CA RiskMinder

Java Developer's Guide

r3.1.01

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) Java classes and methods to enable your online application to programmatically perform risk evaluation and related tasks. This document describes the Java implementation of the RiskMinder SDK.

Sample Code (see page 77) provides a fully-functional sample code that you can run to test the supported user-related operations, risk evaluation, and post-evaluation functionality 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 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.

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

This section introduces you to the Java APIs provided by RiskMinder. It covers the following topics:

- Introduction to the RiskMinder SDK (see page 10)
- RiskMinder SDK Features (see page 11)
- Overview of the Integration Steps (see page 12)
- Before You Begin (see page 14)

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 SDK

The RiskMinder software development kit (SDK) provides a programmatic front-end to RiskMinder. It provides APIs for interacting with RiskMinder Server to perform the tasks needed for assessing each transaction (Risk Evaluation APIs).

After you install RiskMinder, the Java SDKs are available at the following location:

Microsoft Windows:

`<install_location>\Arcot Systems\sdk\java\lib\arcot\`

UNIX-Based Platforms:

`<install_location>/arcot/sdk/java/lib/arcot/`

Tip: If you have not installed RiskMinder yet, you can still access these Javadocs in the Documentation directory of the RiskMinder package you receive.

Risk Evaluation API

The Risk Evaluation API (arcot-riskfort-evaluaterisk.jar) is the interface to RiskMinder Server, which provides the logic for evaluating risk associated with a transaction and returning an appropriate advice.

Based on the various factors collected from user's system and the result of configured rules that are triggered, this API returns a score and a corresponding advice. If RiskMinder recommends additional authentication (which is performed by your application), then this API also 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 API, 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 a system used by the user for a transaction. Users who are not bound are more likely to be challenged in order to be authenticated. You can also list and delete these associations by using this API.

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).

RiskMinder SDK Features

This section discusses the salient features of the Risk Evaluation SDK.

- **Multiple Ways to Initialize the SDKs**

  You can initialize Risk Evaluation SDK either by using properties file or by using a map. See "Before You Begin" (see page 33) for more information on how to do this.

- **Support for SDK Failover**

  Java SDKs support failover mechanism, if an instance of RiskMinder Server is not operational, then the SDKs automatically connect to any of the additional configured instances. See "Configuring a Backup RiskMinder Server Instance" **(see page 68)** for more information on how to do this.

- **Support for SSL**

  You can secure the connection between the Java SDK and RiskMinder Server by using Secure Socket Layer (SSL). To set up SSL between SDK and RiskMinder Server, you must edit the properties files. See "SSL Communication Between RiskMinder Components" (see page 69) for more information on how to do this.

- **Support for Additional Parameters**

  In addition to the mandatory inputs, the APIs also accepts additional input that can be passed as name-value pair. This input can include information, such as locale, calling application details, or additional transaction details.

Overview of the Integration Steps

The RiskMinder SDK offers you multiple degrees of freedom in the available integration methods and the types of risk-based authentication flows. (See "Understanding RiskMinder Workflows" (see page 15) for more information on supported workflows.) This enables you to design the optimal authentication solution that best suits your organization's requirements.

The RiskMinder flows can be integrated with your online application at the points discussed in following subsections.

Before a User Logs in to Your Application (and Just Accesses the Login Page)

In this case, your application must invoke RiskMinder's evaluateRisk() function call from the login page (*before* the user specifies the login credentials) to assess the risk associated with the incoming data. For example, you can evaluate the IP address and/or the country for Negative IP and Negative Country checks.

Note: Negative IP addresses is a collection of IP addresses that have been the origin of known anonymizer proxies or fraudulent or malicious transactions in past. Similarly, Negative countries is a collection of all countries from which fraudulent or malicious transactions have been recorded in past.

In this case, you can also evaluate other RiskMinder rules that do not require user information. These include Device Velocity Check and any custom rules you might have implemented.

After a User Logs in to Your Online Application (By Specifying the User Name and Password to Access Their Account or the Protected Resource)

In this case:

1. Your application must invoke RiskMinder from the main page of your application that appears after successful login. The following scenarios are possible:

   - **User is enrolled with RiskMinder**, and must undergo risk evaluation.

     In this case, your application must invoke RiskMinder's evaluateRisk() function call by passing user, device signature (collected by using the DeviceDNA Javascript provided by RiskMinder), IP address, and transaction details for assessing the risk:

     - If the received user data is assigned a low score after rule execution (based on the incoming data and the data stored for this user or device), then the advice is ALLOW. In this case, your application grants access to the user to the protected resource or Web page and allows the user to continue with the transaction.

        - If the received user data is assigned a high score after rule execution (based on the incoming data and the data stored for this user or device), then the advice is DENY. In this case, your organizational policies determine the outcome. The transaction can be denied or can be forwarded to a security analyst (known as Customer Support Representative (CSR) in RiskMinder terminology) for review and further action.
        - If a transaction is flagged as suspicious, then the advice is INCREASEAUTH. In this case, your application must perform a secondary authentication, which can include industry-standard one-time password (OTP) or security questions (such as mother's maiden name and date of birth).

      Note: You can also use CA AuthMinder for this purpose. See *http://www.ca.com/us/two-factor-authentication.aspx* for more information.

        Only if your application authenticates the user during the secondary authentication, then you must grant the user access to the protected resource or Web page and allow the user to continue with the transaction.

    - **User is new to RiskMinder**, and therefore must be enrolled with it.

      In this case, your application must invoke RiskMinder's createUserRequest message in the ArcotUserRegistrySvc Web service by passing user details to create the user in RiskMinder database.

      Important! Because RiskMinder works "behind the scene" for an end user, it is recommended that you do not change the end-user experience for this enrollment.

      After RiskMinder "knows" the user after this enrollment, then you must call the evaluateRisk() function and allow the user to proceed with the transaction, if the advice is ALLOW.

      See "Enrollment Workflows" (see page 15) for more information on the two different ways in which you can enroll users.

2. Your application invokes RiskMinder's postEvaluate() function after the evaluateRisk() function. RiskMinder Server determines whether to create a user-device association and update the attributes based on the results of this function.

Before You Begin

Before you integrate your application with RiskMinder, RiskMinder must be installed and configured, ensuring that:

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

  Book: Refer to the system requirements in the *CA RiskMinder Installation and Deployment Guide*.

- 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 case of **single-system deployment** of RiskMinder (see "Deploying RiskMinder on a Single System" in the *CA RiskMinder Installation and Deployment Guide* for more information). Ensure that the all the components are up.

In case of **distributed-system deployment** of RiskMinder (see "Deploying RiskMinder on a Distributed System" in the *CA RiskMinder Installation and Deployment Guide* for more information). Ensure that the connection is established between all the components and that they successfully communicate with each other.

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 these workflows and provides an overview of each workflow so you can understand the different processes involved:

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 the next time they undergo risk evaluation, they do not see the same advice.

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 enrollment, you must perform risk evaluation (as discussed in "Risk Evaluation Workflows" (see page 23)).

You can also *implicitly* create the user by setting the **Mode of User Enrollment** 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 (as discussed in "Risk Evaluation Workflows" (see page 23)), the user is automatically created in the system.

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

Explicit Enrollment

In case of *explicit enrollment*, you must explicitly call RiskMinder's createUserRequest message in the ArcotUserRegistrySvc Web service from your application's code to create a user in RiskMinder database. You can call this function either *before* (Scenario 1 (see page 17)) or *after* (Scenario 2 (see page 19)) you perform risk evaluation (by using the evaluateRisk()call.)

Scenario 1

The steps for the explicit enrollment workflow, if you call the createUserRequest message in the ArcotUserRegistrySvc Web service *before* evaluateRisk() function are:

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

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

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

   At this stage, your application must make an explicit call to the createUserRequest message in the ArcotUserRegistrySvc Web service. In this call, you must pass all pertinent user details, such as user's 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 on 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.**

   At this stage, your application collects information from the user's system that will be used by RiskMinder for analyzing 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 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. **Your 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 23) for more information.

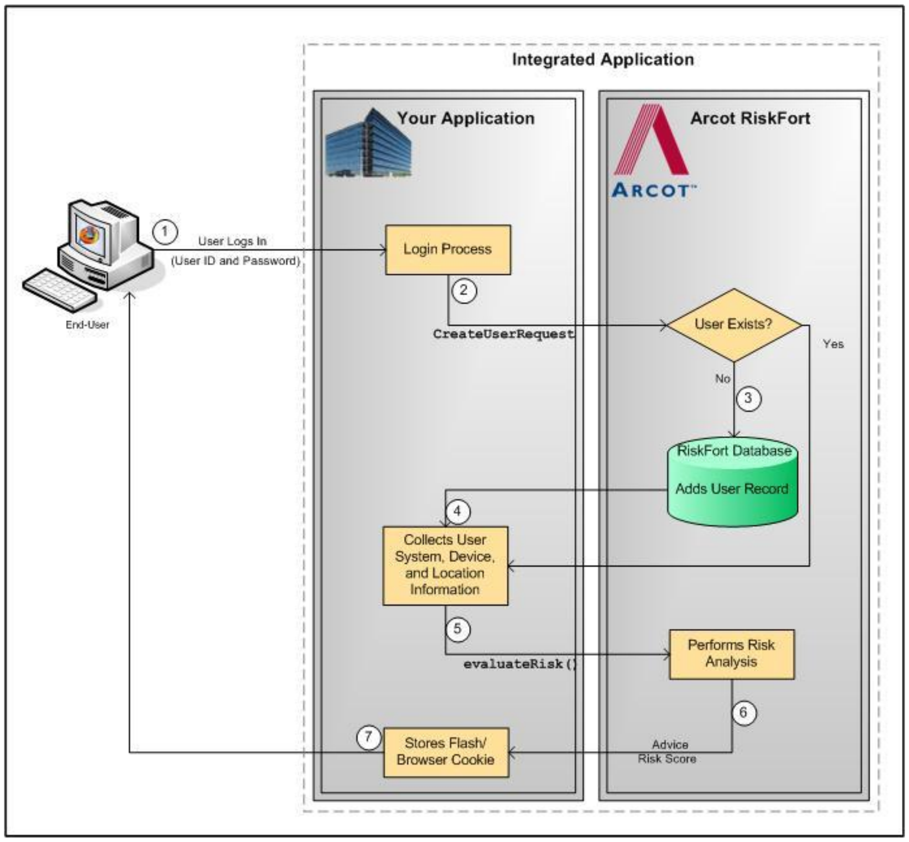
6. **RiskMinder performs risk analysis.**

   RiskMinder generates a risk score and an advice.

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

   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 make the createUserRequest message call before the evaluateRisk() call.



Scenario 2

The steps for the explicit enrollment workflow, if you call the CreateUserRequest message *after* evaluateRisk() function, are:

1. **User logs into your online application.**

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

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

   At this stage, your application collects information from the user's system that will be used by RiskMinder for analyzing 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 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() function.**

   At this stage, your application must call the evaluateRisk() function in riskfortAPI. In this call, you must pass all the user and device information that you collected in the preceding step to RiskMinder.

4. **RiskMinder performs 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.**

   At this stage, your application must make an explicit call to the createUserRequest message in the ArcotUserRegistrySvc Web service. In this call, you must pass all pertinent user details, such as user's name, last name, organization, e-mail, 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 on 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, user is enrolled with RiskMinder.

7. **Your application calls RiskMinder's evaluateRisk() function again.**

At this stage, your application must again call the evaluateRisk() function in riskfortAPI. In this call, you must ensure that you pass all the user and device information that you collected in Step 2 to RiskMinder.

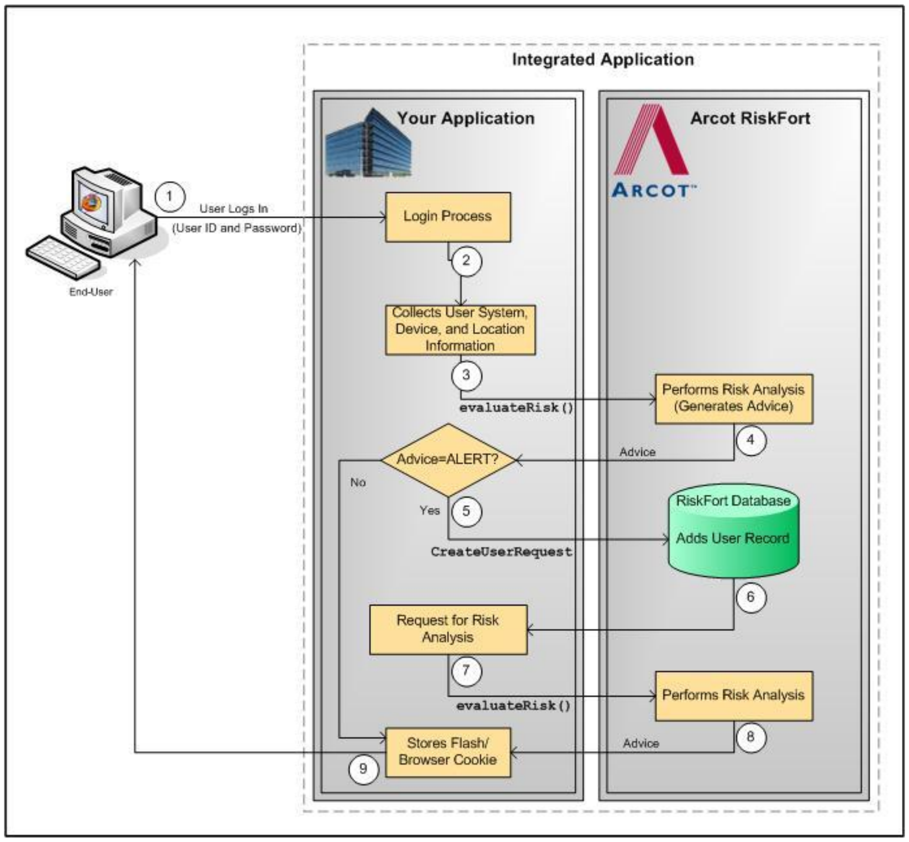
8. **RiskMinder performs risk analysis for the user.**

   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 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 RiskMinder database. Instead when RiskMinder generates the ALERT advice for an "unknown user", it automatically calls the function to enroll the user.

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

The steps for the implicit enrollment workflow are:

1. **User logs into your online application.**

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

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

   At this stage, your application collects information from the user's system that will be used by RiskMinder for analyzing 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 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() function.**

   At this stage, your application must call the evaluateRisk() function in riskfortAPI. In this call, you must pass all the user and device information that you collected in the preceding step 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 database.**

   For an ALERT advice that is generated, RiskMinder 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 on the createUserRequest message.

6. **Your application calls RiskMinder's evaluateRisk() function again.**

   At this stage, your application must again call the evaluateRisk() function in riskfortAPI. In this call, you must ensure that you pass all the user and device information that you collected in Step 2 to RiskMinder.

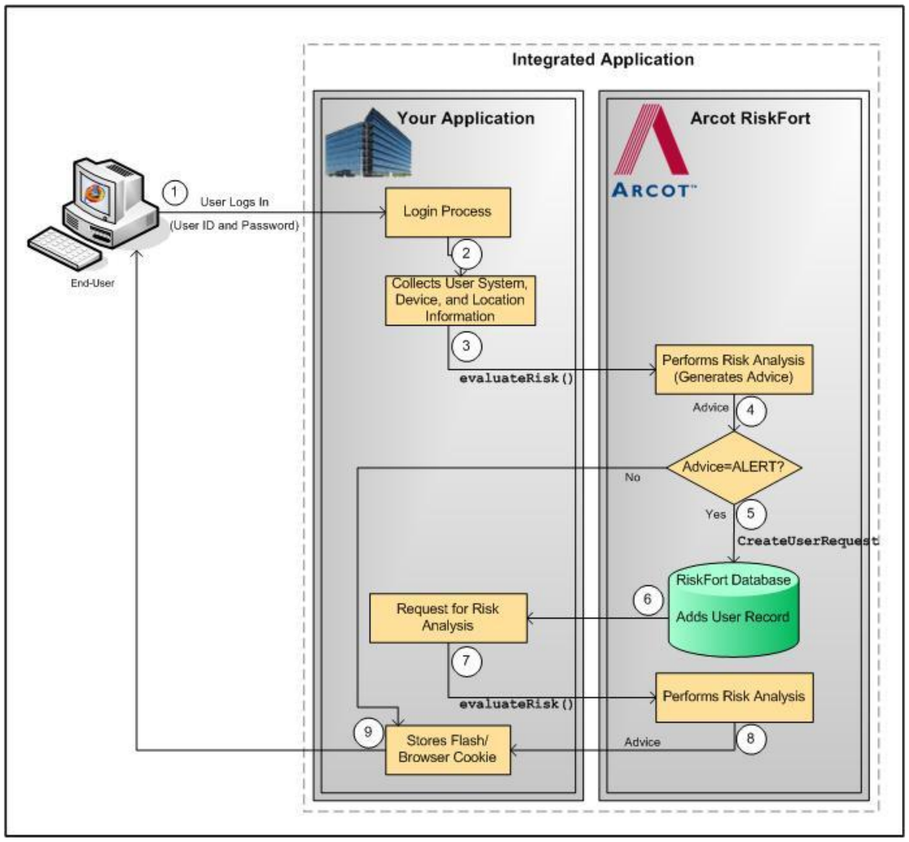
7. **RiskMinder performs risk analysis for the user.**

   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 user is using for the current transaction.

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



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 system.
- 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 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() function, the following workflows are possible:

- Pre-Login Risk Evaluation Workflow (see page 24)
- Post-Login Risk Evaluation Workflow (see page 26)

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 will only use 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.**

   At this stage, your application collects information from the user's system that will be used by RiskMinder for analyzing 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 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() function.**

   At this stage, your application must call the evaluateRisk() function in riskfortAPI. In this call, you must pass the information that you collected in the preceding step to RiskMinder.

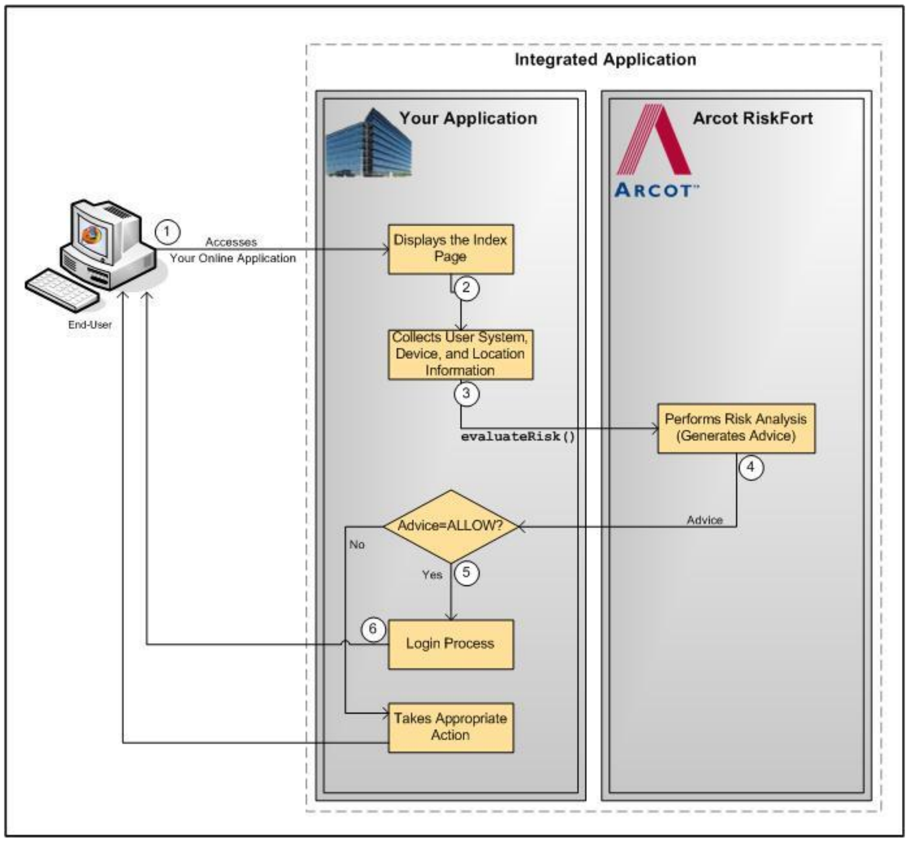
4. **RiskMinder performs risk analysis for the user.**

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

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

   Based on RiskMinder's recommendation, 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.


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 number of factors, such as network information, user information, and (if implemented) transaction information to evaluate users.

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

- In case of ALLOW, the user-device association information is updated.
- In case of ALERT and DENY, the user-device association information is *not* updated at all.
- In 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 29)) 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 into your online application.**

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

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

   At this stage, your application collects information from the user's system that will be used by RiskMinder for analyzing 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 at the client-end.
   - **Location information** that includes 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() function.**

   At this stage, your application must call the evaluateRisk() function in riskfortAPI. In this call, you must pass all the user and device information that you collected in the preceding step 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 29) 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 29) for more information.

6. **Your application calls RiskMinder's postEvaluate() function.**

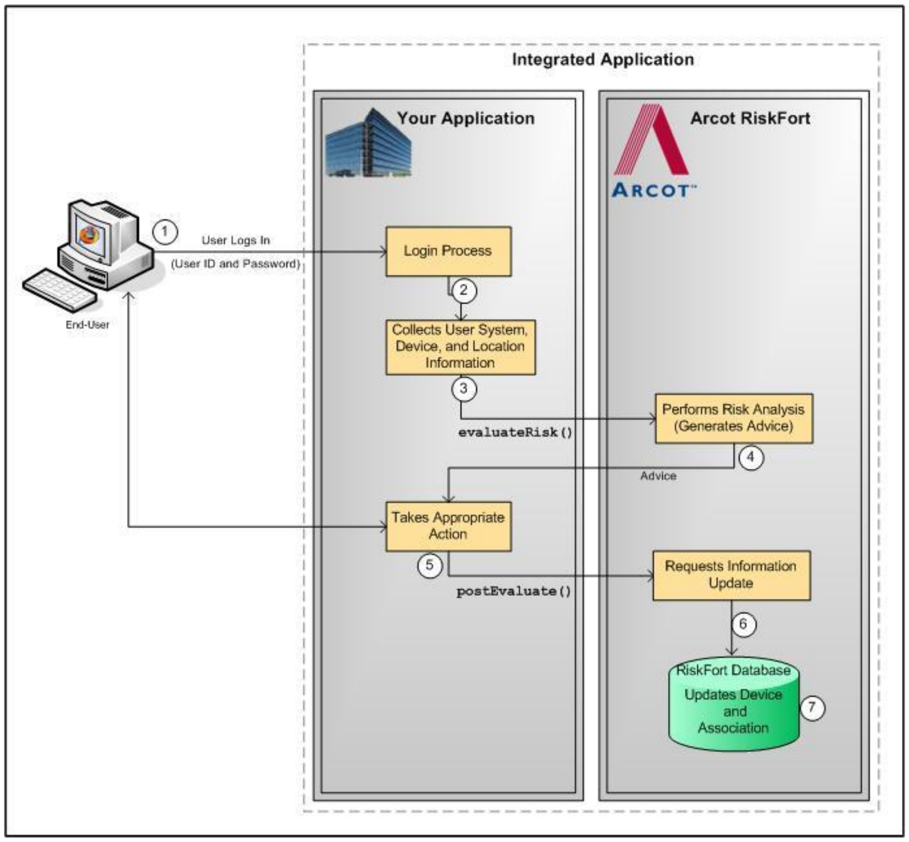
   At this stage, your application must call the postEvaluate() function in riskfortAPI. Based on the output generated by the evaluateRisk() call, this call helps RiskMinder to generate the final advice and update the device and association information.

   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.


Integrated Application
Your Application
Arcot RiskFort
ARCOT
1
User Logs In
(User ID and Password)
End-User
Login Process
2
Collects User System, Device, and Location Information
3
evaluateRisk()
Performs Risk Analysis (Generates Advice)
4
Advice
Takes Appropriate Action
5
postEvaluate()
Requests Information Update
6
RiskFort Database
Updates Device and Association
7

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 to store the feedback to use for risk analysis of future transactions.

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

1. **User logs into your online application.**

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

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

   At this stage, your application collects information from the user's system that will be used by RiskMinder for analyzing 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 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() function.**

   At this stage, your application must call the evaluateRisk() function in riskfortAPI. In this call, you must pass all the user and device information that you collected in the preceding step 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 help further authenticate the user.

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

Based on the secondary authentication mechanism that you are using, your application displays 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 phone authentication.

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

6. **Your application calls RiskMinder’s postEvaluate() function and forwards the result of the secondary authentication to RiskMinder.**

   At this stage, irrespective of the fact 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() function in riskfortAPI. 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.

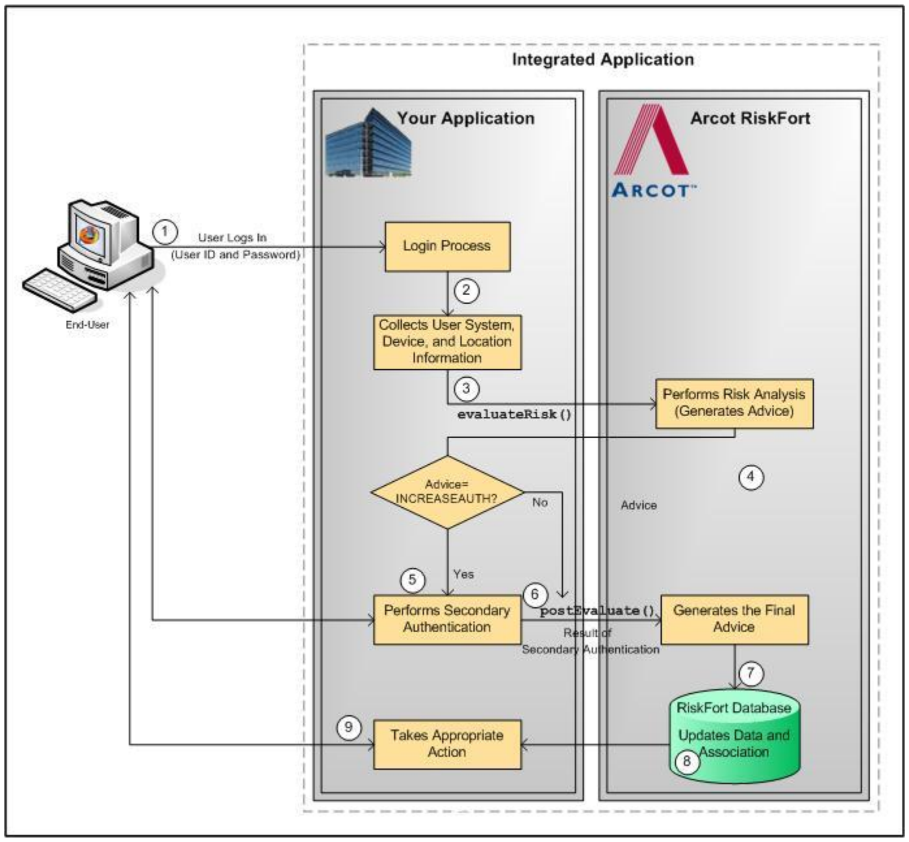
8. **RiskMinder updates the device information and creates the association information.**

   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.


Integrated Application
Your Application
Arcot RiskFort
ARCOT
1 User Logs In (User ID and Password)
End-User
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 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
		Scenario 2: Creates a user in the RiskMinder database, when you call the createUserRequest message **after** evaluateRisk().	■ Post-Login Risk Evaluation

Workflow	**Sub-Type of the Workflow**	**Description**	**Dependant Workflows**
	Implicit	RiskMinder **automatically** creates a user in the RiskMinder database, without you having to make the createUserRequest message call.	■ 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 in case your application performed a secondary authentication after RiskMinder recommended INCREASEAUTH. Also updates user information and device association information.	■ Post-Login Risk Evaluation

Chapter 3: Before You Begin

Before you use the Risk Evaluation API, you must include the related JAR files in the CLASSPATH. If you are using Properties files in your application, then you must also include them in the CLASSPATH. After including the required JAR files or Properties files in CLASSPATH, you must next Initialize the API.

This section covers the following topics:

Configuring Java APIs

To configure RiskMinder Risk Evaluation APIs for using with a J2EE application:

Note: The following instructions are based on Apache Tomcat Server. The configuration process might vary depending on the application server you are using. Refer to the application server documentation for detailed information on these instructions.

1. Before proceeding with the configuration steps in this section, ensure that the JARs required for implementing the Java APIs are installed at:
   - **For Microsoft Windows:** <*install_location*>\Arcot Systems\sdk\java\lib\
   - **For UNIX-Based Platforms:** <*install_location*>/arcot/sdk/java/**lib**/
2. Copy the listed JAR files (*from* the location where they are installed) **to** the appropriate location in your <*APP_SERVER_HOME*> directory. (For example, on Apache Tomcat this location is <*Application_Home*>/WEB-INF/**lib**/.)
   - /sdk/java/lib/arcot/**arcot_core.jar**
   - /sdk/java/lib/arcot/**arcot-pool.jar**
   - /sdk/java/lib/arcot/**arcot-riskfort-evaluaterisk.jar**
   - /sdk/java/lib/arcot/**arcot-riskfort-mfp.jar**
   - /sdk/java/lib/external/**bcprov-jdk14-139.jar**
   - /sdk/java/lib/external/**commons-lang-2.0.jar**
   - /sdk/java/lib/external/**commons-pool-1.4.jar**

   For example, on Apache Tomcat 5.5, you will need to copy these files to C:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps\<*Your_Application*>\WEB-INF\lib\.
3. Configure the log4j.properties.risk-evaluation and riskfort.risk-evaluation.properties files as follows:
   - If the application *already has* a configured log4j.properties.risk-evaluation file, then merge it with the following log configuration files:
     - **On Microsoft Windows:**
       `<`*`install_location`*`>\Arcot Systems\sdk\java\properties\`**`log4j.properties.risk-evaluation`**

       and
       `<`*`install_location`*`>\Arcot Systems\sdk\java\properties\`**`riskfort.risk-evaluation.properties`**
     - **On UNIX-Based Platforms:**

       <*install_location*>/arcot/sdk/java/properties/**log4j.properties.risk-evaluation**

       and

`<install_location>/arcot/sdk/java/properties/`**`riskfort.risk-evaluation.properties`**

- If the application *does not have* the log4j.properties file already configured, then:

a. Rename log4j.properties.risk-evaluation to log4j.properties.

b. Merge riskfort.risk-evaluation.properties with log4j.properties.

c. Copy the log4j.properties file to:
`<Application_Home>/WEB-INF/classes/`**`properties`**`/`

For example, on Apache Tomcat 5.5, you will need to copy log4j.properties to C:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps\<Your_Application>\WEB-INF\classes\.

Including Risk Evaluation JAR Files in CLASSPATH

To use the API, you need to include the Risk Evaluation JAR files in the CLASSPATH variable. To do so:

1. If required, you need to copy the Risk Evaluation JAR files listed in the following tables to the appropriate directory (say lib) in your *<APPLICATION_CONTEXT>*.

For Microsoft Windows

Primary JAR Files	arcot_core.jar arcot-pool.jar arcot-riskfort-evaluaterisk.jar arcot-riskfort-mfp.jar	*<install_location>*\Arcot Systems**sdk****java****lib****arcot**\
Other JAR Files	bcprov-jdk14-131.jar commons-lang-2.0.jar commons-pool-1.4.jar	*<install_location>*\Arcot Systems**sdk****java****lib****external**\

For UNIX-Based Platforms

Primary JAR Files	arcot_core.jar arcot-pool.jar arcot-riskfort-evaluaterisk.jar arcot-riskfort-mfp.jar	*<install_location>*/arcot/**sdk/java/lib/arcot/**
Other JAR Files	bcprov-jdk14-131.jar commons-lang-2.0.jar commons-pool-1.4.jar	*<install_location>*/arcot/**sdk/java/lib/external/**

2. Add the parent directory (in the <*APPLICATION_CONTEXT*>) of the directory to which you copied the JARs to the CLASSPATH environment variable.

Including Properties Files in CLASSPATH

If your application is using Properties files, then the riskfort.risk-evaluation.properties file must be included in CLASSPATH. To ensure this:

1. Copy the **properties** directory from the following location to the appropriate directory (say classes) in your <*APPLICATION_CONTEXT*>.
   - **For Microsoft Windows**
     `<install_location>\Arcot Systems\sdk\java\`
   - **For UNIX-Based Platforms**
     `<install_location>/arcot/sdk/java/`
2. Also add the parent directory of the properties directory to the CLASSPATH environment variable.

Initializing the Risk Evaluation API

Initialize the Risk Evaluation API by using the **RiskFactory** class in the **com.arcot.riskfortAPI** package. The initialization process creates all connection pools, creates the database pool, and initializes loggers. After initialization, it returns an appropriate object to your calling application.

Note: You *cannot* apply any configuration changes *after* you initialize the API. To enable the configuration changes, you must re-initialize the API.

The RiskFactory class provides two methods, as discussed in the following subsections, to initialize the Risk Evaluation APIs:

- Method 1: Initializing the API by Using the Properties File (see page 37)
- Method 2: Initializing the API by Using the Map (see page 38)

Method 1: Initializing the API by Using the Properties File

The initialize(java.lang.String propertyLocation) method initializes the Risk Evaluation API by using the parameters listed in the input Properties file. If you pass NULL, then the parameters are read from the riskfort.risk-evaluation.properties file, which is present in the properties directory of CLASSPATH.

The fields and format of this properties file must be as follows:

```
//RiskFort Server IP address
HOST.1=

//RiskFort Server port number
PORT.1=

//Type of the connection. Possible values include SSL and TCP.
TRANSPORT_TYPE=

//Required if TRANSPORT_TYPE is set to SSL. The file must be in
//PEM format. Provide the complete path for the file.
CA_CERT_FILE=
```

The following table provides the details of the initialize() method.

Description	**Input Parameter**	**Output Value**
Initializes the Risk Evaluation API by using the specified Properties file.	■ propertyLocation The absolute path of the Properties file. **Note:** If you do not pass any value, then the value defaults to the default location of riskfort.risk-evaluation.properties.	Throws RFSDKException if the API was not initialized successfully.

Method 2: Initializing the API by Using the Map

The initialize(java.util.Map initproperty) method in the RiskFactory class initializes the Risk Evaluation API based on the property map provided. The following table provides the details of the map that initialize() method uses.

Description	**Input Value**	**Output Value**
Initializes the Risk Evaluation API by using the provided map.	■ map The key-value pair specifying the configuration information. The supported keys are: – HOST.1 The IP address of the host where RiskMinder Server is available. – PORT.1 The port at which RiskMinder Server is available. Default value is 7680. – TRANSPORT_TYPE The type of the connection. Possible values include SSL and TCP. – CA_CERT_FILE (Required only if TRANSPORT_TYPE is set to SSL.) The path for the CA certificate file of the server. The file must be in PEM format. Provide the complete path for the file.	Throws RFSDKException if the API was not initialized successfully.

Preparing Additional Inputs

In addition to the mandatory inputs, the Risk Evaluation API also accepts additional inputs in form of name-value pairs. These additional inputs are especially useful if you want to create custom rules and Evaluation Callouts. This is because any parameter can be added to this class and can be passed to the custom rules and Evaluation Callouts. These inputs can then be used for evaluating risk.

The input to this class can include information, such as locale, calling application details, or other transaction-related details that can be used for risk evaluation. These additional name-value custom parameters can help you capture the real-time inputs from each transaction, and are processed by deployed custom rules for successful risk evaluation.

RiskMinder's **com.arcot.riskfortAPI.AdditionalInputs** package provides you the **AdditionalInputs** class, which enables you to set the additional information that you plan to use. Some of the pre-defined additional input parameters supported by the Risk Evaluation AdditionalInputs class include:

- AR_RF_LOCALE_ID

  Specifies the locale that RiskMinder will use while returning the messages back to your calling application.

- AR_RF_CALLER_ID

  Specifies the transaction identifier in your calling application. This is useful for end-to-end tracking of transactions.

To implement custom risk evaluation parameters:

1. Use the AdditionalInputs() method to obtain an object that implements the AdditionalInputs class.
2. Set the necessary inputs for the returned object by using the put() method. The syntax of the method is:

   public void put(java.lang.String *name*, java.lang.String *value*)

   If the additional input is *not* present with the given *name* (name of the input parameter), then one is created. Otherwise, it is overwritten with the new *value* (value of the input parameter). It is recommended that you do not use large strings either for *name* or for *value*.

   Note: The *name* and *value* parameters must *not* contain = *and* the newline character (\n). The API behavior is undefined if *name* and *value* contain any of these characters.

Chapter 4: Managing Users

Creating a new user in RiskMinder is a one-time operation, performed only when a new user is to be added to RiskMinder database. Typically, this is an existing user of your application accessing RiskMinder for the first time.

In previous releases, the Issuance Java API provided a programmable interface, which could be used by Java clients (such as Java Servlets and JSP pages) to send Issuance-related requests to RiskMinder Server. In this release (3.1.01), the Issuance API (**Issuance**) has been deprecated. Now, you must use the User Management Web service (**ArcotUserRegistrySvc**) for the purpose.

Book: Refer to "Creating Users" in "Managing Users " in the *CA RiskMinder Administration Guide* for more information on how to create users.

Chapter 5: 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 15)) 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.

- AuthMinder Plugin store: The AuthMinder plugin store is automatically created on the end user’s device when the CA ArcotID OTP Client is installed on the end user’s device. Among the storage options listed in this section, the plugin store is the most persistent storage option. 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\xE2\x80\x99s 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 45) and Device ID (see page 44) 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>*\ Arcot Systems**sdk**\ **devicedna**\ **Solaris:** *<install_location>*/arcot/ **sdk/devicedna/**	riskminder-client.j s	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 47)
- Step 2: Initialize Device ID and DeviceDNA Collection (see page 48)
- Step 3: Collect the Device ID and DeviceDNA (see page 50)
- Step 4: Collect the IP Address (see page 50)

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 device.

Note: See "Enrollment Workflows" (see page 15) 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_location>\Arcot Systems\sdk\`
   - **On UNIX-Based Platforms**
     `<install_location>/arcot/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 "Sample Code Reference" (see page 51) 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 "Understanding the APIs for Retrieving DeviceDNA in the Sample Code" (see page 52) 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 Understanding the APIs for Retrieving DeviceDNA in the Sample Code (see page 52) 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:
<*RISKMINDER_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-3.1.01-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" (see page 57) 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 55) 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 earlier 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: ■ plugin ■ localstorage ■ cookie ■ 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 the table in Understanding the APIs for Retrieving DeviceDNA in the Sample Code (see page 52) 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 the Browser (HTTP) cookie store.

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 6: Performing Risk Evaluation

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

The Risk Evaluation Java API provides a programmable interface, which can be used by Java clients (such as Java Servlets and JSP pages) to send risk evaluation-related requests to RiskMinder Server. This API creates a request message that is sent to RiskMinder Server, receives the response, and packages it as return structures to be read by the client.

This section provides an overview of how to use the Java API to perform risk evaluation and the methods and interfaces it implements. A description of the API interfaces and methods used to list and delete associations in the RiskMinder database is also provided. Each risk evaluation-related and association management task description is followed by a sample code snippet that you can use in your code to perform the task. The section covers following topics:

- Evaluating Risks and Performing Post-Evaluation (see page 58)
- Managing Associations (see page 62)

Evaluating Risks and Performing Post-Evaluation

To evaluate the risk associated with a transaction and to perform the subsequent post-evaluation, you need to use the RiskXActionAPI interface (in the com.arcot.riskfortAPI package. This interface represents the client-side interface to RiskMinder Server's risk evaluation functionality and exposes the supported API for risk evaluation workflows.

To evaluate risk of a transaction and perform post-evaluation tasks:

1. Ensure that you have initialized the Risk Evaluation API by using the RiskFactory class.

   See "Initializing the Risk Evaluation API" (see page 36) in "Before You Begin" (see page 33) for more information.

2. If required, prepare the additional inputs for the transaction, by using the AdditionalInputs class in the com.arcot.riskfortAPI.AdditionalInputs package.

   See "Preparing Additional Inputs" (see page 39) in "Before You Begin" (see page 33) for more information.

3. Use the RiskFactory.getRiskXActionAPI() static method to obtain an object that implements the RiskXActionAPI interface.

   This method returns the RiskXActionAPI object created as a part of the RiskFactory API initialization.

4. Use the RiskXActionAPI.evaluateRisk() method to obtain an object of the RiskAssessment class.

   This method requires the following input parameters:

   a. Define and set the CallerID string variable, which will be used by your application for tracking purposes across calls.

   b. Invoke the buildDeviceSignature() method by using a DeviceContext object to build the JSON signature that you built, as discussed in "Collecting Device ID and DeviceDNA" (see page 43).

   c. Create a LocationContext object, and then invoke the LocationContext object's setIpAddress() method to set the IP address to that of the user's system.

   d. Create a UserContext object, and then use the UserContext object's setUserID() method to set a unique ID for the user.

   e. Create a TransactionContext object and, if required, set the necessary properties for the returned object by using the methods in this class (setAction() and setChannel()).

   f. If you are using extra information, then use the AdditionalInputs object that you created in Step 2. Set the necessary properties for the returned object by using the AdditionalInputs.put(name,value) method.

      These are additional inputs in form of name-value pairs. For example:

MerchantID=id;MerchantCountry=country;MerchantName=name

5. Obtain the resulting RiskAssessment RiskAdvice object by using the RiskAssessment.getRiskAdvice() method.

   Important! If the advice is INCREASEAUTH, then your application *must* perform secondary authentication and pass the result of this authentication to RiskMinder by using the PostEvaluate() method.

6. Finally, use the RiskXActionAPI.postEvaluate() method to determine the final outcome of the transaction and return a PostEvaluateResponse object.

   The postEvaluate() method updates the Device Signature information for the user, if it changed and creates or updates user-device associations, if required.

   You will need to pass the result of secondary authentication to the method, if you performed any.

7. You can query the resulting PostEvaluateResponse object by using the PostEvaluateResponse.isAllowAdvised() method to:

   - Determine whether or not to permit the transaction to proceed.
   - Update the information related to the transaction in the RiskMinder database.

Handling Errors

Any errors that occurred during the execution of any of the Risk Evaluation API methods can result in one of the following two exceptions being thrown:

- RiskException is thrown if the error occurred at RiskMinder Server-end. In this case, the SDK only "relays" the information to your code. The RiskException object has getReasonCode, getResponseCode,and getTransactionId members that contain the reason code, error code, and transaction identifier (generated at RiskMinder Server-end) associated with the error that occurred.
- RFSDKException is, typically, thrown if the error occurred at the SDK-end. The RFSDKException object has getResponseCode and getTransactionId members that contain the error code and transaction identifier (generated at RiskMinder Server-end) associated with the error that occurred.

Sample Code for Risk Evaluation and Post Evaluation

Note: Refer to "Sample Code for Risk Evaluation and Post-Evaluation" (see page 78) in appendix for a detailed working code sample.

You can use the following sample code snippet to understand how to implement the risk evaluation and post-evaluation capability of RiskMinder in your application code.

```
public static void sampleCode() {
        String propertyLocation=
"/properties/riskfort.risk-evaluation.properties";
        try {
             RiskFactory.initialize(propertyLocation);
             RiskXActionAPI riskXActionAPI = RiskFactory.getRiskXActionAPI();
```

```
            String callerId;
            UserContext userContext = new UserContext();
            LocationContext locationContext = new LocationContext();
            DeviceContext deviceContext = new DeviceContext();
            TransactionContext transactionContext = new TransactionContext();
            AdditionalInputs additionalInputs = new AdditionalInputs();

            // string used by the calling application for tracking across
             // calls
            callerId="MyApplicationTrackingId";

            // Unique identifier for the user. In case of a Bank it may be
             // user's bank account number
            // It may be name of the user in some other case.
            userContext.setUserId("USER1");

            // IP address of the user's machine, typically, extracted from
             // the HTTP header
            locationContext.setIpAddress(InetAddress.getByName("10.150.1.1"));

            // JSON Signature comes from mfp_json.js, in this example the
             // signature is hard coded
            // for the sample use.
            String jsonSignature =
"{\"navigator\":{\"platform\":\"Win32\",\"appName\":\"Netscape\",\"appCodeName\":
\"Mozilla\",\"appVersion\":\"5.0 (Windows;
en-US)\",\"language\":\"en-US\",\"oscpu\":\"Windows NT
5.0\",\"vendor\":\"\",\"vendorSub\":\"\",\"product\":\"Gecko\",\"productSub\":\"2
0070312\",\"securityPolicy\":\"\",\"userAgent\":\"Mozilla/5.0 (Windows; U; Windows
NT 5.0; en-US; rv:1.8.0.11) Gecko/20070312
Firefox/1.5.0.11\",\"cookieEnabled\":true,\"onLine\":true},\"plugins\":[{\"name\"
:\"Adobe Acrobat Plugin\",\"version\":\"7.00\"},{\"name\":\"Macromedia
Director\",\"version\":\"10.1\"},{\"name\":\"Windows Media Player Plug-in Dynamic
Link Library\",\"version\":\"\"},{\"name\":\"Macromedia Shockwave
Flash\",\"version\":\"9.0\"},{\"name\":\"Java Virtual
Machine\",\"version\":\"1.6.0\"}],\"screen\":{\"availHeight\":690,\"availWidth\":
1024,\"colorDepth\":32,\"height\":768,\"pixelDepth\":32,\"width\":1024},\"extra\"
:{\"javascript_ver\":\"1.6\",\"timezone\":-330}}";
            deviceContext.buildDeviceSignature(jsonSignature,null,null);
            String
userDeviceId="GPXp+4e0hzzxzh6YLlPZqKgXCGbBXB8E0ghZnFXHq8o3HLRaww6c4g==";
            // The device id collected from the user machine
            deviceContext.setDeviceID("HTTP_COOKIE", userDeviceId);

            // Providing the addition inputs.
            additionalInputs.put("MerchantID","id") ;
            additionalInputs.put("MerchantCountry","country") ;
            additionalInputs.put("MerchantName","name") ;
```

```
             transactionContext.setAction("Login");
             RiskAssessment riskAssessment=null;
             riskAssessment = riskXActionAPI.evaluateRisk(callerId , deviceContext,
locationContext , userContext, transactionContext, additionalInputs);
             boolean secondaryAuthenticationStatus = true;
             String associationName = "USER1inHomePC";

             if
(riskAssessment.getRiskAdvice().equals(RiskAssessment.RISK_ADVICE_INCREASEAUTH))
{
                         // then you may ask for secondary authentication
                         //if( secondaryAuthentication succeeded )
                         //          secondaryAuthenticationStatus = true;
                         //else
                         //          secondaryAuthenticationStatus = false
             }

             PostEvaluateResponse postEvaluateResponse =
                         riskXActionAPI.postEvaluate(callerId, riskAssessment,

secondaryAuthenticationStatus, associationName);
             if( postEvaluateResponse.isAllowAdvised() ) {
                         //Allow the transaction to be completed
             }
             else {
                             //Deny and terminate the transaction
             }

             } catch (IOException e) {
            //Looks like the property file location is not valid
                e.printStackTrace();
             } catch (RiskException e) {
                //One of the RiskFort API calls broke
                e.printStackTrace();
             }
}
```

Managing 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 devices. For example, a user can use a work and a home computer to access your application. Similarly, you can bind a single device to more than one users. 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.

Association management includes:

- Listing Associations (see page 63)
- Deleting Associations (see page 65)

Listing Associations

To list all the stored associations for a specified user:

1. Ensure that you have initialized the Risk Evaluation API by using the RiskFactory class.

   See "Initializing the Risk Evaluation API" (see page 36) in "Before You Begin" (see page 33) for more information on this.

2. If required, prepare the additional inputs for the transaction, by using the AdditionalInputs class in the com.arcot.riskfortAPI.AdditionalInputs package.

   See "Preparing Additional Inputs" (see page 39) in "Before You Begin" (see page 33) for more information on this.

3. Use the RiskFactory.getRiskXActionAPI() static method to obtain an object that implements the RiskXActionAPI interface.

   This method returns the RiskXActionAPI object created as a part of the RiskFactory API initialization.

4. Define and set the CallerID string variable, which will be used by your application for tracking purposes across calls.

5. Create a UserContext object, and then use the UserContext object's setUserID() method to set a unique ID for the user.

6. Set the necessary properties for the returned object.

   For example, you can set the user ID by calling the UserContext.setUserId() method.

7. Call the RiskXActionAPI.listAssociations() method to create a ListAssociationResponse object.

   The following code snippet shows the usage of the method to list all existing associations.

   ```
   public ListAssociationResponse listAssociations(java.lang.String callerId,
          UserContext userContext,
          AdditionalInputs additionalInputs)
          throws RFSDKException, RFSDKException
   ```

   The ListAssociationResponse.getAllAssociations() method returns an array of all known associations for the specified user.

Handling Errors

Any errors that occurred during the execution of any of the Risk Evaluation API methods can result in one of the following two exceptions being thrown:

- RiskException is thrown if the error occurred at RiskMinder Server-end. In this case, the SDK simply "relays" the information to your code. The RiskException object has getReasonCode, getResponseCode,and getTransactionId members that contain the reason code, error code, and transaction identifier (generated at RiskMinder Server-end) associated with the error that occurred.
- RFSDKException is, typically, thrown if the error occurred at the SDK-end. The RFSDKException object has getResponseCode and getTransactionId members that contain the error code and transaction identifier (generated at RiskMinder Server-end) associated with the error that occurred.

Deleting Associations

To delete the specified user-device association for a user:

1. Ensure that you have initialized the Risk Evaluation API by using the RiskFactory class.

   See "Initializing the Risk Evaluation API" (see page 36) in "Performing Risk Evaluation" (see page 57) for more information on this.

2. If required, prepare the additional inputs for the transaction, by using the AdditionalInputs class in the com.arcot.riskfortAPI.AdditionalInputs package.

   See "Preparing Additional Inputs" (see page 39) in "Before You Begin" (see page 33) for more information on this.

3. Use the RiskXActionAPI.evaluateRisk() method to obtain an object of the RiskAssessment class.Use the RiskFactory.getRiskXActionAPI() static method to obtain an object that implements the RiskXActionAPI interface.

   This method returns the RiskXActionAPI object created as a part of the RiskFactory API initialization.

4. Define and set the CallerID string variable, which will be used by your application for tracking purposes across calls.

5. Create a UserContext object, and then use the UserContext object's setUserID() method to set a unique ID for the user.

6. Set the necessary properties for the returned object.

   For example, you can set the user ID by calling the UserContext.setUserId() method.

7. Obtain a ListAssociationsResponse object by invoking RiskXActionAPI.listAssociations. Then invoke ListAssociationResponse's getAllAssociations() method. This method returns an array of type UserDeviceAssociation. You can use UserDeviceAssociation.getAssociationName() for each UserDeviceAssociation object to get the name of the association.

8. Call the RiskXActionAPI.deleteAssociation() method to delete the association.

   The following code snippet shows the usage of the method to delete user-device associations.

   ```
   public DeleteAssociationResponse deleteAssociation(java.lang.String callerId,
          UserContext userContext,
          java.lang.String associationName,
          AdditionalInputs additionalInputs)
          throws RiskException, RFSDKException
   ```

   The DeleteAssociationResponse.DeleteAssociationResponse() method deletes the association for the user that you specified.

Handling Errors

Any errors that occurred during the execution of any of the Risk Evaluation API methods can result in one of the following two exceptions being thrown:

- RiskException is thrown if the error occurred at RiskMinder Server-end. In this case, the SDK simply "relays" the information to your code. The RiskException object has getReasonCode, getResponseCode,and getTransactionId members that contain the reason code, error code, and transaction identifier (generated at RiskMinder Server-end) associated with the error that occurred.
- RFSDKException is, typically, thrown if the error occurred at the SDK-end. The RFSDKException object has getResponseCode and getTransactionId members that contain the error code and transaction identifier (generated at RiskMinder Server-end) associated with the error that occurred.

Appendix A: Additional SDK Configurations

This appendix discusses the following miscellaneous topics:

Note: Case Management Queuing Server does not communicate directly with the Java SDKs. Therefore, you do not need to configure SSL communication between SDKs and Case Management Queuing Server.

Configuring a Backup RiskMinder Server Instance

RiskMinder enables you to configure the Java SDK to communicate with a primary RiskFort Server instance at any given time. However, you can configure it to fail over to a backup instance if the primary instance fails. To do so, you must edit the SDK properties file (riskfort.risk-evaluation.properties). In case the backup is unavailable for some reason, then the JDK fails over back to the primary instance.

By default, this properties file provides the entries to configure *one* RiskFort Server instance. These entries are appended with 1, which indicates that only one instance is configured. To configure the backup instance, you must duplicate these entries and append the instance number (2) accordingly.

To configure the backup server instance:

1. Ensure that you have the required instances of RiskFort up and running.

   Book: Depending on your deployment - Single-System or Distributed - see "Deploying RiskFort On a Single System" or "Deploying RiskFort on a Distributed System", respectively, in *CA RiskMinder Installation and Deployment Guide* for detailed information on how to bring up instances of RiskFort.

2. Navigate to the following location:

   ■ **On Microsoft Windows:**
   `<install_location>\Arcot Systems\sdk\java\properties\`

   ■ **On UNIX-Based Platforms:**
   `<install_location>/arcot/sdk/java/properties/`

3. Open riskfort.risk-evaluation.properties by using an editor of your choice.

4. Ensure that the value of HOST.1 is set to the host name or the IP address of the primary server instance.

5. Also ensure that the value of PORT.1 parameter is set to the port number on which the **RiskFort Native** or the **Transaction Web Service** protocol is listening.

6. Add the following *un-commented* entry:
   ```
   HOST.2=
   PORT.2=
   ```

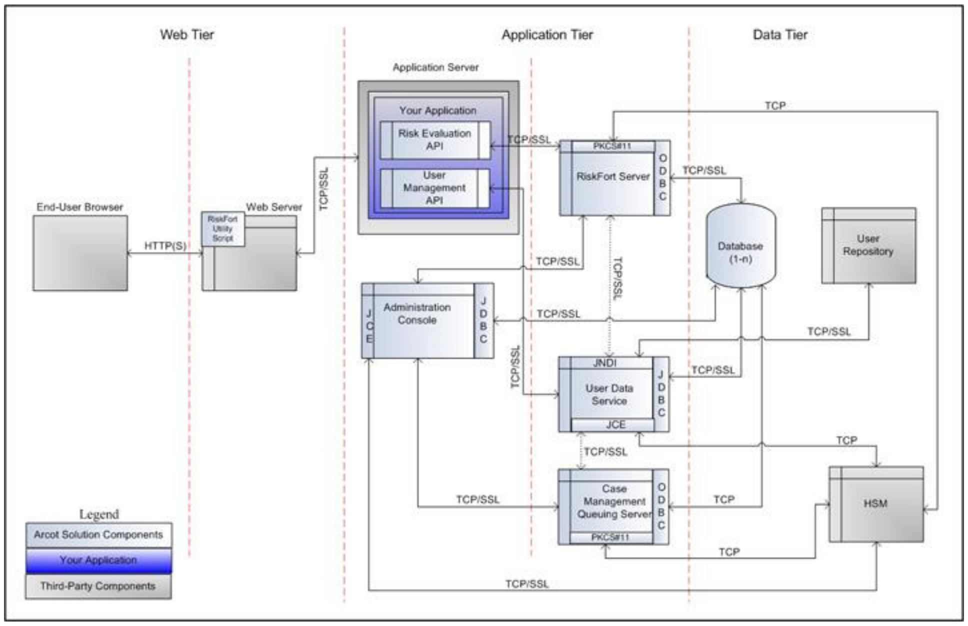
7. Set the value of the corresponding TRANSPORT.<*n*> parameter to the required communication mode. By default, it is set to TCP.

   See "SSL Communication Between RiskMinder Components" (see page 69) if you want to change the communication mode to SSL.

8. Save the changes and close the file.

RM_3.1--SSL Communication Between RiskMinder Components

In addition to supporting TCP-based communication between RiskMinder Server and the SDKs, RiskMinder also 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.



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.

RM_3.1--Setting Up SSL Communication Between Java SDK and RiskMinder Server

To enable RiskMinder Java SDK for SSL communication, you must first configure your client that accesses the SDK for SSL communication, then configure the **Native (SSL)** protocol by using Administration Console.

- Configuring One-Way SSL (see page 71)
- Configuring Two-Way SSL (see page 73)

Configuring One-Way SSL

To set up one-way SSL between the Risk Evaluation SDK and RiskFort Server, you must first configure the RiskFort **Native (SSL)** protocol by using Administration Console and then configure the riskfort.risk-evaluation.properties file.

To configure one-way SSL between Java SDK and RiskFort Server:

1. Ensure that you are logged in as the MA.
2. Activate the **Services and Server Configurations** tab in the main menu.
3. Ensure that the **RiskFort** tab 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.
6. In the **List of Protocols** section, click the **Native (SSL)** protocol link to display the page for configuring the protocol.
7. Configure the following fields:
   - Ensure that the **Protocol Status** is **Enabled**.

     If not, then select the **Change Protocol Status** option and then from the **Action** list, select **Enable**.
   - 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 RiskFort 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 RiskFort Server private key.
8. Click the **Save** button.
9. Restart RiskFort 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.
10. Navigate to the following location:
    - **On Microsoft Windows:**
      `<`*`install_location`*`>\Arcot Systems\sdk\java\`**`properties`**`\`
    - **On UNIX-Based Platforms:**
      `<`*`install_location`*`>/arcot/sdk/java/`**`properties`**`/`

11. Open the riskfort.risk-evaluation.properties file in an editor window of your choice.

    Book: Refer to appendix, "Configuration Files and Options" in *CA RiskMinder Installation and Deployment Guide* for more information on the riskfort.risk-evaluation.properties file.

    a. Set the following parameters:

       ■ TRANSPORT_TYPE= SSL (By default, this parameter is set to TCP.)

       ■ CA_CERT_FILE= <*absolute_path_to_Server_root_certificate_in_PEM_format*>

         For example, you can specify one of the following:

         CA_CERT_FILE=<*install_location*>/certs/<ca_cert>.pem

         CA_CERT_FILE=<*install_location*>\\certs\\<ca_cert>.pem

         For example, you can specify CA_CERT_FILE= <*install_location*>/certs/<ca_cert>.pem.

         Important! In the absolute path that you specify, ensure that you use \\ or / instead of \. This is because the change might not work, if you use the conventional \ that is used in Microsoft Windows for specifying paths.

       ■ CLIENT_P12_FILE=cliencert.p12

       ■ CLIENT_P12_PASSWORD=******

    b. Save the changes and close the file.

12. Restart the application server where your Java SDK is deployed.

Configuring Two-Way SSL

To set up two-way SSL between the Risk Evaluation SDK and RiskMinder Server, you must first upload the root certificates for the CAs trusted by RiskMinder, then configure the RiskFort **Native (SSL)** protocol by using Administration Console, and finally configure the riskfort.risk-evaluation.properties file.

To configure two-way SSL between Java SDK and RiskMinder Server:

1. Enable the application server where Java SDKs are deployed for SSL communication.

   Refer to your application server vendor documentation for detailed information.
2. Log in to Administration Console using a Master Administrator account.
3. Activate the **Services and Server Configurations** tab in the main menu.
4. Ensure that the **RiskFort** tab is active.
5. Under the **Instance Configuration** section, click the **Protocol Configuration** link to display the Protocol Configuration page.
6. Under **System Configuration**, click the **Trusted Certificate Authorities** link to display the RiskMinder Server Trusted Certificate Authorities page.
7. Set the following information on the page:
   - In the **Name** field, enter the name for the SSL trust store.
   - Click the **Browse** button adjacent to the first **Root CAs** field and navigate to and select the root certificate of the application server where Java SDKs are deployed.
8. Click the **Save** button.
9. Under the **Instance Configuration** section, click the **Protocol Configuration** link to display the Protocol Configuration page.
10. Select the **Server Instance** for which you want to configure the SSL.
11. In the **List of Protocols** section, click the **Native (SSL)** protocol link to display the page for configuring the protocol.
12. Configure the following fields:
    - Ensure that the **Protocol Status** is **Enabled**.

      If not, then select the **Change Protocol Status** option and then from the **Action** list, select **Enable**.
    - 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 7.

13. Click the **Save** button.
14. 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.
15. Navigate to the following location:
    - **On Microsoft Windows:** `<`*`install_location`*`>\Arcot Systems\sdk\java\`**`properties`**`\`
    - **On UNIX-Based Platforms:** `<`*`install_location`*`>/arcot/sdk/java/`**`properties`**`/`
16. Open the riskfort.risk-evaluation.properties file in an editor window of your choice.

    Book: Refer to appendix, "Configuration Files and Options" in *CA RiskMinder Installation and Deployment Guide* for more information on the riskfort.risk-evaluation.properties file.

    a. Set the following parameters:
       - TRANSPORT_TYPE= SSL (By default, this parameter is set to TCP.)
       - CA_CERT_FILE= <*absolute_path_to_Server_root_certificate_in_PEM_format*>

       For example, you can specify one of the following:
       - CA_CERT_FILE=<*install_location*>/certs/<ca_cert>.pem
       - CA_CERT_FILE=<*install_location*>\\certs\\<ca_cert>.pem

       For example, you can specify CA_CERT_FILE= <*install_location*>/certs/<ca_cert>.pem.

       Important! In the absolute path that you specify, ensure that you use \\ or / instead of \. This is because the change might not work, if you use the conventional \ that is used in Microsoft Windows for specifying paths.

    b. Save the changes and close the file.
17. Restart the application server where your Java SDK is deployed.
18. Verify that RiskMinder Server is enabled for SSL communication by performing the following steps:

    a. Navigate to the following location:

b. Open the arcotriskfortstartup.log file in a text editor.

c. Check for the following line:

   ```
   Started listener for [RiskFort Native (SSL)] [7681] [SSL]
   [RiskFort]
   ```

   If you located this line, then two-way SSL was set successfully.

d. Close the file.

Appendix B: Sample Code

This appendix provides the sample codes that you can run to test the Risk evaluation and post-evaluation (Sample Code for Risk Evaluation and Post-Evaluation (see page 78)) functionality of RiskMinder.

Important! Before you run this code, ensure that you have completed all the tasks in "Before You Begin" (see page 33).

Sample Code for Risk Evaluation and Post-Evaluation

You can plug the following sample code snippet in to your application code to test the risk evaluation and post-evaluation functionality of RiskFort.

```
/* Packages to be imported for RiskFort Transaction API */

import java.io.IOException;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Map;
import java.util.HashMap;

import com.arcot.riskfortAPI.DeviceContext;
import com.arcot.riskfortAPI.LocationContext;
import com.arcot.riskfortAPI.PostEvaluateResponse;
import com.arcot.riskfortAPI.RiskAssessment;
import com.arcot.riskfortAPI.RiskException;
import com.arcot.riskfortAPI.RFSDKException;
import com.arcot.riskfortAPI.RiskFactory;
import com.arcot.riskfortAPI.RiskXActionAPI;
import com.arcot.riskfortAPI.TransactionContext;
import com.arcot.riskfortAPI.UserContext;
import com.arcot.riskfortAPI.AdditionalInputs;

public class Assess_Risk {

// In this example values are hard coded for sample use.
public static void main(String[] args) {
  /*
  initialize:
    Initializes API object from the input property file.
    initialize() should be called only once at the application startup.

    Following are the fields and format of the property file.
      HOST.1=RiskFort server IP address
      PORT.1=RiskFort server port number
      TRANSPORT_TYPE = Connection Type. Possible values are TLS/TCP
      CA_CERT_FILE = Required if TRANSPORT_TYPE = TLS: CA certificate file. server
CA certificate (in PEM format) file path.

  public static synchronized void initialize(String propertyLocation)
    throws IOException, RiskException;

  Parameters:
      propertyLocation - Represents the location to be passed as a parameter with
respect to the class path.
      If <code>null</code> is passed, it will take default location is
properties/riskfort.risk-evaluation.properties.
  Returns:
```

```
  Throws:
    RiskException - If request fails for any reason.
    IOException
  */


  // Sample code to initialize the API object from the input property file.
  // Create a RiskXActionAPI object.
  RiskXActionAPI api = null;

  String propertiesFileLocation =
"/properties/riskfort.risk-evaluation.properties";

  try {
      System.out.println("Initializing RiskFort API using " +
propertiesFileLocation);

    // Initializes RiskXActionAPI object from the input property file.
      RiskFactory.initialize(propertiesFileLocation);
      //  Get RiskXActionAPI object that has been initialized earlier.
      api = RiskFactory.getRiskXActionAPI();

      System.out.println("RiskFort API initialized.");
  }
  catch (IOException e) {
    // Take suitable action.

  }
  catch (RFSDKException e) {

    /* The following methods on UserRepositoryException object can be used to get the
error codes and error messages as follows:
    * String code = e.getErrorCode();
    * String message = e.getMessage();
    */
    System.out.println("Exception during initialize.");
    /* System.out.println("Error code: " + e.getErrorCode()); */
    System.out.println("Error message: " + e.getMessage());

    /* The following error codes are returned by the API. */

    /* ERRCODE_INVALID_INPUT
    * This can be caused due to two reasons.
    * 1. Possible Reason:
    *    The transport type mentioned in the properties file is invalid.
    *    Possible Action:
    *   Mention the valid transport type in the "riskfort.risk-evaluation.properties"
file.
```

```
    * 2. Possible Reason:
    *    If TRANSPORT_TYPE=TLS in the properties file, the primary key file used for
TLS was not found.
    *     Possible Action:
    *      Check for the correctness of the path to the primary key file.
    */

    /* ERRCODE_RISKXACTIONAPI_ALREADY_INITIALIZED
    * Possible Reason:
    *     The API being initialized has already been initialized.
    * Possible Action:
    *    Get the RiskXActionAPI object and continue with the transaction.
    */

    /* ERROR_CONF_INVALID_POOL
    * Possible Reason:
    *  Inability to create a pool of live connections to RiskFort Server.
    * Possible Action:
    *
    */

    /* ERROR_CONF_NOT_AVAILABLE
    * This can be caused due to two reasons.
    * 1. Possible Reason:
    *     The properties file could not be read.
    *     Possible Action:
    *      Check for the correctness of the path to the properties file.
    * 2. Possible Reason:
    *     The Root CA for the server certificate is invalid.
    *     Possible Action:
    *      Get a valid server certificate.
    */
  }


  /*
  evaluateRisk:
    Evaluates risk associated with the transaction and returns an advice accordingly.
It also provides a new DeviceId.
    Actions to be taken by the calling application are:
      1. The output DeviceId should be stored on the user machine in some form. Most
common way is to store it as a HTTP cookie. Nevertheless, storing it as a HTTP cookie
has the risk of it being deleted when user deletes all cookies on the machine.
      2. Retrieve the DeviceId from user machine and set it using setDeviceID.
      3. If RiskAdvice is equal to INCREASEAUTH, perform second authentication and
pass the result of the second authentication to RiskFort using PostEvaluate.

  public RiskAssessment evaluateRisk(java.lang.String callerId,
        DeviceContext deviceContext,
        LocationContext locationContext,
        UserContext userContext,
        TransactionContext transactionContext,
        AdditionalInputs additionalInputs)
      throws RiskException

  Parameters:
    callerId - An identifier as decided by the application calling the API for it's
own tracking.
    deviceContext - Device contextual information.
    locationContext - Location contextual information(IP address).
    userContext - User contextual information.
    transactionContext - Transaction contextual information.
    additionalInputs - Additional inputs that may be needed for different operations
  Returns:
    RiskAssessment - Contains RiskAdvice, a new DeviceId which should be placed on
the user machine, a RiskScore and other transaction related information.
  Throws:
    RiskException - If request fails for any reason.
  */

  // Sample code to evaluate risk associated with the transaction.

  RiskAssessment riskAssessment = null;
  System.out.println("The following information is used to assess the risk associated
with the transaction.");

  // Build the context to be used for risk evaluation
  String callerId = "MyApplicationTrackingId"; // string used by the calling
application for tracking across calls.

  // input user related information.
  UserContext userContext = new UserContext();

  //Unique identifier for the user. For example, in case of a bank it may be user's
bank account number.
  userContext.setUserId("TestUser");
  userContext.setOrg("DEFAULTORG");
  System.out.println("Username: " + userContext.getUserID());
  System.out.println("Organization Name:" + userContext.getOrg());

  // input device related information
  DeviceContext deviceContext = new DeviceContext();

  // JSON Signature comes from json.js.
  String jsonSignature =
"{\"navigator\":{\"platform\":\"Win32\",\"appName\":\"Netscape\",\"appCodeName\":
\"Mozilla\",\"appVersion\":\"5.0 (Windows;
en-US)\",\"language\":\"en-US\",\"oscpu\":\"Windows NT
5.0\",\"vendor\":\"\",\"vendorSub\":\"\",\"product\":\"Gecko\",\"productSub\":\"2
0070312\",\"securityPolicy\":\"\",\"userAgent\":\"Mozilla/5.0 (Windows; U; Windows
NT 5.0; en-US; rv:1.8.0.11) Gecko/20070312
Firefox/1.5.0.11\",\"cookieEnabled\":true,\"onLine\":true},\"plugins\":[{\"name\"
:\"Adobe Acrobat Plugin\",\"version\":\"7.00\"},{\"name\":\"Macromedia
Director\",\"version\":\"10.1\"},{\"name\":\"Windows Media Player Plug-in Dynamic
Link Library\",\"version\":\"\"},{\"name\":\"Macromedia Shockwave
Flash\",\"version\":\"9.0\"},{\"name\":\"Java Virtual
Machine\",\"version\":\"1.6.0\"}],\"screen\":{\"availHeight\":690,\"availWidth\":
1024,\"colorDepth\":32,\"height\":768,\"pixelDepth\":32,\"width\":1024},\"extra\"
:{\"javascript_ver\":\"1.6\",\"timezone\":-330}}";

  deviceContext.buildDeviceSignature(jsonSignature, null, null);
  System.out.println("Device Signature: " + deviceContext.getDeviceSignature());

  // Set the device id
  String idType = "HTTP_COOKIE";
  /* During the first call to evaluateRisk, deviceId=null as the device is not
recognized by RiskFort server.
   * RiskFort server then sets a deviceId in a cookie on the user's machine which is
passed to RiskFort server during subsequent transactions.
  */
  String deviceId = null;
  deviceContext.setDeviceID(idType, deviceId);

  /* For each transaction, either the deviceId or the aggregatorID but not both should
to be set.

deviceContext.setAggregatorID("LcPywTghrtyed6KDuRcMbWiFFTYR2oFThfdDOtBKqKcdEXsH9d
FIFfrr/dsfdud");

    System.out.println("Aggregator ID: " + deviceContext.getAggregatorID());
  */

  // input location related information.
  LocationContext locationContext = new LocationContext();

  InetAddress ipAddress = null;
  // IP address of the user's machine, typically extracted from the HTTP header.
  try {
      ipAddress = InetAddress.getByName("127.0.0.1");
  } catch (UnknownHostException e) {
      // Take suitable action.
  }
  locationContext.setIpAddress(ipAddress);
  System.out.println("Ip address: " + locationContext.getIPAddress());

  // input transaction related information.
  TransactionContext transactionContext = new TransactionContext();

  transactionContext.setAction("action");
  transactionContext.setChannel("DEFAULT");

  /*For each transaction, either the extensible elements must be set in the transaction
context or the additional inputs must be set.

transactionContext.setExtensibleElements("MerchantID=id;MerchantCountry=country;M
erchantName=name");
  */

  HashMap<String,String> additionalInputs = new HashMap<String,String>();

  String extName1 = "MerchantID";
  String extValue1 = "id";
  String extName2 = "MerchantCountry";
  String extValue2 = "country";
  String extName3 = "MerchantName";
  String extValue3 = "name";
  //Below attributes has to be supplied when organization is configured for using
accounts.(not required otherwise)
  //In this case, USERID attribute inside userContext will be treated as account so
below information is required
  // to identify the actual use to which this account belongs.
  String extName4 = "ACCOUNTTYPE";
  String extValue4 = "accType";
  String extName5 = "PARENTUSERID"; //If organization is configured for using accounts
and implicit user creation is enabled.
  String extValue5 = "parentid";
  if(extName1 != null && extName1 != "" )
    additionalInputs.put(extName1, extValue1);
  if(extName2 != null && extName2 != "" )
    additionalInputs.put(extName2, extValue2);
  if(extName3 != null && extName3 != "" )
    additionalInputs.put(extName3, extValue3);
  if(extName4 != null && extName4 != "" )
    additionalInputs.put(extName4, extValue4);
  if(extName5 != null && extName5 != "" )
    additionalInputs.put(extName5, extValue5);
  try {
    System.out.println("evaluateRisk called.");
    // Call the API to evaluate the risk associated with the transaction.
    riskAssessment  = api.evaluateRisk(callerId, deviceContext, locationContext,
userContext, transactionContext /*, additionalInputs */);
```

```
    System.out.println("evaluateRisk succeeded.");
    System.out.println("Device Id set on the user's machine: " +
riskAssessment.getOutputDeviceId());
  } catch (RFSDKException e) {
    /* The following methods on UserRepositoryException object can be used to get the
error codes and error messages as follows:
    * String code = e.getErrorCode();
    * String message = e.getMessage();
    */
    System.out.println("Exception in 'evaluateRisk'.");
    /*System.out.println("Error code: " + e.getErrorCode());*/
    System.out.println("Error message: " + e.getMessage());

    /* The following error codes are returned by the API. */

    /* ERRCODE_INVALID_PACKET_FROM_SERVER
    * Possible Reason:
    *   Invalid Packet type received from the server.
    * Possible Action:
    *   Report transaction failure and ask for a retry.
    */

    /* ERRCODE_PARSING_DATA
    * Possible Reason:
    *   Error in parsing the xml from the server.
    * Possible Action:
    *   Report transaction failure and ask for a retry.
    */
  }
  catch (RiskException e) {
    /* The following methods on UserRepositoryException object can be used to get the
error codes and error messages as follows:
    * String code = e.getErrorCode();
    * String message = e.getMessage();
    */
    System.out.println("Exception in 'evaluateRisk'.");
    /*System.out.println("Error code: " + e.getErrorCode());*/
    System.out.println("Error message: " + e.getMessage());

    /* The following error codes are returned by the API. */

    /* ERRCODE_INVALID_PACKET_FROM_SERVER
    * Possible Reason:
    *   Invalid Packet type received from the server.
    * Possible Action:
    *   Report transaction failure and ask for a retry.
    */
    /* ERRCODE_PARSING_DATA
    * Possible Reason:
    *   Error in parsing the xml from the server.
    * Possible Action:
    *   Report transaction failure and ask for a retry.
    */

  }


  /*
  postEvaluate:
    Helps to make the final decision on the transaction based on the output of
evaluateRisk and any second authentication that may have been performed by the calling
application.
    Also takes care of updating information in the RiskFort system as needed.

  public PostEvaluateResponse postEvaluate(java.lang.String callerId,
       RiskAssessment riskAssessment,
       boolean secondaryAuthenticationStatus,
       java.lang.String associationName,
       AdditionalInputs additionalInputs)
       throws RiskException;



  Parameters:
    callerId - An identifier as decided by the application calling the API for it's
own tracking.
    riskAssessment - The output from evaluateRisk.
    secondaryAuthenticationStatus - Result of second authentication.
        Pass "true" if secondary authentication succeeded, "false" otherwise.
        If evaluateRisk returned an advice other than INCREASEAUTH (i.e. secondary
authentication was not asked for), pass "false".

    associationName - A value that user chose as the association name for the machine
from where the transaction has been carried out.
        User should be recommended not to choose association for shared machines, in
which case "null" can be passed.
    additionalInputs - Additional inputs that may be needed for different operations.
This has been kept for future use.
  Returns:
    PostEvaluateResponse - Indicates whether or not this transaction should be allowed
to continue. Can be checked using isAllowAdvised() which returns "true" if the
transaction should be allowed and "false" if it should be denied.

  Throws:
    RiskException - If request fails for any reason.
  */
  // Sample code to make the final decision on the transaction based on the output
of evaluateRisk and any second authentication that may have been performed by the
calling application.
  // Here the RiskAssessment object passed as input is the object returned by the call
to evaluateRisk()

  PostEvaluateResponse postEvalResponse = null;

  String associationName; // the association name for the machine from where the
transaction has been carried out

  boolean secondaryAuthenticationStatus; // Result of any second authentication that
may have been performed by the calling application.

  // Build the context to be used in the postEvaluate call.
  associationName = "testAssociationName";
  secondaryAuthenticationStatus = true;

 /*  Map */ additionalInputs = new HashMap<String,String>();

    //Below attributes has to be supplied when organization is configured for using
accounts.(not required otherwise)
    //In this case, USERID attribute inside userContext will be treated as account
so below information is required
    // to identify the actual use to which this account belongs. This has to be supplied
in postEvaluate as well.
    /*String */ extName1 = "ACCOUNTTYPE";
    /*String */ extValue1 = "accType";
    /*String */ extName2 = "PARENTUSERID"; //If organization is configured for using
accounts and implicit user creation is enabled.
    /*String */ extValue2 = "parentid";
    if(extName1 != null && extName1 != "" )
      additionalInputs.put(extName1, extValue1);
    if(extName2 != null && extName2 != "" )
      additionalInputs.put(extName2, extValue2);

  try {
    System.out.println("Calling postEvaluate with Secondary Authentication Status =
" + secondaryAuthenticationStatus );
    System.out.println("Association name passed: " + associationName);

    // Call the API to make the final decision based on evaluateRisk and second
Authentication.
    postEvalResponse = api.postEvaluate(callerId, riskAssessment,
secondaryAuthenticationStatus, associationName /*, additionalInputs*/ );

    System.out.println("postEvaluate succeeded.");
  } catch (RFSDKException e) {
    /* The following methods on UserRepositoryException object can be used to get the
error codes and error messages as follows:
    * String code = e.getErrorCode();
    * String message = e.getMessage();
    */
    System.out.println("Exception in 'postEvaluate'.");
    /* System.out.println("Error code: " + e.getErrorCode()); */
    System.out.println("Error message: " + e.getMessage());

    /* The following error codes are returned by the API. */

    /* ERRCODE_INVALID_PACKET_FROM_SERVER
    * Possible Reason:
    *    Invalid Packet type received from the server.
    * Possible Action:
    *    Report transaction failure and ask for a retry.
    */

    /* ERRCODE_PARSING_DATA
    * Possible Reason:
    *    Error in parsing the xml from the server.
    * Possible Action:
    *    Report transaction failure and ask for a retry.
    */
  }
  catch (RiskException e) {

    /* The following methods on UserRepositoryException object can be used to get the
error codes and error messages as follows:
    * String code = e.getErrorCode();
    * String message = e.getMessage();
    */
    System.out.println("Exception in 'postEvaluate'.");
    /* System.out.println("Error code: " + e.getErrorCode()); */
    System.out.println("Error message: " + e.getMessage());

    /* The following error codes are returned by the API. */

    /* ERRCODE_INVALID_PACKET_FROM_SERVER
    * Possible Reason:
    *    Invalid Packet type received from the server.
    * Possible Action:
    *    Report transaction failure and ask for a retry.
    */

    /* ERRCODE_PARSING_DATA
    * Possible Reason:
    *    Error in parsing the xml from the server.
    * Possible Action:
    *    Report transaction failure and ask for a retry.
    */
  }
  System.out.println("Risk Evaluation done.");
  }
}
```

To Compile on Microsoft Windows

To compile this test program, save it in a file called Assess_Risk.java. Make sure that the arcot-riskfort-evaluaterisk.jar and related jar files arcot_core.jar, arcot-riskfort-mfp.jar, bcprov-jdk14-131.jar, commons-lang-2.0.jar, commons-pool-1.4.jar are in the JAVA compiler's CLASSPATH. Then run the JAVA compiler.

The arcot-riskfort-evaluaterisk.jar file is usually present in the sdk\java\lib\arcot directory in the RiskMinder installation directory. If your Assess_Risk.java file is saved in \Program Files\Arcot Systems\sdk\java, then use the following command shown below. If your JAVA file is not in this directory, provide the full pathname to the sdk\java\lib directory in CLASSPATH.

```
> cd \program files\arcot systems\sdk\java

> javac -classpath
".;lib\arcot\arcot-riskfort-evaluaterisk.jar;lib\arcot\arcot_core.jar;lib\arcot\a
rcot-riskfort-mfp.jar;lib\external\bcprov-jdk14-131.jar;lib\external\commons-lang
-2.0.jar;lib\external\commons-pool-1.4.jar;%CLASSPATH%" Assess_Risk.java
```

This creates the output file Assess_Risk.class in the same directory as the JAVA file.

To Compile on UNIX-Based Platforms

To compile this test program, save it in a file called Assess_Risk.java. Make sure that the arcot-riskfort-evaluaterisk.jar and related jar files arcot_core.jar, arcot-riskfort-mfp.jar, bcprov-jdk14-131.jar, commons-lang-2.0.jar, commons-pool-1.4.jar are in the JAVA compiler's CLASSPATH. Then run the JAVA compiler.

The arcot-riskfort-evaluaterisk.jar file is usually present in the sdk/java/lib/**arcot** directory in the RiskMinder installation directory. If your Assess_Risk.java file is saved in /opt/arcot/sdk/**java**, use the following command shown below. If your JAVA file is not in this directory, provide the full pathname to the sdk/java/**lib** directory in CLASSPATH.

```
> cd /opt/arcot/sdk/java
> javac -classpath
".:./lib/arcot/arcot-riskfort-evaluaterisk.jar:./lib/arcot/arcot_core.jar:./lib/a
rcot/arcot-riskfort-mfp.jar:./lib/external/bcprov-jdk14-131.jar:./lib/external/co
mmons-lang-2.0.jar:./lib/external/commons-pool-1.4.jar:$CLASSPATH"
Assess_Risk.java
```

This creates output file Assess_Risk.class in the same directory as the JAVA file.

To Run on Microsoft Windows

Before you can run the test, the RiskMinder Service must be installed and started. To run the test, make sure that the SDK library is in the path and arcot-riskfort-evaluaterisk.jar is in the CLASSPATH then run the JAVA command as shown below.

To run the test, use the following commands:

```
> cd \program files\arcot systems\sdk\java

> java -classpath
".;lib\arcot\arcot-riskfort-evaluaterisk.jar;lib\arcot\arcot_core.jar;lib\arcot\a
rcot-riskfort-mfp.jar;lib\external\bcprov-jdk14-131.jarlib\external\commons-lang-
2.0.jarlib\external\commons-pool-1.4.jar;%CLASSPATH%" Assess_Risk
```

To Run on UNIX-Based Platforms

Before you can run the test, the RiskMinder Service must be installed and started. To run the test, make sure that the SDK library is in the path and arcot-riskfort-evaluaterisk.jar is in the CLASSPATH then run the JAVA command as shown below.

To run the test, use the following commands:

```
> cd /opt/arcot/sdk/java

> java -classpath
".:./lib/arcot/arcot-riskfort-evaluaterisk.jar:./lib/arcot/arcot_core.jar:./lib/a
rcot/arcot-riskfort-mfp.jar:./lib/external/bcprov-jdk14-131.jar:./lib/external/co
mmons-lang-2.0.jar:./lib/external/commons-pool-1.4.jar:$CLASSPATH" Assess_Risk
```

Expected Output

After running the provided sample code, you should see the following output:

```
Initializing RiskFort API using /riskfort.risk-evaluation.properties
RiskFort API initialized.
The following information is used to assess the risk associated with the transaction.
Username: TestUser
```

```
Organization Name:DEFAULTORG
Device Signature:
{"navigator":{"platform":"Win32","appName":"Netscape","appCodeName":"Mozilla","ap
pVersion":"5.0 (Windows; en-US)","language":"en-US","oscpu":"Windows NT
5.0","vendor":"","vendorSub":"","product":"Gecko","productSub":"20070312","securi
tyPolicy":"","userAgent":"Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US;
rv:1.8.0.11) Gecko/20070312
Firefox/1.5.0.11","cookieEnabled":true,"onLine":true},"plugins":[{"name":"Adobe
Acrobat Plugin","version":"7.00"},{"name":"Macromedia
Director","version":"10.1"},{"name":"Windows Media Player Plug-in Dynamic Link
Library","version":""},{"name":"Macromedia Shockwave
Flash","version":"9.0"},{"name":"Java Virtual
Machine","version":"1.6.0"}],"screen":{"availHeight":690,"availWidth":1024,"color
Depth":32,"height":768,"pixelDepth":32,"width":1024},"extra":{"javascript_ver":"1
.6","timezone":-330}}
Ip address: /127.0.0.1
evaluateRisk called.
evaluateRisk succeeded.
Device Id set on the user's machine:
qcd+Vq3NC6qAafCDbsFCBrup0y+z+ci8qjV5QmZI9GWuFGbbX9oIgw==
Calling postEvaluate with Secondary Authentication Status = true
Association name passed: testAssociationName
postEvaluate succeeded.
Risk Evaluation done.
```

Appendix C: Java API Reference

The RiskFort SDK constitutes a set of Java classes and methods that provide a way for your online application to programmatically integrate with RiskFort objects. The RiskFort Java SDK consists of the following components:

- The Risk Evaluation Java classes
- Javadoc information for the associated Java classes and methods

Note: The Sample Application shipped with RiskFort demonstrates the usage of the Java classes and methods.See appendix, "RiskFort Sample Application" (see page 133) for more information on RiskFort Sample Application.

Accessing the Javadoc HTML Documentation

You can use the Javadoc information provided with the RiskFort SDK along with this guide and other Java reference materials, to add RiskFort Risk Evaluation services to new or existing Java applications.

If you are updating an existing RiskFort application, then you must consult the Release Notes and Javadoc HTML documentation for deprecated Java APIs before making any changes.

You can access the latest Javadocs by installing RisKFort and copying the Javadocs from the docs directory. (You can then copy the Javadocs to another location on your development system.) Alternatively, you can also access the Javadocs directly from the Documentation directory in the RiskMinder installation package, without having to install RiskFort.

See the table in Risk Evaluation API (see page 92) for the installation locations of the RiskFort Javadocs.

Risk Evaluation API

The following table lists the files that are installed as a part of the Risk Evaluation SDK 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-3.1.01-risk-evaluation-sdk-javadocs.zip or Arcot-RiskFort-3.1.01-risk-evaluation-sdk-javadocs.tar.gz	Javadoc HTML documentation for the Java classes and methods provided with the Risk Evaluation Java API.
samples\java\\ (Microsoft Windows) samples/**java**/ (UNIX Platforms)	riskfort-3.1.01-sample-application.war	The demonstration that illustrates the use of Risk Evaluation APIs. See appendix, "RiskFort Sample Application" (see page 133) for more information on how to run and use the Sample Application.
	riskfort-3.1.01-sample-callouts.war	The demonstration that illustrates the use of the Callout feature. **Book:** See appendix, "Working with Sample Callouts" in *CA RiskMinder Administration Guide* for more information on how to run and use the Sample Callouts.
	arcot_core.jar	The proprietary Java Archive (JAR) file containing the set of shared components, toolkits, and services used to build the CA products.

Location	**File Name**	**Description**
sdk\java\lib\arcot\ (Microsoft Windows) sdk/java/lib/**arcot**/ (UNIX Platforms)	arcot-pool.jar	The Java Archive (JAR) file containing the classes and methods required for connection pooling between the RiskFort resource pack and User Data Service (UDS).
	arcot-riskfort-evaluaterisk.jar	The Java Archive (JAR) file containing the classes and methods associated with the Risk Evaluation API.
	arcot-riskfort-mfp.jar	The Java Archive (JAR) file containing the classes and methods associated with Device ID and DeviceDNA collection.
sdk\java\properties\ (Microsoft Windows) sdk/java/**properties**/ (UNIX Platforms)	log4j.properties.risk-evaluation	The properties file used by classes and methods associated with the Risk Evaluation API to specify the logging behavior.
	riskfort.risk-evaluation.properties	The properties file used by classes and methods associated with the Risk Evaluation API to read RiskFort Server information.
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.

Third-Party JARs Used by Risk Evaluation API

The Risk Evaluation API also uses the **third-party JARs** listed in the following table.

- **On Microsoft Windows**
  `<install_location>\Arcot Systems\sdk\java\lib\external\`
- **On UNIX-Based Platforms**
  `<install_location>/arcot/sdk/java/lib/external/`

File Name	Description
bcprov-jdk14-139.jar	The Bouncy Castle APIs for supporting cryptographic operations. *http://www.bouncycastle.org/latest_releases.html*
commons-lang-2.0.jar	The Apache Java utility packages that provide support for string manipulation methods, basic numerical methods, object reflection, creation and serialization, and system properties. *http://commons.apache.org/lang/*
commons-pool-1.4.jar	The Apache package for supporting object-pooling implementations. *http://commons.apache.org/pool/*

Appendix D: Exceptions and Error Codes

This appendix lists all exceptions and error codes thrown by the RiskMinder SDKs:

- RiskMinder SDK Exceptions (see page 95)
- RiskMinder Server Response Codes (see page 97)

RiskMinder SDK Exceptions

This section lists and describes the types of exceptions that you might see:

- Risk Evaluation Server Exceptions (see page 95)
- Risk Evaluation SDK Exceptions (see page 96)

Risk Evaluation Server Exceptions

The RiskMinder Risk Evaluation API throws the RiskException, if a Server operation failed. The getResponseCode() function of the RiskException class returns the string value corresponding to the type of error, the getTransactionId() method returns the Server-generated transaction ID for which the error occurred, and the getReasonCode() method returns the Server's TransactionID for the transaction, if one is provided.

The possible RiskException errors, their corresponding values, and descriptions are listed in the following table.

Response Code	**Value**	**Description**
INVALID_MESSAGE_TYPE	7021	The API identifier (message type) in the response sent from the Server is not valid. The packet might be corrupted.

Risk Evaluation SDK Exceptions

The RiskMinder Risk Evaluation SDK throws RFSDKException, if the incoming configuration information used by the SDK is not correct, and as a result, if a related operation might fail. This exception class also relays Server errors to your application.

The getResponseCode() function of the RFSDKException class returns the string value corresponding to the type of error. If the exception was generated at the Server-end, then the getTransactionId() method returns the Server-generated transaction ID for which the error occurred.

The possible RFSDKException errors, their corresponding values, and descriptions are listed in the following table.

Response Code	**Value**	**Description**
BUILDRESPONSE_ERROR	7016	The SDK encountered an error while interpreting the response XML from the Server.
CONFIGURATION_INVALID_POOL	7010	The connection to RiskMinder Server could not be established.
CONFIGURATION_NOT_AVAILABLE	7009	The configuration keys of properties that are used for configuring the API could not be read correctly from the properties file.
ENCODING_NOT_SUPPORTED	7014	The response that the SDK received from the Server contains encoding that is not supported.
ERROR_PARSING_DATA	7006	The SDK could not parse the response from the Server.
INTERNAL_ERROR	7002	The error was caused due to an internal SDK error.
INVALID_ADVICE_CODE	7019	The Advice code passed to postevaluate() is not valid.
INVALID_INPUT_PARAMETERS	7005	The value of the parameter passed to the API is invalid.
INVALID_MESSAGE_TYPE	7021	The API identifier (message type) in the response sent from the Server is not valid. The packet might be corrupted.
INVALID_PACKET_FROM_SERVER	7003	The Server packet is either malformed or is corrupted.
MISSING_INPUT_PARAMETERS	7013	One or more mandatory parameters needed by the API are missing.

Response Code	Value	Description
RISKXACTIONAPI_ALREADY_INITIALIZED	7008	The RiskXActionAPI interface has already been initialized by the Server.
RISKXACTIONAPI_INITIALIZATION_ FAILURE	7023	The RiskXActionAPI interface could not be successfully initialized.
RISKXACTIONAPI_NOT_INITIALIZED	7007	The RiskXActionAPI interface has not yet been initialized by the Server.
SUCCESS	7000	The operation was successful.
TCP_COMMUNICATION_ERROR	7011	A (TCP-based) communication error occurred between the SDK and the Server.
TCP_CONNECTION_ERROR	7012	The SDK could not establish a (TCP-based) connection to RiskMinder Server.
USER_RECORD_NOT_FOUND	7018	The specified user information was not found in the RiskMinder database.
XML_PARSE_ERROR	7020	One or more XML parameters are either incomplete or are missing.

RiskMinder Server Response Codes

The following table lists the 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 the value of this as 5, then you will get this error. **Solution:** Provide valid value for the parameter. See appendix, "Input Data Validations" (see page 111) for the supported parameter values.
	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 111) 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 111) for the supported parameter values.
	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 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 111) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
1050	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 111) for the supported parameter values.
	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 111) for the supported parameter values.
	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 the value of this as 5, then you will get this error. **Solution:** Provide valid value for the parameter. See appendix, "Input Data Validations" (see page 111) 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.

Response Code	Reason Code	Description	Possible Cause for Failure
1050	2057	One of the parameters used in the operation does not meet the formatting requirements.	**Possible Cause:** The parameter specified by ParameterKey has invalid format. **Solution:** Provide valid format for the parameter that is specified by ParameterKey.
	2061	Value of one of the parameters used in the operation is not allowed.	**Possible Cause:** The parameter specified by ParameterKey has invalid format. **Solution:** Provide 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.
	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 8000	The operation was successful.	NA.
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.

Response Code	Reason Code	Description	Possible Cause for Failure
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 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.
7502	0	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	0	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.

Response Code	Reason Code	Description	Possible Cause for Failure
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 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.
7664	8199	The Case Queue that you are trying to access does not have any cases.	**Possible Cause:** No cases were found in the specified Queue. **Solution:** Try to access the Queue again after some time.
7664	8200	There was an internal error in the Case Management operation.	**Possible Cause:** Unexpected internal error. **Solution:** Most likely cause might be Case Management Server or database failure. Check the Server transaction and database logs for details and ensure appropriate action is taken based on the Server logs.
7664	8201	No Queue is assigned to the Case administrator.	**Possible Cause:** No Queue assigned to the Case administrator. **Solution:** Ensure that the Case administrator is assigned a Queue before the operation.
7664	8202	There was an internal error in the Case Management operation.	**Possible Cause:** The Case was not successfully reassigned. **Solution:** Ensure that the Case is assigned to an administrator.

Response Code	Reason Code	Description	Possible Cause for Failure
7664	8203	There was an internal error in the Case Management operation.	**Possible Cause:** Unexpected internal error. **Solution:** Most likely cause might be Case Management Server or database failure. Check the Server transaction and database logs for details and ensure appropriate action is taken based on the Server logs.
7664	8204	The specified Queue was not found in the system.	**Possible Cause:** The Queue that you are trying t access does not exist. **Solution:** Ensure that you are using the correct Queue name and parameters for the operation.
7664	8205	There was an internal error in the Case Management operation.	**Possible Cause:** The Queue Schedule that you are trying to create could not be successfully created. **Solution:** Ensure that you are using the correct Schedule name and parameters for the operation.
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 valid rule for the parameter. See Appendix E, "Input Data Validations" (see page 111) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
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 invalid value. **Solution:** Provide 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.
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 valid value for the parameter. See appendix, "Input Data Validations" (see page 111) 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 111) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
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 111) 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 111) for the supported parameter values.
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 111) 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 111) 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 111) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
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 111) for the supported parameter values.
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:** Provide valid rule for the parameter. See appendix, "Input Data Validations" (see page 111) 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 valid format and value for the parameter that is specified. See appendix, "Input Data Validations" (see page 111) 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 valid format and value for the parameter. See appendix, "Input Data Validations" (see page 111) 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 valid format and value for the parameter. See appendix, "Input Data Validations" (see page 111) 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 valid format and value for the parameter. See appendix, "Input Data Validations" (see page 111) 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 valid format and value for the parameter. See appendix, "Input Data Validations" (see page 111) 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 valid format and value for the parameter. See appendix, "Input Data Validations" (see page 111) 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 valid format and value for the parameter. See appendix, "Input Data Validations" (see page 111) for the supported parameter values.
7668	8155	The EMAIL ID parameter contains prohibited characters.	**Possible Cause:** The EMAIL ID parameter has prohibited characters. **Solution:** Provide valid format and value for the parameter. See appendix, "Input Data Validations" (see page 111) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
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 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 correct ruleset information.
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.

Response Code	Reason Code	Description	Possible Cause for Failure
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.
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.

Response Code	Reason Code	Description	Possible Cause for Failure
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.
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.

Appendix E: 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.

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.
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

Attribute	Attribute ID	Validation Criteria
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.
		Can contain numbers, alphabets, underscore, and dot.
Start IP	START_IP	Is non-empty.

Attribute	Attribute ID	Validation Criteria
		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.

Attribute	Attribute ID	Validation Criteria
Combination Rule Match2	COMBINATION_RULE_ MATCH2	Is non-empty.
		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.

Attribute	Attribute ID	Validation Criteria
Connection Timeout	CONNECTION_TIMEOUT	Value is between 0 and 1000000.
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 F: 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 119)
- Startup Log Files (see page 119)
- Transaction Log Files (see page 122)
- Administration Console Log File (see page 125)
- UDS Log File (see page 126)

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 RiskMinder log file.) The typical logging configuration options that you can change in these files include:

- **Specifying log file name and path:** RiskMinder enables you to specify the directory for writing the log files and storing the backup log files.
- **Log file size:** The maximum number of bytes 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 lets you specify the configuration option to control the size of diagnostic logging files. This lets 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 128) for more information on the supported log levels.
- **Specifying time zone information:** RiskMinder enables you to either use the local time zone for time stamping the logged information or use GMT for the same.

Installation Log File

When you install RiskMinder, the installer records all the information that you supply during the installation and the actions (such as creating the directory structure and making registry entries) that it performs in the Arcot_RiskFort_Install<*timestamp*>.log file. 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 119)
- Case Management Queuing Server Startup Log File (see page 121)

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 very useful in identifying the source of the 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 do so, you must use the instance-specific configuration page that you can access by clicking the required instance in the **Instance Management** page.

RM_3.1--Changing RiskMinder Startup Logging Parameters

If you want to change the logging parameters that you see when RiskMinder Server starts up, then:

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 which 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 the RiskMinder Case Management Queuing Server, it records all startup (or boot) actions in the arcotriskfortcasemgmtstartup.log file. The information in this file is very useful in identifying the source of the 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 do so, 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

If you want to change the logging parameters that you see when Case Management Queuing Server starts up, then:

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
   ```
   The table in Changing RiskMinder Startup Logging Parameters (see page 120) explains these parameters.
4. Set the required values for the parameters that you want to change.
5. Save and close the file.
6. Restart RiskMinder Server.

Transaction Log Files

The transaction logs consist of:

- RiskMinder Server Log (see page 123)
- Case Management Server Log File (see page 124)

RM_3.1--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 also control whether you want to enable trace logging. See "Format of the RiskMinder Server and Case Management Server Log Files" (see page 127) 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/`

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 (specified under the [arcot/riskfortcasemgmtserver/logger] section) 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 also control whether you want to enable trace logging. See "Format of the RiskMinder Server and Case Management Server Log Files" (see page 127) 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 very useful in identifying the source of the 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 also 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 128) 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 very useful in identifying the source of the 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 also 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 128) for details of the default format used in the file.

RM_3.1--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 "Transaction Log Files" (see page 122).

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 128) 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 loggers:

- arcotuds.log (UDS Log File (see page 126))
- arcotadmin.log (Administration Console Log File (see page 125))

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 128) 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	WARNI NG	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 Administration Console and UDS log files, in the *decreasing* order of severity.

Log Level		Description
0	OFF	Use this 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	WARNI NG	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
```

Appendix G: RiskMinder Sample Application

RiskMinder is shipped with a Sample Application that serves as a "template" of simple Java primitives (code) that demonstrate the usage of RiskMinder Java APIs and how your application can be integrated with RiskMinder. In this manner, Sample Application also serves to standardize the integration process between RiskMinder and your application.

Important! Sample Application *must not* be used in your production environment. It is recommended that you build your own Web application by using Sample Application *only* as a code-reference. In a production environment, Sample Application can only be used to verify if RiskMinder was installed successfully, and if is able to perform risk-evaluation operations.

This appendix introduces you to the RiskMinder Sample Application and walks you through the following sections:

- Understanding the Sample Application (see page 133)
- Installing and Configuring the Sample Application (see page 136)
- Performing Risk Evaluation (see page 139)
- Creating Users (see page 143)

Understanding the Sample Application

To deploy RiskMinder in your environment, you need to integrate its APIs with your online application. Sample Application uses RiskMinder Java APIs to demonstrate the most common functionality of RiskMinder, discussed in detail in "Performing Risk Evaluation" (see page 139).

However, before you explore the risk evaluation capability of Sample Application, you must understand the Sample Application Components (see page 134) and Sample Application Recommendations (see page 135).

Sample Application Components

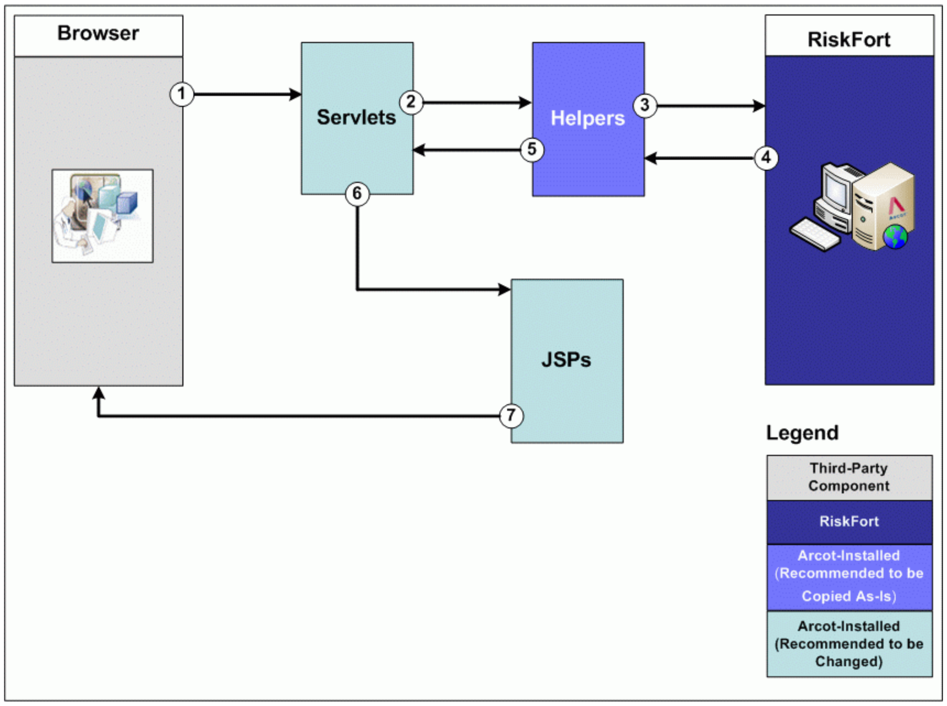
Sample Application constitutes the following:

- **Servlets:** Platform-independent server-side modules that can be used to add new functionality to a Web server. (Servlets are an equivalent of an applets, however, unlike applets, they do not have a user interface. They just enable you to access the existing business logic by the help of user interface provided by JSPs.
- **Helpers:** Classes that provide the extra functionality that is not a part of the class that makes use of them. As a result, helper classes help making the code maintainable and reusable.

  For example, if a class needs to display the value of a number, it could just display it raw, or alternately format it neatly (comma-separated format, for example). To do the formatting instead, the class can make use of another class, which is referred to as a helper class.
- **JSPs:** Java Server Pages (JSPs) that enable you to create and serve dynamic Web content that is server- and platform-independent.

These Sample Application components are organized in a *model-view-controller* (MVC) framework, which separates the user interface of your application from the underlying business logic. As a result, it is much easier to modify either the user interface of the application or the business rules or both without affecting the other.

The following figure illustrates the interaction between the components of Sample Application (*after* a user is successfully authenticated by your application.)


Browser
Servlets
Helpers
RiskFort
JSPs
1
2
3
4
5
6
7
Legend
Third-Party Component
RiskFort
Arcot-Installed (Recommended to be Copied As-Is)
Arcot-Installed (Recommended to be Changed)

As illustrated in the preceding figure, the flow of information between the various components of Sample Application is as follows:

1. User initiates a transaction using the browser window.

   Note: This step assumes that the end user is already successfully logged in after being authenticated by your application.

2. The *servlet* processes the request and invokes the corresponding *helper* function so that it can, in turn, invoke the required RiskMinder SDK.

3. The helper function calls the appropriate API to interact with RiskMinder Server by passing the inputs forwarded by the servlet in the preceding step.

4. RiskMinder Server assesses the input and returns a risk score and an advice. In each case, an object of the relevant type is returned.

   If the assessment failed, an exception is generated.

5. The helper function returns the object created by RiskMinder SDK along with the response to the corresponding servlet.

   In case of a failure, the helper function catches all exceptions and displays a meaningful error message.

6. The servlet receives the response from the helper function, sets the corresponding values in request attribute, and forwards it to a JSP.

7. The JSP parses the request attribute set by the servlet, generates an HTTP response, and forwards it to the browser.

   This response is then displayed to the user.

Sample Application Recommendations

In this integration model, it is recommended that:

- You can use the helper functions *without* any modifications.
- You replace all Sample Application-provided JSPs with your own to control the look and feel of the user interface of the application.

  Note: Read the API documentation *before* changing the code.

- You replace all Sample Application-provided servlets with your own controller logic.
- You *do not use* Sample Application in your production environment. Instead, it is recommended that build your own Web application *by using* Sample Application only as a code-reference.

Installing and Configuring the Sample Application

You can install the Sample Application in the following two ways:

- If you select the **Complete** option while running the RiskMinder installer. When you do so, the Sample Application is also installed along with other RiskMinder components.
- If you select the **Custom** installation option, then you must select the **SDKs and Sample Application** option on the Component Selection screen to install the Sample Application.

Book: For detailed information on prerequisites of the Sample Application and its deployment, refer to the *CA RiskMinder Installation and Deployment Guide*.

On Microsoft Windows

To install (and later configure) RiskMinder and the Sample Application on Microsoft Windows successfully, the user account that you plan to use for installation *must* belong to the Administrators group. Else, some critical steps in the installation, such as DNS creation and configuration and RiskMinder service creation, will not go through successfully, though the installation might complete without any errors.

On UNIX-Based Platforms

The installation UNIX-based platforms is not restricted to the root account. You can install as a non-root user also.

Configuring Sample Application

You can deploy Sample Application in any of the scenarios discussed in the following sub-sections.

Sample Application and RiskMinder on the Same System

Important! If Sample Application and RiskMinder Server are running on the same system, you *do not* have to perform any steps in this section to configure Sample Application.

However if your RiskMinder Server uses a port other than the default (7680), then the only deployment steps you need to perform is to change the port number. To do so:

1. Ensure that the riskfort-3.1.01-sample-application.war is deployed on the application server.

   Book: To ensure that Sample Application is deployed correctly, refer to the *CA RiskMinder Installation and Deployment Guide* for details.

2. Navigate to the location where the WAR file is deployed and open riskfort.risk-evaluation.properties file.

   Note: The deployment procedure (and App_Home) will depend on the application server that you are using.

   Refer to your application server documentation for detailed instructions.

   On Apache Tomcat, riskfort.risk-evaluation.properties is typically available at <*Tomcat_Home*>\webapps\riskfort-3.1.01-sample-application\WEB-INF\classes\properties\riskfort.risk-evaluation.properties.

3. Ensure that the value of HOST.1 is localhost.

4. Change the value of PORT.1.

   PORT.1 represents the port on which RiskMinder Server is listening to the incoming requests. For example, PORT.1=7680.

Sample Application and RiskMinder on Different Systems

If Sample Application and RiskMinder Server are running on different systems, then you *must* complete the following steps to configure Sample Application:

1. Ensure that the riskfort-3.1.01-sample-application.war is deployed on the application server.

   Book: To ensure that Sample Application is deployed correctly, refer to the *CA RiskMinder Installation and Deployment Guide* for details.

2. Navigate to the location where the Sample Application WAR file is deployed and open riskfort.risk-evaluation.properties file.

   Note: The deployment procedure (and App_Home) will depend on the application server that you are using.
   Refer to your application server documentation for detailed instructions.

   On Apache Tomcat, riskfort.risk-evaluation.properties is typically available at <*Tomcat_Home*>\webapps\riskfort-3.1.01-sample-application\WEB-INF\classes\properties\riskfort.risk-evaluation.properties.

3. Change the value of HOST.1.

   HOST.1 represents the host name or IP address of RiskMinder Server. For example, HOST.1=10.150.1.111.

4. Change the value of PORT.1.

   PORT.1 represents the port on which RiskMinder Server is listening to the incoming requests. For example, PORT.1=7680.

5. **(Optional, if RiskMinder Server not running in SSL mode)** Perform this step only if you want to secure the communication between Sample Application and RiskMinder Server over SSL.

   In this case, change the value of TRANSPORT_TYPE to SSL:

   TRANSPORT_TYPE=SSL

6. **(Optional, if RiskMinder Server not running in SSL mode)** Specify server CA certificate location for CA_CERT_FILE=<*Server_CA_certificate_location*>.

   Important! The certificate *must* be in PEM format.

   For example, CA_CERT_FILE=C:\certs\riskfort_ca.pem

7. Save the changes and close the open files.

8. Restart the application server on which the Sample Application is running.

With the completion of these steps, your Sample Application deployment is also complete. You can now start using it to understand the API workflow for each supported operation, as demonstrated in the following sections.

Performing Risk Evaluation

RiskMinder Sample Application demonstrates some of these aspects of the RiskMinder risk-evaluation process. It covers the following:

Performing Risk Evaluation on Gathered Information

Sample Application demonstrates how APIs are used to perform risk evaluation on the gathered data. To view the demonstration, perform the following steps:

1. Access the Sample Application URL in a browser window. This URL typically is:

   http://<app_server_host_name>:<port_number>/riskfort-3.1.01-sample-application /index.jsp

   In the preceding URL, <app_server_host_name> represents the host name or IP address of the application server on which Sample Application has been deployed and <port_number> represents the port number at which Sample Application is available.

   The RiskMinder Sample Application landing page appears.

2. Click **Evaluate Risk** to open the Risk Evaluation page.

3. On the page, specify:

   - The name of the user who is being evaluated for risk in the **User Name** field.

   Note: You can also specify the name of a user who is not yet enrolled with Sample Application, as discussed in "Performing Risk Evaluation" (see page 139).

   - (Optional) The name of the organization to which the user belongs in the **User Organization** field.

   Note: If you leave the **User Organization** field empty, the value is automatically set to DEFAULTORG.

   - (Optional) The name of the **Channel** where the transaction originated.
   - (Optional) The corresponding **Name** and **Value** fields with **Additional Inputs**, such as locale information and transaction amount for the current transaction.

4. Click **Evaluate Risk** to generate the Risk Score and Risk Advice for the specified user.

   The Risk Evaluation Results page appears. If the user is not enrolled with RiskMinder, then the advice is typically, ALERT. This advice changes when you create the user in RiskMinder database, as discussed in "Creating Users" (see page 143) later in this appendix.

5. Click **Store DeviceID** to store the specified type of Device ID information (Step 2) on the end user's device.

6. Click **Next Step** to perform post evaluation for the specified user.

   The Post Evaluation page appears.

   Important! As indicated on the page, if the advice on the previous page was INCREASEAUTH, then it is recommended that you plug-in your secondary authentication mechanism, so that the user performs another authentication before performing RiskMinder post-evaluation steps.

7. Click **Post Evaluate** to generate and display the final Risk Advice for your application.

The final result is displayed in the Post Evaluation Results frame of the page.

Editing the Gathered User Information

Sample Application also allows you to edit the collected Device ID information of an existing user. This enables you to simulate real-life situations a user might run in to. For example, if you change the IP address of an enrolled user, then you can simulate that the user is logging in from a different location.

For demonstration purposes, the data that you can edit by using Sample Application is:

- IP Address
- Device ID

After a user’s information is updated, they must undergo the risk-evaluation procedure. In such cases, RiskMinder typically generates the INCREASEAUTH advice.

To edit a user’s information by using Sample Application, perform the following tasks:

1. On the RiskMinder Sample Application page, Click **Risk Evaluation** to open the Risk Evaluation input page.
2. Specify the **User Name**, and if required the **User organization**.
3. Click **Edit Inputs** to open the Edit Risk-Evaluation Inputs page.
4. Change the IP address in the **IP Address of My Machine** field or the Device ID in the **Device ID of My Machine** field, or both.
5. Click **Evaluate Risk** to generate the Risk Score and Risk Advice for the specified user.

   The Risk Evaluation Results page should appear with an ALERT advice.
6. Click **Store DeviceID** to store the specified type of Device ID information on the end user's device.
7. Click **Next Step** to perform post evaluation for the specified user.

   The Post Evaluation page appears.
8. Specify the result of secondary authentication (to simulate the scenario where RiskMinder receives the result of this additional authentication) in the **Result of Secondary Authentication** field.
9. Click **Post Evaluate** to generate and display the final Risk Advice for your application.

   The final result is displayed in the Post Evaluation Results frame of the page.

API Workflow and Reference

When you use Sample Application for risk evaluation:

1. The ArRFInitHandler.class of the com.arcot.riskfort.sampleapp.initialize package is invoked.

   After the initialization is complete, RiskMinder is ready to serve the requests.

2. RiskFactory, which is responsible for RiskMinder-related operations, is invoked by the ArRFEvaluateHelper.class in the com.arcot.riskfort.sampleapp.helpers package.

   When the evaluateRisk() method in ArRFEvaluateHelper.class is called, the RiskXActionAPI interface is invoked.

3. To perform **risk evaluation**, your application's servlet, .jsp, or any other calling file must call the evaluateRisk() method of the ArRFEvaluateHelper class.

   This method returns the RiskAssessment object as response, which contains riskScore, riskAdvice, deviceID, and related information to your application.

   Important! Based on the generated risk score and advice at this stage, your application must provide logic to perform the required action, such as forward the transaction request to a CSR or force the user to perform additional authentication.

4. To perform post evaluation, your application's servlet, .jsp, or any other calling file must call the postEvaluateHelper() method of the ArRFEvaluateHelper class.

   Important! The postEvaluateHelper() method must be called only after the evaluateRisk() method was executed.

   This method accepts CallerID, RiskAssessment object (returned by the evaluateRisk() function), and result of your secondary authentication, if any, (secondaryAuthenticationStatus), and association name (optional) as input and returns to your application the postEvaluateResponseObj object, which contains the final advice.

   The class and method required in the preceding workflow are described in the following table.

API	Description
Class: ArRFEvaluateHelper	The helper class that contains the methods required for risk evaluation and post evaluation.
Method: evaluateRisk()	The method that generates the Risk Score and Risk Advice.
Method: postEvaluateHelper()	The method that generates the final Risk Advice. This advice is a Boolean value. In case of True, the advice is ALLOW, while in case of False, the advice is to not ALLOW the transaction. This method must be called only *after* evaluateRisk() has been executed.

Creating Users

Creating a new user in RiskMinder is a one-time operation, which is performed only when a new user is to be added to RiskMinder. This user typically is an existing user of your application accessing RiskMinder for the first time.

This section describes how the Sample Application demonstrates the creation of a user and then explains the API workflow for the same:

- Creating a User (see page 143)
- Performing Risk Evaluation for the User that You Created (see page 144)

Creating a User

From this release, the Issuance API that was responsible for user creation has been deprecated. You can create end users in the system:

- Either by using Administration Console.
- Or by making a call to the createUserRequest message in the ArcotUserRegistrySvc Web service.

The following procedure walks you through the steps for creating a user by using Administration Console.

Book: If you want to create a user by using the Web service, then refer to "Creating Users" in "Managing Users and Accounts" in the *CA RiskMinder Web Services Developer’s Guide*.

To create a user:

1. Ensure that you are logged in with the required privileges and scope to create the user.
2. Activate the **Users and Administrators** tab.
3. Under the **Manage Users and Administrators** section, click the **Create User** link to display the Create User page.
4. On the page, enter the required details of the user.
5. Click **Create User** to create the user.

Performing Risk Evaluation for the User that You Created

If now you perform risk evaluation (Step 2 on page G-146 through Step 4 on page G-148, as mentioned in "Performing Risk Evaluation on Gathered Information" (see page 140)) for the user that you just created, you will see the **Risk Score** and **Risk Advice** values as 65 and INCREASEAUTH, respectively, on the Risk Evaluation Results page.

When you click **Next Step** to perform post evaluation, you will see the Post Evaluation page as follows.

If you now select SUCCESS in the **Result of Secondary Authentication** field and click **Post Evaluate**, you will see the Final Risk Advice as **ALLOW.**

After you see the **ALLOW** advice for a user, then for next risk evaluation result you will always see **ALLOW**, unless you changed any on the user IP or Device ID information.