CA RiskMinder™

Web Services Developer's Guide

r3.1

ca technologies

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Getting Started

This guide provides information about how to use CA RiskMinder (referred to as RiskMinder later in the guide) Web services to enable your online application to programmatically perform risk evaluation and related tasks. This document describes the Web services implementation of RiskMinder.

RiskMinder is an adaptive authentication solution that evaluates each online transaction in real time by examining a wide range of collected data against the configured rules. It then assigns each transaction a risk score and advice. The higher the risk score, the greater is the possibility of a fraud, the negative is the advice. Based on your business policies, your application can then use this risk score and advice to approve or decline a transaction, ask for additional authentication, or alert a customer service representative.

RiskMinder offers you the flexibility to modify the configuration parameters of any of the risk evaluation rules in keeping with your policies and risk-mitigation requirements. It also gives you the flexibility to modify the default scoring configurations, scoring priorities, and risk score for any rule and selectively enable or disable the execution of one or more rules.

Besides pre-configured out-of-the-box rules, RiskMinder's field-programmable custom rules capability allows for industry-specific rules to be selectively deployed and augmented based on your requirements.

Note: See "Understanding RiskMinder Basics" in the *CA RiskMinder Installation and Deployment Guide* to understand the basic concepts of RiskMinder and its architecture.

This section discusses the Web services provided by RiskMinder, the checks that you must perform before implementing the Web services, and steps to generate Java client stubs. It covers the following topics:

Note: CA RiskMinder still contains the terms Arcot and RiskFort in some of its code objects and other artifacts. Therefore, you will find occurrences of Arcot and RiskFort in all CA RiskMinder documentation. In addition, some of the topics in this guide do not follow the standard formatting guidelines. These inconsistencies will be fixed in a future release.

Introduction to the RiskMinder Web Services

RiskMinder provides a programmatic interface that you can use to integrate your application with RiskMinder. The RiskMinder Web services are broadly classified, as follows:

- Risk Evaluation Web Service (see page 10)
- Administration Web Service (see page 11)
- Organization Management Web Service (see page 11)
- User Management Web Service (see page 12)
- Configuration Management Web Service (see page 12)

Risk Evaluation Web Service

The Risk Evaluation Web service (evaluateRisk in ArcotRiskFortEvaluateRiskService.wsdl) is the interface to RiskMinder Server. This Web service provides the logic for evaluating the risk associated with a transaction and returning an appropriate advice. Based on various factors collected from the user's system and the result of configured rules that are triggered, this Web service returns a score and a corresponding advice, in addition to other related details.

If RiskMinder recommends additional authentication (which must be performed by your application), the Post Evaluation Web service (postEvaluate in ArcotRiskFortEvaluateRiskService.wsdl) returns a final advice based on the feedback of this secondary authentication received from your application.

During risk evaluation, a Device ID is passed to the Web service, which is then used by RiskMinder Server to form a user-device association in the database. The Device ID is stored on the end user's device.

This *association* (or device binding) helps identify the risk for transactions originating from the user's system for a transaction. Users who are not bound are more likely to be challenged before they are authenticated. You can list and delete these associations by using the listAssociations and deleteAssociation Web services (in ArcotRiskFortEvaluateRiskService.wsdl), respectively.

Note: Users can be bound to more than one device (for example, someone using a work and home computer) and a single device can be bound to more than one user (for example, a family sharing a computer).

Refer to "Performing Risk Evaluation and Managing Associations" (see page 165) for more information on how to use the Risk Evaluation Web service.

Administration Web Service

The RiskMinder Administration Web service (ArcotRiskFortAdminWebService in ArcotRiskFortAdminWebService.wsdl) provides a limited number of administrative operations that you can perform either by using Administration Console or by using this Web service. The operations that you can perform using this Web service are:

- Add a user to Exception User List (addUserToExceptionList)
- Remove a user from Exception User List (deleteUserFromExceptionList)
- Fetch user profile information (getUserProfile)
- Fetch location and connection information for an IP address (getLocationAndConnectionInfo)

Refer to "Performing Selected Administration Tasks" (see page 185) for more information on how to use the Administration Web service.

Organization Management Web Service

Organization is a RiskMinder unit that can either map to a complete enterprise (or a company) or a specific division, department, or other entities within the enterprise. The Organization Management Web service enables you to programmatically create and manage these organizations. You can perform the following operations by using this Web service:

- Create organizations
- Fetch organization information
- Fetch default organization
- Update organization information
- Update organization status
- Refresh organization cache
- Fetch user attributes that RiskMinder supports
- Fetch user attributes that the directory service supports

Refer to "Managing Organizations" (see page 41) for more information on how to use the Organization Management Web service.

User Management Web Service

The User Management Web service enables you to manage users, user accounts, Personal Assurance Message (PAM), and authentication operations for LDAP users. You can perform the following operations by using this Web service:

- Create users and user accounts
- Search users
- Fetch users and user accounts
- Update user information
- Update user account information
- Fetch and update user status
- Authenticate administrators (username-password and QnA authentication mechanisms *only*)
- Set and fetch PAM

Refer to "Managing Users and Accounts" (see page 87) for more information on how to use the User Management Web service.

Configuration Management Web Service

The Configuration Management Web service can be used to perform the following operations:

- Create account types
- Update account types
- Fetch account types
- Fetch email ID and telephone number types configured at the global level
- Fetch the user attributes that are configured to be stored in the encrypted format

Refer to "Managing Additional User Configurations" (see page 69) for more information on how to use the Configuration Management Web service.

RiskMinder Web Services Features

This section discusses the salient features of RiskMinder Web services:

- **Web Services Authentication and Authorization**

  RiskMinder Web services are protected from rogue requests through authentication and authorization of all Web service requests. Authentication ensures that the incoming request to the Web service has valid credentials to access the Web service, while authorization ensures that the authenticated request has appropriate privileges to access the Web service.

  See "Managing Web Services Security" (see page 37) for more information.

  Book: A Master Administrator (MA) can also enable Web services authentication and authorization by using the Authentication and Authorization page in Administration Console.

  See "Getting Started" in *CA RiskMinder Administration Guide* for more information on how to do this.

- **Support for Additional Parameters**

  In addition to the mandatory inputs, Web services accept additional input that can be passed as a name-value pair. This input can include information, such as locale, calling application details, or profile.

Before You Begin

Before you integrate your application with RiskMinder, you must install and configure RiskMinder, ensuring that:

- The systems on which you plan to install RiskMinder meet the system requirements.

  Book: Refer to "System Requirements" in the *CA RiskMinder Installation and Deployment Guide* for a complete list of installation prerequisites.

- You have completed the configuration and planning-related information:
  - You have installed and configured the required number of RiskMinder database instances.

    Book: See "Configuring Database Server" and "Database-Related Post-Installation Tasks" in the *CA RiskMinder Installation and Deployment Guide* for detailed instructions.

  - You have installed the applicable version of JDK on the system where you plan to install RiskMinder components that use JDK.
  - You have also installed the required application server.

    Book: See "Requirements for Java-Dependent Components" in the *CA RiskMinder Installation and Deployment Guide*.

In the case of **single-system deployment** of RiskMinder, ensure that all the components are up and running.

Book: See "Deploying RiskMinder on a Single System" in the *CA RiskMinder Installation and Deployment Guide* for more information.

In the case of **distributed-system deployment** of RiskMinder, ensure that the connection is established between all the components and that they successfully communicate with each other.

Book: See "Deploying RiskMinder on a Distributed System" in the *CA RiskMinder Installation and Deployment Guide* for more information

Using RiskMinder WSDL Files

To generate client applications, you must use the WSDL documents that are shipped with RiskMinder. These documents define the request and response messages that are exchanged between your application and RiskMinder Server to perform an operation.

The following table lists the WSDL documents that RiskMinder provides. These WSDLs are available at the following location:

- **On Microsoft Windows:** *install_location*\Arcot Systems\wsdls\
- **On UNIX-Based platforms:** *install_location*/arcot/wsdls/

You can use any tool of your choice, such as Apache Axis or .NET SOAP Framework, to generate client stub classes by using the WSDL files listed in the following table. You can then use the generated stub classes to build your application and access the required Web services.

WSDL File	**Description**
riskfort/**ArcotRiskFortEvaluateRiskService.wsdl**	Used to perform risk evaluation and post evaluation.
riskfort/**ArcotRiskFortAdminWebService.wsdl**	Used to manage a few administrative operations.
uds/**ArcotOrganizationManagementSvc.wsdl**	Used to create and manage organizations in your setup.
uds/**ArcotConfigManagementSvc.wsdl**	Used to create and manage user account types.
uds/**ArcotUserManagementSvc.wsdl**	Used to create and manage users and user accounts.

Important! If you are using .NET SOAP Framework to generate the client stubs, then you must include the following line in your code *before* you invoke the RiskMinder operations (ArcotRiskFortAdminSvc and RiskFortEvaluateRiskSvc):

ServicePointManager.Expect100Continue = false; // which is available in System.Net;

If you do not include this line, you might see errors.

Quick Summary

The following steps provide a quick recap of the steps that you must perform to set up your environment to use RiskMinder Web service:

1. Access the WSDL by navigating to the following location:

   On Microsoft Windows: *install_location*\Arcot Systems\wsdls\

   On UNIX-Based platforms: *install_location*/arcot/wsdls/

2. Generate the client stub classes by using the WSDL files.

   You can use a SOAP Framework, such as Apache Axis or Microsoft.NET, to generate client stub classes from a WSDL file.

3. Create the client application by using the stub classes generated in Step 2.

   Depending on the software that you choose, refer to the respective vendor documentation for more information on writing the client and the files required for the client to connect to the RiskMinder Web service.

4. Connect the client to the RiskMinder Web service end point by using the default URLs listed in the following table.

   Note: The following table lists the default URLs on which the Web services listen for client requests. If you change the service end point URL, then ensure that you connect your client to the new location that you have configured.

Web Service	**URL**
Organization Management Web Service	*http://Apphost:Port/arcotuds/services/ArcotUserRegistryMgmtSvc* ■ **Apphost**: Host name or the IP address of the system where User Data Service (UDS) is deployed. ■ **Port**: The port number at which the application server (on which UDS is deployed) is listening.
User Management Web Service	*http://Apphost:Port/arcotuds/services/ArcotUserRegistrySvc* ■ **Apphost**: Host name or the IP address of the system where UDS is deployed. ■ **Port**: The port number at which the application server (on which UDS is deployed) is listening.

Web Service	**URL**
Configuration Management Web Service	*http://Apphost:Port/arcotuds/services/ArcotConfigRegistrySvc* ■ **Apphost**: Host name or the IP address of the system where User Data Service (UDS) is deployed. ■ **Port**: The port number at which the application server (on which UDS is deployed) is listening.
Risk Evaluation Web Service	*http://Apphost:Port/services/RiskFortEvaluateRiskSvc* ■ **Apphost**: Host name or the IP address of the system where RiskMinder Server is installed. ■ **Port**: The port number at which the Transaction Web Services protocol is listening. By default, this is **7778**.
Administration Web Service	*http://Apphost:Port/services/ArcotRiskFortAdminSvc* ■ **Apphost**: Host name or the IP address of the system where RiskMinder Server is installed. ■ **Port**: The port number at which the Administration Web Services protocol is listening. By default, this is **7777**.

Note: To secure the connection using SSL, enable the Web Services protocols for SSL connection. See appendix, "Additional Configurations" (see page 197) for detailed instructions.

1. Send the requests to the RiskMinder Web service through the client.

   The RiskMinder Web service processes the request and returns the message, response code, reason code, and transaction ID in the response.

Chapter 2: Understanding RiskMinder Workflows

RiskMinder provides many workflows that can be integrated and used by your online application. Based on your organizational requirements, you can integrate these workflows, without changing the existing online experience for your users in most cases, except when RiskMinder generates the INCREASEAUTH advice.

This section describes the RiskMinder workflows and provides an overview of each workflow so that you can understand the different processes involved. This section covers the following topics:

Enrollment Workflows

Every time your application forwards a request for risk analysis, RiskMinder uses the **Unknown User Check** rule to determine if the user details exist in the RiskMinder database. If this information is not found, then RiskMinder treats the incoming request as a first-time (or unknown) user request and recommends the ALERT advice. In such cases, you must enroll the user so that they do not see the same advice the next time they undergo risk evaluation.

Enrollment is the process of creating a new user in the RiskMinder database. As discussed in the following subsections, you can enroll the user *explicitly* by calling the createUserRequest message in the ArcotUserRegistrySvc Web service from your application. After user enrollment, you must perform risk evaluation (as discussed in "Risk Evaluation Workflows" (see page 27)).

You can also *implicitly* create the user by setting the **User Enrollment Mode** as **Implicit** in the Miscellaneous Configurations page of Administration Console. If you enable this option, then every time you perform risk evaluation for an unknown user, the user is automatically created in the system.

However, if the user is not registered with your application (in other words, the user is unknown to *your* application), then you must take action according to your organizational policies.

Explicit Enrollment

In the case of *explicit enrollment*, you must explicitly call RiskMinder's createUserRequest message in the ArcotUserRegistrySvc Web service from your application to create a user in the RiskMinder database. You can call this operation either *before* (Scenario 1 (see page 21)) or *after* (Scenario 2 (see page 23)) you perform risk evaluation (by using the evaluateRisk call.)

Scenario 1

If you call the createUserRequest message in the ArcotUserRegistrySvc Web service *before* the evaluateRisk operation, then the steps for the explicit enrollment workflow are:

1. **User logs in to your online application.**

   Your system validates if the user exists in the system. If the user name is not valid, then your application must take appropriate action.

2. **Your application calls RiskMinder's createUserRequest message.**

   Your application must make an explicit call to the createUserRequest message in the ArcotUserRegistrySvc Web service. In this call, you must pass all relevant user details, such as the user's first name, last name, organization, email, and their personal assurance message (PAM) to RiskMinder.

   Book: See "Managing Users and Accounts" in the *CA RiskMinder Web Services Developer's Guide* for detailed information about the createUserRequest message.

3. **RiskMinder creates the user in the database.**

   If the createUserRequest call was successful, then RiskMinder creates the user record in the RiskMinder database. With this, user is enrolled with RiskMinder.

4. **Your application collects information required by RiskMinder.**

   Your application collects the following information from the user's system that will be used by RiskMinder for analyzing the risk:

   - **User system information** that includes operating system, platform, browser information (such as browser language, HTTP header information), locale, and screen settings. Your application uses RiskMinder's Utility Script called riskminder-client.js to collect this information.
   - **Device information** that includes Device ID, which is stored on the end user's device.
   - **Transaction information** that includes the name of the channel being used by the user, a numeric identifier for the transaction, and some other information about the transaction.
   - **Location information** that includes the IP address and Internet Service Provider related information.
   - (**Optionally, if you are using additional information**) **Additional Inputs** that are specific to custom rules or the channel selected.

5. **You application calls RiskMinder's evaluateRisk for risk analysis.**

   In this case, because you enrolled the user *before* performing risk analysis, the RiskMinder system "knows" the user and does not generate the ALERT advice. Refer to "Risk Evaluation Workflows" (see page 27) for more information.

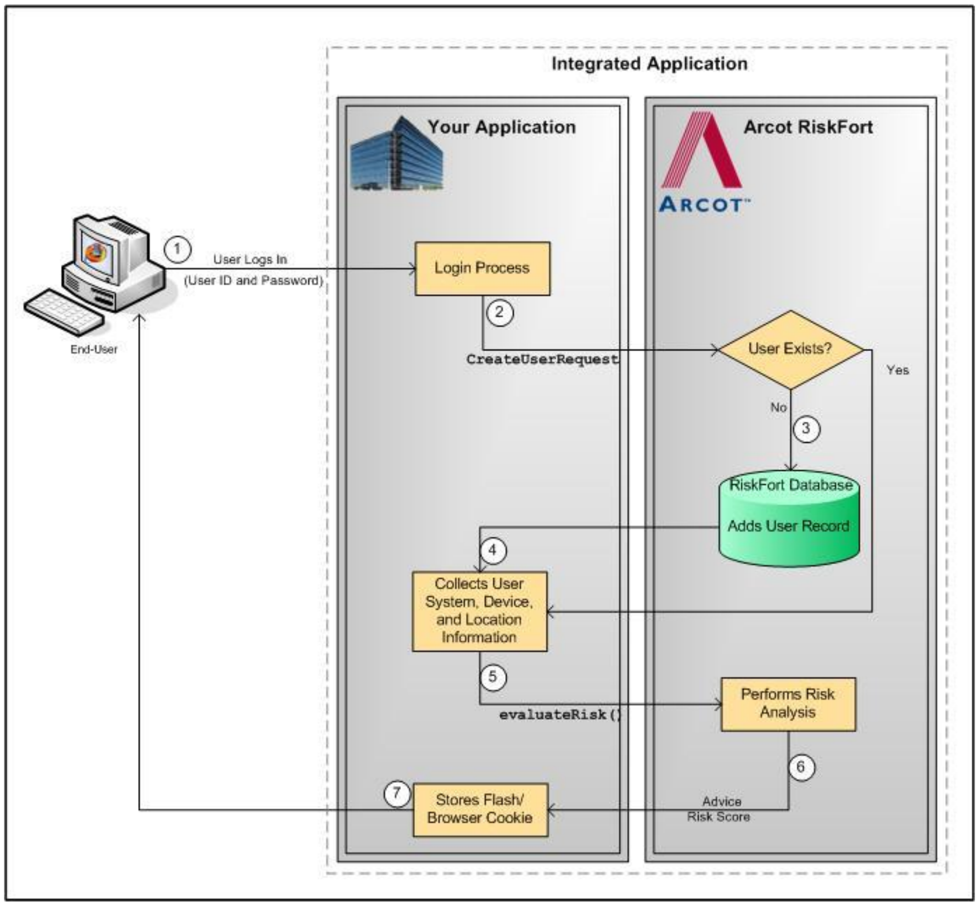
6. **RiskMinder performs risk analysis.**

   RiskMinder generates a risk score and an advice.

7. **Your application stores the Device ID on the end-user's system.**

   Your application must store the Device ID returned by evaluateRisk as a cookie on the device that the end user is using for the current transaction.

The following figure illustrates the explicit enrollment workflow when you call the createUserRequest message before the evaluateRisk call.



Scenario 2

If you call the CreateUserRequest message *after* the evaluateRisk operation, the steps for the explicit enrollment workflow are:

1. **User logs in to your online application.**

   Your system validates if the user exists in the system. If the user name is not valid, then your application must take appropriate action.

2. **Your application collects information required by RiskMinder.**

   Your application collects the following information from the user’s system that will be used by RiskMinder for analyzing the risk:

   ■ **User system information** that includes operating system, platform, browser information (such as browser language, HTTP header information), locale, and screen settings. Your application uses RiskMinder's Utility Script called riskminder-client.js to collect this information.

   ■ **Device information** that includes Device ID, which is stored on the end user's device.

   ■ **Transaction information** that includes the name of the channel being used by the user, a numeric identifier for the transaction, and some other information about the transaction.

   ■ **Location information** that includes the IP address and Internet Service Provider related information.

   ■ (**Optionally, if you are using additional information**) **Additional Inputs** that are specific to custom rules or the channel selected.

3. **Your application calls RiskMinder’s evaluateRisk operation.**

   Your application must call the evaluateRisk operation in RiskFortEvaluateRiskSvc. In this call, you must pass all the user and device information that you collected in Step 2 to RiskMinder.

4. **You application calls RiskMinder’s evaluateRisk for risk analysis.**

   RiskMinder performs risk analysis for the user and generates an advice. In this case, because the user is not yet "known" to the RiskMinder system, the ALERT advice is generated.

5. **Your application calls RiskMinder’s createUserRequest message.**

   Your application must make an explicit call to the createUserRequest message in the ArcotUserRegistrySvc Web service. In this call, you must pass all relevant user details, such as the user’s first name, last name, organization, email, and their personal assurance message (PAM) to RiskMinder.

   Book: See "Managing Users and Accounts" in the *CA RiskMinder Web Services Developer’s Guide* for detailed information about the createUserRequest message.

6. **RiskMinder creates the user in the database.**

If the createUserRequest call was successful, then RiskMinder creates the user record in the RiskMinder database. With this, the user is enrolled with RiskMinder.

7. **Your application calls RiskMinder's evaluateRisk operation again.**

   Your application must call the evaluateRisk operation in RiskFortEvaluateRiskSvc. In this call, you must ensure that you pass all the user and device information that you collected in Step 2 to RiskMinder.

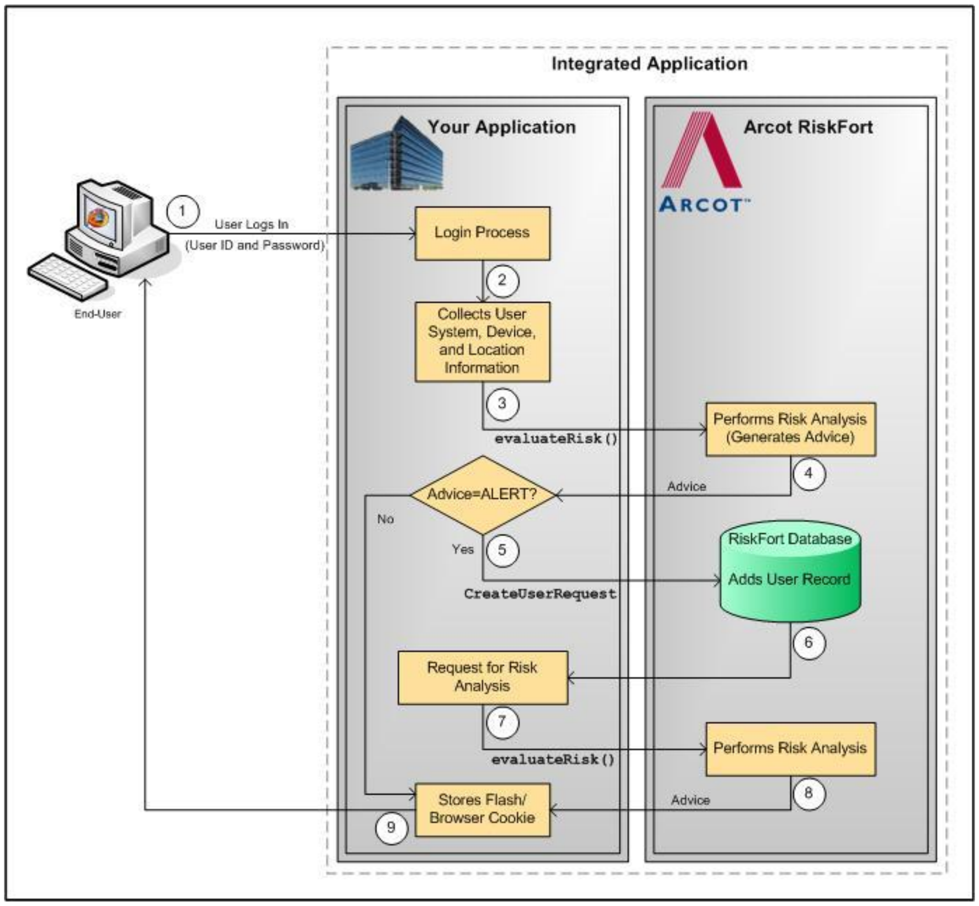
8. **RiskMinder performs risk analysis for the user.**

   In this case, RiskMinder executes the rules and generates the risk score and the advice.

9. **Your application stores the Device ID on the end-user's system.**

   Your application must store the Device ID returned by evaluateRisk as a cookie on the device that the end user is using for the current transaction.

The following figure illustrates the explicit enrollment workflow when you call the createUserRequest message before the evaluateRisk call.



Implicit Enrollment

In the case of *implicit enrollment*, you do not need to call RiskMinder’s createUserRequest message explicitly from your application’s code to create a user in the RiskMinder database. Instead, when RiskMinder generates the ALERT advice for an "unknown user", it automatically calls the operation to enroll the user.

For this enrollment to work, it is important that you first set the value of **User Enrollment Mode** field in the Miscellaneous Configurations page of Administration Console to **Implicit**.

The steps for the implicit enrollment workflow are:

1. User logs in to your online application.

   Your system validates if the user exists in the system. If the user name is not valid, then your application must take appropriate action.

2. Your application collects information required by RiskMinder.

   Your application collects the following information from the user’s system that will be used by RiskMinder for analyzing the risk:

   - **User system information** that includes operating system, platform, browser information (such as browser language, HTTP header information), locale, and screen settings. Your application uses RiskMinder's Utility Script called riskminder-client.js to collect this information.
   - **Device information** that includes Device ID, which is stored on the end user's device.
   - **Location information** that includes the IP address and Internet Service Provider related information.
   - (**Optionally, if you are using additional information**) **Additional Inputs** that are specific to custom rules or the channel selected.

3. Your application calls RiskMinder’s evaluateRisk operation.

   Your application must call the evaluateRisk operation in RiskFortEvaluateRiskSvc. In this call, you must pass all the user and device information that you collected in Step 2 to RiskMinder.

4. RiskMinder performs risk analysis for the user.

   In this case, because the user is not yet "known" to the RiskMinder system, the default ALERT advice is generated.

5. RiskMinder creates the user in the database.

   For every ALERT advice that is generated, RiskMinder automatically uses the createUserRequest message in the ArcotUserRegistrySvc Web service to create the user record in the RiskMinder database. With this, the user is enrolled with RiskMinder.

Book: See "Managing Users and Accounts" in the *CA RiskMinder Web Services Developer's Guide* for detailed information about the createUserRequest message.

6. Your application calls RiskMinder's evaluateRisk operation again.

   Your application must call the evaluateRisk operation in RiskFortEvaluateRiskSvc. In this call, you must ensure that you pass all the user and device information that you collected in Step 2 to RiskMinder.

7. RiskMinder performs risk analysis for the user.

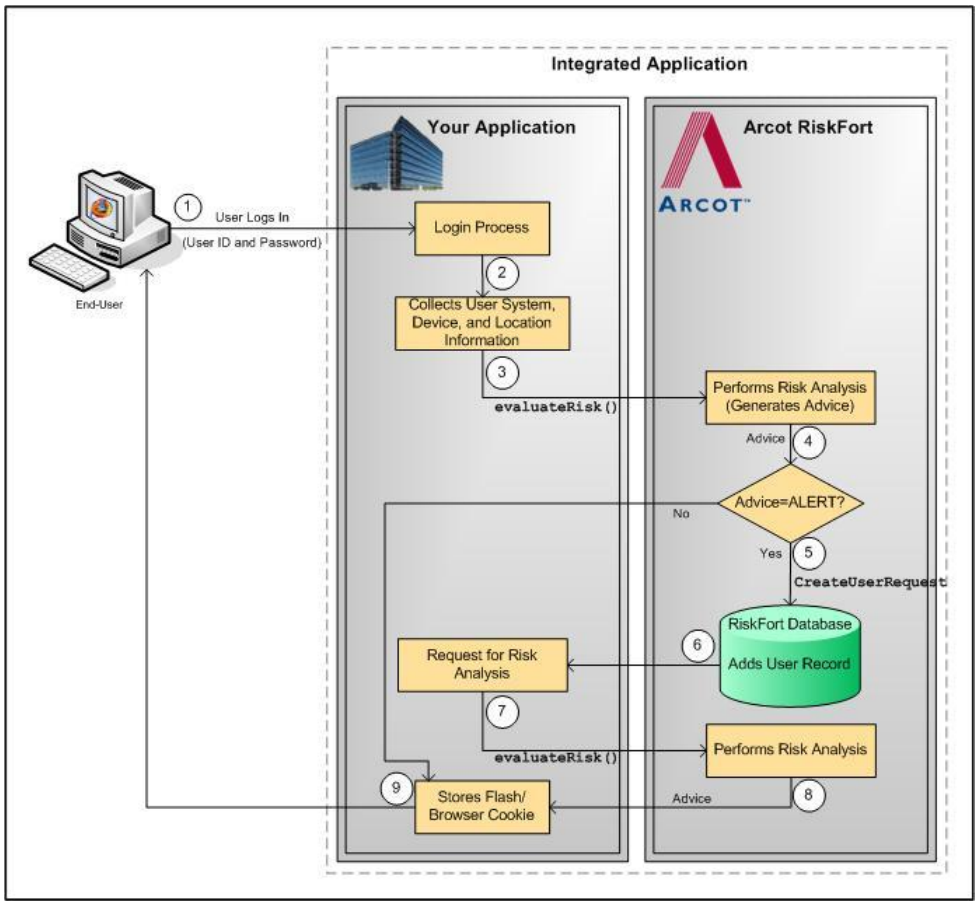
   In this case, RiskMinder executes the rules and generates the risk score and the advice.

8. Your application stores the Device ID on the end-user's system.

   After the user has been created, your application must store the Device ID returned by evaluateRisk as a cookie on the device that the end user is using for the current transaction.

The following figure illustrates the implicit enrollment workflow when RiskMinder automatically creates the user.

Implicit Enrollment Workflow



Risk Evaluation Workflows

The risk evaluation workflows enable your online application to determine if the incoming user request is potentially risky or not:

- If the risk is low, the user is allowed to access your online application.
- If the risk is high, the user is denied access to your online application.
- If the transaction is tagged as suspicious, this workflow also prompts your application to challenge users for additional authentication to prove their identity.

  If the user fails this additional authentication, then it is strongly recommended that you do not allow the user to access the protected resource(s).

You can implement RiskMinder's risk analysis capability either *before* the user logs in to your online application or *after* they have successfully logged in and are performing a transaction. Depending on when you call RiskMinder's evaluateRisk operation, the following workflows are possible:

- Pre-Login Risk Evaluation Workflow (see page 28)
- Post-Login Risk Evaluation Workflow (see page 30)

Pre-Login Risk Evaluation Workflow

When a user accesses your online application, you can assess them for potential risk even before they log in, by implementing this workflow. This workflow only uses inputs related to device identification and location information (such as IP address, Device ID, and DeviceDNA) and rules that do not require user-specific information as the criterion for risk evaluation.

If you call RiskMinder's risk analysis capability even before a user logs in to your online application, then the risk evaluation workflow is as follows:

1. **User accesses your online application.**

   When a user accesses your online application, you can assess them for potential risk even before they log in.

2. **Your application collects information required by RiskMinder.**

   Your application collects the following information from the user's system that will be used by RiskMinder for analyzing the risk:

   - **User system information** that includes operating system, platform, browser information (such as browser language, HTTP header information), locale, and screen settings. Your application uses RiskMinder's Utility Script called riskminder-client.js to collect this information.
   - **Device information** that includes Device ID, which can be stored on the end user's device.
   - **Location information** that includes the IP address and Internet Service Provider related information.
   - (**Optionally, if you are using additional information**) **Additional Inputs** that are specific to custom rules or the channel selected.

3. **Your application calls RiskMinder's evaluateRisk operation.**

   Your application must call the evaluateRisk operation in RiskFortEvaluateRiskSvc. In this call, you must pass the information that you collected in Step 2 to RiskMinder.

4. **RiskMinder performs risk analysis for the user.**

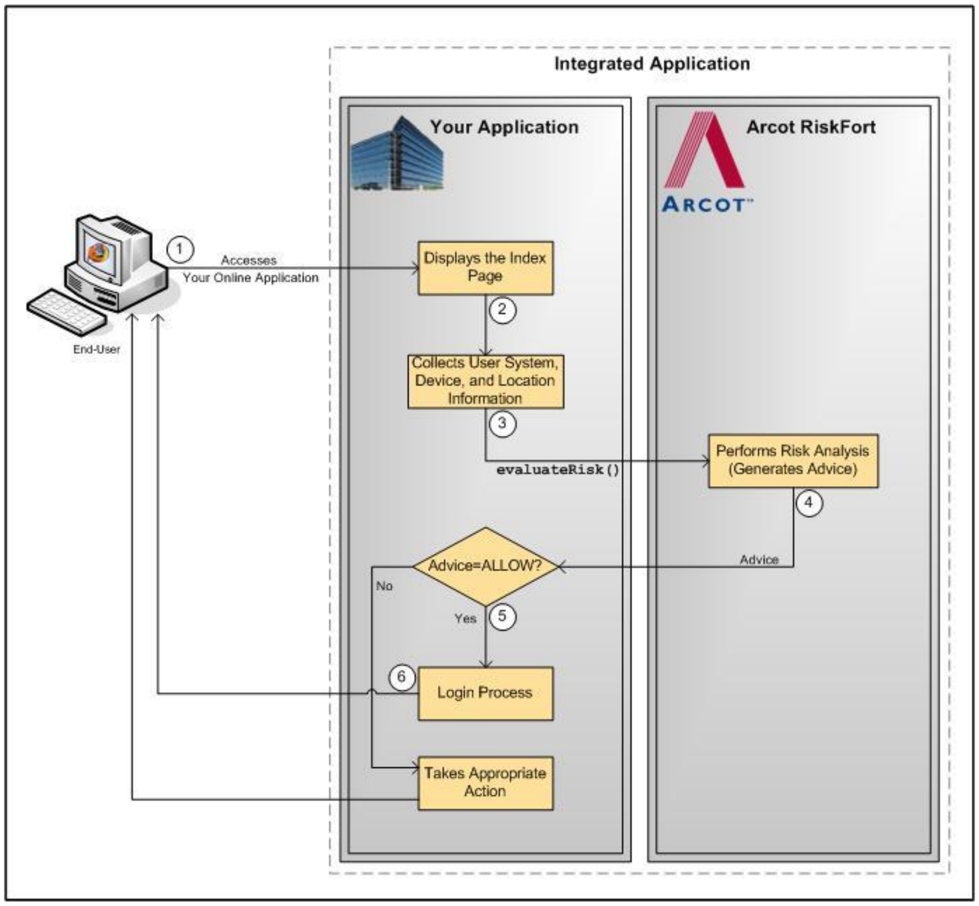
   RiskMinder generates the appropriate risk score and advice based on the user inputs and configured rules.

5. **Your application validates the user.**

   Based on risk advice generated by RiskMinder, your application can allow the user to proceed with the login process or can deny access to your system.

The following figure illustrates the pre-login risk evaluation workflow.

Pre-Login Risk Evaluation Workflow


Integrated Application
Your Application
Arcot RiskFort
ARCOT
End-User
1
Accesses Your Online Application
Displays the Index Page
2
Collects User System, Device, and Location Information
3
evaluateRisk()
Performs Risk Analysis (Generates Advice)
4
Advice
Advice=ALLOW?
No
Yes
5
6
Login Process
Takes Appropriate Action

Post-Login Risk Evaluation Workflow

When a user accesses your online application, you can first log them in and then comprehensively assess them for potential risks by implementing this workflow. This workflow uses device identification information and other factors, such as network information, user information, and (if implemented) transaction information to evaluate users.

Based on the result of the evaluateRisk operation, RiskMinder determines whether to create an association and update the attributes during the postEvaluate operation:

- In the case of ALLOW, the user-device association information is updated.
- In the case of ALERT and DENY, the user-device association information is *not* updated at all.
- In the case of INCREASEAUTH, the user-device association information is updated, but the user association information is created *only if* the result of the additional authentication ("Secondary Authentication Workflow" (see page 33)) was successful.

If you call RiskMinder's risk analysis capability after you authenticate a user in to your online application, then the risk evaluation workflow is as follows:

1. **User logs in to your online application.**

   Your system validates if the user exists in the system. If the user is not valid, then your application must take appropriate action.

2. **Your application collects information required by RiskMinder.**

   Your application collects the following information from the user's system that will be used by RiskMinder for analyzing the risk:

   - **User system information** that includes operating system, platform, browser information (such as browser language, HTTP header information), locale, and screen settings. Your application uses RiskMinder's Utility Script called riskminder-client.js to collect this information.
   - **Device information** that includes Device ID, which is stored on the end user's device.
   - **Location information** that includes the IP address and Internet Service Provider related information.
   - (**Optionally, if you are using additional information**) **Additional Inputs** that are specific to custom rules or the channel selected.

3. **Your application calls RiskMinder's evaluateRisk operation.**

   Your application must call the evaluateRisk operation in RiskFortEvaluateRiskSvc. In this call, you must pass all the user and device information that you collected in Step 2 to RiskMinder.

4. **RiskMinder performs risk analysis for the user.**

RiskMinder evaluates the risk using the incoming inputs and the configured rules. Based on the result of rules that were executed and whether the information matched, RiskMinder generates:

- ALERT, if the information for the user does not exist in the RiskMinder database.
- ALLOW, if the risk score is low.
- DENY, if the risk score is high.
- INCREASEAUTH, if the incoming information is suspicious.

If the advice is INCREASEAUTH, then refer to "Secondary Authentication Workflow" (see page 33) for more information on how to proceed.

5. **Your application takes the appropriate action by using RiskMinder's recommendation.**

   Based on the result of the evaluateRisk call, your application either allows the user to continue with the transaction, denies them access to the protected resource, or performs secondary authentication.

   See "Secondary Authentication Workflow" (see page 33) for more information.

6. **Your application calls RiskMinder's postEvaluate operation.**

   At this stage, your application must call the postEvaluate operation in RiskFortEvaluateRiskSvc. Based on the output generated by the evaluateRisk call, this call helps RiskMinder generate the final advice and update the device and association information.

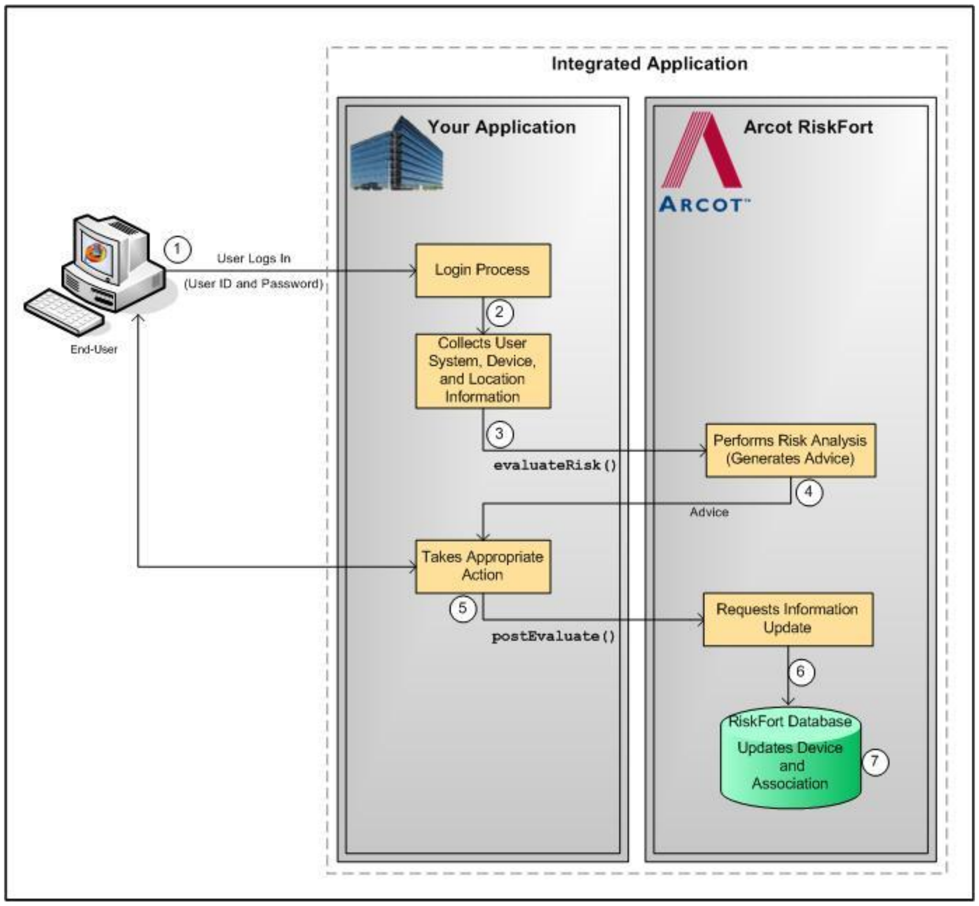
   In this call, you must pass the risk score and advice from the evaluateRisk call, the result of secondary authentication (if the advice in the previous step was INCREASEAUTH), and any association name, if the user specified one.

7. **RiskMinder updates the device and association information.**

   If any change is detected in the incoming data, RiskMinder updates the data and association information in the RiskMinder database.

The following figure illustrates the post-login risk evaluation workflow.

Post-Login Risk Evaluation Workflow



Secondary Authentication Workflow

When RiskMinder generates the INCREASEAUTH advice, it transfers the control back to your application temporarily for secondary authentication. In this case, your application must implement some mechanism for performing additional authentication. For example, your application can display industry-standard security (or challenge) questions to the user (such as mother’s maiden name and date of birth) or make them undergo out-of-band phone authentication.

After you determine whether the user authenticated successfully or not, you must forward the result to RiskMinder, which uses this feedback to generate the final advice, update device information, create association information, and store the feedback to use for risk analysis of future transactions.

The risk evaluation workflow in the case of secondary authentication is as follows:

1. **User logs in to your online application.**

   Your system validates if the user exists in the system. If the user is not valid, then your application must take appropriate action.

2. **Your application collects information required by RiskMinder.**

   Your application collects the following information from the user\xE2\x80\x99s system that will be used by RiskMinder for analyzing the risk:

   - **User system information** that includes operating system, platform, browser information (such as browser language, HTTP header information), locale, and screen settings. Your application uses RiskMinder's Utility Script called riskminder-client.js to collect this information.
   - **Device information** that includes Device ID, which is stored on the end user's device.
   - **Location information** that includes the IP address and Internet Service Provider related information.
   - (**Optionally, if you are using additional information**) **Additional Inputs** that are specific to custom rules or the channel selected.

3. **Your application calls RiskMinder’s evaluateRisk operation.**

   Your application must call the evaluateRisk operation in RiskFortEvaluateRiskSvc. In this call, you must pass all the user and device information that you collected in Step 2 to RiskMinder.

4. **RiskMinder performs risk analysis for the user.**

   If RiskMinder flags the transaction as suspicious, it generates the INCREASEAUTH advice. This implies that extra credentials are required to further authenticate the user.

5. **Your application performs secondary authentication.**

Based on the secondary authentication mechanism that you are using, your application displays the appropriate pages to the user. For example, you can prompt the user to:

- Answer the security questions that they selected while enrolling with your application.
- Perform One-Time Password (OTP) authentication.
- Perform out-of-band phone authentication.

After receiving the user input, your application determines the outcome of the additional authentication.

6. **Your application calls RiskMinder's postEvaluate operation and forwards the result of the secondary authentication to RiskMinder.**

   Irrespective of whether the user failed or cleared the secondary authentication, your application *must* pass the result back to RiskMinder. This information helps RiskMinder build an up-to-date and accurate user history.

   To do so, your application must call the postEvaluate operation in RiskFortEvaluateRiskSvc. In this call, you must pass the risk score and advice from the evaluateRisk call, the result of secondary authentication, and any association name, if the user specified one.

7. **RiskMinder generates the final advice.**

   By using your application's feedback regarding the secondary authentication, RiskMinder generates the final advice.

8. **RiskMinder updates the device information and creates the association information.**

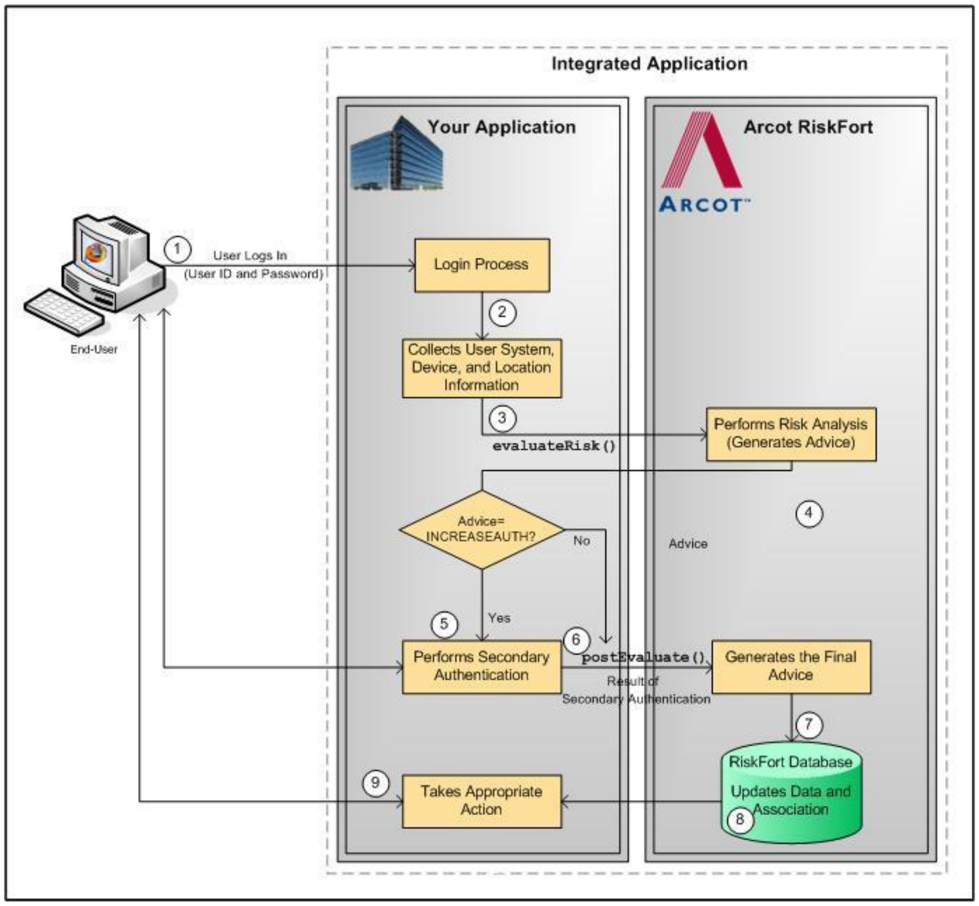
   Based on the result of the postEvaluate call, RiskMinder also updates the device attributes and creates the association information in the RiskMinder database.

9. **Your application takes the appropriate action.**

   Based on the result of the postEvaluate call, your application either allows the user to continue with the transaction or denies them access to the protected resource.

The following figure illustrates the secondary authentication risk evaluation workflow.

Secondary Authentication Risk Evaluation Workflow


Integrated Application
Your Application
Arcot RiskFort
ARCOT
End-User
1 User Logs In (User ID and Password)
Login Process
2
Collects User System, Device, and Location Information
3 evaluateRisk()
Performs Risk Analysis (Generates Advice)
4 Advice
Advice= INCREASEAUTH?
No
Yes
5
Performs Secondary Authentication
6 postEvaluate() Result of Secondary Authentication
Generates the Final Advice
7
RiskFort Database Updates Data and Association
8
9
Takes Appropriate Action

Workflow Summary

The following table provides a brief summary of the workflows provided by RiskMinder.

Note: All these workflows, except for the secondary authentication workflow, are implemented "behind the scenes" and do not change the user experience.

Workflow	Sub-Type of the Workflow	Description	Dependant Workflows
Enrollment	**Explicit**	**Scenario 1:** Creates a user in the RiskMinder database, when you call the createUserRequest message **before** evaluateRisk. In this case, the end user never gets an ALERT advice.	■ Post-Login Risk Evaluation

Workflow	**Sub-Type of the Workflow**	**Description**	**Dependant Workflows**
		Scenario 2: Creates a user in the RiskMinder database, when you call the createUserRequest message **after** evaluateRisk.	■ Post-Login Risk Evaluation
	Implicit	RiskMinder **automatically** creates a user in the RiskMinder database, without you having to call the createUserRequest message.	■ Post-Login Risk Evaluation
Risk Evaluation	**Pre-Login**	Analyzes the risk of a transaction **before** the user logs in to your online application system.	None
	Post-Login	Analyzes the risk of a transaction **after** the user logs in to your online application system. Also updates user information and device association information.	■ Enrollment ■ Secondary Authenticatio n
	In case of **Secondary Authentication**	Provides the final advice if your application performed secondary authentication after RiskMinder recommended INCREASEAUTH. Also updates user information and device association information.	■ Post-Login Risk Evaluation

Chapter 3: Managing Web Services Security

RiskMinder Web services are protected from rogue requests through authentication and authorization of all Web service requests. Authentication ensures that the incoming request to the Web service has valid credentials to access the Web service, while authorization ensures that the authenticated request has appropriate privileges to access the Web service. To enable the authentication and authorization feature, you must ensure that your calling application includes the required details in the incoming call header.

The Web services authentication and authorization works as follows:

1. The calling application authenticates to the RiskMinder Web services by including the required credentials in the call header.
2. The RiskMinder Web services authenticate these credentials and, if valid, provide your calling application with an authentication token.
3. The calling application includes the authentication token and the authorization elements in the header of the subsequent calls.

This section covers the following information:

- Authentication Header Elements (see page 37)
- Authorization Header Elements (see page 38)
- SOAP Header Namespace (see page 38)

Authentication Header Elements

The following table lists the elements that have to be included in the call header for authentication.

Element	Mandatory	Description
userID	Yes	The unique identifier of the user whose account has to be authenticated.
orgName	Yes	The organization name to which the authenticating user belongs.
credential	Yes	The credential of the user that is to be used for authentication.

Authorization Header Elements

The following table lists the elements that you must pass in the call header for authorization.

Element	Mandatory	Description
authToken	Yes	The authentication token that is returned after successful user verification. This token indicates that the user is already authenticated, and thereby eliminates the need for user credentials for successive authentication attempts. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Note: You can set any *one* of the following elements.		
targetorg	No	The organization to which your calling application must authorize before performing any operation. **Note:** If you want to enable authorization for more than one organization, then repeat this entry for every organization.
targetAllOrgs	No	Indicates whether authorization is required before operations on all organizations can be performed. Set the value of this element to TRUE to enable authorization for all organizations.
globalEntity	No	Indicates whether authorization is required for performing global configurations. Set this value to TRUE if you want to enable authorization for the global configuration operations, such as fetching attributes for users and fetching UDS attributes.

SOAP Header Namespace

The authentication and authorization header elements explained in the previous two sections must use the namespace, as mentioned in the following table.

Web Service	Namespace
User Data Service Web Services	
■ User Management ■ User Registry Management ■ Configuration Registry	*http://ws.arcot.com/UDSTransaction/1.0*
RiskMinder Web Services	

Web Service	**Namespace**
Risk Evaluation	*http://ws.arcot.com/RiskFortEvaluateRiskAPI/2.0/wsdl*
Administration	*http://ws.arcot.com/ArcotRiskFortAdminSvc/1.0/wsdl*

Chapter 4: Managing Organizations

Important! To use the Web service operations that are discussed in this section, you *must* deploy the User Data Service (**arcotuds.war**) file.
See "Deploying User Data Service" in the *CA RiskMinder Installation and Deployment Guide* for more information.

In RiskMinder, an *organization* can either map to a complete enterprise (or a company) or a specific division, department, or other entities within the enterprise. The organization structure provided by RiskMinder is flat. In other words, organizational hierarchy (in the form of parent and child organizations) is *not* supported, and all organizations are created at the same level as the Default Organization.

This section discusses the following Web service operations that RiskMinder provides to create and manage organizations:

You must use the ArcotOrganizationManagementSvc.wsdl file to perform the operations discussed in this section.

Creating Organizations

When you deploy Administration Console, an organization is created by default. This out-of-the-box organization is referred to as *Default Organization* (DEFAULTORG). For a single organization setup, instead of creating an organization you can rename this default organization, change its configurations, and then continue to use it.

For a multi-organization setup, you must create additional organizations. You can do this either by using Administration Console or by using Web services.

This section walks you through the following steps for creating organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you create an organization, you *must* refresh the system cache for the new organization to take effect. See "Refreshing the Organization Cache" (see page 51) for more information on how to refresh the cache.

Preparing the Request Message

The createOrgRequest message is used to create organizations in the RiskMinder database. The following table lists the elements of this request message.

Element	Mandatory	Description
orgName	Yes	The unique name of the organization that you want to create. This name will be used to log in to Administration Console.
displayName	Yes	A descriptive name for the organization.
keyLabel	No	The label for the key used to encrypt the sensitive organization data. Setting the key label is a one-time operation. After you set this value, you *cannot* modify it. **Note:** If this value is not specified, then the Master Key is used as the key label.

Element	Mandatory	Description
repositoryType	No	The repository where the accounts of the users who belong to the organization will reside. This repository can be one of the following: ■ **ARUSER**: Indicates that the user accounts will be created in a Relational Database Management System (RDBMS). RiskMinder supports MS SQL, MySQL, and Oracle Database. ■ **LDAP**: Indicates that the user accounts existing in your directory service will be used. **Note**: If you choose this option, then ensure that you have successfully deployed User Data Service (UDS) and configured it to connect to your directory service.
ldapDetails	No Required *only* if repositoryType =LDAP	The details of the directory service where the user information is available: ■ **host** The host name of the system where your directory service is available. ■ **port** The port number at which the directory service is listening. ■ **schemaName** The LDAP schema used by the directory service. This schema specifies the types of objects that a directory service can contain, and specifies the mandatory and optional attributes of each object type. Typically, the schema name for Active Directory is user and for SunOne Directory, it is inetorgperson. ■ **baseDN** The name-value key pairs of the base *Distinguished Name* (DN) of the directory service. This value indicates the starting node in the LDAP hierarchy to search in the directory service. For example, to search or retrieve a user with a DN of cn=rob laurie, ou=sunnyvale, o=arcot, c=us, you must specify the base DN as the following: ou=sunnyvale, o=arcot, c=us Typically, these values are case sensitive and search all sub-nodes under the specified base DN.

Element	Mandatory	Description
connectionCredential	No Required *only* if repositoryType =LDAP	The information required to connect to the directory service: ■ **ssl** The type of connection to be established with the directory service: – **TCP**: Indicates that the directory service will listen to incoming requests over TCP. – **1WAY**: Indicates that the directory service will listen to incoming requests over one-way SSL. – **2WAY**: Indicates that the directory service will listen to incoming requests over two-way SSL. ■ **loginName** The complete distinguished name of the LDAP repository user who has the privilege to log in to the repository sever and manage the base DN. For example, uid=gt,dc=arcot,dc=com ■ **loginPassword** The password of the user provided in loginName. ■ (Optional) **serverTrustCert** The base64-encoded trusted root certificate of the server that issued the SSL certificate to the directory service. This parameter is required *only* if ssl is set to 1WAY or 2WAY. ■ (Optional) **clientKeyStore** The password for the client key store and the base64-encoded root certificate of UDS. This parameter is required *only* if ssl is set to 2WAY.
redirectSearchSchema	No Required *only* if repositoryType =LDAP	The schema to be used when searching for values whose attributes are in a different node.
redirectSearchAttribute	No Required *only* if repositoryType =LDAP	The value of the attribute to be searched in redirectSearchSchema.

Element	Mandatory	Description
repositoryattribute	No Required *only* if repositoryType =LDAP	The user attribute in the directory service that has to be mapped to the RiskMinder attribute. Based on this mapping, UDS searches for the user in the directory service.
arcotattribute	No Required *only* if repositoryType =LDAP	The RiskMinder attribute to which the directory service attribute must be mapped. For example, you can map the UID attribute in the directory service to the USERNAME RiskMinder attribute.
status	No	The status of the organization in the database: ■ INITIAL Indicates that the organization is not yet activated and *cannot* be used for any operations. ■ ACTIVE Indicates that the organization has been successfully created and activated. You can perform any supported operation on the organization. ■ INACTIVE Indicates that the organization has been deactivated. To perform any further operation, you must first activate the organization. ■ DELETED Indicates that the organization has been deleted and cannot be used anymore. **Note:** If you do not set the status element for the organization, then the organization is created with the INITIAL state.
description	No	A description for the organization that helps the administrators managing the organization to easily identify the organization.
customAttribute	No	Name-value pairs that you can use to set any additional user or organization information.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking related transactions.

Invoking the Web Service

To create organizations:

1. (Optional) Include the authentication and authorization details in the header of the createOrg operation.

   See "Managing Web Services Security" (see page 37) for more information on the header elements.

2. Use the createOrgRequest elements to set the organization information, as listed in the table.

3. Use the createOrgRequest message and construct the input message by using the details specified in the preceding step.

4. Invoke the createOrg operation of the ArcorUserRegistryMgmtSvc service to create the organization.

   This operation returns the createOrgResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, createOrgResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table.

The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating Organizations

The updateOrg operation enables you to update the following organization information:

- Display name
- Description
- Custom attributes

Note: In addition to the elements that are required to perform these tasks, updateOrgRequest contains other elements for repository (directory service or RiskMinder database) configuration and user attribute mapping. After you create an organization, *you ca*nnot change the repository type and the related settings. Therefore, these elements are *not* applicable when you update an organization. Even if you set these elements, they will *not* be considered.

This section walks you through the following steps for updating organizations:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you update an organization, you *must* refresh the system cache for the changes to take effect. See "Refreshing the Organization Cache" (see page 51) for more information on how to refresh the system cache.

Preparing the Request Message

The updateOrgRequest message is used to update organizations in the RiskMinder database. The following table lists the elements of this request message.

Note: The following table lists *only* the elements that you can use to update the organization information. You can ignore other additional updateOrgRequest elements, such as repository type (repositoryDetails) configuration, user attribute mapping (mappingDetails) configuration, and status.

Element	**Mandatory**	**Description**
orgName	Yes	The name of the organization that has to be updated.
displayName	No	The descriptive name of the organization.
description	No	A description for the organization that will help the administrators managing the organization.
customAttribute	No	The name-value pairs that you can use to set any additional user or organization information.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update an organization:

1. (Optional) Include the authentication and authorization details in the header of the updateOrg operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the upateOrgRequest elements to update the organization information, as listed in the table.
3. Use the upateOrgRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the updateOrg operation of the ArcorUserRegistryMgmtSvc service to update the organization.

   This operation returns the updateOrgResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateOrgResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating Organization Status

The updateOrgStatus operation is used to update the status of the organization in the RiskMinder database.

This section walks you through the following steps for updating the organization status:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you update the organization status, you *must* refresh the system cache for the changes to take effect. See "Refreshing the Organization Cache" (see page 51) for more information on how to refresh the system cache.

Preparing the Request Message

The updateOrgStatusRequest message is used to update the organization status. The following table lists the elements of this request message.

Element	Mandatory	Description
status	Yes	The status of the organization in the database: ■ INITIAL Indicates that the organization is not yet activated and *cannot* be used for any operations. ■ ACTIVE Indicates that the organization has been successfully created and activated. You can perform any operation on the organization. ■ INACTIVE Indicates that the organization has been deactivated. To perform any further operation, you must *first* activate the organization. ■ DELETED Indicates that the organization has been deleted and cannot be used anymore.
OrgName	Yes	The unique name with which the organization is identified.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update the organization status:

1. (Optional) Include the authentication and authorization details in the header of the updateOrgStatus operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the upateOrgStatusRequest elements to update the organization status, as listed in the table.
3. Use the upateOrgStatusRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the updateOrgStatus operation of the ArcorUserRegistryMgmtSvc service to update the organization status.

   This operation returns the updateOrgStatusResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateOrgStatusResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Refreshing the Organization Cache

The refreshCache operation is used to refresh the organization configurations that are stored in the cache. This section walks you through the following steps for refreshing the organization cache:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The refreshCacheRequest message is used to refresh the organization cache. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
systemCache	No	Specifies whether you want to refresh the system cache: ■ True: Indicates that the system cache will be refreshed. ■ False: Indicates that the system cache will not be refreshed.
Note: You can set any *one* of the following elements.		
allOrganizations	No	Specifies whether the cache of all organizations has to be refreshed. Set the value of this element to TRUE to refresh the cache of all organizations.
OrgName	Yes	The unique name with which the organization is identified.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To refresh the organization cache:

1. (Optional) Include the authentication and authorization details in the header of the refreshCache operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the refreshCacheRequest elements for updating the organization configurations, as listed in the table.
3. Use the refreshCacheRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the refreshCache operation of the ArcorUserRegistryMgmtSvc service to refresh the organization cache.

   This operation returns the refreshCacheResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, refreshCacheResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching Default Organization Details

The Master Administrator (MA) sets the default organization details in the system. Typically, when you create administrators or enroll users without specifying their organization, they are created in this default organization. The retrieveDefaultOrg operation is used to fetch the details of the default organization.

This section walks you through the following steps for fetching the details of the default organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The retrieveDefaultOrgRequest message is used to fetch the default organization information. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the default organization information:

1. (Optional) Include the authentication and authorization details in the header of the retrieveDefaultOrg operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the retrieveDefaultOrgRequest elements for fetching the default organization information, as listed in the table.
3. Use the retrieveDefaultOrgRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the retrieveDefaultOrg operation of the ArcorUserRegistryMgmtSvc service to fetch the default organization details.

   This operation returns the retrieveDefaultOrgResponse message that includes the transaction identifier, authentication token, and default organization details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, retrieveDefaultOrgResponse, returns the transaction identifier, authentication token, and other details in the SOAP envelope header. The SOAP body includes the default organization details for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
orgName	The unique name of the organization.
displayName	The descriptive name of the organization.

Element	**Description**
repositoryDetails	The repository where the accounts of the users who belong to the organization reside: ■ ARUSER ■ LDAP
dateCreated	The timestamp when the organization was created.
dateModified	The timestamp when the organization configuration was last modified.
description	The description for the organization that helps the administrators managing the organization.
status	The status of the default organization in the database: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
preferredLocale	The locale configured for the organization. If you do not specify the locale, then the default locale is set to en-US.
customAttribute	The name-value pairs of the custom attributes that have been set for the organization.

Fetching Organization Details

The retrieveOrg operation is used to read the details of an organization.

Note: If you want to fetch details of multiple organizations at the same time, then use the listOrgs operation. See "Searching Organizations" (see page 58) for more information on how to use this.

This section walks you through the following steps for fetching the details of an organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The retrieveOrgRequest is used to fetch the details of an organization. The following table lists the elements of this request message.

Element	Mandatory	Description
orgName	Yes	The unique name with which the organization is identified.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the organization details:

1. (Optional) Include the authentication and authorization details in the header of the retrieveOrg operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the retrieveOrgRequest elements for fetching the organization details, as listed in the table.
3. Use the retrieveOrgRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the retrieveOrg operation of the ArcorUserRegistryMgmtSvc service to fetch the organization details.

   This operation returns the retrieveOrgResponse message that includes the transaction identifier, authentication token, and organization details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, retrieveOrgResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the organization details for a successful transaction and the Fault response for an error condition.

See the second table in Fetching Default Organization Details (see page 53) for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Searching Organizations

The listOrgs operation is used to simultaneously read the details of multiple organizations. You can search organizations by their organization name, status, and partial or complete display name.

This section walks you through the following steps for searching organizations:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listOrgsRequest operation is used to fetch the details of multiple organizations. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
namePattern	No	The search pattern that you want to use to search organizations. You can enter the partial or complete display name of an organization. If you enter the partial name, then all organizations with the display name matching the search pattern will be fetched.
orgName	No	The unique name with which the organization is identified. **Note:** If you want to search for more than one organization, then repeat this element for different organizations.
OrgStatus	No	The status of the organization in the database: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To search multiple organizations:

1. (Optional) Include the authentication and authorization details in the header of the listOrgs operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listOrgsRequest elements for fetching the organization details, as listed in the table.
3. Use the listOrgsRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listOrgs operation of the ArcorUserRegistryMgmtSvc service to fetch the organization details.

   This operation returns the listOrgsResponse message that includes the transaction identifier, authentication token, and organization details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listOrgsResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the organization details for a successful transaction and the Fault response for an error condition.

See the second table in Fetching Default Organization Details (see page 53) for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Fetching RiskMinder Database Attributes

The listArcotAttributes operation is used to fetch the user attributes that are used to store the user information in the RiskMinder database.

This section walks you through the following steps for fetching the user attributes supported by the RiskMinder database:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listArcotAttributesRequest message is used to fetch the user attributes. The following table lists the elements of this request message.

Element	Mandatory	Description
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the RiskMinder database attributes:

1. (Optional) Include the authentication and authorization details in the header of the listArcotAttributes operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listArcotAttributesRequest elements to fetch the user attributes, as listed in the table.
3. Use the listArcotAttributesRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listArcotAttributes operation of the ArcorUserRegistryMgmtSvc service to fetch the user attributes supported by the RiskMinder database.

   This operation returns the listArcotAttributesResponse message that includes the transaction identifier, authentication token, and user attributes. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listArcotAttributesResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user attributes for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
DATECREATED	The timestamp when the user account was created.
DATEMODIFIED	The timestamp when the user account was last modified.
EMAILADDR	The email address of the user.
FNAME	The first name of the user.

Element	**Description**
IMAGE	The personal assurance image that the user selected.
LNAME	The last name of the user.
MNAME	The middle name of the user.
PAM	The Personal Assurance Message (PAM) that is displayed when the user tries to access any resource protected by RiskMinder. PAM is the text string that serves as server verification to the client and is set by the user during enrollment.
PAMURL	The URL that lists the images, which can be used by the user to select their personal assurance image.
STATUS	The status of the user in the database: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
TELEPHONENUMBER	The telephone number of the user.
USERID	The unique identifier for the user.

Fetching Directory Service Attributes

The listRepositoryAttributes operation is used to fetch the directory service user attributes that are mapped to RiskMinder-supported user attributes.

This section walks you through the following steps for fetching the user attributes that the directory service supports:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listRepositoryAttributesRequest message is used to fetch directory service user attributes that are mapped to RiskMinder-supported user attributes. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
repositoryType	Yes	The directory service where the user information resides: ■ ARUSER: For organizations that are created in the RiskMinder database. ■ LDAP: For organizations that are mapped with LDAP repository.

Element	Mandatory	Description
ldapDetails	No	The details of the directory service where the user information is available: ■ **host** The host name of the system where your directory service is available. ■ **port** The port number at which the directory service is listening. ■ **schemaName** The LDAP schema used by the directory service. This schema specifies the types of objects that a directory service can contain, and specifies the mandatory and optional attributes of each object type. Typically, the schema name for Active Directory is user and for SunOne Directory, it is inetorgperson. ■ **baseDN** The name-value key pairs of the base Distinguished Name (DN) of the directory service. This value indicates the starting node in the LDAP hierarchy to search in the directory service. For example, to search or retrieve a user with a DN of cn=rob laurie, ou=sunnyvale, o=arcot, c=us, you must specify the base DN as the following: ou=sunnyvale, o=arcot, c=us Typically, these values are case sensitive and search all sub-nodes under the specified base DN.

Element	Mandatory	Description
connectionCredential	No	The information required to connect to the directory service: ■ **ssl** The type of connection that has to be established with the directory service: – **TCP**: Indicates that the directory service will listen to incoming requests on TCP. \xE2\x80\x93 **1WAY**: Indicates that the directory service will listen to incoming requests on one-way SSL. – **2WAY**: Indicates that the directory service will listen to incoming requests on two-way SSL. ■ **loginName** The complete distinguished name of the LDAP repository user who has the privilege to log in to the repository sever and manage the base DN. For example, uid=gt,dc=arcot,dc=com ■ **loginPassword** The password of the user provided in loginName. ■ (Optional) **serverTrustCert** The base64-encoded trusted root certificate of the server that issued the SSL certificate to the directory service. This parameter is required *only* if ssl is set to 1WAY or 2WAY. ■ (Optional) **clientKeyStore** The password for the client key store and the base64-encoded root certificate of UDS. This parameter required *only* if ssl is set to 2WAY.
redirectSearchSchema	No	The schema to be used to search for the values whose attributes are in a different node.
redirectSearchAttribute	No	The value of the attribute to be searched in the redirectSearchSchema.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the user attributes:

1. (Optional) Include the authentication and authorization details in the header of the listRepositoryAttributes operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listRepositoryAttributesRequest elements to set the directory service information, as listed in the table.
3. Use the listRepositoryAttributesRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listRepositoryAttributes operation of the ArcorUserRegistryMgmtSvc service to fetch the user attributes.

   This operation returns the listRepositoryAttributesResponse message that includes the transaction identifier, authentication token, and user attributes. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listRepositoryAttributesResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user attributes for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
The user attributes used to store user information.	

Deleting Organizations

The deleteOrg operation is used to delete organizations. After you delete an organization, the information related to that organization is still maintained in the system. Therefore, you cannot create an organization with the same name as that of the deleted organization.

This section walks you through the following steps for deleting organizations:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you delete an organization, you *must* refresh the system cache for the changes to take effect. See "Refreshing the Organization Cache" (see page 51) for more information on how to refresh the system cache.

Preparing the Request Message

The deleteOrgRequest message is used to delete organizations. The following table lists the elements of this request message.

Element	Mandatory	Description
orgName	Yes	The unique name with which the organization is identified.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To delete organizations:

1. (Optional) Include the authentication and authorization details in the header of the deleteOrg operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the deleteOrgRequest elements for fetching the organization details, as listed in the table.
3. Use the deleteOrgRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the deleteOrg operation of the ArcorUserRegistryMgmtSvc service to delete the organization.

   This operation returns the deleteOrgResponse message that includes the transaction identifier, authentication token, and organization details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteOrgResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Chapter 5: Managing Additional User Configurations

Important! To use the Web service operations that are discussed in this section, you *must* deploy the User Data Service (**arcotuds.war**) file.
See "Deploying User Data Service" in the *CA RiskMinder Installation and Deployment Guide* for more information.

This section describes the operations that are used to manage account types, fetch the email and telephone types configured for the users, and fetch the user attributes that are configured for encryption. This section covers the following topics:

You must use the ArcotConfigManagementSvc.wsdl file to perform the operations discussed in this section.

Managing Account Types

All RiskMinder users are identified in the system by a unique user name. RiskMinder now supports the concept of an account or account ID, which is an alternate ID to identify the user in addition to the user name. A user can have none or one or more accounts or account IDs.

An account type is an attribute that qualifies the account ID and provides additional context about the usage of the account ID. To assign multiple accounts to a user, you must first create an account type, and then create an account for each account type.

For example, consider a financial institution that identifies the customers by their unique customer identifier. If the customer enhances their portfolio with a fixed deposit, then the financial institution can create an account type called *FIXED_DEPOSIT* and create an account in this account type with the fixed deposit number, for example 000203876544.

Now the customer can log in either with their unique customer identifier or the account type and account ID (FIXED_DEPOSIT and 000203876544) combination.

You can configure the account type to be available to specific organizations only or to all organizations, including those that will be created in the future. At the organization level, each organization can choose to support a set of account types.

This section covers the following operations related to account type:

- Creating Account Types (see page 71)
- Updating Account Types (see page 73)
- Fetching Account Types (see page 75)
- Deleting Account Types (see page 77)

Creating Account Types

This section walks you through the steps for creating account types:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you create an account type, you *must* refresh the system cache for the new account type to take effect. See "Refreshing the Organization Cache" (see page 51) for more information on how to refresh the cache.

Preparing the Request Message

The createAccountTypeRequest message is used to create account types in the RiskMinder database. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
accountType/name	Yes	The name of the account type that you want to create.
accountType/displayName	Yes	A descriptive name for the account type.
accountType/customAttribute	No	Name-value pairs that you can use to specify additional information related to account types.
targetAllOrgs	No	Indicates whether the account type should be assigned to all organizations: ■ true: Account type is assigned to all organizations. ■ false: Account type is assigned only to the organizations that are listed in the ListOfOrganizations element. **Note:** By default, the value of this element is set to false.
ListofOrganizations/Organization/ orgName	No	The name of the organization to which the account type must be assigned.
ListofOrganizations/Organization/ customAttribute	No	The custom attribute that you have set for the organization to which you want to assign the account type.

Element	Mandatory	Description
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To create account types:

1. (Optional) Include the authentication and authorization details in the header of the createAccountType operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the createAccountTypeRequest elements to set the account information, as listed in the table.
3. Use the createAccountTypeRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the createAccountType operation of the ArcotConfigRegistrySvc service to create the account type.

   This operation returns the createAccountTypeResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, createAccountTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating Account Types

The updateAccountType operation is used to update the account type information and the list of organizations to which the account type belongs.

This section walks you through the following steps for updating existing account types:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you update an account type, you *must* refresh the system cache for the new account type to take effect. See "Refreshing the Organization Cache" (see page 51) for more information on how to refresh the cache.

Preparing the Request Message

The updateAccountTypeRequest message is used to update account types in the RiskMinder database. The following table lists the elements of this request message.

Element	Mandatory	Description
name	Yes	The name of the account type that you want to update.
displayName	No	The descriptive name of the account type.
customAttribute	No	Name-value pairs that you can use to specify additional user or organization information.
removeCustomAttribute	No	The name of the account type custom attribute that you want to delete.
targetAllOrgs	No	Indicates whether the updated account type should be assigned to all organizations: ■ true: Updated account type is assigned to all organizations. ■ false: Updated account type is assigned only to the organizations that are listed in the ListOfOrganizations element. **Note:** By default, the value of this element is set to false.
ListofOrganizations/orgName	No	The name of the organization to which the account type must be assigned.
ListofOrganizations/customAttribute	No	The custom attribute that you have specified for the organization.

Element	Mandatory	Description
RemoveOrganizations/orgName	No	The name of the organization that you want to disassociate with the account type.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update account types:

1. (Optional) Include the authentication and authorization details in the header of the updateAccountType operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the updateAccountTypeRequest elements to set the account information, as listed in the table.
3. Use the updateAccountTypeRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the updateAccountType operation of the ArcotConfigRegistrySvc service to update the account type.

   This operation returns the updateAccountTypeResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateAccountTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching Account Types

The listAccountTypes operation is used to fetch the account types that are associated with an organization.

This section walks you through the following steps for fetching the account types:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listAccountTypeRequest message is used to fetch account types that are associated with an organization. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
targetAllOrgs	Yes	Indicates whether to fetch the account types assigned to all organizations: ■ true: Account types assigned to all organizations are fetched. ■ false: Account types assigned to the organizations that are listed in the orgName element are fetched.
orgName	No	The name of the organization to which the account types to be fetched belongs.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To list the account types of an organization:

1. (Optional) Include the authentication and authorization details in the header of the listAccountTypes operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listAccountTypeRequest elements to set the account information, as listed in the table.
3. Use the listAccountTypeRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listAccountTypes operation of the ArcotConfigRegistrySvc service to update the account type.

   This operation returns the listAccountTypeResponse message that includes the transaction identifier, authentication token, and the account types associated with an organization. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listAccountTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the account type details for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
AccountType/name	The name of the account type.
AccountType/displayName	The descriptive name of the account type.

Element	Description
AccountType/customAttribute	Name-value pairs that are used to specify additional account type information.

Deleting Account Types

The deleteAccountType operation is used to delete the account types that are associated with an organization.

This section walks you through the following steps for deleting account types:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you delete an account type, you *must* refresh the system cache for the new account type to take effect. See "Refreshing the Organization Cache" (see page 51) for more information on how to refresh the cache.

Preparing the Request Message

The deleteAccountTypeRequest message is used to delete account types in the RiskMinder database. The following table lists the elements of this request message.

Element	Mandatory	Description
accountType	Yes	The name of the account type that you want to delete.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To delete account types:

1. (Optional) Include the authentication and authorization details in the header of the deleteAccountType operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the deleteAccountTypeRequest elements to get the account type that has to be deleted, as listed in the table.
3. Use the deleteAccountTypeRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the deleteAccountType operation of the ArcotConfigRegistrySvc service to delete the account type.

   This operation returns the deleteAccountTypeResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteAccountTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching Email and Telephone Types

RiskMinder enables you to specify multiple email addresses and telephone numbers while creating users in an organization. Email and telephone types are used to define multiple email addresses and telephone numbers. These types can be defined globally or can be specific to an organization. If the email address or telephone number types are mandatory for an organization, then you must provide these values while you create users in that organization.

This section covers the following topics that discuss how to fetch the email and telephone types that are configured for an organization:

- Fetching Email Types (see page 79)
- Fetching Telephone Types (see page 81)

Fetching Email Types

The listEmailTypes operation is used to fetch the email address types that are configured for an organization.

This section walks you through the following steps for fetching the email address types configured for an organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listEmailTypeRequest message is used to fetch email address types that are configured for the organization. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
orgName	No	The name of the organization for which the email address types have to be fetched.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch email address types:

1. (Optional) Include the authentication and authorization details in the header of the listEmailTypes operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listEmailTypeRequest elements to get the organization name, as listed in the table.
3. Use the listEmailTypeRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listEmailTypes operation of the ArcotConfigRegistrySvc service to fetch the email address types.

   This operation returns the listEmailTypeResponse message that includes the transaction identifier, authentication token, and email address types. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listEmailTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the email address types for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
isGlobal	Specifies whether the email type is configured at the global level: ■ True: Indicates that the email type is configured at the global level. ■ False: Indicates that the email type is configured at the organization level.

Element	Description
emailType/name	The name of the email address type.
emailType/display Name	The display name of the email address type.
emailType/priority	The priority of the email type if more than one email type has been configured.
emailType/isMand atory	Indicates whether the email type is mandatory.

Fetching Telephone Types

The listTelephoneTypes operation is used to fetch the telephone types that are configured for an organization.

This section walks you through the following steps for fetching the telephone types configured for an organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listTelephoneTypeRequest message is used to fetch the telephone types that are configured for the organization. The following table lists the elements of this request message.

Element	Mandatory	Description
orgName	No	The name of the organization for which the telephone address types have to be fetched.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch telephone types:

1. (Optional) Include the authentication and authorization details in the header of the listTelephoneTypes operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listTelephoneTypeRequest elements to get the organization name, as listed in the table.
3. Use the listTelephoneTypeRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listTelephoneTypes operation of the ArcotConfigRegistrySvc service to fetch the telephone types.

   This operation returns the listTelephoneTypeResponse message that includes the transaction identifier, authentication token, and telephone types. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listTelephoneTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the telephone types for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
isGlobal	Specifies whether the telephone type is configured at the global level: ■ True: Indicates that the telephone type is configured at the global level. ■ False: Indicates that the telephone type is configured at the organization level.

Element	Description
TelephoneType/name	The name of the telephone type.
TelephoneType/displayName	The display name of the telephone type.
TelephoneType/priority	The priority of the telephone type if more than one telephone type has been configured.
TelephoneType/isMandatory	Indicates whether the telephone type is mandatory.

Fetching User Attributes Configured for Encryption

The administrators of an organization can choose to store the user attributes in an encrypted format. To fetch such attributes that are configured to be stored in encrypted format, you need to use the listConfiguredAttributesForEncryption operation.

This section walks you through the following steps for fetching the user attributes that are configured for encryption:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listConfiguredAttributesForEncryptionRequest message is used to fetch the user attributes that are configured for encryption. The following table lists the elements of this request message.

Element	Mandatory	Description
orgName	No	The name of the organization for which the user attributes have to be fetched.
clientTxId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch user attributes configured for encryption:

1. (Optional) Include the authentication and authorization details in the header of the listConfiguredAttributesForEncryption operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listConfiguredAttributesForEncryptionRequest elements to get the organization name, as listed in the table.
3. Use the listConfiguredAttributesForEncryptionRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listConfiguredAttributesForEncryption operation of the ArcotConfigRegistrySvc service to fetch the user attributes.

   This operation returns the listConfiguredAttributesForEncryptionResponse message that includes the transaction identifier, authentication token, and user attributes. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listConfiguredAttributesForEncryptionResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user attributes for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
isGlobal	Specifies whether the user attributes are configured for encryption at the global level: ■ True: Indicates that the user attributes are configured for encryption at the global level. ■ False: Indicates that the user attributes are configured for encryption at the organization level.

Element	Description
attribute	The name of the user attribute.

Chapter 6: Managing Users and Accounts

Important! To use the Web service operations that are discussed in this section, you *must* deploy the User Data Service (**arcotuds.war**) file.
See "Deploying User Data Service" in the *CA RiskMinder Installation and Deployment Guide* for more information.

For RiskMinder to authenticate users, users have to be created in the database, which is a one-time process. The user can either be created in the RiskMinder database or RiskMinder can be configured to connect to LDAP for user information.

This section discusses the Web service operations that are used to create and manage users, create and manager user accounts, and authenticate LDAP users. This section covers the following topics:

You must use the ArcotUserManagementSvc.wsdl file to perform the operations discussed in this section.

Before You Proceed

This section lists the supported user states, transitions supported between the user states, and the user operations that are possible on a particular organization and user status combination. Before you proceed with the user and user account operations that are discussed in this section, read this section to understand whether the operation can be performed based on the organization and user status.

The following topics are covered in this section:

User States

RiskMinder supports the following states for users in the system:

- **INITIAL**

  Indicates that the user has been created in the system, but cannot perform any operation. To create a user in this state, you need to specify the status in the createUser operation.

- **ACTIVE**

  Indicates that the user can perform any operation in the system. This is the default status of the user when you create a user in the system.

- **INACTIVE**

  Indicates that the user has been deactivated and cannot perform any operation. You can deactivate a user permanently or for a specific period. You might need to deactivate the user for a specified period in situations where an employee goes for a long vacation and you want to disable their logins during this period to prevent any unauthorized access.

  To deactivate the user for a specific period, you must specify the startLockTime and endLockTime elements. If you do not specify these values, then the user will be permanently deactivated.

- **DELETED**

  Indicates that the user no longer exists in the system.

Supported User State Transitions

The following table lists the transitions possible between the supported user states.

Current State	**Change State to**				
	INITIAL	**ACTIVE**	**INACTIVE (Temporary)**	**INACTIVE (Permanent)**	**DELETED**
INITIAL	Yes	Yes	No	No	Yes
ACTIVE	No	Yes	Yes	Yes	Yes
INACTIVE	No	Yes	Yes	Yes	Yes
DELETED	No	No	No	No	Yes

User Operations and States

The following table lists the user operations and whether each operation is allowed on a specific combination of the organization and user status.

User Operation	**Organization Status**	**User Status**	**Allowed**
Create User	INITIAL	NA	No
	ACTIVE	NA	Yes
	INACTIVE	NA	No
	DELETED	NA	No
Update User	INITIAL	NA	No
	ACTIVE	Any User State	Yes
	INACTIVE	Any User State	Yes
	DELETED	Any User State	Yes
Update User Status	INITIAL	NA	No
	ACTIVE	INITIAL ACTIVE INACTIVE DELETED	Yes
	INACTIVE	INITIAL ACTIVE INACTIVE DELETED	Yes
	DELETED	Any User State	No
Delete User	INITIAL	NA	No
	ACTIVE	INITIAL ACTIVE INACTIVE	Yes
	INACTIVE	INITIAL ACTIVE INACTIVE	Yes
	DELETED	Any User State	No

User Account Operations and States

The following table lists the user account operations and whether each operation is allowed on a specific combination of the organization and user status.

User Account Operation	Organization Status	User Status	Allowed
Add User Account	INITIAL	NA	No
	ACTIVE/INACTIVE	INITIAL	Yes
		ACTIVE	Yes
		INACTIVE	Yes
		DELETED	No
	DELETED	Any User State	No
Update User Account	INITIAL	NA	No
	ACTIVE/INACTIVE	INITIAL	Yes
		ACTIVE	Yes
		INACTIVE	Yes
		DELETED	No
	DELETED	Any User State	No
Update User Account	INITIAL	NA	No
	ACTIVE/INACTIVE	INITIAL	Yes
		ACTIVE	Yes
		INACTIVE	Yes
		DELETED	No
	DELETED	Any User State	No
Delete User Account	INITIAL	NA	No
	ACTIVE/INACTIVE	INITIAL	Yes
		ACTIVE	Yes
		INACTIVE	Yes
		DELETED	No
	DELETED	Any User State	No

Performing User Operations

This section covers the following operations:

Creating Users

This section walks you through the following steps for creating users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The createUserRequest message is used to create users in the RiskMinder database. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
userId/orgName	No	The name of the organization to which the user must belong. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userID/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The unique identifier assigned to the user when they are created. This identifier is used as a reference to track different operations performed by a user.
dateCreated	No	The timestamp when the user was created in the system. **Note:** Not applicable for the createUser operation.

Element	Mandatory	Description
dateModified	No	The timestamp when the user details were last modified. **Note:** Not applicable for the createUser operation.
emailId	Yes	The email ID of the user that has to be registered. The default qualifier is EMAILID. **Note:** You can repeat this entry if you want to configure multiple email IDs for a user, and accordingly use the qualifier based on the email types configured using Administration Console.
telephoneNumber	Yes	The telephone number of the user that has to be registered. The default qualifier is TELEPHONE. **Note:** You can repeat this entry if you want to configure multiple telephone numbers for a user, and accordingly use the qualifier based on the telephone types configured using Administration Console.
firstName	No	The first name of the user.
middleName	No	The middle name of the user.
lastName	No	The last name of the user.
pam	No	The Personal Assurance Message (PAM) displayed to the user, when they try to access a resource protected by RiskMinder.
pamImageURL	No	The URL that contains the image displayed to the user, when they try to access a resource protected by RiskMinder.
image	No	The picture that the user wants to upload to identify themselves.
status	No	The status of the user: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED **Note:** If you do not pass a value, then by default the status is set as ACTIVE.

Element	Mandatory	Description
customAttribute	No	The additional user information that you want to pass as name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	No	The timestamp when the user has to be deactivated.
endLockTime	No	The timestamp when the deactivated user has to be activated.
account/accountType	Yes *Only* if the account element is defined.	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.
account/accountStatus	No	The status of the account: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account status is UNKNOWN.
account/accountIDAttribute	No	The alternate identifier used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
account/dateCreated	No	The timestamp when the account ID was created. **Note:** Not applicable for the createUser operation.
account/dateModified	No	The timestamp when the account ID was last modified. **Note:** Not applicable for the createUser operation.

Element	Mandatory	Description
account/accountCustomAttribute	No	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To create users in the RiskMinder database:

1. (Optional) Include the authentication and authorization details in the header of the createUser operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the createUserRequest elements to provide the user information, as listed in the table.
3. Use the createUserRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the createUser operation of the ArcorUserRegistrySvc service to create users.

   This operation returns the createUserResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, createUserResponse, returns the transaction identifier and authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction performed by using UDS.

Element	Description
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating Users

This section walks you through the following steps for updating the user information:

- Preparing the Request Message
- Interpreting the Response Message
- Interpreting the Response Message

Preparing the Request Message

The updateUserRequest message is used to update the user information in the RiskMinder database. The following table lists the elements of this request message.

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userID/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
dateCreated	No	The timestamp when the user was created in the system.
dateModified	No	The timestamp when the user details were last modified.
emailId	No	The email ID of the user that has to be registered. The default qualifier is EMAILID. **Note:** You can repeat this entry if you want to configure multiple email IDs for a user, and accordingly use the qualifier based on the configured email types.

Element	Mandatory	Description
telephoneNumber	No	The telephone number of the user that has to be registered. The default qualifier is TELEPHONE. **Note:** You can repeat this entry if you want to configure multiple telephone numbers for a user, and accordingly use the qualifier based on the configured telephone types.
firstName	No	The first name of the user.
middleName	No	The middle name of the user.
lastName	No	The last name of the user.
pam	No	The Personal Assurance Message (PAM) displayed to the user, when they try to access a resource protected by RiskMinder.
pamImageURL	No	The URL that contains the image displayed to the user when they try to access a resource protected by RiskMinder.
image	No	The picture that the user wants to upload to identify themselves.
status	No	The status of the user: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
customAttribute	No	The additional user information that you want to pass as a name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	No	The timestamp when the user has to be deactivated.
endLockTime	No	The timestamp when the deactivated user has to be activated.

Element	Mandatory	Description
account/accountType	Yes *Only* if the account element is defined.	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.
account/accountStatus	No	The status of the account: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account status is UNKNOWN.
account/accountIDAttribute	No	The alternate identifier used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
account/dateCreated	No	The timestamp when the account ID was created. **Note:** Not applicable for the createUser operation.
account/dateModified	No	The timestamp when the account ID was last modified. **Note:** Not applicable for the createUser operation.
account/accountCustomAttribute	No	The additional account information that you want to pass as name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.

Element	Mandatory	Description
updateUserFlags/updateImage	No	The flag to indicate whether the user image can be changed. Supported values are: ■ 0: Indicates that the image cannot be changed. ■ 1: Indicates that the image can be changed.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update users in the RiskMinder database:

1. (Optional) Include the authentication and authorization details in the header of the updateUser operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the updateUserRequest elements to update the user information, as listed in the table.
3. Use the updateUserRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the updateUser operation of the ArcorUserRegistrySvc service to update the user information.

   This operation returns the updateUserResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateUserResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating User Status

The updateUserStatus operation is used to change the status of the user. In a single call, you can update the status of multiple users.

The status of a user can be any of the following:

- INITIAL
- ACTIVE
- INACTIVE
- DELETED

This section walks you through the following steps for changing the user status:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the updateUserStatusRequest message.

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userID/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
Note: If want to update the status of more than one user, then repeat the userID element with the user details.		
status	Yes	The status that you want to assign to the user: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
startLockTime	No	The timestamp when the user has to be deactivated.
endLockTime	No	The timestamp when the deactivated user has to be activated.

Element	Mandatory	Description
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update user status in the RiskMinder database:

1. (Optional) Include the authentication and authorization details in the header of the updateUserStatus operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the updateUserStatusRequest elements to update the user status, as listed in the table.
3. Use the updateUserStatusRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the updateUserStatus operation of the ArcorUserRegistrySvc service to update the user status.

   This operation returns the updateUserStatusResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateUserStatusResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching User Details

The retrieveUser operation is used to search the details of a particular user.

This section walks you through the following steps for reading the user details:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the retrieveUserRequest message.

Element	**Mandatory**	**Description**
userIdentifier	Yes	The unique identifier (user name) with which the user is identified in the system.
orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
accountType	No	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
filter/includeImage	No	The flag to indicate whether the user image has to be retrieved or not. Supported values are: ■ 0: Indicates that the image must not be retrieved. ■ 1: Indicates that the image must be retrieved.
filter/includeAccounts	No	The flag to indicate whether the user accounts have to be retrieved or not. Supported values are: ■ 0: Indicates that the user accounts must not be retrieved. ■ 1: Indicates that the user accounts must be retrieved.

Element	**Mandatory**	**Description**
filter/deepSearch	No	The flag to indicate whether the user must be searched based on more than one parameters. Supported values are: ■ 0: Indicates that the users will be searched based on their user names *only*. ■ 1: Indicates that the users will be searched using the following details: First search attribute: User name Second search attribute: Account ID Third search attribute: Account ID attribute If the user details are not found using the first search attribute, then the second attribute is used. If both the first and second attributes fail to fetch the user details, then the third attribute is used to search the user details.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To retrieve the details of a user:

1. (Optional) Include the authentication and authorization details in the header of the retrieveUser operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the retrieveUserRequest elements to collect the user details, as listed in the table.
3. Use the retrieveUserRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the retrieveUser operation of the ArcorUserRegistrySvc service to fetch the user details.

   This operation returns the retrieveUserResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, retrieveUserResponse returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user details for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
userId/orgName	The name of the organization to which the user belongs.
userId/userName	The unique identifier with which the user is identified in the system.
userId/userRefId	The identifier used as a reference to track different operations performed by a user.
dateCreated	The timestamp when the user was created in the system.

Element	**Description**
dateModified	The timestamp when the user details were last modified.
emailId	The email ID of the user that has been registered. If multiple email IDs are configured for the user, then all email IDs are fetched.
telephoneNumber	The telephone number of the user that has been registered. If multiple telephone numbers are configured for the user, then all numbers are fetched.
firstName	The first name of the user.
middleName	The middle name of the user.
lastName	The last name of the user.
pam	The Personal Assurance Message (PAM) displayed to the user, when they try to access a resource protected by RiskMinder.
pamImageURL	The URL which contains the image displayed to the user, when they try to access a resource protected by RiskMinder.
image	The picture that the user wants to upload to identify themselves.
status	The status of the user: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED **Note:** If you do not pass a value, then by default the status is set as ACTIVE.
customAttribute	The additional user information in name-value pairs. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	The timestamp when the user has to be deactivated.
endLockTime	The timestamp when the deactivated user has to be activated.
account/accountType	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.

Element	Description
account/accountStatus	The status of the account: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account status is UNKNOWN.
account/accountIDAttribute	The alternate identifier used to identify the user in the system.
account/dateCreated	The timestamp when the account ID was created.
account/dateModified	The timestamp when the account ID was last modified.
account/accountCustomAttribute	The additional account information that you want to pass as a name-value pair. ■ attributename Indicates the name of the custom attribute. ■ attributevalue Indicates the corresponding value for the name.

Searching Users by Using Pagination

When you search for users in the RiskMinder database or directory service, the information is fetched and displayed in the alphabetical order of the user names. If you have a large setup with many users, then you will have to navigate through the search result to search for a particular user. To increase the search efficiency in such cases, you can search the users by specifying the start and end index range.

Note: If you are searching for the users in the LDAP organization, then ensure that the LDAP supports pagination search.

This section walks you through the following steps for searching the active users based on the search index:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the listUsersRequest message.

Element	Mandatory	Description
orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
startIndex	Yes	The index entry starting from which the user information has to be fetched. For example, if the complete search fetches 60 results and if the startIndex is set to 45, then the user information from search result entry 45 is returned.
endIndex	Yes	The index page where the user search must end. For example, if the complete search fetches 60 results and if the startIndex is set to 45 and endIndex is set to 55, then the user information from the search result entry 45 to 55 is returned.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To search for users based on pagination:

1. (Optional) Include the authentication and authorization details in the header of the listUsers operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listUsersRequest elements to collect the start and end index, as listed in the table.
3. Use the listUsersRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listUsers operation of the ArcorUserRegistrySvc service to fetch the user details for the specified start and end index.

   This operation returns the listUsersResponse message that includes the transaction identifier, authentication token, and user details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listUsersResponse, returns the transaction identifier and authentication token in the SOAP envelope header. The SOAP body includes the user details and status for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
count	The total number of users returned in the search result.
userId/orgName	The name of the organization to which the user belongs.
userId/userName	The unique identifier with which the user is identified in the system.

Element	Description
userId/userRefId	The identifier used as a reference to track different operations performed by a user.
dateCreated	The timestamp when the user was created in the system.
dateModified	The timestamp when the user details were last modified.
emailId	The email ID of the user that has been registered. If multiple email IDs are configured for the user, then all email IDs are fetched.
telephoneNumber	The telephone number of the user that has been registered. If multiple telephone numbers are configured for the user, then all numbers are fetched.
firstName	The first name of the user.
middleName	The middle name of the user.
lastName	The last name of the user.
pam	The Personal Assurance Message (PAM) displayed to the user when they try to access a resource protected by RiskMinder.
pamImageURL	The URL which contains the image displayed to the user, when they try to access a resource protected by RiskMinder.
image	The picture that the user wants to upload to identify themselves.
status	The status of the user: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
customAttribute	The additional user information that you want to pass as a name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	The timestamp when the user has to be deactivated.
endLockTime	The timestamp when the deactivated user has to be activated.

Searching All Users

You must use the searchUsers operation to search for all users in the system.

This section walks you through the following steps for searching the users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the searchUsers message.

Element	**Mandatory**	**Description**
orgPattern	No	The pattern used to search the organizations.
orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
searchExpression	Yes	The search expression to use to search for users.
count	No	If the search result exceeds this value, then only the search results equal to this value are fetched.
filter/includeImage	No	The flag to indicate whether the user image has to be retrieved or not. Supported values are: ■ 0: Indicates that the image must not be retrieved. ■ 1: Indicates that the image must be retrieved.
filter/includeAccounts	No	The flag to indicate whether the user accounts have to be retrieved or not. Supported values are: ■ 0: Indicates that the user accounts must not be retrieved. ■ 1: Indicates that the user accounts must be retrieved.

Element	Mandatory	Description
filter/deepSearch	No	The flag to indicate whether the user must be searched based on more than one parameter. Supported values are: ■ 0: Indicates that the users will be searched based on their user names *only*. ■ 1: Indicates that the users will be searched using the following details: **First search attribute**: User name **Second search attribute**: Account ID **Third search attribute**: Account ID attributes If the user details are not found using the first search attribute, then the second attribute is used. If both the first and second attributes fail to fetch the user details, then the third attribute is used to search the user details.
status	No	The status of the user: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED **Note:** If you do not pass the value, then by default the status is set as ACTIVE.
account/accountType	Yes *Only* if the account element is defined.	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.

Element	Mandatory	Description
account/accountStatus	No	The status of the account: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account status is UNKNOWN.
account/accountIDAttribute	No	The alternate identifier used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
account/dateCreated	No	The timestamp when the account ID was created.
account/dateModified	No	The timestamp when the account ID was last modified.
account/accountCustomAttribute	No	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.
RepositoryUserAttributes/attributeName	No	The name of the user attribute used to store the user information. For example, First Name or Email Address.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To search users:

1. (Optional) Include the authentication and authorization details in the header of the searchUsers operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the searchUsersRequest elements to collect the user information, as listed in the table.
3. Use the searchUsersRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the searchUsers operation of the ArcorUserRegistrySvc service to fetch the information of all users.

   This operation returns the searchUsersResponse message that includes the transaction identifier, authentication token, and user details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, searchUsersResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user details and status for a successful transaction, and the Fault response for an error condition.

The elements returned for searchUsersResponse are the same as those for retrieveUserResponse. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Checking the User Status

You must use the getUserStatus operation to know the current status of the user in the database.

This section walks you through the following steps for checking the user status:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the getUserStatusRequest message.

Element	**Mandatory**	**Description**
userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userId/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To check the user status:

1. (Optional) Include the authentication and authorization details in the header of the getUserStatus operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the getUserStatusRequest elements to collect the user details, as listed in the table.
3. Use the getUserStatusRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the getUserStatus operation of the ArcorUserRegistrySvc service to check the user status.

   This operation returns the getUserStatusResponse message that includes the transaction identifier, authentication token, and user details and status. See the following section for more information on the response message.

Interpreting the Response Message

The response message, getUserstatusResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user details and status for a successful transaction, and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
userId/orgName	The name of the organization to which the user belongs.
userId/userName	The unique identifier with which the user is identified in the system.

Element	Description
userId/userRefId	The identifier used as a reference to track different operations performed by a user.
status	The status of the user: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED

Updating the User Status

You must use the updateUserStatus operation to change the current status of the user in the database.

This section walks you through the following steps for updating the user status:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the updateUserStatusRequest message.

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userId/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
status	Yes	The status that you want to assign to the user: ■ INITIAL ■ ACTIVE ■ INACTIVE **Note:** If the current status of the user is DELETED, then you cannot update the status of that user.

Element	Mandatory	Description
startLockTime	No	The timestamp when the user has to be deactivated.
endLockTime	No	The timestamp when the deactivated user has to be activated.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update the user status:

1. (Optional) Include the authentication and authorization details in the header of the updateUserStatus operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the updateUserStatusRequest elements to collect the user details, as listed in the table.
3. Use the updateUserStatusRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the updateUserStatus operation of the ArcorUserRegistrySvc service to check the user status.

   This operation returns the updateUserStatusResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateUserStatusResponse returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Deleting Users

This section walks you through the following steps for deleting users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The deleteUserRequest message is used to delete users in the RiskMinder database. The following table lists the elements of this request message.

Element	Mandatory	Description
userId/orgNam e	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userID/userNa me	Yes	The unique identifier with which the user is identified in the system.
userId/userRefI d	No	The unique identifier assigned to the user when they are created. This identifier is used as a reference to track different operations performed by a user.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To delete users in the RiskMinder database:

1. (Optional) Include the authentication and authorization details in the header of the deleteUser operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the deleteUserRequest elements to provide the user information, as listed in the table.
3. Use the deleteUserRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the deleteUser operation of the ArcorUserRegistrySvc service to delete users.

   This operation returns the deleteUserResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteUserResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Performing User Account Operations

In addition to the user name, which is the unique user identifier, users can be identified by their accounts (also known as account ID*).* A user can have multiple accounts. To define an account for the user, an account type has to be first configured for the organization to which the user belongs.

An account type provides additional context about the usage of the account. An account type can have only *one* account ID. If you want to assign multiple account IDs for a user, then you need to first configure the account type for each account ID that you plan to create for the user.

This section covers the following steps related to user account operations:

- Adding User Accounts (see page 120)
- Updating User Accounts (see page 123)
- Fetching All Accounts of a User (see page 125)
- Fetch User Account Details (see page 127)
- Fetching User Details Using Accounts (see page 130)
- Deleting User Accounts (see page 134)

Note: Accounts are dependent on user name and account type. Before adding user accounts, you must ensure that the user has already been created in the system, as discussed in "Performing User Operations" (see page 91), and that the account type has been defined for the organization to which the user belongs, as discussed in "Managing Additional User Configurations" (see page 69).

Adding User Accounts

You must use the addUserAccount operation to add accounts for users. This section walks you through the following steps for adding user accounts.

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the addUserAccountRequest message.

Element	**Mandatory**	**Description**
userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userId/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
account/accountType	Yes	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.
account/accountStatus	No	The status of the account: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account status is UNKNOWN.
account/accountIDAttribute	No	The alternate identifier used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.

Element	Mandatory	Description
account/dateCreated	No	The timestamp when the account ID was created.
account/dateModified	No	The timestamp when the account ID was last modified.
account/accountCustomAttribute	No	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To add user accounts:

1. (Optional) Include the authentication and authorization details in the header of the addUserAccount operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the addUserAccountRequest elements to collect the user details, as listed in the following table.
3. Use the addUserAccountRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the addUserAccount operation of the ArcorUserRegistrySvc service to add accounts for the user.

   This operation returns the addUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, addUserAccountResponse, returns the transaction identifier and authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating User Accounts

You must use the updateUserAccount operation to update the existing accounts of users. This section walks you through the steps for updating the user accounts.

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The updateUserAccountRequest message elements are the same as those for addUserAccountRequest. See the first table in Adding User Accounts (see page 120) for more information.

Invoking the Web Service

To update user accounts:

1. (Optional) Include the authentication and authorization details in the header of the updateUserAccount operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the updateUserAccountRequest elements to collect the user account details, as listed in Adding User Accounts (see page 120).
3. Use the updateUserAccountRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the updateUserAccount operation of the ArcorUserRegistrySvc service to update accounts of the user.

   This operation returns the updateUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateUserAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header, these elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.

Element	Description
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching All Accounts of a User

To fetch the details of all accounts that are created for a user, you must use the listUserAccounts operation. This section walks you through the following steps for fetching the accounts of a user.

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: If you want to fetch details of a particular account, then use the retrieveUserAccount operation. See "Fetch User Account Details" (see page 127) for more information.

Preparing the Request Message

The listUserAccountRequest message elements are same as those for addUserAccountRequest. See the first table in Adding User Accounts (see page 120) for more information.

Invoking the Web Service

To fetch user accounts:

1. (Optional) Include the authentication and authorization details in the header of the listUserAccounts operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listUserAccountRequest elements to collect the user account details, as listed in Adding User Accounts (see page 120).
3. Use the listUserAccountRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listUserAccounts operation of the ArcorUserRegistrySvc service to update accounts of the user.

   This operation returns the listUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listUserAccountResponse returns the transaction identifier and authentication token in the SOAP envelope header. The SOAP body includes the user account details for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. The authentication token by default is valid for *one* day, after which you need to authenticate again.
Body Elements	
account/accountType	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.
account/accountStatus	The status of the account: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account status is UNKNOWN.
account/accountIDAttribute	The alternate identifier used to identify the user in the system.
account/dateCreated	The timestamp when the account ID was created.
account/dateModified	The timestamp when the account ID was last modified.
account/accountCustomAttribute	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.

Fetch User Account Details

You must use the retrieveUserAccount operation to fetch the details of a particular user account.

This section walks you through the following steps to fetch the details of a single user account:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the retrieveUserAccountRequest message.

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userId/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
account/accountType	Yes	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.
account/accountStatus	No	The status of the account: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account status is UNKNOWN.

Element	Mandatory	Description
account/accountIDAttribute	No	The alternate identifier used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
account/dateCreated	No	The timestamp when the account ID was created.
account/dateModified	No	The timestamp when the account ID was last modified.
account/accountCustomAttribute	No	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch user account details:

1. (Optional) Include the authentication and authorization details in the header of the retrieveUserAccount operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the retrieveUserAccountRequest elements to collect the user and account details, as listed in the table.
3. Use the retrieveUserAccountRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the retrieveUserAccount operation of the ArcorUserRegistrySvc service to fetch the user details based on the account information.

   This operation returns the retrieveUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, retrieveUserAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user account details for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
account/accountType	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.

Element	Description
account/accountStatus	The status of the account: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account status is UNKNOWN.
account/accountIDAttribute	The alternate identifier used to identify the user in the system.
account/dateCreated	The timestamp when the account ID was created.
account/dateModified	The timestamp when the account ID was last modified.
account/accountCustomAttribute	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.

Fetching User Details Using Accounts

To fetch the user details using their account information, you must use the listUsersForAccount operation. This section walks you through the following steps for fetching the user information based on the user accounts:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the listUsersForAccountRequest message.

Element	Mandatory	Description
orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.

Element	Mandatory	Description
accountType	No	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
Note: The accountID and accountIDAttribute elements are optional, but you must pass at least one element.		
accountID	No	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.
accountIDAttri bute	No	The alternate identifier used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
filter/includeI mage	No	The flag to indicate whether the user image has to be retrieved or not. Supported values are: ■ 0: Indicates that the image must not be retrieved. ■ 1: Indicates that the image must be retrieved.
filter/includeAc counts	No	The flag to indicate whether the user accounts have to be retrieved or not. Supported values are: ■ 0: Indicates that the user accounts must not be retrieved. ■ 1: Indicates that the user accounts must be retrieved.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the user details using their account information:

1. (Optional) Include the authentication and authorization details in the header of the listUsersForAccount operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listUsersForAccountRequest elements to collect the user account information, as listed in the table.
3. Use the listUsersForAccountRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the listUsersForAccount operation of the ArcorUserRegistrySvc service to fetch the user details based on the account information.

   This operation returns the listUsersForAccountResponse message that includes the transaction identifier, authentication token, and user details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listUsersForAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user details for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
userId/orgName	The name of the organization to which the user belongs.
userId/userName	The unique identifier with which the user is identified in the system.
userId/userRefId	The identifier used as a reference to track different operations performed by a user.
dateCreated	The timestamp when the user was created in the system.

Element	Description
dateModified	The timestamp when the user details were last modified.
emailId	The email ID of the user that has been registered. If multiple email IDs are configured for the user, then all email IDs are fetched.
telephoneNumber	The telephone number of the user that has been registered. If multiple telephone numbers are configured for the user, then all numbers are fetched.
firstName	The first name of the user.
middleName	The middle name of the user.
lastName	The last name of the user.
pam	The Personal Assurance Message (PAM) displayed to the user, when they try to access a resource protected by RiskMinder.
pamImageURL	The URL that contains the image displayed to the user when they try to access a resource protected by RiskMinder.
image	The picture that the user wants to upload to identify themselves.
status	The status of the user: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
customAttribute	The additional user information that you want to pass as a name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	The timestamp when the user has to be deactivated.
endLockTime	The timestamp when the deactivated user has to be activated.
account/accountType	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	The alternate identifier used to identify the user in addition to the user name. The account ID is also known as account.

Element	Description
account/accountStatus	The status of the account: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account status is UNKNOWN.
account/accountIDAttribute	The alternate identifier used to identify the user in the system.
account/dateCreated	The timestamp when the account ID was created.
account/dateModified	The timestamp when the account ID was last modified.
account/accountCustomAttribute	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.

Deleting User Accounts

You must use the deleteUserAccount operation to delete accounts for users. This section walks you through the following steps for deleting user accounts:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the deleteUserAccountRequest message.

Element	Mandatory	Description
accountType	Yes	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To delete user accounts:

1. (Optional) Include the authentication and authorization details in the header of the deleteUserAccount operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the deleteUserAccountRequest elements to collect the user details, as listed in the table.
3. Use the deleteUserAccountRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the deleteUserAccount operation of the ArcorUserRegistrySvc service to delete accounts for the user.

   This operation returns the deleteUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteUserAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Setting the Personal Assurance Message

The Personal Assurance Message (PAM) is a text string that is displayed to the user, when they try to access a resource protected by RiskMinder. This string assures the user that they are connected to the genuine network or resource.

To set the PAM for a user, you must use the setPAM operation. This section walks you through the following steps for setting the PAM for the users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the setPAMRequest message.

Element	Mandatory	Description
UserId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
UserId/userName	Yes	The unique identifier with which the user is identified in the system.
UserId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
PAM	No	The Personal Assurance Message (PAM) displayed to the user, when they try to access a resource protected by RiskMinder. **Note:** If you do not pass the PAM element, then an empty value is set as PAM.
pamImageURL	No	The URL that contains the image displayed to the user when they try to access a resource protected by RiskMinder.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To set the PAM for a user:

1. (Optional) Include the authentication and authorization details in the header of the setPAM operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the setPAMRequest elements to collect the user information, as listed in the table.
3. Use the setPAMRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the setPAM operation of the ArcorUserRegistrySvc service to set the PAM for the user.

   This operation returns the setPAMResponse message that includes the transaction identifier and authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, setPAMResponse returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching the Personal Assurance Message

To read the PAM that is set for a user, you must use the getPAM operation. This section walks you through the following steps for fetching the PAM of the users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the getPAMRequest message.

Element	**Mandatory**	**Description**
UserId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
UserId/userName	Yes	The unique identifier with which the user is identified in the system.
UserId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the PAM of a user:

1. (Optional) Include the authentication and authorization details in the header of the getPAM operation. See"Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the getPAMRequest elements to collect the user information, as listed in the table.
3. Use the getPAMRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the getPAM operation of the ArcorUserRegistrySvc service to get the PAM for the user.

   This operation returns the getPAMResponse message that includes the transaction identifier, authentication token, and PAM. See the following section for more information on the response message.

Interpreting the Response Message

The response message, getPAMResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the PAM for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
UserId/orgName	The name of the organization to which the user belongs.
UserId/userName	The unique identifier with which the user is identified in the system.
UserId/userRefId	The identifier used as a reference to track different operations performed by a user.

Element	Description
PAM	The Personal Assurance Message (PAM) displayed to the user, when they try to access a resource protected by RiskMinder.
pamImageURL	The URL that contains the image displayed to the user when they try to access a resource protected by RiskMinder.

Setting Custom User Attributes

In addition to the standard user information that RiskMinder supports, you can set additional user information by using custom attributes. You must pass the additional information as name-value pairs.

To set the custom user attributes, you must use the setCustomAttributes operation. This section walks you through the following steps for setting custom attributes:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the setCustomAttributesRequest message.

Element	Mandatory	Description
UserId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
UserId/userName	Yes	The unique identifier with which the user is identified in the system.
UserId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
customAttribute	No	The additional user information that you want to pass as a name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.

Element	Mandatory	Description
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To set additional information for a user:

1. (Optional) Include the authentication and authorization details in the header of the setCustomAttributes operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the setCustomAttributesRequest elements to collect the user information, as listed in the table.
3. Use the setCustomAttributesRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the setCustomAttributes operation of the ArcorUserRegistrySvc service to set the user information.

   This operation returns the setCustomAttributesResponse message that includes the transaction identifier and authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, setCustomAttributesResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Authenticating LDAP Users

This section discusses the operations used for authenticating users whose accounts are present in the directory service. It covers the following topics:

- Using the LDAP Password (see page 142)
- Using Directory Service Attributes (see page 144)

Important! The operations discussed in this section are applicable *only* for organizations with repository type as **LDAP**.

Using the LDAP Password

Administration Console uses the LDAP authentication mechanism to authenticate the users whose accounts are available in the LDAP repository. In this case, users log in to the Console by specifying their LDAP user name and password.

To use the LDAP authentication mechanism to authenticate users, you must use the authenticateUser operation. This section walks you through the following steps for authenticating users using the LDAP authentication mechanism:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the authenticateUserRequest message.

Element	**Mandatory**	**Description**
UserCredential/userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
UserCredential/userId/userName	Yes	The unique identifier with which the user is identified in the system.
UserCredential/userId/userRefId	No	The identifier used as a reference to track different operations performed by a user.
UserCredential/userCredential/type	Yes	The credential that has to be used to authenticate the user. You must set the type as password.

Element	Mandatory	Description
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To authenticate users using the LDAP authentication mechanism:

1. (Optional) Include the authentication and authorization details in the header of the authenticateUser operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the authenticateUserRequest elements to collect the user and credential information, as listed in the table.
3. Use the authenticateUserRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the authenticateUser operation of the ArcorUserRegistrySvc service to set the user information.

   This operation returns the authenticateUserResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, authenticateUserResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the authentication status for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, Exceptions and Error Codes" (see page 209) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	

Element	Description
AuthResult/status	The authentication status of the user in the LDAP. Possible values are: ■ SUCCESS ■ FAILURE

Using Directory Service Attributes

This section discusses the following operations that are used to authenticate users using their directory service attributes:

- Fetching User Attributes (see page 144)
- Fetching User Attribute Values (see page 146)
- Verifying User Attributes (see page 148)

Fetching User Attributes

The attributes that are used to store the user information in the directory service can be read using the getQnAAttributes operation. This section walks you through the following steps related to this operation:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the QnAAttributesRequest message.

Element	Mandatory	Description
orgName	Yes	The name of the LDAP organization to which the user attributes that you want to fetch belong.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the user attributes:

1. (Optional) Include the authentication and authorization details in the header of the getQnAAttributes operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the getQnAAttributesRequest elements to collect the organization information, as listed in the table.
3. Use the QnAAttributesRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the getQnAAttributes operation of the ArcorUserRegistrySvc service to fetch the user attributes of the LDAP organization.

   This operation returns the QnAAttributesResponse message that includes the transaction identifier, authentication token, and user attributes. See the following section for more information on the response message.

Interpreting the Response Message

The response message, QnAAttributesResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user attributes for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token by default is valid for *one* day, after which you need to authenticate again.
Body Elements	
The user attributes configured in the LDAP.	

Fetching User Attribute Values

The getQnAValues operation is used to read the values that are set for the user attributes present in the directory service. You can fetch the values for one or more attributes. This section walks you through the following steps related to this operation:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the QnAValuesRequest message.

Element	**Mandatory**	**Description**
username	Yes	The unique identity of the user whose attribute values you want to fetch.
orgname	Yes	The name of the LDAP organization to which the user attribute values that you want to fetch belong.
attributes/attribute	Yes	The name of the attributes whose value you want to fetch.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the values of user attributes:

1. (Optional) Include the authentication and authorization details in the header of the getQnAValues operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the getQnAValuesRequest elements to collect the user, organization, and attribute information, as listed in the table.
3. Use the QnAValuesRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the getQnAValues operation of the ArcorUserRegistrySvc service to fetch the values of the user attributes that are stored in directory service.

   This operation returns the QnAValuesResponse message that includes the transaction identifier, authentication token, and attribute values. See the following section for more information on the response message.

Interpreting the Response Message

The response message, QnAValuesResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user attribute values for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
The values for the user attributes that are configured in the directory service.	

Verifying User Attributes

You can authenticate the users of an organization (mapped to LDAP repository) by using their LDAP attributes. You must use the performQnAVerification operation to perform this authentication. This section walks you through the following steps related to this operation:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the QnAVerificationRequest message.

Element	**Mandatory**	**Description**
username	Yes	The unique identifier of the user whose attributes you want to verify.
orgname	Yes	The name of the LDAP organization to which the user belongs.
attributes/attribute	Yes	The name (attrName) and value (attrValue) of the attribute that has to be verified.
ignorecase	Yes	Specifies whether the case of the attribute values passed in the input must match the case of the values stored in the directory service: ■ 0: Indicates that the case must match. ■ 1: Indicates that the case of the input values will be ignored.
clientTxId	No	The unique transaction identifier that your calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To authenticate users with their LDAP attributes:

1. (Optional) Include the authentication and authorization details in the header of the performQnAVerification operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the performQnAVerificationRequest elements to collect the user, organization, and attribute information, as listed in the table.
3. Use the QnAVerificationRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the performQnAVerification operation of the ArcorUserRegistrySvc service to fetch the values of the user attributes that are stored in directory service.

   This operation returns the QnAVerificationResponse message that includes the transaction identifier, authentication token, and verification result. See the following section for more information on the response message.

Interpreting the Response Message

The response message, QnAAVerificationResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the verification result for each attribute and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Exceptions and Error Codes" (see page 209) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction performed by using UDS.
authToken	The authentication token that is returned if the credential verification to access the Web service was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web service. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
QnAResponseAttribute/name	The name of the attribute that was verified.

Element	**Description**
QnAResponseAttribute/result	The result of the verification: ■ MATCHED ■ NOT_MATCHED ■ NOT_VERIFIED ■ NOT_FOUND

Chapter 7: Collecting Device ID and DeviceDNA

Important! If you are an existing customer of RiskMinder and have integrated your application with a previous release of RiskMinder, then it is strongly recommended that you use the new APIs to leverage the full benefit of enhanced Device ID and DeviceDNA. In addition, the older APIs will be deprecated soon.

RiskMinder uses user-device (desktop computers, laptops, and notebooks) information as one of the parameters to determine the risk associated with a login attempt or a transaction. As a result, the verification of the online identity of the end user is a challenge. RiskMinder also uses Device ID and DeviceDNA technologies (in addition to other inputs, as discussed in "Understanding RiskMinder Workflows" (see page 19)) for this purpose. These technologies enable RiskMinder to build the user profile and to transparently provide accurate results by using the hardware that users already possess, without changing the end-user experience significantly.

This section provides detailed information on how to get and set Device ID and collect the DeviceDNA data from the end user's device and pass it to RiskMinder. It covers the following topics:

End-User Device Identification Basics

This section introduces you to the techniques RiskMinder uses to gather the end-user device identification information.

Device ID

The *Device ID* is a device identifier string that RiskMinder generates on the end user's device to identify and track the device that the end user uses for logging into your online application and performing transactions. The Device ID information is in encrypted format.

The following are the options for storing the Device ID on the end user's device. The plugin store is the most persistent storage option.

- Plugin store: The plugin store is a permanent store on the end user's device. A Device ID that is placed in the plugin store cannot be deleted by common end user actions such as clearing browser cache and deleting browser cookies. The plugin store is supported from CA RiskMinder Client release 2.1 onward.
- Local storage provided in HTML5
- UserData store: This store is available only in Microsoft Internet Explorer
- Cookie store: Typically, on Microsoft Windows, the Device ID is stored in one of the following folders:
  - **Internet Explorer on Microsoft Windows 7 or 2008:**
    `C:\Documents and Settings\`*`user_profile`*`\Application Data\Microsoft\Windows\`**`Cookies`**`\`
  - **Internet Explorer on Microsoft Windows 2003 or XP:**
    `C:\Documents and Settings\`*`user_profile`*`\`**`Cookies`**`\`
  - **Mozilla Firefox:**
    `C:\Documents and Settings\`*`user_profile`*`\Application Data\Mozilla\Firefox\Profiles\`*`random_dirname`*`\`**`cookies.sqlite`**
  - **Safari:**
    `C:\Documents and Settings\`*`user_name`*`\Application Data\Apple Computer\Safari\`**`cookies.plist`**

Important! From CA RiskMinder Client version 2.0 onward, the Device ID is not stored as a Flash cookie. If you have existing Flash cookies from an earlier release, then these cookies are automatically migrated to one of the stores listed earlier in this section.

Machine FingerPrint (MFP)

Machine FingerPrint (also referred to as Device fingerprinting or PC fingerprinting in industry terms) represents the browser information and device identification attributes (such as operating system, installed software applications, screen display settings, multimedia components, and other attributes) that are gathered from the end user's system and are analyzed to generate a risk profile of a device in real time. Some of the attributes that are collected from the end user's device include:

- Browser information (such as name, UserAgent, major version, minor version, JavaScript version, HTTP headers)
- Operating system name and version
- Screen settings (such as height, width, color depth)
- System information (such as time zone, language, system locale)

For every transaction performed by the end user, RiskMinder matches the corresponding MFP stored in its database with the incoming information. If this match percentage (%) is equal to or more than the value specified for the Device-MFP Match rule in Administration Console, then it is considered "safe".

DeviceDNA

DeviceDNA is a device identification and analytics technique that uses both Machine FingerPrint (MFP) (see page 153) and Device ID (see page 152) for more accurate information analyses. For accuracy, more information is collected than in case of MFP. For example:

- Additional system information (such as platform, CPU, MEP, system fonts, camera, and speaker information)
- Additional browser information (such as vendor, VendorSubID, BuildID)
- Additional screen settings (such as buffer depth, pixel depth, DeviceXDPI, DeviceYDPI)
- Plug-in information (such as QuickTime, Flash, Microsoft Windows Media Player, ShockWave, Internet Explorer plug-ins)
- Network information (such as connection type)

File that You Will Need

You will need the file listed in the following table, available when you install RiskMinder, to collect the Device ID and DeviceDNA information from the end user's device.

Location	File Name	Description
Microsoft Windows: *install_location*\ RiskMinder Systems\sdk\ **devicedna**\ **Solaris:** *install_location*/RiskMinder/**sdk**/**devicedna**/	riskminder-client.js	This file contains the functions to gather the Device ID- and DeviceDNA-related information from the end user's device and to generate the single-encoded String with all the DeviceDNA values.

Note: In the same location as riskminder-client.js, you will also see a file called riskminder-client.swf. This latter file is internally used by riskminder-client.js. So, you will not need to explicitly use this file.

However, riskminder-client.swf *must* always be present in the same location as riskminder-client.js, when you include it.

Configuring Device ID and DeviceDNA

To implement the functionality of the DeviceDNA and Device ID collection, you must implement corresponding code snippets into each page of your application that contains an event that requires risk assessment. For example, for risk assessment of a login event, your application must implement the required JavaScript files and code snippets into the login page. Similarly for a pre-login event, the steps discussed in this section must trigger when a user accesses the first page of your online application.

The steps to build the DeviceDNA and collect the Device ID from the end user's device are:

- Step 1: Include the Javascript File (see page 155)
- Step 2: Initialize Device ID and DeviceDNA Collection (see page 156)
- Step 3: Collect the Device ID and DeviceDNA (see page 158)
- Step 4: Collect the IP Address (see page 158)

You can implement these steps either in a single page of your online application, or across multiple pages (depending on how many pages you show during the login process) *before* you call the evaluateRisk method.

Step 1: Include the Javascript File

You will need to modify the appropriate Web pages, such as the login or index page (say, index.jsp or login.jsp) to enable them to gather MFP and DeviceDNA-related information, and collect the Device ID (cookie) from the end user's computer.

Note: See "Enrollment Workflows" (see page 19) for more information on when and how RiskMinder sets the Device ID on the end user's device.

To implement the script codes:

1. Copy the entire **devicedna** directory *from* the following location *to* the appropriate Web application folder (say, *APP_SERVER_HOME*/*Your_Application_Home*/**devicedna**/):
   - **On Microsoft Windows**
     `install_locatio>\Arcot Systems\sdk\`
   - **On UNIX-Based Platforms**

     install_location/Arcot Systems/**sdk**/
2. Include the **riskminder-client.js** file in the required application pages. We assume that these files are located in a folder that is relative to the folder containing **index.jsp**.
   ```
   <script type="text/javascript"
   src="devicedna/riskminder-client.js"></script>
   ```

Step 2: Initialize Device ID and DeviceDNA Collection

Note: Refer to the code in the "Sample Code Reference" (see page 159) section to understand this step better.

To implement the Device ID and DeviceDNA collection, include (declare) the following parameters in your HTML code *before* processing anything related to DeviceDNA:

```
<html>

<script type="text/javascript" src="devicedna/riskminder-client.js"></script>
<script type="text/javascript">

var client;

window.onload = function()
{
    init();

}

function init(){
    client = new ca.rm.Client();
    var contextPath = "<%=request.getContextPath()%>";

    client.setProperty("baseurl", contextPath);

    client.loadFlash(readyCallback);
}

function readyCallback(flag)
{

    // set desired configurations...
    configureClient();
    client.processDNA();


}

function configureClient() {

    // set the desired name for the cookie
    client.setProperty("didname", "rmclient");

    // turn off flash
    client.setProperty("noFlash", true);

    /// configure MESC values
    client.setProperty("mescmaxIterations", 2);
    client.setProperty("mesccalibrationduration", 150);
    client.setProperty("mescintervaldelay", 45);
    // etc...
    //Refer to the setProperty() API description in section, "Understanding the APIs
for Retrieving DeviceDNA in the Sample Code (see page 160)" for the complete list of
configuration parameters that you can use according to your requirements.
}

<body>

  //Your HTML code here

</body>

</html>
}
```

Note: Refer to the setProperty() API description in setProperty(key,val) (see page 161) for the complete list of configuration parameters that you can use.

Sample Application Reference

You can also refer to index.jsp, which is a part of the RiskMinder Sample Application. This file showcases the collection of DeviceDNA and other required information and sets these parameters for the session. After you deploy the Sample Application, this file is available at:
<*RISKFORT_SAMPLEAPP_HOME*>**index.jsp**

For example, if you are using Apache Tomcat 5.5, then the location of index.jsp will be <*Tomcat_Home*>\webapps\riskfort-3.1-sample-application\index.jsp.

Step 3: Collect the Device ID and DeviceDNA

You must now ensure that you now get the Device ID along with the DeviceDNA, as follows:

1. Ensure that on click of the **Login** (or **Submit**) button on the page, the following code snippet is called:
   ```
   <input type="button" value="Login"
   onClick="collectSystemInfo();">
   ```
2. Ensure that you have defined the collectSystemInfo() function. For example, you can use the following code snippet:
   ```
   function collectSystemInfo()
   {
     client.processDNA();

     var json = client.getDNA();
     var did = client.getDID();

     document.CollectMFPToEvaluate.DDNA = json;
     document.CollectMFPToEvaluate.DeviceID = did ;

     //post to server, both the DeviceDNA and Device ID values for risk eval

   }
   ```
3. After you have collected the DeviceDNA and the Device ID, as required, you must pass this collected information as input to evaluateRisk() method.

   See "Performing Risk Evaluation and Managing Associations" (see page 165) for more information.

Step 4: Collect the IP Address

RiskMinder does not provide any mechanism to collect the IP address of the end-user device. As a result, you must implement your own logic to do so.

See "Collecting the IP Address" (see page 163) for recommendations.

Sample Code Reference

The following sample code illustrates how to implement RiskMinder's DeviceDNA and Device ID collection mechanism. It showcases the collection logic in one file (say, index.jsp). However, you can implement appropriate code snippets in different pages, depending on the number of pages you show *before* you call the evaluateRisk() method.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<script type="text/javascript"
src="<%=request.getContextPath()%>/devicedna/riskminder-client.js"></script>
<script language="javascript">

var client;

 function init(){
try{
client = new ca.rm.Client();
var contextPath = "<%=request.getContextPath()%>";
client.setProperty("baseurl", contextPath);
client.loadFlash(readyCallback);
}catch(e){
alert(e.message);
}
}

 function collectingSystemInfo() {
try{
client.setProperty("externalip", "<%=request.getRemoteHost()%>");
computeDDNA();
}catch(e){
alert(e);
}
}

 function readyCallback(flag){
 configureClient();
 client.processDNA();
 }

 function configureClient(flag){
 //configure the client properties.
 client.setProperty("format", "json");
 client.setProperty("didname", "RISKFORT_COOKIE");
 }
 function computeDDNA() {

client.processDNA();

var dna = client.getDNA();
var did = client.getDID();

//forward this info to appropriate servlet to perform risk eval
document.CollectMFPToEvaluate.IpAddress.value = '<%=request.getRemoteHost()%>';
document.CollectMFPToEvaluate.CallerID.value = "MyCallerID";
document.CollectMFPToEvaluate.DeviceID.value = did;
document.CollectMFPToEvaluate.MFP.value = dna;

document.CollectMFPToEvaluate.submit();
 }
 </script>
</head>

<body onload="init()">
<form name="CollectMFPToEvaluate" method="POST" action="ArRFMFPCollectionServlet">
<input type="hidden" name="MFP" value="">
<input type="hidden" name="IpAddress">
<input type="hidden" name="CallerID">
<input type="hidden" name="DeviceID">
<h1 align="center">Arcot RiskFort Sample Application</h1>
<input type="button" style="width: 150px" name="Login" value="Login"
onclick="collectingSystemInfo();"/>
</form>
</body>
</html>
```

Understanding the APIs for Retrieving DeviceDNA in the Sample Code

The RiskMinder Client runs on the client browser and collects the device signature and Device ID. All the client-side controls for RiskMinder are provided in the RiskMinder Client Javascript API. This API allows you to program the functionality of the client using JavaScript.

This section describes the RiskMinder Client APIs that are used to retrieve the DeviceDNA.

ca.rm.Client()

Main JavaScript class that exposes all the published APIs of the RiskMinder Client.

getVersion()

Returns a String that specifies the version of the RiskMinder Client. The current supported version is 2.1.

setProperty(key,val)

Specifies the configuration values for the RiskMinder Client. The following table describes the properties that you can set for this method.

Property Key	**Description**
baseurl	The context path of the Web application that is using DeviceDNA. This value *must* be set immediately after creating an instance of ca.rm.Client JavaScript object. No default value is supported.
didname	The cookie or local storage item name. Device ID cookie is set by using this name. The value should be a string.
flashdatastorename	The name of the Flash local store where the Flash Device ID was stored (in the previous releases).
flashPath	Not being used currently. This property is reserved for future use.
format	The format in which DeviceDNA results should be returned. The value should be one of the following a strings: ■ HTML ■ JSON
jobs	Not being used currently. This property is reserved for future use.
store	The storage area for Device ID. The value should be one of the following strings: ■ cookie ■ localstorage ■ plugin ■ default
externalIP	The IP address of the system from which the page containing the Client was served.
noFlash	The indication whether the Flash movie bundled with the Client should be used for gathering additional attributes for DeviceDNA. The value should be boolean - true or false. By default, noFlash is set to false, which implies that the Flash movie will be used.

Property Key	Description
MESC-Related Configurations **MESC** stands for Machine Effective Speed Calculations. An attempt is made to estimate processor speed by executing several runs of batched arithmetic operations. In our case, it is integer addition for specified intervals of time.	
mescmaxIterations	Specifies how many runs of the batched arithmetic operations should be executed. Default value is 2.
mesccalibrationDuration	Specifies the duration for which each batch of arithmetic operations should run. The value is specified in milliseconds. The default value is 200ms.
mescintervalDelay	Specifies the delay (in number of milliseconds) between successive runs. The value is specified in milliseconds. The default value is 50ms.

getProperty(key)

This API returns the currently defined value for the property represented by the *key*. Key values are same as for setProperty(). See setProperty(key,val) (see page 161) for more information.

loadFlash(callback)

This API loads the flash movie that is part of the RiskMinder Client and initializes it. The Callback function should be a JavaScript function taking a boolean flag as parameter and defined in the Web page that is calling this method.

At the end of initialization, the Callback function is invoked with parameter set to true, if the Flash movie initialization was successful. Else, the Callback function is invoked with parameter set to false.

processDNA()

This is the main API of the RiskMinder Client. It retrieves a number of system attributes from the end-user system and from the software installed on this system. It then computes the corresponding DeviceDNA using these values.

All the configuration settings are taken into consideration by the processDNA function while computing the DeviceDNA.

getDNA()

This API returns a string that represents the end-user system’s DeviceDNA, as computed by the RiskMinder Client. The DeviceDNA string can either be in the HTML format or JSON format. This is controlled by the value that is specified for the format property.

getTimeTaken()

This API returns the time taken (in milliseconds) by the processDNA() call to compute the end-user system’s DeviceDNA.

setDID(value)

This function stores the Device ID on the end user's device. The Device ID string must be specified in the value parameter of the function.

getDID()

This function returns the Device ID that has been stored on the end user's device. This function also migrates older Flash cookie with the same cookie name (if present) to one of the supported stores.

deleteDID()

This function deletes the Device ID that has been set on the end user's device by the RiskMinder Client.

Collecting the IP Address

The end user accessing your online application might be a home user or might be accessing it from their corporate network. In case of latter category of users, chances are that they might be "hidden" behind a proxy server. As a result, the way you will collect the IP address of an end user who is accessing your online application from behind a proxy will be different from the user who accesses it directly from home.

If the End User is Accessing Your Application Directly

If the end user is accessing your application directly, then you can use the getRemoteAddr() method of the HttpServletRequest interface in your JSP. This method returns a string that contains the IP address of the client that sent the request.

Chapter 8: Performing Risk Evaluation and Managing Associations

When a user accesses your online application, the application forwards the request to RiskMinder for risk analysis. RiskMinder evaluates the risk for all users, irrespective of whether they are first-time users (and therefore not "known" to RiskMinder) or they are already enrolled with the RiskMinder system.

The Risk Evaluation Web service enables you to send risk evaluation requests to RiskMinder Server. This Web service creates a request message and sends it to RiskMinder Server, receives the response back from the Server, and packages it as return structures to be read by the client.

This section provides an overview of how to use the Risk Evaluation Web service to perform risk evaluation, post-evaluation, and association management operations that the Web service implements. It covers the following topics:

To perform the operations discussed in this section, you must use the **ArcotRiskFortEvaluateRiskService.wsdl** file. This service represents the client-side interface to RiskMinder Server’s risk evaluation functionality and exposes the supported operations for risk evaluation workflows.

Evaluating Risk

To evaluate the risk associated with a transaction, you need to use the RiskFortEvaluateRiskSvc service (available through ArcotRiskFortEvaluateRiskService.wsdl.)

This section walks you through the following topics:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

You must use the evaluateRiskRequest message to evaluate the risk associated with a transaction. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
callerId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking related transactions.
Device Context Elements		
deviceContext	No	The end-user device details, as described by aggregatorID, deviceIDs, deviceSignature, and shortDeviceSignature.
deviceContext/ aggregatorID	No	The unique ID of the third-party vendor who provides account aggregation services by collating specified information of users across multiple enterprises.
deviceContext/ deviceIDs	No	Defined by the DeviceIDItem element, this element describes the unique identifier information to identify and track the device that the end user uses to log in to your online application and perform transactions: ■ deviceIDType: The string that identifies the storage type used to store the Device ID. ■ DeviceIDValue: The corresponding value for deviceIDType. **Note:** You can add more than one DeviceIDItem element, with deviceIDType and DeviceIDValue pairs.

Element	Mandatory	Description
deviceContext/ deviceSignature	No	The Machine FingerPrint (MFP) that RiskMinder's MFP Collector builds this on the client side. This signature contains information related to the end-user's device, such as browser details, system details, plug-in details, and screen width.
deviceContext/ shortDeviceSignature	No	The short form of the deviceSignature.
Location Context Elements		
locationContext	No	The transaction location details, as described by clientIPAddress, longitude, latitude, continent, country, countryISO2, region, state, city, connectionType, and lineSpeed.
locationContext/ clientIPAddress	No	The Internet Protocol (IP) address of the end-user system in the public address space.
locationContext/ longitude	No	A floating point number, with positive numbers representing East and negative numbers representing West.
locationContext/ latitude	No	A floating point number, with positive numbers representing North and negative numbers representing South.
locationContext/ continent	No	The continent from where the transaction originated: ■ Africa ■ Antarctica ■ Asia ■ Australia ■ Europe ■ North America ■ Oceania (Melanesia, Micronesia, Polynesia) ■ South America
locationContext/ country	No	The country from where the transaction originated.
locationContext/ countryISO2	No	The two-letter country code (as defined in ISO 3166-1) from where the transaction originated.

Element	Mandatory	Description
locationContext/ region	No	The district or territory from where the transaction originated.
locationContext/ state	No	The first-level administrative division within each country (if one exists) from where the transaction originated.
locationContext/ city	No	The city from where the transaction originated.
locationContext/ connectionType	No	The type of data connection between the end-user's device and their Internet Service Provider (ISP): ■ **Satellite:** High-speed broadband links between a user and a geosynchronous satellite. ■ **OCX:** The OC-3 circuits and OC-48 circuits that are used by large backbone carriers. ■ **TX:** Old links of type T-3 circuits and T-1 circuits. ■ **Frame Relay:** High-speed alternatives to TX. ■ **Dialup:** Modems that operate at 56kbps. ■ **Cable:** Cable modem broadband circuits, primarily offered by cable TV companies. ■ **DSL:** Digital Subscriber Line broadband circuits that include aDSL, iDSL, and sDSL. ■ **ISDN:** High-speed Integrated Services Digital Network technology with specialized modems and switches. ■ **Fixed Wireless:** Wireless connections where the location of the receiver is fixed. ■ **Mobile Wireless:** Wireless connections where the location of the receiver is mobile.
locationContext/ lineSpeed	No	The speed of the user's Internet connection. This is based on connectionType.
User Context Elements		
userContext	No	The user details, as described by orgName and userName.
userContext/ orgName	No	The name of the organization to which the end user belongs.

Element	Mandatory	Description
userContext/ userName	Yes	The name of the user who performed the transaction.
Transaction Context Elements		
transactionContext	No	The transaction details, as described by action and channel.
transactionContext/ action	No	The type of transaction performed by the user, which can be: ■ Login ■ Wire Transfer ■ Any other value that you specify through your application
transactionContext/ channel	No	The channel from which the transaction originated: ■ **Web:** Transactions initiated through a Web browser. The originator may be a computer, smart phone, tablet, or set-top box. ■ **SMS:** Transactions initiated through SMS messaging. ■ **App:** Transactions initiated through smart phone, tablet application, or set-top box embedded applications. ■ **3DSecure:** Online transactions initiated using credit card or debit card. ■ **ATM:** Transactions initiated through an Automated Teller Machine. ■ **PoS:** Transactions initiated at physical point of sale.
Administration Context Type Elements		
adminContextType	No	The administrator details, as described by orgName, adminName, and locale, who initiated the Web service call.
adminContextType/ orgName	No	The name of the organization to which the administrator who initiated the Web service call belongs.
adminContextType/ adminName	No	The name of the administrator who initiated the Web service call.

Element	Mandatory	Description
adminContextType/locale	No	The locale used by the administrator. The output message will be converted to this locale.
Additional Input Elements		
additionalInput	No	Enables you to set additional inputs if you want to augment RiskMinder's risk evaluation capability by specifying additional information. In such cases, you must set the extra information in *name-value* pairs. ■ name: The name with which you want to create the key pair. ■ value: The corresponding value for name. **Note:** You can add more than one of these elements.

Invoking the Web Service

To evaluate the risk associated with a transaction:

1. (Optional) Include the authentication and authorization details in the header of the evaluateRisk operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the evaluateRiskRequest elements to set the required information, as listed in the table.
3. Use the evaluateRiskRequest message and construct the input message by using the details specified in preceding step.
4. Invoke the evaluateRisk operation of the RiskFortEvaluateRiskSvc service to perform risk evaluation.

   This operation returns the evaluateRiskResponse message that includes the risk assessment elements and the success result. See the following section for more information on the response message.

Interpreting the Response Message

The response message, evaluateRiskResponse, returns the risk assessment elements and the success result in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the riskfortFault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
Risk Assessment Elements	

Element	Description
riskAssessment	Contains the following details of the transaction: ■ advice An action (ALERT, ALLOW, DENY, INCREASEAUTH) suggested by the Risk Assessment module after evaluating the score of the transaction. ■ outputDeviceID The Device ID (cookie) information for the device. ■ score The score generated based on device details, location details, and user details. ■ matchedRuleMnemonic The rules that matched and for which RiskMinder flagged the transaction as risky. ■ ruleAnnotation The result of execution of all rules (or the reason for score and advice). ■ transactionID The unique identifier of the transaction. ■ deviceContext The gathered Machine FingerPrint (MFP) of the end-user's device. ■ locationContext The gathered location details where the device was used to perform the transaction. ■ userContext The gathered details of the user who performed the transaction.
RiskFort Success Elements	
riskFortSuccess	Contains the following information related to the result of the operation: ■ successMessage A string that defines the status of the operation. ■ transactionID The unique transaction identifier.

Performing Post Evaluation

The Post Evaluation operation accepts input from the Risk Evaluation operation and updates the device signature and other information for the specified user, if it changed. This operation also creates or updates user-device associations, if required.

To perform the subsequent post-evaluation *after* you have completed the risk evaluation of a transaction, you must use the RiskFortEvaluateRiskSvc service (available through ArcotRiskFortEvaluateRiskService.wsdl). This service represents the client-side interface to RiskMinder Server's post-evaluation functionality and exposes the supported operations.

This section walks you through the following topics:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

You must use the postEvaluateRequest message to perform post-evaluation tasks. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
callerId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking related transactions.
Risk Assessment Elements		
advice	Yes	An action (ALERT, ALLOW, DENY, INCREASEAUTH) obtained from the riskAssessment element of the evaluateRiskResponse message.
outputDeviceID	Yes	The Device ID (cookie) obtained from the riskAssessment element of the evaluateRiskResponse message.
score	Yes	The score obtained from the riskAssessment element of the evaluateRiskResponse message.
matchedRuleMnemonic	Yes	Obtained from the riskAssessment element of the evaluateRiskResponse message, the rules that matched and for which RiskMinder flagged the transaction as risky.
ruleAnnotation	Yes	The result of execution of all rules (or the reason for score and advice), as obtained from the riskAssessment element of the evaluateRiskResponse message.

Element	Mandatory	Description
transactionID	Yes	Unique transaction identifier received from the riskFortSuccess element of evaluateRiskResponse message. Your application can include this identifier for tracking purposes.
deviceContext	No	The MFP details of the end-user's device obtained from the riskAssessment element of the evaluateRiskResponse message. This element is further described by the following attributes: ■ aggregatorID The unique ID of the third-party vendor who provides account aggregation services by collating specified information of users across multiple enterprises. ■ deviceIDs Defined by the DeviceIDItem element, this element describes the unique identifier information to identify and track the device that the end user uses to log in to your online application and perform transactions: – deviceIDType: The string that identifies the storage type that is used to store the Device ID. -- DeviceIDValue: The corresponding value for deviceIDType. **Note:** You can add more than one DeviceIDItem element, with deviceIDType and DeviceIDValue pairs. ■ deviceSignature The Machine FingerPrint (MFP) that RiskMinder's MFP Collector builds this on the client-side. This signature contains information related to the end-user's device, such as browser details, system details, plug-in details, and screen width. ■ shortDeviceSignature The short form of the deviceSignature.

Element	Mandatory	Description
locationContex t	No	The transaction location details, obtained from the riskAssessment element of the evaluateRiskResponse message. This element is further described by the following attributes: ■ clientIPAddress The Internet Protocol (IP) address of the end-user system in the public address space. Not mandatory. ■ longitude A floating point number, with positive numbers representing East and negative numbers representing West. Not mandatory. ■ latitude A floating point number, with positive numbers representing North and negative numbers representing South. Not mandatory. ■ continent The continent from where the transaction originated: – Africa – Antarctica – Asia – Australia – Europe – North America – Oceania (Melanesia, Micronesia, Polynesia) – South America ■ country The country from where the transaction originated. Not mandatory. ■ countryISO2 The two-letter country code (as defined in ISO 3166-1) from where the transaction originated. Not mandatory. ■ region The district or territory from where the transaction originated. Not mandatory. ■ state The first-level administrative division within each country (if one exists) from where the transaction originated. Not mandatory. ■ city The city from where the transaction originated. Not mandatory. ■ connectionType The type of data connection between the end-user's device and their Internet Service Provider (ISP): – **Satellite:** High-speed broadband links between a user and a geosynchronous satellite.

Element	Mandatory	Description
userContext	Yes (userName is mandatory)	The user details, obtained from the riskAssessment element of the evaluateRiskResponse message. This element is further described by the following attributes: ■ orgName The name of the organization to which the end user belongs. This attribute is optional. ■ userName The name of the user who performed the transaction. This attribute is mandatory.
transactionContext	No	The transaction details, obtained from the riskAssessment element of the evaluateRiskResponse message. This element is further described by the following attributes: ■ action The type of transaction performed by the user, which can be: – Login – Wire Transfer – Any other value that you specify through your application ■ channel The channel from which the transaction originated: – **Web:** Transactions initiated through a Web browser. The originator may be a computer, smart phone, tablet, or set-top box. – **SMS:** Transactions initiated through SMS messaging. – **App:** Transactions initiated through smart phone, tablet application, or set-top box embedded applications. – **3DSecure:** Online transactions initiated using credit card or debit card. – **ATM:** Transactions initiated through an Automated Teller Machine. – **PoS:** Transactions initiated at physical point of sale.

Element	Mandatory	Description
additionalOutput	No	Enables you to set additional outputs that you got from RiskMinder's risk evaluation request. In such cases, you need to set the extra information in *name-value* pairs. ■ name: The name with which you want to create the key pair. ■ value: The corresponding value for name. **Note:** You can add more than one of these elements.
Secondary Authentication Status Element		
secondaryAuthenticationStatus	Yes	The result of the additional authentication that your application might have performed based on the advice obtained from the riskAssessment element of evaluateRiskResponse: ■ 0: Indicates that your application denied the transaction. ■ 1: Indicates that the transaction was allowed.
Association Element		
associationName	No	The string identifier for the user-to-device association in the system.
Administration Context Type Elements		
adminContextType	No	The administrator details, as described by orgName, adminName, and locale, who initiated the Web service call.
adminContextType/ orgName	No	The name of the organization to which the administrator who initiated the Web service call belongs.
adminContextType/ adminName	No	The name of the administrator who initiated the Web service call.
adminContextType/ locale	No	The locale used by the administrator.
Additional Input Elements		

Element	Mandatory	Description
additionalInput	No	Enables you to set additional inputs if you want to augment RiskMinder's post-evaluation capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements.

Invoking the Web Service

To perform post-evaluation tasks:

1. (Optional) Include the authentication and authorization details in the header of the postEvaluate operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use postEvaluateRequest elements to set the required information, as listed in the table.
3. Use the postEvaluateRequest message and construct the input message by using the details specified in the preceding step.
4. Invoke the postEvaluate operation of the RiskFortEvaluateRiskSvc service for post evaluation of a transaction.

   This operation returns the postEvaluateResponse message that includes the final risk advice, indicating whether the result was updated successfully, and the transactionID. See the following section for more information on the response message.

Interpreting the Response Message

The response message, postEvaluateResponse, returns the final risk advice, indicating whether the result was updated successfully, and the transactionID in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the riskfortFault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
isAllowAdvised	Contains the final risk advice, generated as a result of post evaluation: ■ true: Indicates the final advice was ALLOW. ■ false: Indicates the final advice was DENY.
RiskFort Success Elements	

Element	Description
riskFortSuccess	Contains the string that indicates whether the information was successfully updated in the database or not.
transactionID	The unique transaction identifier.

Listing Associations

RiskMinder uniquely identifies a user as a valid user of your system by automatically associating (or binding) a user to the device that they use to access your application. This is referred to as an *association* (or device binding) in RiskMinder terminology. Users who are not bound are more likely to be challenged in order to be authenticated.

RiskMinder also allows users to be bound to more than one device. For example, a user can use a work computer and a home computer to access your application. Similarly, you can bind a single device to more than one user. For example, members of a family can use one computer to access your application.

Important! It is recommended that you discourage users from creating associations with publicly shared devices, such as systems in an Internet cafe or kiosk.

This section walks you through the following tasks for listing stored user-device associations for a specified user:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

You must use the listAssociationsRequest message to view all known associations for the specified user. The following table lists the elements of this request message.

Element	Mandatory	Description
callerId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking related transactions.
User Context Elements		
userContext	No	The user details, as described by orgName and userName.
userContext/ orgName	No	The name of the organization to which the end user belongs.

Element	**Mandatory**	**Description**
userContext/ userName	Yes	The name of the user who performed the transaction.
Administration Context Type Elements		
adminContextType	No	The administrator details, as described by orgName, adminName, and locale, who initiated the Web service call.
adminContextType/ orgName	No	The name of the organization to which the administrator who initiated the Web service call belongs.
adminContextType/ adminName	No	The name of the administrator who initiated the Web service call.
adminContextType/ locale	No	The locale used by the administrator. The output message is converted to this locale.
Additional Input Elements		
additionalInput	No	Enables you to set additional inputs if you want to augment RiskMinder's risk evaluation capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name: The name with which you want to create the key pair. ■ value: The corresponding value for name. **Note:** You can add more than one of these elements.

Invoking the Web Service

To list all the stored associations for a specified user:

1. (Optional) Include the authentication and authorization details in the header of the listAssociations operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the listAssociationsRequest elements to set the required information, as listed in the table.
3. Use the listAssociationsRequest message and construct the input message by using the details specified in preceding step.
4. Invoke the listAssociations operation of the RiskFortEvaluateRiskSvc service to list all associations for the given user.

   This operation returns the listAssociationsResponse message that includes the association details and the success result. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listAssociationsResponse, returns the list and details of all known associations for the specified user in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the riskfortFault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
Association Elements	
associationName	The name(s) of device association(s) found for the specified user.
creationDate	The date and time when the association was created.
deviceID	The corresponding Device ID(s) extracted from the Device ID store on the end user's computer.
status	The status of the association: ■ 1: Indicates that the association is valid and active. ■ 0: Indicates that the association is not valid any more and has been deleted.
RiskFort Success Elements	
successMessage	Contains the string that indicates whether the operation was successful or not.
transactionID	The unique transaction identifier.

Deleting Associations

RiskMinder also enables you to delete user-device associations. However internally, RiskMinder does not remove the association entry. It merely sets this value to 0 for the given association.

This section walks you through the following tasks for deleting stored user-device associations for a specified user:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

You must use the deleteAssociationRequest message to delete the specified associations for a user. The following table lists the elements of this request message.

Element	Mandatory	Description
callerId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking related transactions.
User Context Elements		
userContext	No	The user details, as described by orgName and userName.
userContext/ orgName	No	The name of the organization to which the end user belongs.
userContext/ userName	Yes	The name of the user who performed the transaction.
Association Element		
associationName	Yes	The name of the association that you want to delete.
Administration Context Type Elements		
adminContextType	No	The administrator details, as described by orgName, adminName, and locale, who initiated the Web service call.
adminContextType/ orgName	No	The name of the organization to which the administrator who initiated the Web service call belongs.

Element	Mandatory	Description
adminContextType/ adminName	No	The name of the administrator who initiated the Web service call.
adminContextType/ locale	No	The locale used by the administrator. The output message is converted to this locale.
Additional Input Elements		
additionalInput	No	Enables you to set additional inputs if you want to augment RiskMinder's risk evaluation capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name: The name with which you want to create the key pair. ■ value: The corresponding value for name. **Note:** You can add more than one of these elements.

Invoking the Web Service

To delete the listed associations for a specified user:

1. (Optional) Include the authentication and authorization details in the header of the deleteAssociation operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the deleteAssociationRequest elements to set the required information, as listed in the table.
3. Use the deleteAssociationRequest message and construct the input message by using the details specified in preceding step.
4. Invoke the deleteAssociation operation of the RiskFortEvaluateRiskSvc service to delete an association.

   This operation returns the deleteAssociationResponse message that includes the details about deleted association(s) and the success result. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteAssociationResponse, returns the list and details of all deleted associations for the specified user in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the riskfortFault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
Association Elements	
associationName	The name of device association that was deleted for the specified user.
creationDate	The date and time when the association was created.
deviceID	The corresponding Device ID(s) extracted from the Device ID store on the end user's device.
status	The status of the association: ■ 0: Indicates that the association is not valid any more and has been deleted. ■ 1: Indicates that the association is valid and active.
RiskFort Success Elements	
successMessage	Contains the string that indicates whether the operation was successful or not.
transactionID	The unique identifier of the transaction.

Chapter 9: Performing Selected Administration Tasks

This section provides an overview of how to use the RiskMinder Administration Web service to perform the following tasks:

- Adding a User to Exception List (see page 186)
- Deleting a User from Exception List (see page 188)
- Fetching User Profile Information (see page 190)
- Fetching Location and Connection Information (see page 192)

To perform the operations discussed in this section, you must use the **ArcotRiskFortAdminWebService.wsdl** file. This service provides a limited client-side interface to Administration Console functionality and exposes the supported operations for risk evaluation workflows.

Adding a User to Exception List

You might want to temporarily exclude a user in your organization from risk evaluation during a specific time interval. For example, if a user travels to a country that is configured as negative in RiskMinder, then for the specified interval while they are there, RiskMinder's advice will always be DENY. To prevent that, their status can be changed to an *exception user*. In this case, if they perform a transaction during this interval, despite their IP address being negative, RiskMinder will return a low risk score and the advice will typically be ALLOW.

Book: You can also perform this operation by using the Case page in Administration Console. See *CA RiskMinder Administration Guide* for detailed instructions to do so.

To add a user to Exception User List, you must use the ArcotRiskFortAdminSvc service (available through ArcotRiskFortAdminWebService.wsdl) for:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

You must use the addUserToExceptionListRequest message to add a user to the Exception User List. The following table lists the elements of this request message.

Element	Mandatory	Description
userName	Yes	The name of the user who you want to add to the Exception User List.
groupName	Yes	The name of the organization to which the user belongs.
startDate	Yes	The date (in *yyyy-mm-dd* format) and time from which you want the user to be exempted from RiskMinder risk evaluation. For example: 2012-10-04+05:30
endDate	Yes	The date (in *yyyy-mm-dd* format) and time till which you want the user to be exempted from RiskMinder risk evaluation.
reason	Yes	The reason for which the user is being added to the Exception User List.
callerId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking related transactions.

Invoking the Web Service

To add a user to the Exception User List:

1. (Optional) Include the authentication and authorization details in the header of the addUserToExceptionList operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the addUserToExceptionListRequest elements to set the required information, as listed in the table.
3. Use the addUserToExceptionListRequest message and construct the input message by using the details specified in preceding step.
4. Invoke the addUserToExceptionList operation of the ArcotRiskFortAdminSvc service to add the user to the list.

   This operation returns the addUserToExceptionListResponse message that includes the status of the operation and success result. See the following section for more information on the response message.

Interpreting the Response Message

The response message, addUserToExceptionListResponse, returns the status of the operation and the success result in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the AdminFault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
code	The status of the operation: ■ 0: Indicates that the user was successfully added to the Exception User List. ■ 1: Indicates that the operation failed.
message	Contains the string that indicates whether the information was successfully updated in the database or not.
transactionID	The unique transaction identifier.

Deleting a User from Exception List

To delete a user from the Exception User List, you must use the ArcotRiskFortAdminSvc service (available through ArcotRiskFortAdminWebService.wsdl). This section walks you through the following topics:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

You must use the deleteUserFromExceptionListRequest message to remove a user from the Exception User List. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
userName	Yes	The name of the user who you want to delete from the Exception User List.
groupName	Yes	The name of the organization to which the user belongs.
moveReason	Yes	The reason for which the user is being deleted from the Exception User List.
callerId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking related transactions.

Invoking the Web Service

To delete a user from the Exception User List:

1. (Optional) Include the authentication and authorization details in the header of the deleteUserFromExceptionList operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the deleteUserFromExceptionListRequest elements to set the required information, as listed in the table.
3. Use the deleteUserFromExceptionListRequest message and construct the input message by using the details specified in preceding step.
4. Invoke the deleteUserFromExceptionList operation of the ArcotRiskFortAdminSvc service to add the user to the list.

   This operation returns the deleteUserFromExceptionListResponse message that includes the status of the operation and success result. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteUserFromExceptionListResponse, returns the status of the operation and the success result in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the AdminFault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	Description
code	The status of the operation: ■ 0: Indicates that the user was successfully removed from the Exception User List. ■ 1: Indicates that the operation failed.
message	Contains the string that indicates whether the information was successfully updated in the database or not.
transactionID	The unique transaction identifier.

Fetching User Profile Information

To view the details of a specified user, you must use the ArcotRiskFortAdminSvc service (available through ArcotRiskFortAdminWebService.wsdl). This section covers the following topics:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

You must use the getUserProfileRequest message to view detailed information about the specified user. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
userName	Yes	The name of the user whose details you want to see.
groupName	Yes	The name of the organization to which the user belongs.
callerId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking related transactions.

Invoking the Web Service

To view a user’s information:

1. (Optional) Include the authentication and authorization details in the header of the getUserProfile operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the getUserProfileRequest elements to set the required information, as listed in the table.
3. Use the getUserProfileRequest message and construct the input message by using the details specified in preceding step.
4. Invoke the getUserProfile operation of the ArcotRiskFortAdminSvc service to fetch the user information.

   This operation returns the getUserProfileResponse message that includes the status of the operation, user details, and success result. See the following section for more information on the response message.

Interpreting the Response Message

The response message, getUserProfileResponse, returns the status of the operation, user details, and the success result in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the AdminFault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
code	The status of the operation: ■ 0: Indicates that the user details were successfully fetched from the database. ■ 1: Indicates that the operation failed.
message	Contains the string that indicates whether the information was successfully retrieved from the database or not.
transactionID	The unique transaction identifier.
userName	The name of the specified user.
firstName	The first name of the specified user.
lastName	The last name of the specified user.
emailAddress	The email ID of the specified user.
isExceptionUser	Indicates whether the user is an exception user or not: ■ true: The user is an exception user. ■ false: The user is not an exception user.

Fetching Location and Connection Information

To view the connection details by using the specified IP address, you must use the ArcotRiskFortAdminSvc service (available through ArcotRiskFortAdminWebService.wsdl). This section covers the following topics:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

You must use the getLocationAndConnectionInfoRequest message to view the detailed connection information for the specified IP address. The following table lists the elements of this request message.

Element	Mandatory	Description
ip	Yes	The IP address of the user whose details you want to see.
callerId	No	Unique transaction identifier that your calling application can include. This identifier helps in tracking related transactions.

Invoking the Web Service

To view the connection information for the specified IP address:

1. (Optional) Include the authentication and authorization details in the header of the getLocationAndConnectionInfo operation. See "Managing Web Services Security" (see page 37) for more information on the header elements.
2. Use the getLocationAndConnectionInfoRequest elements to set the IP address information, as listed in the table.
3. Use the getLocationAndConnectionInfoRequest message and construct the input message by using the details specified in preceding step.
4. Invoke the getLocationAndConnectionInfo operation of the ArcotRiskFortAdminSvc service to view the connection information.

   This operation returns the getLocationAndConnectionInfoResponse message that includes the status of the operation, connection details, and success result. See the following section for more information on the response message.

Interpreting the Response Message

The response message, getLocationAndConnectionInfoResponse, returns the status of the operation, connection details, and success result in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the AdminFault response is returned. See appendix, "Exceptions and Error Codes" (see page 209) for more information on the SOAP error messages.

Element	**Description**
code	The status of the operation: ■ 0: Indicates that the connection details were successfully fetched from the database. ■ 1: Indicates that the operation failed.
message	Contains the string that indicates whether the information was successfully retrieved from the database or not.
transactionID	The unique transaction identifier.
locationContext	The connection and location details, as described by city, cityCF, state, stateCF, country, countryCF, countryISOCode, postalCode, timeZone, longitude, latitude, aolFlag, connectionType, routingType, and lineSpeed.
locationContext/ city	The city from where the IP address originated.

Element	Description
locationContext/ cityCF	The city code from where the IP address originated.
locationContext/ state	The first-level administrative division within each country (if one exists) from where the IP address originated.
locationContext/ stateCF	The state code from where the IP address originated.
locationContext/ country	The country from where the IP address originated.
locationContext/ countryCF	The country code from where the IP address originated.
locationContext/ countryISOCode	The two-letter country code (as defined in ISO 3166-1) from where the IP address originated.
locationContext/ postalCode	The postal (ZIP) code of the location from where the IP address originated.
locationContext/ timeZone	The timezone of the location from where the IP address originated.
locationContext/ longitude	A floating point number (with positive numbers representing East and negative numbers representing West) from where the IP address originated.
locationContext/ latitude	A floating point number (with positive numbers representing North and negative numbers representing South) from where the IP address originated.
locationContext/ aolFlag	The indication whether the specified IP is part of the AOL network: ■ Y: The user with the specified IP address is a member of the AOL service. ■ N: The user with the specified IP address is not a member of the AOL service.

Element	Description
locationContext/ connectionType	The type of data connection between the end-user's device and their Internet Service Provider (ISP): ■ **Satellite:** High-speed broadband links between a user and a geosynchronous satellite. ■ **OCX:** The OC-3 circuits, OC-48 circuits that are used by large backbone carriers. ■ **TX:** Old links of type T-3 circuits and T-1 circuits. ■ **Frame Relay:** High-speed alternatives to TX. ■ **Dialup:** Modems that operate at 56kbps. ■ **Cable:** Cable modem broadband circuits, primarily offered by cable TV companies. ■ **DSL:** Digital Subscriber Line broadband circuits that include aDSL, iDSL, and sDSL. ■ **ISDN:** High-speed Integrated Services Digital Network technology with specialized modems and switches. ■ **Fixed Wireless:** Wireless connections where the location of the receiver is fixed. ■ **Mobile Wireless:** Wireless connections where the location of the receiver is mobile.
locationContext/ routingType	The IP routing method used for the connection: ■ **Fixed:** Cable, DSL, OCX ■ **AOL:** AOL users ■ **POP:** Dial up to regional ISP ■ **Super POP:** Dial up to multi-state ISP ■ **Cache Proxy:** Accelerator proxy, content distribution service ■ **Regional Proxy:** Proxy for multiple states in a country ■ **Anonymizer:** Anonymizing proxy ■ **Satellite:** Consumer satellite or backbone satellite ISP ■ **International Proxy:** Proxy funneling international traffic ■ **Mobile Gateway:** Mobile device gateway to Internet ■ **Unknown:** Cannot currently be determined
locationContext/ lineSpeed	The speed of the user's Internet connection. This is based on connectionType.

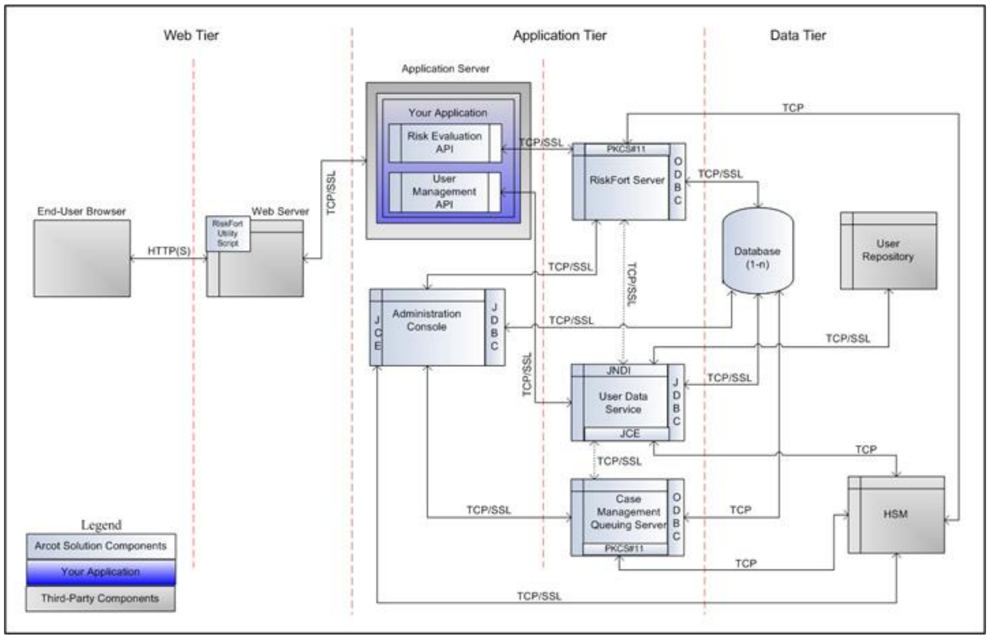
Appendix A: Additional Configurations

This appendix discusses the following miscellaneous topics:

SSL Communication Between RiskMinder Components

In addition to supporting TCP-based communication between RiskMinder Server and the SDKs, RiskMinder supports Secure Socket Layer (SSL) for secure communication between these components. RiskMinder can be configured for one-way Secure Socket Layer (SSL) with server-side certificates or two-way SSL with server-side and client-side certificates between the Server and SDKs, as shown in the following figure.

Communication Modes



Although the default mode of communication is TCP, RiskMinder Server supports SSL communication (two-way as well as one-way) with the following components to ensure integrity and confidentiality of the data being exchanged during a transaction:

- Case Management Queuing Server
- RiskMinder Database
- User Data Service
- RiskMinder Risk Evaluation SDK
- Sample Application
- Evaluation Callout
- Scoring Callout

Note: RiskMinder enables you to write your own custom Evaluation rule, based on your business requirements. This custom rule is called **Evaluation Callout**. Similarly, RiskMinder also enables you to write your own custom Scoring logic called **Scoring Callout**.

Refer to *CA RiskMinder Administration Guide* for more information on these Callouts.

Setting Up SSL Communication Between Risk Evaluation Web Service and RiskFort Server

To enable RiskMinder Web services for SSL communication, you must first configure your client that accesses the Web service for SSL communication, and then configure the Transaction Web Service protocol by using Administration Console. This section describes how to set up the following between the Risk Evaluation Web service and RiskMinder Server:

- One-Way SSL (see page 199)
- Two-Way SSL (see page 200)

One-Way SSL

To set up one-way SSL between the Risk Evaluation Web service and RiskMinder Server:

1. Ensure that you are logged in as the Master Administrator (MA).
2. Activate the **Services and Server Configurations** tab.
3. Ensure that the **RiskFort** tab in the submenu is active.
4. Under the **Instance Configuration** section, click the **Protocol Configuration** link to display the Protocol Configuration page.
5. Select the **Server Instance** for which you want to configure the SSL communication.
6. In the **List of Protocols** section, click the **Transaction Web Service** link.

   The page to configure the Transaction Web Service protocol appears.
7. Configure the following fields:
   - Ensure that the **Protocol Status** is **Enabled**.

     If not, then select the **Change Protocol Status** option and select **Enable** from the **Action** list.
   - Ensure that the **Port** is set to the correct SSL port value.
   - Select **SSL** from the **Transport** list.
   - If you want to store the SSL key on an HSM, then select the **Key in HSM** option.
   - Click the **Browse** button adjacent to the **Server Certificate Chain** field to select the RiskMinder Server root certificate.
   - (*Only* if you did not select the **Key in HSM** option) Click the **Browse** button adjacent to the **Server Private Key** field to select the RiskMinder Server private key.
8. Click **Save**.
9. Restart RiskMinder Server:
   - **On Microsoft Windows:** Click the **Start** button**,** navigate to **Settings**, **Control Panel**, **Administrative Tools**, and **Services**. Double-click **Arcot RiskFort Service** from the listed services.
   - **On UNIX-Based Platforms:** Navigate to *install_location*/arcot/bin/ and specify the ./riskfortserver start command in the console window.

Two-Way SSL

To enable two-way SSL communication between the Risk Evaluation Web service and RiskMinder Server:

1. Log in to Administration Console as the MA.
2. Activate the **Services and Server Configurations** tab in the main menu.
3. Ensure that the **RiskFort** tab in the submenu is active.
4. Under **System Configuration**, click the **Trusted Certificate Authorities** link to display the Riskfort Server Trusted Certificate Authorities page.
5. Set the following information on the page:
   - In the **Name** field, enter the name for the SSL truststore.
   - Click the **Browse** button adjacent to the first **Root CAs** field and navigate to and select the root certificate of the application server where your Web services client is deployed.
6. Click **Save**.
7. Under **Instance Configuration**, click the **Protocol Configuration** link to display the Protocol Configuration page.
8. Select the **Server Instance** for which you want to configure the SSL communication.
9. In the **List of Protocols** section, click the **Transaction Web Service** link.

   The page to configure the Transaction Web Service protocol appears.
10. Configure the following fields:
    - Ensure that the **Protocol Status** is **Enabled**.

      If not, then select the **Change Protocol Status** option and select **Enable** from the **Action** list.
    - Ensure that the **Port** is set to the correct SSL port value.
    - Select **SSL** from the **Transport** list.
    - If you want to store the SSL key on an HSM, then select the **Key in HSM** option.
    - Click the **Browse** button adjacent to the **Server Certificate Chain** field to select the RiskMinder Server root certificate.
    - (*Only* if you did not select the **Key in HSM** option) Click the **Browse** button adjacent to the **Server Private Key** field to select the RiskMinder Server private key.
    - Select the **Client Store** that you created in Step 5.
11. Click **Save**.
12. Restart RiskMinder Server:

- **On Microsoft Windows:** Click the **Start** button**,** navigate to **Settings**, **Control Panel**, **Administrative Tools**, and **Services**. Double-click **Arcot RiskFort Service** from the listed services.
- **On UNIX-Based Platforms:** Navigate to *install_location*/arcot/bin/ and specify the ./riskfortserver start command in the console window.

13. Verify that RiskMinder Server is enabled for SSL communication by performing the following steps:

    a. Open the arcotriskfortstartup.log file in a text editor.

    b. Check for the following line:
    ```
    Started listener for [RiskFort Trans WS] [7778] [SSL]
    [transwsprotocol]
    ```
    If you located this line, then two-way SSL was set up successfully.

    c. Close the file.

Setting Up SSL Communication Between Administration Web Service and RiskFort Server

To enable Administration Web service for SSL communication, you must first configure your client that accesses the Web services for SSL communication, and then configure the Administration Web service protocol by using Administration Console. This section describes how to set up the following between the Administration Web service and RiskMinder Server:

- One-Way SS (see page 202)
- Two-Way SSL (see page 203)

One-Way SSL

To set up one-way SSL between the Administration Web service and RiskMinder Server:

1. Ensure that you are logged in as the MA.
2. Activate the **Services and Server Configurations** tab.
3. Ensure that the **RiskFort** tab in the submenu is active.
4. Under the **Instance Configuration** section, click the **Protocol Configuration** link to display the Protocol Configuration page.
5. Select the **Server Instance** for which you want to configure the SSL communication.
6. In the **List of Protocols** section, click the **Administration Web Service** link.

   The page to configure the Administration Web service protocol appears.
7. Configure the following fields:
   - Ensure that the **Protocol Status** is **Enabled**.

     If not, then select the **Change Protocol Status** option and select **Enable** from the **Action** list.
   - Ensure that the **Port** is set to the correct SSL port value.
   - Select **SSL** from the **Transport** list.
   - If you want to store the SSL key on an HSM, then select the **Key in HSM** option.
   - Click the **Browse** button adjacent to the **Server Certificate Chain** field to select the RiskMinder Server root certificate.
   - (*Only* if you did not select the **Key in HSM** option) Click the **Browse** button adjacent to the **Server Private Key** field to select the RiskMinder Server private key.
8. Click **Save**.
9. Restart RiskMinder Server:
   - **On Microsoft Windows:** Click the **Start** button**,** navigate to **Settings**, **Control Panel**, **Administrative Tools**, and **Services**. Double-click **Arcot RiskFort Service** from the listed services.
   - **On UNIX-Based Platforms:** Navigate to *install_location*/arcot/bin/ and specify the ./riskfortserver start command in the console window.

Two-Way SSL

To enable two-way SSL communication mode between the Administration Web service and RiskMinder Server:

1. Log in to Administration Console as the MA.
2. Activate the **Services and Server Configurations** tab in the main menu.
3. Ensure that the **RiskFort** tab in the submenu is active.
4. Under **System Configuration**, click the **Trusted Certificate Authorities** link to display the Riskfort Server Trusted Certificate Authorities page.
5. Set the following information on the page:
   - In the **Name** field, enter the name for the SSL truststore.
   - Click the **Browse** button adjacent to the first **Root CAs** field and navigate to and select the root certificate of the application server where your Web services client is deployed.
6. Click **Save**.
7. Under **Instance Configuration**, click the **Protocol Configuration** link to display the Protocol Configuration page.
8. Select the **Server Instance** for which you want to configure the SSL communication.
9. In the **List of Protocols** section, click the **Administration Web Service** link.

   The page to configure the Administration Web service protocol appears.
10. Configure the following fields:
    - Ensure that the **Protocol Status** is **Enabled**.

      If not, then select the **Change Protocol Status** option and select **Enable** from the **Action** list.
    - Ensure that the **Port** is set to the correct SSL port value.
    - Select **SSL** from the **Transport** list.
    - If you want to store the SSL key on an HSM, then select the **Key in HSM** option.
    - Click the **Browse** button adjacent to the **Server Certificate Chain** field to select the RiskMinder Server root certificate.
    - (*Only* if you did not select the **Key in HSM** option) Click the **Browse** button adjacent to the **Server Private Key** field to select the RiskMinder Server private key.
    - Select the **Client Store** that you created in Step 5.
11. Click the **Save** button.
12. Restart RiskMinder Server:

- **On Microsoft Windows:** Click the **Start** button**,** navigate to **Settings**, **Control Panel**, **Administrative Tools**, and **Services**. Double-click **Arcot RiskFort Service** from the listed services.
- **On UNIX-Based Platforms:** Navigate to *install_location*/arcot/bin/ and specify the ./riskfortserver start command in the console window.

13. Verify that RiskMinder Server is enabled for SSL communication by performing the following steps:

    a. Open the arcotriskfortstartup.log file in a text editor.

    b. Check for the following line:
    ```
    Started listener for [RiskFort Admin WS] [7777] [SSL]
    [aradminwsprotocol]
    ```
    If you located this line, then two-way SSL was set up successfully.

    c. Close the file.

Appendix B: Web Services Reference

RiskMinder provides a set of Web services that your online application can use to programmatically integrate with RiskMinder, irrespective of the coding technology you use. RiskMinder consists of the following Web services components:

- The Risk Evaluation Web services
- The Administration Console Web services for Risk Evaluation
- The UDS Web services
- WSDLDoc information for the associated Web services

Accessing the WSDL Documentation

You can use the WSDLDoc information provided with the RiskMinder Web services along with this guide and other Web services reference materials, to add RiskMinder Risk Evaluation services to new or existing applications.

If you are updating an existing RiskMinder application, then you must consult the Release Notes and WSDLDoc documentation for deprecated APIs before making any changes.

You can access the latest WSDLDocs by installing RiskMinder and copying the WSDLDocs from the docs directory. (You can then copy the WSDLDocs to another location on your development system.) Alternatively, you can also access the WSDLDocs directly from the Documentation directory in the RiskMinder installation package, without having to install RiskMinder.

See the tables in the next two sections for the installation locations of the RiskMinder and associated WSDLDocs.

Risk Evaluation Web Services

The following table lists the files that are installed as a part of the Risk Evaluation Web services component. The base location for these files is:

- **Microsoft Windows**
  `*install_location*\Arcot Systems\`
- **UNIX-Based Platforms**
  `*install_location*/arcot/`

Location	File Name	Description
docs\riskfort\\ (Microsoft Windows) docs/**riskfort**/ (UNIX Platforms)	Arcot-RiskFort-3.1-AdminWebService-wsdl docs.zip or Arcot-RiskFort-3.1-AdminWebService-wsdl docs.tar.gz	WSDL documentation for the Administration Web service.
	Arcot-RiskFort-3.1-risk-evaluation-wsdl docs.zip or Arcot-RiskFort-3.1-risk-evaluation-wsdl docs.tar.gz	WSDL documentation for the Risk Evaluation Web service.
wsdls\admin\\ (Microsoft Windows) wsdls/**admin**/ (UNIX Platforms)	ArcotRiskFortAdminWebService.wsdl	This WSDL describes the Administration Web services and how to access them.

Location	File Name	Description
wsdls\riskfort\ (Microsoft Windows) wsdls/**riskfort**/ (UNIX Platforms)	ArcotRiskFortEvaluateRiskService.wsdl	This WSDL describes the Risk Evaluation Web services and how to access them.
sdk\devicedna\ (Microsoft Windows) sdk/**devicedna**/ (UNIX Platforms)	riskminder-client.js	This file contains the functions to gather the Device ID- and DeviceDNA-related information from the end user's device and to generate the single-encoded String with all the DeviceDNA values.

User Data Service (UDS) Web Services

The following table lists the files that are installed as a part of the UDS Web services component. The base location for these files is:

- **Microsoft Windows**
  `install_location\Arcot Systems\`
- **UNIX-Based Platforms**
  `install_location/arcot/`

Location	File Name	Description
docs\uds\ (Microsoft Windows) docs/**uds**/ (UNIX Platforms)	arcot-uds-2_0-wsdl-docs.zip or arcot-uds-2_0-wsdl-docs.tar.gz	WSDL documentation for the User Data Service (UDS) Web service. This WSDL describes the UDS Web services and how to access them.

Location	**File Name**	**Description**
wsdls\uds\\ (Microsoft Windows) wsdls/**uds**/ (UNIX Platforms)	ArcotConfigManagementSvc.wsdl	This WSDL describes the UDS Configuration management Web services and how to access them. These Web services are used to create and manage user account types.
	ArcotOrganizationManagementSvc.wsdl	This WSDL describes the UDS Organization management Web services and how to access them. These Web services are used to create and manage organizations in the system.
	ArcotUserManagementSvc.wsdl	This WSDL describes the UDS User management Web services and how to access them. These Web services are used to create and manage users in the system.
	ArcotUserSchema.xsd	This is the XML Schema Definition that serves as the reference library that can be uses by your code for working with the preceding three UDS Web services.

Appendix C: Exceptions and Error Codes

This appendix lists the following error codes thrown by the RiskMinder Web services:

- User Data Service (UDS) Error Codes (see page 209)
- RiskMinder Response Codes (see page 229)

User Data Service (UDS) Error Codes

The following table lists the **AdminFault** error codes and messages that are returned by the Web services used to manage organizations, users, and account types.

Error Code	Error Message	Possible Cause for Failure
31201	Unable to process the database query, {0}. **Note:** This is a critical error.	**Possible Causes:** ■ Invalid input parameter was specified. ■ Error occurred during encryption. ■ Database is down. **Solution:** 1. Verify that the database information in arcotcommon.ini is correct. 2. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 3. If the issue is not resolved, then you must contact CA Support.
31002	UDS is not initialized. **Note:** This is a critical error, and is typically seen when UDS or Administration Console are restarted.	**Possible Causes:** ■ ARCOT_HOME is not correctly set. ■ Database is down. ■ Hardware encryption initialization failed. **Solution:** 1. Verify that the database information in arcotcommon.ini is correct. 2. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 3. If the issue is not resolved, then you must contact CA Support.

Error Code	Error Message	Possible Cause for Failure
31003	Fatal error, restart UDS. This error is expected when the UDS application has encountered an unexpected error.	**Possible Causes:** ■ UDS did not start up correctly. ■ Database is down. ■ Hardware encryption initialization failed. **Solution:** 1. Verify that the database information in arcotcommon.ini is correct. 2. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 3. Restart UDS.
31006	Configuration parameter, {0} not found. This error occurs if the specified UDS configuration was not found in the ARUDSCONFIG table.	**Possible Causes:** ■ Database was manually updated. ■ Information in database tables was not correctly populated. **Solution:** 1. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.
31007	Invalid configuration parameter value, {0} for parameter name, {1}. This error occurs if the specified UDS configuration contains invalid value(s).	**Possible Causes:** ■ Database was manually updated. ■ Information in database tables was not correctly populated. **Solution:** 1. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.
31008	Unknown error. This message appears if an unexpected internal error occurred.	**Possible Cause:** Unexpected internal error. **Solution:** 1. See the arcotadmin.log and arcotuds.log files and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.

Error Code	Error Message	Possible Cause for Failure
31009	General error: {0}. This message appears if an unexpected internal error occurred.	**Possible Cause:** Unexpected internal error. **Solution:** 1. See the arcotadmin.log and arcotuds.log files and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.
35100	Error communicating with data store. **Note:** This is a critical error that occurs either when the connectivity with the database server is lost or when processing a database query.	**Possible Causes:** ■ The database is down. ■ There was an error during encryption or decryption of data. **Solution:** 1. See the arcotadmin.log and arcotuds.log files and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.
35101	Error while loading configuration file, {0}. **Note:** This is a critical error and occurs while reading configuration files in the conf directory of ARCOT_HOME.	**Possible Cause:** The configuration files are corrupted. **Solution:** 1. Ensure that the configuration files contain the required details. 2. Retry reading from the files.
35102	Configuration file, {0} not found.	**Possible Causes:** 1. ARCOT_HOME is not set. 2. The required configuration files are not found in the conf directory of ARCOT_HOME. **Solution:** 1. Verify if ARCOT_HOME points to the right location. 2. Verify if the required configuration files exist in the conf directory of ARCOT_HOME.
35103	ARCOT_HOME environment variable is not set.	**Possible Cause:** ARCOT_HOME is not set. **Solution:** Set the ARCOT_HOME to point to your installation directory.

Error Code	Error Message	Possible Cause for Failure
35105	Invalid input parameter.	**Possible Cause:** The input value provided for the specified parameter is not valid. **Solution:** Refer to the section, "User Data Service Validations" (see page 261) in appendix for valid input set.
35106	Missing input parameter, {0}.	**Possible Cause:** The specified input parameter is missing in the API request. **Solution:** Provide the required parameter.
35107	Cannot update global chosen encryption set, because the organization already contains users.	**Possible Cause:** One or more organizations refer to the global encryption set, but users have already been created in these organizations. **Solution:** Global encryption set cannot be updated when there are users in the referring organizations.
35108	Error while audit logging.	**Possible Causes:** 1. Connection to the database is lost. 2. Invalid input provided for the audit logs. **Solution:** See the arcotadmin.log and arcotuds.log files and take corrective action.
35109	Field, {0} exceeded maximum length, {1}.	**Possible Cause:** The specified field exceeded the allowed length. **Solution:** Provide a value within the expected range. Refer to the section, "User Data Service Validations" (see page 261) in appendix for the allowed length of input fields.

Error Code	Error Message	Possible Cause for Failure
35110	Field, {0} contains invalid characters.	**Possible Causes:** The value provided for the specified field contains unsupported characters. **Solution:** Retry with valid inputs. Refer to the section, "User Data Service Validations" (see page 261) in appendix for the allowed character set for different fields.
31125	User, {0} not found.	**Possible Causes:** 1. The user identifier specified in the request is not valid. 2. The user does not exist in the system. 3. The user has been deleted. **Solution:** 1. Provide valid user details. 2. Search for deleted users to verify whether the user has been deleted.
31126	User, {0} not unique. More than one user found.	**Possible Cause:** The specified user is not unique in the system. As a result, more than one user is returned for the given UserID. **Solution:** 1. Ensure that the UserID is unique. 2. Check whether the UserID mapping attribute exists in the LDAP organization. 3. If the issue is not resolved, then you must contact CA Support.
31118	Search field, {0} not permitted.	**Possible Cause:** User search is not permitted on the specified field. For example, searching for an unmapped LDAP attribute is not allowed. **Solution:** 1. Search based on other fields. 2. Ensure that the required attributes are correctly mapped.

Error Code	Error Message	Possible Cause for Failure
31127	Operation, {0} not supported. Invalid current state {1} of User, {2}.	**Possible Causes:** ■ The current operation is not supported for the given user status. ■ The status of the specified user is either INITIAL or INACTIVE. For example, authentication operations are not supported for INACTIVE users. **Solution:** Update the user status to a valid status and then perform the operation. Refer to the section, "User Data Service Validations" (see page 261) in appendix for valid user status for operations.
31104	Operation, {0} not supported for repository.	**Possible Cause:** The current operation is not supported for the repository. For example, Write operations are not supported for LDAP repository. **Solution:** Unsupported operations must be independently performed on the repository and must not go through RiskMinder flows. For example, users must be created in LDAP through LDAP user interface or APIs.
31128	User, {0} already exists. **Note:** The createUser API throws this error.	**Possible Cause:** The specified user already exists in the system. You cannot create another user with the same UserID. **Solution:** UserID must be unique in an organization. Create an user with a different UserID.
31119	User identifier, {0} is mandatory.	**Possible Cause:** API was called without providing the user identifier, which is mandatory for the given API call. **Solution:** Provide a valid user identifier in the API request.

Error Code	Error Message	Possible Cause for Failure
31129	PAM is not set. This is a C++ error code.	**Possible Causes:** 1. Specified user was not found. 2. Connection to the database is lost. **Solution:** 1. See the arcotadmin.log and arcotuds.log files and take corrective action. 2.If the issue is not resolved, then you must contact CA Support.
31131	Invalid authentication token. **Note:** This error is observed when the Web service is enabled for authentication and authorization.	**Possible Causes:** 1. The authentication token is not provided in the request. 2. The specified authentication token is not valid or has been tampered with. **Solution:** Provide a valid authentication token, if the API is enabled for authentication and authorization.
31132	Invalid authentication request. **Note:** This error is observed when the Web service is enabled for authentication and authorization.	**Possible Causes:** 1. The authentication token provided in the request has expired. 2. The SOAP request is invalid. **Solution:** 1. Provide a valid authentication token in the request. 2. Obtain a new authentication token, if required.

Error Code	Error Message	Possible Cause for Failure
31135	Hashing of authentication token failed. **Note:** This error is observed when the Web service is enabled for authentication and authorization.	**Possible Causes:** 1. The authentication token provided in the request is invalid or has been tampered with. 2. The token could not be processed. 3. The securestore.enc file is not found in the conf directory of ARCOT_HOME. **Solution:** 1. Provide a valid authentication token. 2. Re-create securestore.enc. **Book:** See *CA RiskMinder Administration Guide* for more information on how to create this file. 3. If the issue is not resolved, then you must contact CA Support.
70611	Authentication failed.	**Possible Causes:** 1. Incorrect username or password has been specified in the request. 2. The administrator status is not valid. 3. The account is locked. 4. The account has expired. **Solution:** Retry with valid password or contact the administrator to unlock or activate the account.
70300	Administrator {0} (organization: {1}) does not have the privilege to perform administration operations for organization, {2}. **Note:** This error is observed when the Web service is enabled for authentication and authorization.	**Possible Cause:** The administrator performing the operation has a limited scope, and does not have the required permissions to perform the current task. **Solution:** 1. Contact an administrator at a higher level for the required permissions.

Error Code	Error Message	Possible Cause for Failure
31136	Delete operation for product {0} failed. **Note:** This error is seen when the Cascade Delete feature is enabled.	**Possible Cause:** A database error occurred during a delete operation. **Solution:** 1. Retry the operation. 2.If the error persists, then you must contact CA Support.
31137	Invalid search expression. **Note:** This error is thrown by the searchUsers API.	**Possible Cause:** The search expression that you provided is not valid. **Solution:** Provide valid search expressions. Refer to the WSDL documentation to know more about valid search expressions.
31138	Invalid start ({0}) or end ({1}) index specified. **Note:** This error is thrown by the listUsers API.	**Possible Causes:** 1. The start or end index that you provided is not valid (probably a a negative integer). 2. End index is less than the start index. **Solution:** Provide valid positive integers for the start and end indexes.
31139	Page size, {0} exceeded the configured default search count, {1}. **Note:** This error is thrown by the listUsers API.	**Possible Cause:** Number of users that you are trying to retrieve exceeds the configured search count. **Solution:** Limit the number of users within the search count. Or, increase the search count.
31151	Start lock time and End lock time are not allowed for ACTIVE user status.	**Possible Cause:** You have provided Start lock time and End lock time as inputs along with the user status, ACTIVE. **Note:** User status cannot be updated to ACTIVE if Start lock time and End lock time are also specified in the request. **Solution:** Do not provide Start lock and End lock dates for ACTIVE user status. These inputs are valid only if the user status is INACTIVE.

Error Code	Error Message	Possible Cause for Failure
31152	Invalid lock period. Start lock time must be before End lock time.	**Possible Cause:** Start lock time is greater than the End lock time. **Solution:** Ensure that the Start lock time is always less than the End lock time.
31153	Invalid lock Period. Start lock time cannot be before current time.	**Possible Cause:** Start lock time is less than the current time. **Solution:** Ensure that the Start lock time is always greater than the current time.
36100	Invalid input parameter: name, {0} value, {1}.	**Possible Cause:** The input value provided for the given parameter name is not valid. For example, the specified email contains multi-byte characters. **Solution:** Provide valid input values. Refer to the section, "User Data Service Validations" (see page 261)in appendix for the set of allowed inputs for fields.
36101	Unsupported encoding exception: {0}.	**Possible Cause:** Error while encoding or decoding user or organization custom attributes. Invalid custom attributes. **Solution:** Retry with valid inputs.
36102	Parser exception: {0}.	**Possible Cause:** Error occurred while parsing user details from datastore. For example, the error occurred while retrieving information from the LDAP Date field. **Solution:** 1. Retry the operation. 2.If the error persists, then you must contact CA Support

Error Code	Error Message	Possible Cause for Failure
36103	{0}.	**Possible Causes:** Error occurred while encrypting or decrypting user data, because: ■ HSM is not reachable ■ Invalid Key Label is provided in the input. ■ Server Cache has not been updated. **Solution:** 1. Retry after refreshing the cache. 2. Verify if the specified Key Label exists in the HSM. 3. Ensure that you can successfully connect to the HSM.
36106	User {0} not updated.	**Possible Causes:** 1. Specified user not found. 2. Connection to the database is lost. **Solution:** 1. Retry the operation. 2. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 3. If the issue is not resolved, then you must contact CA Support.
36107	Provided image size, {0} KB, exceeds the maximum supported size, {1} KB.	**Possible Cause:** The user image has exceeded the supported size. **Solution:** Ensure that the image size is within the supported range. Refer to the section, "User Data Service Validations" (see page 261) in appendix for more information.
36108	Invalid image format, {0}. Supported image formats are JPEG, GIF, BMP, and PNG only.	**Possible Cause:** The format of the user image provided in the request is not valid. **Solution:** Ensure that the image format you are using is supported by UDS. Refer to the section, "User Data Service Validations" (see page 261) in appendix for the supported image formats.

Error Code	Error Message	Possible Cause for Failure
31134	Invalid input parameter. {0} is not an enterprise LDAP organization.	**Possible Cause:** You passed non-LDAP organization details to the LDAP-only API, such as API=performQnaVerification. **Solution:** Ensure that you use the correct non-LDAP APIs.
31108	Invalid input parameter: name, {0} and value, {1}.	**Possible Cause:** The value provided for the specified parameter is not valid. **Solution:** Provide valid inputs. Refer to the section, "User Data Service Validations" (see page 261) in appendix for the supported input set.
31109	Organization with name {0} already exists.	**Possible Cause:** Organization with the specified name already exists. As a result, another organization with the same name cannot be created. **Solution:** Provide a unique organization name.
31110	Organization with the display name {0} already exists.	**Possible Cause:** Organization with the specified display name already exists. As a result, another organization with the same display name cannot be created. **Solution:** Provide a unique display name for the organization.
31114	Operation, {0} is not supported for organization {1} with status {2}.	**Possible Cause:** The specified operation is not supported for the given status of an organization. For example, setting an INACTIVE organization as default is not allowed. **Solution:** Update the organization status and then perform the operation.

Error Code	Error Message	Possible Cause for Failure
31115	Organization, {0} with status, {1} does not exist.	**Possible Cause:** The specified organization with the given status does not exist. **Solution:** Verify if the organization exists and check if its status has changed.
31116	Organization {0} is already deleted.	**Possible Cause:** The organization that you are trying to delete cannot be deleted, because it has already been deleted. **Solution:** Ensure that you specify the correct organization details.
31117	Unable to connect to the repository, {0}.	**Possible Causes:** ■ Repository is down. ■ Specified repository connection details are not correct. **Solution:** Check the repository connection details and retry.
31121	Invalid organization status, {0}.	**Possible Cause:** The organization status is not valid for the given operation. For example, the organization cannot be created with status INACTIVE. **Solution:** Update the organization status to a valid one.
31122	Operation, {0} not supported for default organization {1}.	**Possible Cause:** The specified operation is not supported on Default organization. For example, you cannot delete the Default organization.

Error Code	Error Message	Possible Cause for Failure
31124	Organization, {0} does not exist.	**Possible Causes:** ■ The organization name provided as input is not valid. ■ The Server cache has not been updated. ■ The organization with the specified name does not exist. **Solution:** Retry the operation after refreshing the cache.
31140	Attribute encryption failed.	**Possible Causes:** ■ The organization display name provided as input is not valid. ■ The Server cache has not been updated. ■ The organization with the specified display name does not exist. **Solution:** Retry the operation after refreshing the cache.
31142	Invalid key label. No key with alias, {0} exists. **Note:** This error is thrown if the system is configured for hardware encryption.	**Possible Causes:** ■ The Key Label specified in the operation does not exist in the HSM. ■ The HSM connection failed. **Solution:** 1. Check the HSM connectivity. 2. Verify if the Key Label is present in the HSM.
31143	Organization {0} exist with the same LDAP configuration. **Note:** The createOrg APIs throw this error.	**Possible Cause:** An LDAP organization with the same configuration already exists. **Solution:** LDAP organizations must have unique configurations. Update the existing organization or create a new organization with different details.

Error Code	Error Message	Possible Cause for Failure
31146	Error saving custom attributes for user {0}. **Note:** This error is thrown while updating users' custom attributes.	**Possible Causes:** ■ The specified user was not found. ■ The database connection failed. **Solution:** 1. Retry the operation. 2. If the issue is not resolved, then you must contact CA Support.
38100	Resource, {0} of type, {1} does not exist. **Note:** This error is thrown when the processing of an account type fails.	**Possible Causes:** ■ The specified account type does not exist. ■ The Server cache has not been updated. **Solution:** Refresh the cache and retry.
38101	Unknown or unexpected error.	**Possible Cause:** A critical internal error occurred. **Solution:** Contact CA Support.
39100	User account, {0} not found for account type, {1}.	**Possible Causes:** The specified user account was not found for the given account type, because: ■ The specified user does not have an account. ■ The account has been deleted. **Solution:** Verify if the input account ID is valid.
39101	The custom attribute, {0} is invalid for account type, {1}.	**Possible Cause:** ■ The specified account custom attribute does not exist for the account type. ■ The Server cache has not been refreshed. **Solution:** 1. Ensure that you provide a valid input. 2. Refresh the cache and retry.

Error Code	Error Message	Possible Cause for Failure
39102	User identifier, {0} not found for organization, {1}.	**Possible Cause:** The user identifier was not found for the organization during deep search. In other words, the specified identifier did *not* match any of the userid, accountid, and accountid attributes. **Solution:** Provide a valid user identifier.
39103	Account(s) creation failed for user identifier, {0} belonging to the organization, {1}. **Note:** This is a C++ error code.	**Possible Causes:** ■ The connection to the database failed. ■ An unexpected error occurred while creating account(s). **Solution:** 1. See if there are any related errors in arcotadmin.log and arcotuds.log and take corrective action. 2. Retry the operation.
39104	The specified user account already exists for user {0}.	**Possible Cause:** The specified user account already exists. **Solution:** Ensure that you specify a unique account name.
39105	Account types do not exist for organization, {0}.	**Possible Causes:** ■ The specified account type is not available for the organization. ■ The Server cache has not been refreshed. **Solution:** 1. Ensure that you specify the correct account type name. 2. Add the organization to the account type scope. 3. Retry after refreshing the cache.

Error Code	Error Message	Possible Cause for Failure
39106	Account type already exists. **Note:** The APIs for creating and updating account types throw this error.	**Possible Cause:** The specified account type exists in the system. As a result, another account type with the same name or display name cannot be created. **Solution:** Create an account type with a unique name or display name.
39107	Account ID, {0} already created for the account type, {1}.	**Possible Cause:** The specified user account has already been created for the given account type. **Solution:** The account ID must be unique for a given account type for an organization. Provide a different account ID and retry.
39201	SOAP action is null.	**Possible Cause:** The incoming SOAP request is not valid. **Solution:** 1. Provide a valid SOAP request. 2. If problem persists, then contact CA Support.
55000	Invalid input parameter, {0}.	**Possible Cause:** The input value provided for the given parameter is not valid. **Solution:** Refer to the section, "User Data Service Validations" (see page 261) in appendix for valid input set.
55001	Missing input parameter, {0}.	**Possible Cause:** The required parameter is missing from the API request. **Solution:** Provide the required parameter.
55002	Insufficient input parameters. **Note:** This is a C++ error code.	**Possible Cause:** The API inputs are incomplete. **Solution:** Provide all the required inputs.

Error Code	Error Message	Possible Cause for Failure
55003	Resource bundles not found. **Note:** This is a C++ error code.	**Possible Causes:** ■ ARCOT_HOME is not set. ■ The required properties files are missing from the resourcebundles subdirectory in the conf directory. **Solution:** Ensure that the ARCOT_HOME environment variable is set correctly.
55004	Database error. **Note:** This is a C++ error code.	**Possible Causes:** ■ The connection to the database failed. ■ An unexpected error occurred while writing to or reading from the database. **Solution:** 1. Refer to arcotadmin.log and arcotuds.log for more information. 2. Retry the connection.
55010	Error while encrypting the data. **Note:** This is a C++ error code.	**Possible Causes:** ■ The connection to the HSM failed. ■ An unexpected error occurred. ■ The Server cache has not been refreshed. **Solution:** 1. Refer to arcotadmin.log and arcotuds.log for more information. 2. Check HSM connection details. 3. Refresh the Server cache.
55011	Error while decrypting the data. **Note:** This is a C++ error code.	**Possible Causes:** ■ The connection to the HSM failed. ■ An unexpected error occurred. ■ The Server cache has not been refreshed. **Solution:** 1. Refer to arcotadmin.log and arcotuds.log for more information. 2. Check HSM connection details. 3. Refresh the Server cache.

Error Code	Error Message	Possible Cause for Failure
55012	Internal error occurred. **Note:** This is a C++ error code.	**Possible Cause:** An unexpected error occurred. **Solution:** Refer to arcotadmin.log and arcotuds.log for more information.
55100	Error while retrieving organization configuration data. **Note:** This is a C++ error code.	**Possible Causes:** ■ The specified organization was not found in the system. ■ The connection to the database failed. ■ The Server cache has not been refreshed. **Solution:** 1. Refer to arcotadmin.log and arcotuds.log for more information. 2. Check the database connection details. 3. Refresh the Server cache.
50030	Search size limit exceeded the maximum value. **Note:** This is a C++ error code.	**Possible Cause:** Search returned more than the maximum configured limit. **Solution:** Refine your search criteria.
50031	Search base node context needs to be bound.	**Possible Causes:** ■ The connection to the LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.

Error Code	Error Message	Possible Cause for Failure
50032	Search is not expected to return more than set limit values. **Note:** This error is thrown by the retrieveUser API in context of LDAP organizations.	**Possible Causes:** 1. The user you specified is not unique in the system (LDAP). As a result, more than one user details were returned for the given UserID. 2. The UserID attribute is not mapped to the correct LDAP attribute. **Solution:** Ensure that the UserID is unique. Also, verify the UserID attribute mappings.
50033	Search criteria is not valid. **Note:** This is a C++ error code.	**Possible Cause:** The search input that you provided is not valid. **Solution:** Provide valid search inputs.
50034	Unable to get supporting data access class.	**Possible Cause:** ■ The connection to the LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.
50035	LDAP node has to be created before referencing.	**Possible Cause:** ■ The connection to the LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.

Error Code	Error Message	Possible Cause for Failure
50036	LDAP repository has to be initialized.	**Possible Cause:** ■ The connection to the LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.
50037	Context is not bound or cannot be created.	**Possible Cause:** ■ The connection to the LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.

RiskMinder Response Codes

The following table lists the **riskfortFault** response codes, reason codes, the possible cause for the failure, and solution wherever applicable.

Response Code	Reason Code	Description	Possible Cause for Failure
0	0	The operation was successful.	NA.
1000	2002	There was an internal error.	**Possible Cause:** Unexpected internal error.

Response Code	Reason Code	Description	Possible Cause for Failure
1050	0	Value of one of the parameters used in the operation is invalid.	**Possible Cause:** The value of the parameter passed to the API is invalid. For example, the allowed values for user status are 0 and 1. If you set this value as 5, then you will get this error. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
1050	2050	Value of one of the parameters used in the operation is empty.	**Possible Cause:** The parameter passed to the API is empty. **Solution:** Provide a non-empty value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
1050	2051	The length of one of the parameters used in the operation has exceeded the maximum allowed value. **Note:** Length here refers to length of the parameter, for example password length.	**Possible Cause:** The length of the parameter passed to the API has exceeded the maximum value. **Solution:** Provide the parameter such that its length is less than or equal to the maximum allowed value. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
	2052	The length of one of the parameters used in the operation is less than minimum allowed value.	**Possible Cause:** The length of the parameter passed to the API is less than the minimum value. **Solution:** Provide the parameter such that the length of the parameter is greater than or equal to the minimum allowed value. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
	2053	Value of one of the parameters used in the operation exceeded the maximum allowed value. **Note:** Value here refers to the value of the parameter.	**Possible Cause:** The value of the parameter passed to the API has exceeded the maximum allowed value. **Solution:** Provide the parameter such that the value of the parameter is less than or equal to the maximum allowed value. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
1050	2054	Value of one of the parameters used in the operation is less than the minimum allowed value.	**Possible Cause:** The value of the parameter passed to the API is less than the minimum allowed value. **Solution:** Provide the parameter such that the value of the parameter is greater than or equal to the minimum allowed value. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
	2055	Value of one of the parameters used in the operation is invalid.	**Possible Cause:** The value of the parameter passed to the API is invalid. For example, the allowed values for user status are 0 and 1. If you set this value as 5, then you will get this error. **Solution:** Provide valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
	2056	Value of one of the parameters used in the operation contains invalid characters.	**Possible Cause:** The parameter specified by ParameterKey contains invalid characters. **Solution:** Provide valid characters for the parameter that is specified by ParameterKey.
	2057	One of the parameters used in the operation does not meet the formatting requirements.	**Possible Cause:** The parameter specified by ParameterKey has an invalid format. **Solution:** Provide a valid format for the parameter that is specified by ParameterKey.
1050	2061	Value of one of the parameters used in the operation is not allowed.	**Possible Cause:** The parameter specified by ParameterKey has an invalid format. **Solution:** Provide a valid format for the parameter that is specified by ParameterKey.
	8104	The specified Callout URL is not valid.	**Possible Cause:** The specified URL is incorrect. **Solution:** Provide the valid URL.

Response Code	Reason Code	Description	Possible Cause for Failure
	8105	The specified duration is not valid.	**Possible Cause:** The Start Date is greater than the End Date. The specified value for Start Date and/or the End Date is in the past. **Solution:** The Start Date must be greater than the End Date and these should be the current or future dates (as this is the duration for the exception user).
7000	0	The operation was successful.	NA.
	8000		
7001	0	There was an internal error.	**Possible Cause:** Unexpected internal error.
	8000	There was an internal error.	**Possible Cause:** Unexpected internal error.
	8108	There was an internal error.	**Possible Cause:** Unexpected internal error.
	8122	There was an internal error.	**Possible Cause:** Unexpected internal error.
7501	0	The current operation on the database failed.	**Possible Cause:** Database is not running. **Solution:** Start the database. **Possible Cause:** Connection between the Server and database is not complete. **Solution:** Establish the connection between the Server and database again. **Possible Cause:** The operation failed because of an internal error. **Solution:** Check the database logs for details and ensure appropriate action is taken based on these logs.

Response Code	Reason Code	Description	Possible Cause for Failure
7502		An exception occurred because of an unexpected internal error.	**Possible Cause:** Internal error because of unexpected Server behavior. **Solution:** Most likely cause might be Server or database failure. Check the Server transaction and database logs for details and ensure appropriate action is taken based on the Server logs.
7503		The time could not be successfully fetched.	**Possible Cause:** The database settings are not set correctly in arcotcommon.ini. **Solution:** Verify and correct the database-related parameters in the file.
7511	8000	The received Device Signature is not valid.	**Possible Cause:** The Server cannot parse the Device Signature. Either the packet was corrupted or there was an issue while building the signature. Solution: Ensure that the Device Signature is correctly built.
7601	0	Value of one of the parameters used in the operation does not exist.	**Possible Cause:** The value of the parameter passed to the API does not exist. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7602		The name specified for the new ruleset already exists.	**Possible Cause:** The value of the parameter passed to the API already exists. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7603		No Active ruleset of the specified name was found.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7604		No Proposed ruleset of the specified name was found.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7605	0	The name specified for the ruleset (Active or Proposed) does not exist.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7606	8101	The *data* sharing type refers to another ruleset. Therefore, the addition operation is not allowed.	**Possible Cause:** The rule or the ruleset refers to another ruleset. Therefore, the data cannot be added to the other ruleset.
7607	0	The *data* sharing type refers to another ruleset. Therefore, the operation to set the value(s) is not allowed.	**Possible Cause:** The rule or the ruleset refers to another ruleset. Therefore, the data cannot be set to another ruleset.
7608		The *data* sharing type refers to another ruleset. Therefore, the delete operation is not allowed.	**Possible Cause:** The rule or the ruleset refers to another ruleset. Therefore, the data cannot be deleted from the other ruleset.
7609		The *data* sharing type refers to another ruleset. Therefore, the update operation is not allowed.	**Possible Cause:** The rule or the ruleset refers to another ruleset. Therefore, the data cannot be updated.
7610		The *data* sharing type refers to another ruleset. Therefore, the value rotation is not allowed.	**Possible Cause:** The rule or the ruleset refers to another ruleset. Therefore, the data values cannot be rotated.
7611		The *parameter* sharing type refers to another ruleset. Therefore, the operation to set the value(s) is not allowed.	**Possible Cause:** The rule or the ruleset refers to another ruleset. Therefore, the data cannot be set to another ruleset.
7612	0	No Active or Proposed data was found for the specified Negative Country.	**Possible Cause:** The value of the parameter passed to the API was not found in the ARRFNEGATIVECOUNTRYLIST table. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7613		No Active or Proposed data was found for the specified Negative IP address.	**Possible Cause:** The value of the parameter passed to the API was not found in the ARRFUNTRUSTEDIPLIST table. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7614		No Active or Proposed data was found for the specified Trusted IP address.	**Possible Cause:** The value of the parameter passed to the API was not found in the ARRFTRUSTEDIPLIST table. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7615	0	No Active or Proposed parameters were found for the specified rule.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values. **Possible Cause:** The parameter specified by ParameterKey contains invalid characters. **Solution:** Provide a valid characters for the parameter that is specified by ParameterKey.

Response Code	Reason Code	Description	Possible Cause for Failure
7616		No Proposed data was found for the specified Trusted IP address.	**Possible Cause:** The value of the parameter passed to the API was not found in the ARRFTRUSTEDIPLIST table. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7617		No Proposed data was found for the specified Negative IP address.	**Possible Cause:** The value of the parameter passed to the API was not found in the ARRFUNTRUSTEDIPLIST table. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7618		No Active or Proposed data was found for the specified Scoring configuration.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7619	0	No Active or Proposed data was found for the specified Execution configuration.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7620		The value of the configuration state parameter used in the operation is invalid.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7621		The ruleset cannot be created because the specified parameters and their values for the rule do not match.	**Possible Cause:** The parameters and their corresponding values passed to the API are invalid. **Solution:** Provide a valid value for the parameter. See appendix,"Input (see page 261)Data Validations" (see page 261) for the supported parameter values.
7622		No Active ruleset parameters found for the specified otherOrgName and otherConfigName.	**Possible Cause:** The value of the otherOrgName and otherConfigName parameters passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix,"Input Data Validations" (see page 261) for the supported parameter values.
7623		No Active ruleset found for the specified data for otherOrgName and otherConfigName parameters.	**Possible Cause:** The value of the otherOrgName and otherConfigName parameters passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7624	0	The update is not allowed because it will create a cyclic dependency.	**Possible Cause:** The rule or the ruleset refers to another ruleset. **Solution:** Change the ruleset(s) that have not yet been migrated, so that the cyclic dependency is eliminated.
7625		No Active ruleset found for the specified otherOrgName and otherConfigName parameters. Therefore, the update operation is not allowed.	**Possible Cause:** The value of the otherOrgName and otherConfigName parameters passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7626		No Active ruleset found for the specified data for otherOrgName and otherConfigName parameters. Therefore, the update operation is not allowed.	**Possible Cause:** The value of the otherOrgName and otherConfigName parameters passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7627	8102	The specified country cannot be deleted because no corresponding Proposed data was found.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data (see page 261)Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7628		The specified IP range cannot be deleted because no corresponding Proposed data was found.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7629	0	The specified IP range cannot be deleted from the Trusted Aggregator list because no corresponding Proposed data was found.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7630		The specified IP or range cannot be deleted for the Trusted Aggregator because no corresponding Proposed data was found.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7631		The specified IP range does not exist in the Negative IP Address list.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7632		No Active or Proposed data for Trusted Aggregator was found for the specified Trusted IP address.	**Possible Cause:** The value of the parameter passed to the API was not found in the ARRFTRUSTEDIPLIST table. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7633		No Proposed data for the specified Trusted Aggregator was found.	**Possible Cause:** The value of the parameter passed to the API was not found in the ARRFTRUSTEDIPLIST table. **Solution:** Provide a valid value for the parameter. See appendix,"Input Data Validations" (see page 261) for the supported parameter values.
7634	0	The specified Trusted Aggregator does not exist in the Trusted Aggregators list.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7635		The specified Trusted Aggregator cannot be added, because it already exists in the Trusted Aggregators list.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7636		The specified Aggregator ID could not be generated because of an internal Server error.	**Possible Cause:** Aggregator ID generation failed due to an internal error in the Server. **Solution:** Most likely cause might be because of database failure. Check the Server transaction logs for details and ensure appropriate action is taken based on the Server logs.
7637		The specified encryption key was not found in the database.	**Possible Cause:** The encryption key does not exist. **Solution:** Ensure that the key that you are using is correct.
7638	0	The supported port types could not be tokenized.	**Possible Cause:** Server host, port, or both might not be configured correctly. **Solution:** Provide the correct host and port number. **Possible Cause:** Server might not be running. **Solution:** Start the Server. **Possible Cause**: If SSL is configured, then certificates might not be configured correctly. **Solution:** Configure the TLS certificates correctly.

Response Code	Reason Code	Description	Possible Cause for Failure
7639		The specified SSL Trust Store does not exist.	**Possible Cause:** The value of the parameter passed to the API was not found. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values. **Possible Cause:** The specified Trust Store name is not valid. **Solution:** You must provide a valid Trust Store name. **Possible Cause:** The specified organization name is not valid. **Solution:** You must provide a valid organization name.
7640	0	The specified score ranges were not found.	**Possible Cause:** No Score ranges or Advice were found in the ARRFADVICECONFIG and ARRFADVICECODE tables. The RiskMinder database scripts were not run properly. **Solution:** Contact CA Support to resolve this issue.
7641		The Scoring Callout does not have the highest execution priority.	**Possible Cause:** The priority set in the Web service call is higher than the existing highest execution priority in the RiskMinder database. **Solution:** Set the priority in the Web service call correctly.

Response Code	Reason Code	Description	Possible Cause for Failure
7642		The Score does not have the highest execution priority.	**Possible Cause:** The priority set in the Web service call is higher than the existing highest execution priority in the RiskMinder database. **Solution:** Set the priority in the Web service call correctly.
7643	8103	The required custom rule details were not specified. Therefore, the rule was not added.	**Possible Cause:** The custom rule details were not specified correctly in the Web service call. **Solution:** Set the Web service call input parameters correctly.
7644	0	The specified rule type does not exist or is not valid.	**Possible Cause:** The specified rule type is either not present in the ARRFADDONRULETYPE table or is an invalid rule type. **Solution:** You must provide a valid rule type.
7645	0	The custom rule was not added because it will create a cyclic dependency of *data*.	**Possible Cause:** The creation of the custom rule referring to another rule creates a cyclic dependency for the data that you added. Therefore, this is not allowed. **Solution:** Change the required rule such that the cyclic dependency is eliminated.
7646		The custom rule was not added because it will create a cyclic dependency of *parameters*.	**Possible Cause:** The creation of the custom rule referring to another rule creates a cyclic dependency for the parameters that you added. Therefore, this is not allowed. **Solution:** Change the required rule such that the cyclic dependency is eliminated.

Response Code	**Reason Code**	**Description**	**Possible Cause for Failure**
7647		The rule that you specified is not available in the ruleset.	**Possible Cause:** The value of the rule passed to the API was not found in the specified ruleset. **Solution:** Provide a valid rule for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7648		Error occurred while fetching the parameters.	**Possible Cause:** The parameter specified by ParameterKey contains invalid characters. **Solution:** Provide valid characters for the parameter that is specified by ParameterKey.
7649		Other rules are dependant on this rule. Therefore, this rule cannot be deleted.	**Possible Cause:** Rules in another ruleset are currently referring to this rule. Therefore, you cannot delete this rule. **Solution:** If required, delete the depending rulesets before you delete this rule.
7650	0	The TypeName input is not specified in the call.	**Possible Cause:** The TypeName has not been specified in the Web service call. **Solution:** Specify the correct input value for TypeName in the input.
7651		Specified TypeName already exists.	**Possible Cause:** The TypeName in the input is already present in the system. **Solution:** Specify a different TypeName in the input.

Response Code	Reason Code	Description	Possible Cause for Failure
7652		A cyclic redundancy is created with this update. Therefore, the update is not allowed. Also, because of this action only some rulesets were migrated.	**Possible Cause:** A cyclic dependency will be created after the migration of *all* the required rulesets. As a result, only a few rulesets have been migrated. **Solution:** Change the ruleset(s) that have not yet been migrated, so that the cyclic dependency is eliminated.
7653		A cyclic redundancy is created with this update. Hence this update is not allowed. No rulesets have been migrated.	**Possible Cause:** A cyclic dependency will be created after the migration of *all* the required rulesets. **Solution:** Change the required ruleset(s) such that the cyclic dependency is eliminated.
7654	0	The database operation failed while migrating to production. Some rulesets were migrated	**Possible Cause:** A cyclic dependency was found during migration of a ruleset. Therefore, only a few rulesets have been migrated. Some ruleset(s), including the failed ruleset, have not been migrated. ■ **Solution:** Check the RiskMinder logs for details and ensure that appropriate action is taken based on these logs. ■ Remove the cyclic dependency of the failed ruleset and migrate it. ■ Migrate the remaining rulesets that were not migrated because of the failed ruleset.
7655		The value could not be converted to uppercase.	**Possible Cause:** The input given cannot be converted to uppercase. **Solution:** Check the input.

Response Code	Reason Code	Description	Possible Cause for Failure
7656		The User Profile information could not be retrieved.	**Possible Cause:** UDS is not up and running. **Solution:** Check if UDS is deployed properly and is up and running. **Book:** See the *CA RiskMinder Installation and Deployment Guide* for more information to do so. **Possible Cause:** The user does not exist in the RiskMinder system. **Solution:** Add the user to the RiskMinder system.
7656	8116	The User Profile information could not be retrieved because the user status is currently disabled.	**Possible Cause:** The user account is disabled. **Solution:** Activate the user.
7656	8117	The User Profile information could not be retrieved because the user account has been deleted.	**Possible Cause:** The user does not exist in the RiskMinder system. **Solution:** Add the user to the RiskMinder system.
7656	8118	The User Profile information could not be retrieved because the user does not exist.	**Possible Cause:** The user does not exist in the RiskMinder system. **Solution:** Add the user to the RiskMinder system.

Response Code	Reason Code	Description	Possible Cause for Failure
7657	0	The Location and Connection information was not found in the database.	**Possible Cause:** The Quova data has not been uploaded in the RiskMinder database. **Solution:** Upload the Quova data into the RiskMinder database by using the arrfupload.exe tool. **Book:** See the *CA RiskMinder Administration Guide* for more information on this.
7658	0	The Exception User does not exist in the system.	**Possible Cause:** The user does not exist in the RiskMinder system. **Solution:** Add the user to the RiskMinder system. **Possible Cause:** The specified user is not an Exception User. **Solution:** Ensure that you have provided the correct details for the user, or add the user to the Exception User List before proceeding.
7659		The custom rule will create cyclic dependency. Therefore, the rule cannot be added.	**Possible Cause:** A cyclic dependency will be created when you create the custom rule. **Solution:** Change the rule such that the cyclic dependency is eliminated.
7660		Other rules have cyclic dependency on the custom rule. Therefore, the rule cannot be deleted.	**Possible Cause:** Other rules have cyclic dependency on the custom rule you are trying to delete. **Solution:** Change the rule such that the cyclic dependency is eliminated.

Response Code	Reason Code	Description	Possible Cause for Failure
7661	8000	The XML input is invalid.	**Possible Cause:** Value of one or more of the parameters used in the operation is invalid. **Solution:** Provide a valid input for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7662	0	The specified version of the parameter used in the operation is not Active.	**Possible Cause:** The version of the parameter passed to the API was not found. **Solution:** Provide a valid input for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7663		The operation to refresh the specified configuration failed.	**Possible Cause:** The specified configuration does not exist. **Solution:** Ensure that the configuration exists or that you have specified the correct details.
7664	8000	There was an internal error in the operation.	**Possible Cause:** Unexpected internal error. **Solution:** Most likely cause might be Server or database failure. Check the Server transaction and database logs for details and ensure that appropriate action is taken based on the Server logs.
7664	8122	The Device ID could not be generated.	**Possible Cause:** This could be because of a possible internal error at the server end or because of corrupted data.

Response Code	Reason Code	Description	Possible Cause for Failure
7664	8207	There was an internal error because the Server is shutting down.	**Possible Cause:** The Server shutdown is in progress. **Solution:** Wait for sometime for the Server to come up again and then try performing the operation.
7666	8000	The length of one of the parameters used in the operation has exceeded the maximum allowed value.	**Possible Cause:** The length of the specified parameter has exceeded the maximum value. **Solution:** Provide a valid input for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7666	8113	No Association Name parameter was not found in the request.	**Possible Cause:** The Association Name parameter is missing from the request. **Solution:** Ensure that the request contains a valid Tag Name.
7666	8114	The value of one of the parameters used in the operation is not valid.	**Possible Cause:** The parameter specified by ParameterKey has an invalid value. **Solution:** Provide a valid format and value for the parameter that is specified by ParameterKey.
7666	8135	The Device ID used in the operation is not valid.	**Possible Cause:** The specified Device ID is not valid. **Solution:** Ensure that you provide a valid value for the Device ID.

Response Code	Reason Code	Description	Possible Cause for Failure
7667	8000	The length of one of the parameters used in the operation has exceeded the maximum allowed value.	**Possible Cause:** The length of the parameter passed to the API has exceeded the maximum value. **Solution:** Provide a valid value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7667	8140	The length of the User Name parameter has exceeded the maximum allowed value.	**Possible Cause:** The length of the User Name parameter has exceeded the maximum value. **Solution:** Ensure that the parameter is of valid length. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7667	8144	The length of the Association Name parameter has exceeded the maximum allowed value.	**Possible Cause:** The length of the Association Name parameter has exceeded the maximum value. **Solution:** Ensure that the parameter is of valid length. See appendix,"Input Data Validations" (see page 261) for the supported parameter values.
7667	8146	The length of the Transaction Type Name parameter has exceeded the maximum allowed value.	**Possible Cause:** The length of the Transaction Type Name parameter has exceeded the maximum value. **Solution:** Ensure that the parameter is of valid length. See appendix,"Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7667	8148	The length of the First Name parameter has exceeded the maximum allowed value.	**Possible Cause:** The length of the First Name parameter has exceeded the maximum value. **Solution:** Ensure that the parameter is of valid length. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7667	8150	The length of the Last Name parameter has exceeded the maximum allowed value.	**Possible Cause:** The length of the Last Name parameter has exceeded the maximum value. **Solution:** Ensure that the parameter is of valid length. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7667	8152	The length of the PAM parameter has exceeded the maximum allowed value.	**Possible Cause:** The length of the PAM parameter has exceeded the maximum value. **Solution:** Ensure that the parameter is of valid length. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7667	8154	The length of the Email ID parameter has exceeded the maximum allowed value.	**Possible Cause:** The length of the Email ID parameter has exceeded the maximum value. **Solution:** Ensure that the parameter is of valid length. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7667	8156	The length of the Organization Name parameter has exceeded the maximum allowed value.	**Possible Cause:** The length of the Organization Name parameter has exceeded the maximum value. **Solution:** Ensure that the parameter is of valid length. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7668	8000	One of the parameters specified for the operation contains prohibited characters.	**Possible Cause:** A specified parameter contains prohibited characters. **Solution:** Provide a valid format and value for the parameter that is specified. See appendix,"Input Data Validations" (see page 261) for the supported parameter values.
7668	8141	The User Name parameter contains prohibited characters.	**Possible Cause:** The User Name parameter has prohibited characters. **Solution:** Provide a valid format and value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7668	8145	The Association Name parameter contains prohibited characters.	**Possible Cause:** The Association Name parameter has prohibited characters. **Solution:** Provide a valid format and value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7668	8147	The Transaction Type parameter contains prohibited characters.	**Possible Cause:** The Transaction Type parameter has prohibited characters. **Solution:** Provide a valid format and value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7668	8149	The First Name parameter contains prohibited characters.	**Possible Cause:** The First Name parameter has prohibited characters. **Solution:** Provide a valid format and value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7668	8151	The Last Name parameter contains prohibited characters.	**Possible Cause:** The Last Name parameter has prohibited characters. **Solution:** Provide a valid format and value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7668	8153	The PAM (Personal Assurance Message) parameter contains prohibited characters.	**Possible Cause:** The PAM parameter has prohibited characters. **Solution:** Provide a valid format and value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
7668	8155	The EMAIL ID parameter contains prohibited characters.	**Possible Cause:** The EMAIL ID parameter has prohibited characters. **Solution:** Provide a valid format and value for the parameter. See appendix, "Input Data Validations" (see page 261) for the supported parameter values.
7669	8158	The EMAIL ID parameter used in the operation does not meet the formatting requirements.	**Possible Cause:** The EMAIL ID parameter specified by ParameterKey has invalid format. **Solution:** Provide a valid format for the parameter.
7670	8000	The value of one of the configurations used in the operation does not exist.	**Possible Cause:** The specified configuration for the organization is not correct. **Solution:** Ensure that the specified organization configuration is correct. **Possible Cause:** There is no configured ruleset for the specified transaction. **Solution:** Ensure that the specified ruleset exists or that you specify the correct ruleset information.

Response Code	Reason Code	Description	Possible Cause for Failure
7670	8120	The value of ruleset configuration used in the operation does not exist for the specified Channel.	**Possible Cause:** The specified configuration for the organization and/or channel is not correct. **Solution:** Ensure that the configuration information that you specify is correct. **Possible Cause:** There is no ruleset configured for the specified channel of the organization. **Solution:** Ensure that the specified ruleset exists or that you specify correct ruleset information.
7670	8121	The value of ruleset configuration used in the operation does not exist.	**Possible Cause:** The specified configuration for the organization is not correct. **Solution:** Ensure that the specified organization configuration is correct. **Possible Cause:** There is no configured ruleset for the specified transaction. **Solution:** Ensure that the specified ruleset exists or that you specify correct ruleset information.
7671	8000	There was a failure in creating the association.	**Possible Cause:** The specified information is not valid. **Solution:** Ensure that the inputs you specify are correct.
7671	8109	There was a failure in deleting the association.	**Possible Cause:** The specified association does not exist. **Solution:** Ensure that the specified association exists.

Response Code	Reason Code	Description	Possible Cause for Failure
7672	8115	The details for the Organization Name that you specified for the operation is not active.	**Possible Cause:** The specified organization has been deactivated. **Solution:** Ensure that the organization is valid and active.
7672	8139	The details for the Organization Name that you specified for the operation were not found.	**Possible Cause:** The specified organization is unknown and was not found in the system. **Solution:** Ensure that the organization is valid and active.
7673	8000	The specified input is not valid.	**Possible Cause:** The specified input is not valid. **Solution:** You must provide a valid input in the required format.
7673	8184	The specified input is not valid.	**Possible Cause:** The user information is not provided. **Solution:** Ensure that you provide the required information for a valid and active user.
7678	8000	The Organization Name that you specified for the operation was not found.	**Possible Cause:** The specified organization does not exist. **Solution:** You must provide a valid organization name.
7679	8000	The Organization Name that you specified for the operation is not valid.	**Possible Cause:** The specified input is not valid. **Solution:** You must provide a valid input in the required format.

Response Code	**Reason Code**	**Description**	**Possible Cause for Failure**
7681	8000	The User Name that you specified for the operation was not found.	**Possible Cause:** The specified user does not exist. **Solution:** You must provide a valid user name.
7683	8000	The User Name that you specified for the operation already exists.	**Possible Cause:** The specified user already exists in the system. **Solution:** You must provide a distinct user name.
7684	8000	The user account for the corresponding User Name that you specified for the operation has been disabled.	**Possible Cause:** The specified user account has been deactivated. **Solution:** Ensure that the user is active.
7690	8000	The user account for the User Name that you specified for the operation already exists.	**Possible Cause:** The specified user account already exists in the system. **Solution:** You must provide a distinct user account name.
7691	8000	Web Service authentication or authorization failed.	**Possible Cause:** Sent to the client if one of the following Web Services is enabled for Authentication and Authorization (AnA), and the supplied credential is not valid: ■ getUserProfile ■ addUserToExceptionList ■ deleteUserFromExceptionList ■ getLocationAndConnectionInfo **Solution:** Ensure that you send the correct credential or authorization token while making the Web Service call.

Appendix D: Input Data Validations

To ensure that the system does not process invalid data, to enforce business rules, and to ensure that user input is compatible with internal structures and schemas, RiskMinder validates the data that it receives from the APIs.These validations can be grouped as:

- User Data Service Validations (see page 261)
- RiskMinder Validations (see page 264)

User Data Service Validations

The following table explains the criteria that the User Data Service (UDS) uses to validate the input data.

Attribute	Attribute ID	Validation Criteria
User Name	UserName	Is non-empty.
		Length is between 1 and 256 characters.
		Does not contain invalid characters (ASCII 0-31).
First Name	FirstName	Is non-empty.
		Length is between 1 and 32 characters.
		Does not contain invalid characters (ASCII 0-31).
Middle Name	MiddleName	Length is between 0 and 32 characters.
		Does not contain invalid characters (ASCII 0-31).
Last Name	LastName	Is non-empty.
		Length is between 1 and 32 characters.
		Does not contain invalid characters (ASCII 0-31).
Email	Email	Is non-empty.
		Length is between 1 and 128 characters.
		Does not contain invalid characters. All default regular expressions are allowed.
Telephone	TelephoneNumber	Is non-empty.

Attribute	**Attribute ID**	**Validation Criteria**
Number		Length is between 1 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Personal Assurance Message	PAM	Length is between 0 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Personal Assurance Message URL	PAM URL	Length is between 0 and 128 characters.
		Does not contain invalid characters, although alphabets, number, and + / \ \ # $ % & - _ : . are allowed.
Image	Image	Size is between 0 and 1024 KB.
		Is of one of the following formats: ■ JPEG ■ JPG ■ GIF ■ BMP ■ PNG
Account ID	AccountID	Length is between 0 and 256 characters.
		Is non-empty if the AccountType attribute is enabled.
		Does not contain invalid characters (ASCII 0-31).
Account ID Attribute1	AccountIDAttribut e1	Length is between 0 and 256 characters.
		Does not contain invalid characters (ASCII 0-31).
Account ID Attribute2	AccountIDAttribut e2	Length is between 0 and 256 characters.
		Does not contain invalid characters (ASCII 0-31).
Account ID Attribute3	AccountIDAttribut e3	Length is between 0 and 256 characters.
		Does not contain invalid characters (ASCII 0-31).

Attribute	Attribute ID	Validation Criteria
User Custom Attributes	User Custom Attributes--	Maximum supported database column size for the field is 2000 KB. The maximum length is dependent on number of custom attributes, multi-byte character support, and encryption.
		Does not contain invalid characters (ASCII 0-31).
Organization Name	OrgName	Is non-empty.
		Length is between 0 and 64 characters.
		Does not contain invalid characters. **Note:** All keyboard characters are supported.
Display Name	DisplayName	Is non-empty.
		Length is between 0 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Description	Description	Length is between 1 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Account Type	AccountType	Length is between 0 and 64 characters.
		Does not contain invalid characters. **Note:** All keyboard characters are supported.
Account Type Display Name	AccountType-DisplayName	Is non-empty if the AccountType attribute is enabled.
		Length is between 0 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Organization Custom Attributes	Org Custom Attributes	Length is between 0 and (2000 -(2 * No of custom attributes - 1)) characters.
		Does not contain invalid characters (ASCII 0-31).
Account Type Custom Attribute	Account Type Custom Attribute	Length is between 0 and (2000 -(2 * No of custom attributes - 1)) characters.
		Does not contain invalid characters (ASCII 0-31).
User Account	User Account	Length is between 0 and 64 characters.

Attribute	Attribute ID	Validation Criteria
Custom Attributes - Name	Custom Attributes -Name	Does not contain invalid characters (ASCII 0-31).
User Account Custom Attributes - Value	User Account Custom Attributes - Value	Length is between 0 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Key Label	Key Label	Is non-empty.
		Does not contain invalid characters. **Note:** All keyboard characters are supported.

RiskMinder Validations

The following table explains the criteria that RiskMinder Server uses to validate this input data.

Note: Attribute length mentioned in the following table corresponds to the character length. Attribute ID is referred to as paramName in the Java APIs.

Attribute	Attribute ID	Validation Criteria
User Name	USER_NAME	Is non-empty.
		Length is between 1 and 256 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Organization Name	ORG_NAME	Is non-empty.
		Length is between 1 and 64 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Display Organization Name	DISPLAY_ORG_NAME	Is non-empty.
		Length is between 1 and 1024 characters.
Machine Fingerprint	DEVICESIGNATURE	No validation, except that RiskMinder Server must be able to parse it.
Action	ACTION	Is non-empty.
		Length is between 1 and 32 characters.
		Does not contain whitespace characters.

Attribute	Attribute ID	Validation Criteria
DeviceID	DEVICEIDVALUE	Is generated by RiskMinder Server, so that the Server is able to parse it.
Device Type	DEVICEIDTYPE	Has one of the following values: ■ HTTP
Rule Annotation	RULEANNOTATION	No validation, except that RiskMinder Server must be able to parse it.
Start Time	START_TIME	Is non-empty.
End Time	END_TIME	Is non-empty.
Create Time	CREATE_TIME	Is non-empty.
		Is less than or equal to the current time.
Last Modified Time	LAST_MODIFIED_TIM E	Is non-empty.
		Is less than or equal to the current time.
Configuration Name	CONFIG_NAME	Configuration name is non-empty.
		Length is between 1 and 64 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Channel Name	CHANNEL_NAME	Channel name is non-empty.
		Length is between 1 and 64 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Configuration State	CONFIG_STATE	Is non-empty.
		Length is between 0 and 2 characters.
Configuration State for Administration Web Service	CONFIG_STATE_WS	Is non-empty.
		Length is not more than 5 characters.
Country Name	COUNTRY_NAME	Is non-empty.
		Length is between 0 and 50 characters.
		Does not contain invalid characters (ASCII 0-31), although ASCII 32-127 are allowed.
Country Code	COUNTRY_CODE	Is non-empty.
		Length is between 1 and 2 characters.

Attribute	Attribute ID	Validation Criteria
		Can contain numbers, alphabets, underscore, and dot.
Start IP	START_IP	Is non-empty.
		Length is between 0 and 4294967295 characters.
		Follows the IP address format.
End IP	END_IP	Length is between 0 and 4294967295 characters.
		Follows the IP address format.
Mask	MASK	Length is between 0 and 4294967295 characters.
		Follows the IP address format.
Start IP	START_IP_STR	Is non-empty.
		Length is between 7 and 15 characters.
		Follows the IP address format.
End IP	END_IP_STR	Length is between 7 and 15 characters.
		Follows the IP address format.
Mask	MASK_STR	Length is between 7 and 15 characters.
		Follows the IP address format.
Type	TYPE	--
Start IP Filter	START_IP_FILTER	Is non-empty.
		Length is between 7 and 15 characters.
		Follows the IP address format.
Source IP Filter	SOURCE_IP_FILTER	Is non-empty.
		Length is between 7 and 15 characters.
		Follows the IP address format.
Rule Name	RULE_NAME	Is non-empty.
		Length is between 1 and 128 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Rule Mnemonic	RULE_MNEMONIC	Is non-empty.
		Length is between 1 and 25 characters.

Attribute	Attribute ID	Validation Criteria
		Does not contain invalid characters, although numbers, alphabets, underscore (_), and hyphen (-) are allowed.
Rule Description Name	RULE_DESCR_NAME	Length is between 1 and 128 characters.
Rule Description	RULE_DESCRIPTION	Length is between 1 and 1024 characters.
Rule Library Name	RULE_LIB	Is non-empty.
Parameter Name	RULE_PARAM_NAME	Is non-empty.
		Length is between 1 and 64 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Parameter Value	RULE_PARAM_VALUE _STR	Is non-empty.
		Length is between 1 and 512 characters.
Parameter Value	RULE_PARAM_VALUE _BIN	Is non-empty.
Parameter Type	RULE_PARAM_TYPE	Is non-empty.
		Length is between 1 and 4 characters.
Aggregator Name	AGGREGATOR_NAME	Length is between 1 and 64 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Aggregator ID	AGGREGATOR_ID	Is non-empty.
		Length is between 1 and 128 characters.
Combination Rule Name1	COMBINATION_RULE _ NAME1	Is non-empty.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Combination Rule Name2	COMBINATION_RULE _ NAME2	Is non-empty.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Combination Rule Match1	COMBINATION_RULE _ MATCH1	Is non-empty.
		Length is between 0 and 1 characters.
Combination Rule	COMBINATION_RULE	Is non-empty.

Attribute	Attribute ID	Validation Criteria
Match2	_MATCH2	Length is between 0 and 1 characters.
Advice	ADVICE	Length is between 1 and 64 characters.
Score	SCORE	Value is between 1 and 100.
Scoring Priority	SCORING_PRIORITY	Value is between 1 and 2147483647.
Execution Enabled	EXECUTIONENABLED	Value must either be 0 or 1.
Scoring Enabled	SCORINGENABLED	Value must either be 0 or 1.
Other Organization Name	OTHERORGNAME	Length is between 1 and 64 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Other Configuration Name	OTHERCONFIGNAME	Length is between 1 and 64 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Sharing Type	SHARINGTYPE	Is non-empty.
		Value is between 1 and 3.
Callout Type	CALLOUT_TYPE	Value is between 0 and 2.
Callout URL	CALLOUT_URL	Is non-empty.
		Length is between 0 and 150 characters.
		Does not contain invalid characters, although alphabets, number, and + / \ \ # $ % & - _ : . are allowed.
Callout Timeout	CALLOUT_TIMEOUT	Value is between 0 and 1000000.
Instance Name	INSTANCE_NAME	Length is between 0 and 32 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Protocol Module Name	PROTOCOL_MODULE_NAME	Length is between 0 and 128 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Client SSL TrustStore Name	CLIENT_SSL_TRUST_STORE_NAME	Length is between 0 and 64 characters.
		Does not contain invalid characters, although ASCII 32-127 are allowed.
Connection Timeout	CONNECTION_TIMEOUT	Value is between 0 and 1000000.

Attribute	Attribute ID	Validation Criteria
Client Certificate	CLIENT_CERT	--
Client Key	CLIENT_KEY	--
Server Root CA Certificate	SERVER_ROOT_CA_CERT	--
Server Private Key	SERVER_PRIVATE_KEY	--
Read Timeout	READ_TIMEOUT	--
Minimum Connections	MIN_CONNECTION	--
Maximum Connections	MAX_CONNECTION	--
Full Distinguished Name of the Certificate	CERT_SUBJECT	--
Issuer Name	ISSUER_NAME	--
Server SSL Authentication	SERVER_AUTH_SSL	--
Client SSL Authentication	CLIENT_AUTH_SSL	--
Action	TRANS_ACTION	Is non-empty.
		Length is between 1 and 32 characters.
		Does not contain invalid characters (ASCII 0-31).
Association Name	ASSOC_NAME	Is non-empty.
		Length is between 1 and 32 characters.
		Does not contain invalid characters (ASCII 0-31).

Appendix E: RiskMinder Logging

To effectively manage the communication between RiskMinder Server and your application, it is necessary to get information about the activity and performance of the Server and other components, as well as any problems that might have occurred.

This appendix describes the various log files supported by RiskMinder, the severity levels that you will see in these files, and the formats of these log files. It covers the following topics:

About the Log Files

The RiskMinder log files can be categorized as:

- Installation Log File (see page 273)
- Startup Log Files (see page 273)
- Transaction Log Files (see page 276)
- Administration Console Log File (see page 278)
- UDS Log File (see page 279)

The parameters that control logging in these files can be configured either by using the relevant INI files (as is the case with UDS, Administration Console, and Server Startup log files) or by using Administration Console itself (as is the case with the RiskMinder log file.) The typical logging configuration options that you can change in these files include:

- **Specifying the log file name and path:** RiskMinder enables you to specify the directory for writing the log files and storing the backup log files.
- **Specifying the Log file size:** You can specify the maximum number of bytes that the log file can contain. When the log files reach this size, a new file with the specified name is created and the old file is moved to the backup directory.
- **Using log file archiving:** As RiskMinder components run and generate diagnostic messages, the size of the log files increases. If you allow the log files to keep increasing in size, then the administrator must monitor and clean up the log files manually. RiskMinder enables you to specify configuration options that limit how much log file data is collected and saved. RiskMinder allows you specify the configuration option to control the size of diagnostic logging files. This helps you determine a maximum size for the log files. When the maximum size is reached, older log information is moved to the backup file before the newer log information is saved.
- **Setting logging levels:** RiskMinder also allows you to configure logging levels. By configuring logging levels, the number of messages saved to diagnostic log files can be reduced; or reversely, the number of messages can be increased to obtain greater details. For example, you can set the logging level so that the system only reports and saves critical messages. See "Supported Severity Levels" (see page 281) for more information on the supported log levels.
- **Specifying time zone information:** RiskMinder enables you to use either the local time zone or GMT for time stamping the logged information.

Installation Log File

When you install RiskMinder, the installer records in the Arcot_RiskFort_Install<*timestamp*>.log file all the information that you provide during the installation and the actions (such as creating the Arcot directory structure and making registry entries) that it performs. The information in this file is very useful in identifying the source of the problems if the RiskMinder installation did not complete successfully.

The default location of this file is at the same level as the *install_location*.

Startup Log Files

Because RiskMinder comprises two server modules, RiskMinder Server and Case Management Queuing Server, you will see two startup log files:

- RiskMinder Server Startup Log File (see page 273)
- Case Management Queuing Server Startup Log File (see page 275)

The default location of these files is:

On Microsoft Windows:
`install_location\Arcot Systems\logs\`

On UNIX-Based Platforms:
`install_location/arcot/logs/`

RiskMinder Server Startup Log File

When you start RiskMinder Server, it records all startup (or boot) actions in the arcotriskfortstartup.log file. The information in this file is useful in identifying the source of problems if the RiskMinder service does not start up.

In this file, all logging-related parameters (specified under the [arcot/riskfort/logger] section) are controlled by Administration Console. To configure these parameters, you must use the instance-specific configuration page that you can access by clicking the required instance in the **Instance Management** page.

Changing RiskMinder Startup Logging Parameters

To change the logging parameters that you see when RiskMinder Server starts up:

1. Navigate to the conf directory in ARCOT_HOME.
2. Open arcotcommon.ini in a text editor of your choice.
3. Add the following section at the end of the file:

```
[arcot/riskfort/startup]
LogFile=
LogFileSize=10485760
BackupLogFileDir=
LogLevel=
LogTimeGMT=0
```

The following table explains these parameters.

Parameter	**Default**	**Description**
LogFile		The file path to the default directory and the file name of the log file. **Note:** This path is relative to ARCOT_HOME, *install_location*\Arcot Systems\ for Microsoft Windows and *install_location*/arcot/ for UNIX-based platforms.
LogFileSize	10485760	The maximum number of **bytes** the log file can contain. When a log file reaches this size, a new file is started and the old file is moved to the location specified for BackupLogFileDir.
BackupLogFileDir		The location of the directory where backup log files are maintained, after the current file exceeds LogFileSize bytes. **Note:** This path is relative to ARCOT_HOME, *install_location*\Arcot Systems\ for Microsoft Windows and *install_location*/arcot/ for UNIX-based platforms.

Parameter	Default	Description
LogLevel		The default logging level for the server, unless an override is specified. The possible values are: ■ 0 FATAL ■ 1 WARNING ■ 2 INFO ■ 3 DETAIL
LogTimeGMT	0	The parameter that indicates the time zone of the time stamp in the log files. The possible values are: ■ 0 Local Time ■ 1 GMT

1. Set the required values for the parameters that you want to change.
2. Save and close the file.
3. Restart RiskMinder Server.

Case Management Queuing Server Startup Log File

When you start Case Management Queuing Server, it records all startup (or boot) actions in the arcotriskfortcasemgmtstartup.log file. The information in this file is useful in identifying the source of problems if the Case Management Queuing service does not start up.

In this file, all logging-related parameters (specified under the [arcot/riskfortcasemgmtserver/logger]section) are controlled by Administration Console. To configure these parameters, you must use the instance-specific configuration page that you can access by clicking the required instance in the **Instance Management** page.

Changing Case Management Queuing Server Startup Logging Parameters

To change the logging parameters that you see when Case Management Queuing Server starts up:

1. Navigate to the conf directory in ARCOT_HOME.
2. Open arcotcommon.ini in a text editor of your choice.
3. Add the following section at the end of the file:

   ```
   [arcot/riskfortcasemgmtserver/startup]
   LogFile=
   LogFileSize=10485760
   BackupLogFileDir=
   LogLevel=
   LogTimeGMT=0
   ```

   RiskMinder Server Startup Log File (see page 273) explains these parameters.
4. Set the required values for the parameters that you want to change.
5. Save and close the file.
6. Restart Case Management Queuing Server.

Transaction Log Files

The transaction logs consist of:

- RiskMinder Server Log (see page 277)
- Case Management Server Log File (see page 277)

RiskMinder Server Log

RiskMinder records all requests processed by the server and related actions in the arcotriskfort.log file. The default location of this file is:

On Microsoft Windows:
`install_location\Arcot Systems\logs\`

On UNIX-Based Platforms:
`install_location/arcot/logs/`

Note: You cannot use the RiskMinder logger to configure your application's logs. You can access these logs by using the tool used by the third-party application server (such as Apache Tomcat or IBM Websphere) that is hosting your application.

All logging-related parameters can be configured by using Administration Console. To do so, you must use the instance-specific configuration page that you can access by clicking the required instance in the **Instance Management** page.

In addition to the log file path, the maximum log file size (in bytes), backup directory, logging level, and timestamp information, you can control whether you want to enable trace logging. See "Format of the RiskMinder Server and Case Management Server Log Files" (see page 280) for details of the default format used in the file.

Case Management Server Log File

When you deploy the Case Management Server module and subsequently start it, the details of all its actions and processed requests are recorded in the arcotriskfortcasemgmtserver.log file. The default location of this file is:

On Microsoft Windows:
`install_location\Arcot Systems\logs\`

On UNIX-Based Platforms:
`install_location/arcot/logs/`

All logging-related parameters (specified under the [arcot/riskfortcasemgmtserver/logger] section) can be configured by using Administration Console. To configure these parameters, you must use the instance-specific configuration page that you can access by clicking the required instance in the **Instance Management** page.

In addition to the log file path, the maximum log file size (in bytes), backup directory, logging level, and timestamp information, you can control whether you want to enable trace logging. See "Format of the RiskMinder Server and Case Management Server Log Files" (see page 280) for the details of the default format used in the file.

Administration Console Log File

When you deploy Administration Console and subsequently start it, the details of all its actions and processed requests are recorded in the arcotadmin.log file. This information includes:

- Database connectivity information
- Database configuration information
- Instance information and the actions performed by this instance
- UDS configuration information
- Other Administration Console information specified by the Master Administrator, such as cache refresh

The information in this file is useful in identifying the source of problems if Administration Console does not start up. The default location of this file is:

On Microsoft Windows:
`install_location``\Arcot Systems\`**`logs`**`\`

On UNIX-Based Platforms:
`install_location``/arcot/`**`logs`**`/`

The parameters that control logging in these files can be configured by using the adminserver.ini file, which is available in the conf folder in ARCOT_HOME.

In addition to the logging level, log file name and path, the maximum log file size (in bytes), log file archiving information, you can control the layout of the logging pattern for the Console by specifying the appropriate values for log4j.appender.debuglog.layout.ConversionPattern.

See "Format of UDS and Administration Console Log Files" (see page 281) for details of the default format used in the file.

UDS Log File

Important! This file is generated only if you deployed the arcotuds.war file to enable LDAP connectivity.

All User Data Service (UDS) information and actions are recorded in the arcotuds.log file. This information includes:

- UDS database connectivity information
- UDS database configuration information
- UDS instance information and the actions performed by this instance

The information in this file is useful in identifying the source of problems if Administration Console could not connect to the UDS instance. The default location of this file is:

On Microsoft Windows:
`install_location\Arcot Systems\logs\`

On UNIX-Based Platforms:
`install_location/arcot/logs/`

The parameters that control logging in this files can be configured by using the udsserver.ini file, which is available in the conf folder in ARCOT_HOME.

In addition to the logging level, log file name and path, the maximum file size (in bytes), and archiving information, you can control the layout of the logging pattern for UDS by specifying the appropriate values for log4j.appender.debuglog.layout.ConversionPattern.

See "Format of UDS and Administration Console Log Files" (see page 281) for details of the default format used in the file.

Format of the RiskMinder Server and Case Management Server Log Files

The following table describes the format of the entries in the RiskMinder logger, arcotriskfort.log and arcotriskfortcasemgmtserver.log, as discussed in the section, "Transaction Log Files" (see page 276).

Column	Description
Time Stamp	The time the entry was logged, translated to the specified time zone. The format of logging this information is: www mmm dd HH:MM:SS.mis yy z In the preceding format: ■ www represents weekday. ■ mis represents milliseconds. ■ z represents the time zone you specified in the arcotcommon.ini file.
Log Level (or Severity)	The severity level of the logged entry. See "Supported Severity Levels" (see page 281) for detailed information.
Process ID (pid)	The ID of the process that logged the entry.
Thread ID (tid)	The ID of the thread that logged the entry.
Transaction ID	The ID of the transaction that logged the entry.
Message	The message logged by the Server in the free-flowing format. **Note:** The granularity of this message depends on the Log Level that you set in arcotcommon.ini.

Format of UDS and Administration Console Log Files

The following table describes the format of the entries in the following log files:

- arcotuds.log (UDS Log File (see page 279))
- arcotadmin.log (Administration Console Log File (see page 278))

Column	Associated Pattern (In the Log File)	Description
Time Stamp	%d{yyyy-MM-dd hh:mm:ss,SSS z} :	The time when the entry was logged. This entry uses the application server time zone. The format of logging this information is: yyyy-MM-dd hh:mm:ss,mis z Here: ■ mis represents milliseconds. ■ z represents the time zone.
Thread ID	[%t] :	The ID of the thread that logged the entry.
Log Level (or Severity)	%-5p :	The severity level of the logged entry. See "Supported Severity Levels" (see page 281) for more information.
Logger Class	%-5c{3}(%L) :	The name of the logger that made the log request.
Message	%m%n :	The message logged by the Server in the log file in the free-flowing format. **Note:** The granularity of the message depends on the Log Level that you set in the log file.

Refer to the following URL for customizing the PatternLayout parameter in the UDS and Administration Console log files:

http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html

Supported Severity Levels

A *log level* (or *severity level*) enables you to specify the level of detail of the information stored in the RiskMinder logs. This also enables you to control the rate at which the log file will grow.

Server Log File Severity Levels

The following table describes the log levels that you see in server log files, in the *decreasing* order of severity.

Log Level		Description
0	FATAL	Use this log level for serious, non-recoverable errors that can cause the abrupt termination of the RiskMinder service. At the FATAL level, only situations which indicate a fatal problem will be logged.
1	WARNING	Use this log level for undesirable run-time exceptions, potentially harmful situations, and recoverable problems that are not yet FATAL.
2	INFO	Use this log level for capturing information on run-time events. In other words, this information highlights the progress of the application, which might include changes in: ■ Server state, such as start, stop, and restart. ■ Server properties. ■ State of services. ■ State of processes on the Server. For example, there are some logs that will always be printed to indicate that requests are being received and that they are being processed. These logs appear at the INFO level.
3	LOW DETAIL	Use this log level for logging detailed information for debugging purposes. This might include process tracing and changes in Server states.

Note: When you specify a log level, messages from all other levels of *higher* significance are reported as well. For example if the LogLevel is specified as 3, then messages with log levels of FATAL, WARNING, and INFO level are also captured.

Administration Console and UDS Log File Severity Levels

The following table describes the log levels that you see in the Administration Console and UDS log files, in the *decreasing* order of severity.

Log Level		Description
0	OFF	Use this log level to disable all logging.
1	FATAL	Use this log level for serious, non-recoverable errors that can cause the abrupt termination of the RiskMinder service.

Log Level		Description
2	WARNING	Use this log level for undesirable run-time exceptions, potentially harmful situations, and recoverable problems that are not yet FATAL.
3	ERROR	Use this log level for recording error events that might still allow the application to continue running.
4	INFO	Use this log level for capturing information on run-time events. In other words, this information highlights the progress of the application, which might include changes in: ■ Server state, such as start, stop, and restart. ■ Server properties. ■ State of services. ■ State of a processes on the Server.
5	TRACE	Use this log level for capturing finer-grained informational events than DEBUG.
6	DEBUG	Use this log level for logging detailed information for debugging purposes. This might include process tracing and changes in Server states.
7	ALL	Use this log level to enable all logging.

Note: When you specify a log level, messages from all other levels of *higher* significance are reported as well. For example if the LogLevel is specified as 4, then messages with log levels of FATAL, WARNING, ERROR, and INFO level are also captured.

Sample Entries for Each Log Level

The following subsections show a few sample entries (based on the Log Level) in the **RiskMinder log files.**

FATAL

```
May 27 18:31:01.585 2010 GMT FATAL: pid 4756 tid 5152: 0: 0: Cannot continue due to
ARRF_LIB_init failure, SHUTTING DOWN
```

WARNING

```
May 24 14:47:39.756 2010 GMT WARNING: pid 5232 tid 5576: 0: 110000: EVALHTTPCALLOUT
: Transport Exception : create: No Transports Available
```

INFO

```
May 24 14:41:43.758 2010 GMT INFO: pid 3492 tid 4904: 0: 109002: Error in
ArPFExtRuleSetEval::evaluate Could not get user context (two parallel requests)
```

```
May 25 10:01:28.131 2010 GMT WARNING: pid 1048 tid 3104: 8: 0: Error in
ArRFCaseStatus::startInit: No data found
```

DETAIL

```
May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : Entering USERRISKEVALVELOCITY Rule Evaluation function

May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE: VELOCITY_DURATION=[60],
VELOCITY_DURATION_UNIT=[MINUTES],                 VELOCITY_TRANSACTION_COUNT=[5]

May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : Entering UserRiskEvalVelocityRule
durationToTimeConvertor
May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : Exiting UserRiskEvalVelocityRule durationToTimeConvertor

May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : Entering UserRiskEvalVelocityRule
callUserEvalVelocityRule

May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : Entering
ArUserRiskEvalVelocityDBO::decisionLogicForUserVelocity

May 24 14:52:01.219 2010 GMT INFO: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : Entering decisionLogicForUserVelocity

May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : Exiting
ArUserRiskEvalVelocityDBO::decisionLogicForUserVelocity

May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : Exiting UserRiskEvalVelocityRule
callUserEvalVelocityRule

May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : USERRISKEVALVELOCITY.RESULT=[0]

May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : USERRISKEVALVELOCITY.DETAIL=[RESULT=0;TCOUNT=2;
ACT=mection]

May 24 14:52:01.219 2010 GMT LOW: pid 2132 tid 1356: 0: 111004:
USERRISKEVALVELOCITYRULE : Exiting USERRISKEVALVELOCITY Rule Evaluation function
```