password.
- **Trust**

  In the trust authentication mode, the JSP creates a session then uses a trust API using the user name and the trusted login and password for authentication.

When using the trust API to create trusted sessions, you use the default `TRUSTED` user role to authenticate sessions.

Using the trust mode involves:

- Authenticating the user session in the JSP using the appropriate API and login

To authenticate a user session

1. In your JSP, create a user session.
2. Authenticate the session:
   - For Normal mode, use:
     ```
     boolean authenticate (String userLogin, String
     userPassword);
     ```
   - For Trust mode, use:
     ```
     boolean authenticate(String userLogin, String trustedLogin,
     String trustedPassword);
     ```

By default, the `trustedLogin` and `trustedPassword` are `trusted/trusted.` For production applications, be sure to change the password in the `LOGINS` table.

Example of TRUST Mode Authentication

```
//The following code shows how to use trusted mode api

// The first parameter is the login you want to be.

// The second and third are login/password of a user declared as trusted in the cid.



boolean blnAuthenticate;

...

String trustedLogin, trustedPassword;

// Here you can set the login and password of the trusted user to the trustedLogin and trustedPassword
variables.

//The following code shows how to use the standard authenticate api

blnAuthenticate = blmSession.authenticate(request.getParameter("managedLogin"), trustedLogin,
trustedPassword);
```

CHAPTER 3

BLM Object Reference

In This Section

About the BLM APIs

In web applications, the web server passes specific types of user requests to the application server for processing by the channel.

A channel is the presentation layer of TSM. This means that it generates the user interface of application. Parts of the user interface include information from the CID as it contains the data users want to access and modify in your application.

For example, if a customer wants to change addresses, the application has to extract and display the current address. Not only does the application display the information, it also has to give the customer the means to modify the information and submit changes. The application also has to check and see if the customer has the required authorizations to make such changes, check consistency and coherence, and insert the information into a request queue for processing or make the changes directly on the CID.

Instead of having to program all of the different processes from scratch, you can use the set of published BLM APIs. These APIs help you program your application quickly and easily. You can use them to manage BLM objects such as contracts and customers. Your server side presentation tools, such as JSPs, can access the BLM using these APIs. By using these APIs, you can create applications that manage information in the BLM in a simple and predictable way.

Based on UML, the BLM APIs are built on top of BLM runtime objects. These BLM objects represent one to one relationships with business objects such as contracts, customers, and so on. There are other APIs in the BLM that are internal objects that manage application objects such as sessions, requests, and so on.

About the Object Types in the BLM APIs

The BLM APIs access the following objects:

- `<object_type>F`

  A Facade object. A Facade object corresponds to a core business object that is in the CID. You use Facade objects to carry out actions that you can only do when the object is in the CID. These objects are classes and always have a default implementation.

  For example, you use the `ContractF` to view and change information about an existing contract that cannot be done using the standard `ContractIF`.

- `<object_type>IF`

  An Interface object. An Interface object corresponds to a business object. The Interface object gives you a set of methods to access and modifying information about the object. They also have accessors you use to retrieve attributes or members of the object. These objects do not necessarily have to exist in the CID and can be considered data containers for business objects.

  For example, when your workflow creates a new contract, you use `ContractIF` to create an instance of a contract and add all of the required information before saving it to the CID.

The BLM also comes with some object managers that make working with BLM objects easy. These object managers include:

- `ObjectRefMgr`

  This object manages access to reference information such as lists of countries, rate plans and other reference information.

- `ObjectMgr`

  This object manages common functions that are used to manage customer objects. For instance, it handles creating contacts, action manager, and so on.

- `ActionMgrIF`

  This object manages how the BLM manages changes to objects. You can send the changes directly back to the BLM which then inserts the request in the CID. Or you can create a shopping cart which allows users to add, remove, and modify objects then submit them for processing.

The BLM APIs are not distributed APIs. The server side presentation tool and the BLM object must run in the same Java Virtual Machine.

About the BLM Packages

The following packages make up the BLM API:

- `com.netonomy.blm.api` package

  This package contains the following technical and business packages:

  - `BillingAccount`
  - `Contract`
  - `Facade`
  - `Hierarchy`
  - `Organization`
  - `Request`
  - `TroubleTicket`
  - `User`
  - `utils`

- `com.netonomy.blm.interfaces` package

  This package contains the following technical and business packages:

  - `billing`
  - `contact`
  - `contract`
  - `documentation`
  - `event`
  - `notification`
  - `offer`
  - `organization`
  - `orgview`
  - `parameters`
  - `personalizationdata`
  - `rateplan`
  - `request`
  - `search`
  - `service`
  - `trouble`
  - `util`

- `validation`

These packages are in the `<home_dir>/lib/nmycore.jar` archive.

APPENDIX A

BLM Configuration File Reference

In This Section

About the BLM Configuration Files

You use several different configuration and customization files to set up and modify TSM's behavior.

The files are divided into the following categories:

CATEGORY	DESCRIPTION
Configuration	These files contain the default configuration. You can modify the information in these files to modify the default configuration to meet your deployment needs. Some of the configuration files also contain information that you enter during installation.
Customization	These files contain settings that you change to customize TSM. You can modify the information in these files to modify the default configuration to meet your specific application needs. Some of these files may be empty or contain sample information.

BLM Configuration Files

FILE NAME	LOCATION	DESCRIPTION
config.xml	classes/nmycfg/blm channels/WEB-INF/classes/nmycfg/blm	Specifies various BLM settings
instance_route.properties	channels/WEB-INF/classes/nmycfg/dal/instances config/approvalsequencer/nmycfg/dal/instances	Specifies the properties of the database connection
jsp_parameters.xml	classes/nmycfg/util/formatter channels/WEB-INF/classes/nmycfg/util/formatter	Specifies the localization formats for the JSPs
LDAP.xml	classes/nmycfg/dal channels/WEB-INF/classes/nmycfg/dal	Specifies the LDAP configuration
policy.properties	classes/nmycfg/blm/util channels/WEB-INF/classes/nmycfg/blm/util	Specifies the location of the error settings for each language
translator.properties	classes/nmycfg/util channels/WEB-INF/classes/nmycfg/util	Specifies the names of the configuration files containing the localization of BLM error messages
<language>.properties	classes/nmycfg/errors channels/WEB-INF/classes/nmycfg/errors	Specifies the location of the localized BLM error message file
core_<language>.properties	classes/nmycfg/errors channels/WEB-INF/classes/nmycfg/errors	Specifies the localized strings for the core BLM error messages. The filename is specified in the <language>.properties configuration file

BLM Customization Files

FILE NAME	LOCATION	DESCRIPTION
containers_customization.xml	classes/nmycfg/dal channels/WEB-INF/classes/nmycfg/dal	Specifies the DAL container routing properties
core_containers.xml	classes/nmycfg/dal channels/WEB-INF/classes/nmycfg/dal	Specifies the configuration of access to the DAL
core_queries.xml	classes/nmycfg/dal/instances channels/WEB-INF/classes/nmycfg/dal/instances	Specifies the SQL queries (each supported database has its own subdirectory and file)
functions_routing.properties	classes/nmycfg/dal channels/WEB-INF/classes/nmycfg/dal	Specified the DAL binding properties
external_custom.xml	classes/nmycfg/blm channels/WEB-INF/classes/nmycfg/blm	Specifies the BLM external classes and their custom extensions
instances.properties	classes/nmycfg/dal/instances channels/WEB-INF/classes/nmycfg/dal/instances	Specifies the location of the DAL instance connection configuration file
objectID_aliases.xml	classes/nmycfg/dal channels/WEB-INF/classes/nmycfg/dal	Specifies the DAL aliases for objects
security.xml	classes/nmycfg/blm channels/WEB-INF/classes/nmycfg/blm	Specifies access rights to BLM functions

Index