Arcot RiskFort™
Java 開発者ガイド
バージョン 2.2.6

455 West Maude Avenue, Sunnyvale, CA 94085

Arcot RiskFort 開発者ガイド
バージョン 2.2.6
2010 年 9 月
部品番号：RF-0226-0DGJ-10

商標

特許

Arcot Systems, Inc., 455 West Maude Avenue, Sunnyvale, CA 94085

サードパーティ ソフトウェア

RiskFort および関連するコンポーネントによって使用されるサードパーティ ソフトウェアはすべて、「Arcot RiskFort 2.2.6 インストールおよび展開ガイド」の付録 G 「サードパーティ ソフトウェア ライセンス」にリスト表示されています。

目次

序文

このガイドでは、Arcot RiskFort Java クラスおよびメソッドを使用して、オンライン アプリケーションによるリスク評価と関連タスクをプログラムで実行する方法について説明します。本書では、RiskFort SDK の Java 実装について説明します。

付録 B の 「サンプル コード」では、RiskFort のサポートされるユーザ関連操作、リスク評価、事後評価機能のテストに実行できる、完全に機能するサンプル コードを提供します。

対象読者

このガイドは、Java プログラムを実装するアプリケーション プログラマを対象とします。このプログラムでは、Arcot RiskFort で提供されている API および関数を使用する必要があります。

読者は以下に精通している必要があります。

- データベースのアーキテクチャおよび概念
- セキュリティ管理の概念
- 認証と許可の概念
- HTTP および TCP/IP をはじめとするインターネット プロトコル
- SSL（Secure Sockets Layer）通信および関連する概念（公開キーと秘密キーの交換、デジタル証明書およびシグネチャ、および証明機関）

このガイドに含まれている情報

このガイドは、以下のように編成されています。

- 第 1 章の「はじめに」では RiskFort SDK およびその機能について説明します。また、アプリケーションが RiskFort と統合することができる「ポイント」のおおまかなアウトラインを提供し、統合のための前提条件について説明します。
- 第 2 章の「RiskFort ワークフローについて」では、オンプレミス RiskFort ソリューションによって提供されるプロセスとワークフローについて説明します。

- 第 3 章の「事前準備」では、CLASSPATH に含める API 固有の JAR ファイルとプロパティ ファイル、および API を初期化する方法について説明します。
- 第 4 章の「ユーザの管理」では、RiskFort に新規ユーザを作成する手順、データベースからユーザの詳細を読み取る手順、および RiskFort Issuance API の適切なクラス、インターフェース、およびメソッドを使用することによりこれらの詳細を更新する手順について説明します。
- 第 5 章の「デバイス ID とマシン FingerPrint の収集」では、オンライン アプリケーション全体に RiskFort Device ID およびマシン Fingerprint （MFP）のコレクション コードを実装する方法を説明します。RiskFort は、cookie コレクションおよびマシン Fingerprint を広範囲で利用して、エンド ユーザのログインに関連するリスクを特定します。
- 第 6 章の「リスク評価の実行」では、RiskFort Risk Evaluation API の適切なクラス、インターフェースおよびメソッドを使用することにより、RiskFort がアプリケーションから転送されたリスク分析リクエストのリスク評価および事後評価を実行する方法について説明します。
- 付録 A の 「追加の SDK 設定」では、複数の RiskFort サーバ インスタンスをセットアップする方法、ならびに Java SDK および RiskFort サーバの間に SSL 通信を設定する方法について説明します。
- 付録 B の 「サンプル コード」では、RiskFort のユーザ関連操作、リスク評価および事後評価機能のテストに実行できる、完全に機能するサンプル コードを提供します。
- 付録 C の 「Java API 関連資料」では、Issuance API と Risk Evaluation Java API の各ファイル、およびそれらにアクセスする方法をリスト表示します。
- 付録 D の 「例外コードおよびエラー コード」では、RiskFort SDK によって返されたエラー コードをリスト表示します。
- 付録 E の 「入力データの検証」では、SDK 入力パラメータをチェックするために使用する条件をリスト表示します。
- 付録 F の 「RiskFort ロギング」では、RiskFort ログ ファイル、それを制御する設定ファイル、記録された重大度レベル、およびログ ファイルのエントリの形式に関する情報について説明します。
- 付録 G の 「RiskFort サンプル アプリケーション」では、RiskFort Sample Application の概要を説明します。これは、RiskFort API を使用するサンプル Web アプリケーションで、RiskFort の最も一般的な機能を実証します。
- 付録 H の 「用語集」では、RiskFort の重要な用語をリスト表示して説明します。

関連出版物

その他の関連出版物を以下に示します。

Arcot RiskFort 2.2.6 管理ガイド	このガイドには、RiskFort を管理して設定するための情報が含まれます。
Arcot RiskFort 2.2.6 インストールおよび展開ガイド	このガイドは、Arcot RiskFort を単一システムおよびマルチシステム環境にインストールして設定する方法について説明します。これは、インストール前、インストール時、およびインストール後のタスクに関する主な情報源です。
Arcot RiskFort 2.2.6 Java 開発者ガイド	このガイドは、RiskFort によって提供される Java API について説明し、それらを使用する方法についても説明します。
Arcot RiskFort 2.2.6 クイック インストール ガイド	このガイドは、RiskFort のインストールに必要なタスクのサマリを提供します。

本書で使用されている規則

このマニュアルの表記規則、形式、および範囲を以下のパラグラフで説明します。

表記規則

このマニュアルでは以下の表記規則を使用します。

斜体	強調、ガイド名
太字	ユーザ入力、GUI 画面のテキスト
固定幅フォント	ファイル名およびディレクトリ名、拡張子、コマンド プロンプト、CLI のテキスト、実行時のテキストのコード
固定幅太字フォント	パスのターゲット ファイルまたはディレクトリ名
固定幅フォント	コマンド プロンプト、CLI のテキスト、コード
固定幅斜体フォント	ユーザごとに異なる場合があるファイルまたはディレクトリ名
リンク	このガイド内のリンク、URL リンク

形式

このマニュアルでは、以下の形式を使用して特別なメッセージを強調表示しています。

	**注：** 重要な情報、または特に注目すべき情報を強調表示します。
	**ヒント：**時間やリソースを節約する手順を強調表示します。
	**警告：** このタイプの注意を無視すると、機器に誤動作または損傷が発生する場合があります。
	**重要：**操作を実行する前に知っておくべき情報を示します。
	**注意：**考えられる危険性に対してユーザの注意を喚起します。
	**関連文書：** 他のガイドへの参照情報を提供します。

サポートへのお問い合わせ

サポートが必要な場合は、以下の Arcot サポートにお問い合わせください。

E-MAIL （電子メール）	*support@arcot.com*
Web サイト	*http://www.arcot.com/support/index.html*

第 1 章
はじめに

Arcot RiskFort （本マニュアルでは以下「RiskFort」）は、広範囲にわたって収集したデータを既定のルールに照らして検証することにより、オンライン トランザクションをリアル タイムで評価する順応性の高い認証ソリューションです。評価後、各トランザクションにはリスク スコアとアドバイスが割り当てられます。リスク スコアが高いほど、不正行為である可能性が高くなります。このリスク スコアとアドバイスをアプリケーションで利用し、自社のビジネス ポリシーに基づいて、トランザクションを承認または拒否したり、追加の認証を要求したり、テクニカル サポート担当者にアラートを発行したりすることができます。

RiskFort は設定の自由度が高く、ポリシーやリスク緩和要件との整合性を保ちながら、任意のリスク評価ルールの設定パラメータを柔軟に変更することができます。また、個々のルールでデフォルトのリスク スコア、スコアリング設定、およびスコアリング優先度を変更したり、1 つ以上のルールに対して実行の有効化と無効化を選択的に指定したりすることもできます。

事前設定済みのすぐに使えるルールに加えて、RiskFort は作業環境でプログラムが可能なアドオン ルール機能を備えており、業界特有のルールを選択的に展開できます。

関連文書： RiskFort とそのアーキテクチャの基本概念については、「Arcot RiskFort2.2.6 インストールおよび展開ガイド」の第 1 章「RiskFort の基本について」を参照してください。

この章では、以下の内容について説明します。

RiskFort SDK の概要

RiskFort ソフトウェア開発キット（SDK）は RiskFort にプログラムのフロントエンドを提供します。これは、RiskFort サーバと対話するための API を提供して、各ログイン トランザクションを評価し（Risk Evaluation API）、RiskFort データベースでユーザを作成するのに必要なタスクを実行します（発行 API）。

RiskFort のインストール後、Java SDK は以下の場所で利用できます。

Windows の場合

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

UNIX プラットフォームの場合

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

RiskFort をまだインストールしていない場合でも、RiskFort パッケージの `Documentation` ディレクトリにある Javadoc にアクセスできます。

Risk Evaluation API

Risk Evaluation API （`arcot-riskfort-evaluaterisk.jar`）は RiskFort サーバのインターフェースで、トランザクションに関連するリスクを評価し、適切なアドバイスを返すロジックを提供します。

ユーザのシステムから収集されたさまざまな要因、およびトリガされた設定済みルールの結果に基づいて、この API はスコアと対応するアドバイスを返します。RiskFort が 2 次認証を推奨（アプリケーションによって実行）した場合は、アプリケーションから受信したこの 2 次認証のフィードバックに基づいて最終アドバイスを返します。

さらに、cookie または Flash Shared Object （FSO）を使用して、この API は、アプリケーションにアクセスするために使用するデバイスにユーザを関連付けます（またはバインドします）。この関連付け（またはデバイス バインディング）は、ユーザがトランザクションに使用するシステムから発信されるトランザクションのリスクを識別するのに役立ちます。バインドされていないユーザは、認証のチャレンジが行われる可能性が高くなります。また、この API により、これらの関連付けのリスト表示および削除を実行できます。

注： ユーザは、複数のデバイスにバインドできます（たとえば仕事用と自宅用のコンピュータを使用するユーザ）。また、単一のデバイスを複数のユーザにバインドすることもできます（たとえば 1 台のコンピュータを使用する家族）。

発行 API

RiskFort 発行 API （arcot-riskfort-issuance.jar）によって、RiskFort データベースにユーザの詳細を作成し、リスト表示し、更新できます。RiskFort データベースにユーザを作成することによって、RiskFort がユーザのリスク履歴を記録して、さらにリスク分析に使用できます。

RiskFort SDK の機能

このセクションでは、リスク評価 SDK および発行 SDK の特徴的な機能について説明します。

- **SDK を初期化する複数の方法**

  プロパティ ファイルまたはマップを使用することによりリスク評価 SDK および発行 SDK を初期化できます。これを実行する方法の詳細については、第 3 章の「事前準備」を参照してください。

- **SDK フェイルオーバのサポート**

  Java SDK はフェイルオーバ メカニズムをサポートします。RiskFort サーバのインスタンスが動作しない場合、SDK は追加で設定されたインスタンスのうちのいずれかに自動的に接続します。これを実行する方法の詳細については、A-79 ページの「複数の RiskFort サーバ インスタンスの設定」を参照してください。

- **SSL のサポート**

  Secure Socket Layer （SSL）の使用により Java SDK と RiskFort サーバとの間の接続を保護することができます。SDK と WebFort サーバの間の SSL をセットアップするには、RiskFort プロパティ ファイルを編集する必要があります。これを実行する方法の詳細については、A-80 ページの「SDK と RiskFort サーバ間の SSL 通信のセットアップ」を参照してください。

- **追加のパラメータのサポート**

  必須入力の他に、API では名前と値のペアとして渡すことができる追加の入力も使用できます。この入力には、ロケール、呼び出しアプリケーションの詳細、または追加のトランザクション詳細などの情報を含めることができます。

統合手順の概要

RiskFort SDK では、ユーザが利用可能な統合メソッドおよびリスク ベースの認証フローのタイプを、さまざまな組み合わせで自由に設定できます （サポートされるワークフローの詳細については、第 2 章の「RiskFort ワークフローについて」を参照)。これにより、組織の要件に最も適した最適な認証ソリューションを設計できます。

RiskFort フローは、以下で説明する段階でユーザのオンライン アプリケーションに統合できます。

ユーザがアプリケーションにログインする前（単にログイン ページにアクセスした時）

この場合、アプリケーションはログイン ページから RiskFort の `evaluateRisk()` 関数を呼び出して、(ユーザがログイン クレデンシャルを指定する前に）受信データに関連付けられたリスクを評価できます。たとえば、拒否 IP チェックおよび拒否国チェックについて、IP アドレスや国を評価できます。

注 : 拒否 IP アドレスは、過去に既知のアノニマイザ プロキシまたは不正行為や悪意のあるトランザクションの発信元となった IP アドレスの集合体です。同様に拒否国は、過去に不正行為や悪意のあるトランザクションが記録されているすべての国の集合体です。

この場合、ユーザ情報が必要でない他の RiskFort ルールを評価することもできます。これには、デバイス頻度チェック、ゾーン ホッピングのチェック、および実装されているすべてのカスタム アドオン ルールなどが含まれます。

ユーザが（ユーザ名とパスワードを指定し、アカウントまたは保護されたリソースにアクセスして）オンライン アプリケーションにログインした後

この場合は、以下が行われます。

1. アプリケーションは、ログイン成功後に表示されるアプリケーションのメイン ページから RiskFort を呼び出す必要があります。以下のような場合が考えられます。

- **ユーザが RiskFort に登録されている場合**、リスク評価を受ける必要があります。

  この場合、アプリケーションは、RiskFort の `evaluateRisk()` 関数を呼び出す必要があります。それには、リスク評価のため、ユーザ、デバイス シグネチャ（RiskFort が提供する MFP Javascript によって収集）、IP アドレス、トランザクション詳細を渡します。

  - ルールの実行後に、受信されたユーザ データに（受信データとこのユーザまたはデバイスの保存データに基づいて）低いスコアが割り当てられた場合、アドバイスとして`許可`が与えられます。

    この場合、アプリケーションは保護されたリソースまたは Web ページへのアクセスをユーザに許可し、ユーザがトランザクションを続行することを許可する必要があります。

  - ルールの実行後に、受信されたユーザ データに（受信データとこのユーザまたはデバイスの保存データに基づいて）高いスコアが割り当てられた場合、アドバイスとして`否認`が与えられます。

    この場合、結果は組織のポリシーによって決定されます。そのトランザクションは拒否することも、セキュリティ アナリスト（RiskFort ではテクニカル サポート担当者（CSR））に転送して確認と次のアクションを実行することもできます。

  - トランザクションに疑わしいとのフラグが立てられた場合、アドバイスとして `INCREASEAUTH` が与えられます。

    この場合、アプリケーションは 2 次認証を実行する必要があります。それには業界標準の秘密の質問（母親の旧姓や生年月日など）を含めることができます。

注： また、この目的には Arcot WebFort を使用することもできます。

詳細については、*http://www.arcot.com/* を参照してください。

    アプリケーションが 2 次認証でユーザを認証した場合にのみ、保護されたリソースまたは Web ページへのアクセスをユーザに許可し、ユーザがトランザクションを続行することを許可する必要があります。

- **ユーザが RiskFort では新規である場合**、登録する必要があります。

  この場合、アプリケーションはユーザ詳細を渡すことにより RiskFort の `createUser()` 関数を呼び出して、RiskFort データベースにユーザを作成する必要があります。

重要 : RiskFort の動作はエンド ユーザに認識されないため、この登録によってエンド ユーザ エクスペリエンスを変更しないように推奨します。

RiskFort がこの登録後にユーザを認識したら、アドバイスが`許可`である場合は、`evaluateRisk()` 関数を呼び出してユーザにトランザクションの続行または保護されたリソースへのアクセスを許可する必要があります。

2. ここで、アプリケーションは RiskFort's `postEvaluate()` 関数を呼び出す必要があります。`evaluateRisk()` の結果が `INCREASEAUTH` であった場合、この呼び出しを使用することにより、RiskFort は最終アドバイスを生成し、関連付けを作成します。その他のアドバイスの場合には、関連付けと属性のみが更新されます。このデータは、将来のトランザクションのリスク分析に使用されるフィードバックとして役立ちます。

事前準備

アプリケーションを RiskFort に統合する前に、RiskFort がインストールされ設定されている必要があります。以下のことを確認してください。

- RiskFort をインストールするシステムがシステム要件を満たしていること。

関連文書 : 「Arcot RiskFort 2.2.6 クイック インストール ガイド」の「システム要件の確認」を参照してください。

- 以下に示す設定および計画関連の情報が完了していること。
  - RiskFort データベース インスタンスの必要な数がインストールされ設定されている。

関連文書 : 詳細な手順については、「Arcot RiskFort 2.2.6 インストールおよび展開ガイド」の「データベース サーバの設定」およびデータベース関連の「インストール後のタスク」を参照してください。

- JDK を使用する RiskFort コンポーネントをインストールするシステムに、JDK の該当するバージョンがインストールされている。
- また、必要なアプリケーション サーバもインストールされている。

関連文書 : 「Arcot RiskFort2.2.6 インストールおよび展開ガイド」の「Java 依存コンポーネントの要件」を参照してください。

RiskFort を**単一システムで展開**する場合は、すべてのコンポーネントがオンになっていることを確認します（詳細については、「Arcot RiskFort 2.2.6 インストールおよび展開ガイド」の第 4 章「単一システムへの RiskFort の展開」を参照）。

RiskFort を**分散システムで展開**する場合は、すべてのコンポーネント間で接続が確立され、それらが相互に通信していることを確認します（詳細については、「Arcot RiskFort 2.2.6 インストールおよび展開ガイド」の第 5 章「分散システムへの RiskFort の展開」を参照）。

第 2 章
RiskFort ワークフローについて

RiskFort は、オンライン アプリケーションによって統合され使用できる多くのワークフローを提供します。RiskFort が `INCREASEAUTH` アドバイスを生成するとき以外は、ほとんどの場合、組織の要件に基づいて、ユーザの既存のオンライン エクスペリエンスを変更せずにこれらのワークフローを統合できます。

この章ではこれらのワークフローについて説明し、関連するさまざまなプロセスについて理解できるように、各ワークフローの概要について説明します。

登録ワークフロー

アプリケーションがリスク分析のリクエストを転送するたびに、RiskFort は **Unknown User Check** ルールを使用してユーザの詳細が RiskFort データベースに存在するかどうか特定します。この情報が見つからない場合、RiskFort は受信リクエストを初めての（不明な）ユーザ リクエストとして扱い、`ALERT` アドバイスを提言します。このような場合は、ユーザを登録する必要があります。

*登録*は RiskFort データベースに新規ユーザを作成するプロセスです。以下のサブセクションで説明されているように、アプリケーションから発行 SDK の `createUser()` メソッドを呼び出すことにより、ユーザを明示的に登録できます。また、Administration Console のその他のルールの設定ページで［**Implicit**］オプションを選択することにより、ユーザを暗黙的に作成することもできます。

登録の後で、リスク評価（2-18 ページの「リスク評価ワークフロー」で詳述）を実行する必要があります。これにより、システムでユーザが暗黙的に作成されます。ただし、ユーザがアプリケーションに登録されていない場合は、組織のポリシーに応じて操作を行う必要があります。

明示的な登録

明示的な登録の場合には、アプリケーション コードから RiskFort の `createUser()` 関数を明示的に呼び出して RiskFort データベースにユーザを作成する必要があります。リスク評価を実行する前（シナリオ 1）、または後（シナリオ 2）で、（`evaluateRisk()` 呼び出しを使用して）この関数を呼び出すことができます。

シナリオ 1

`evaluateRisk()` 関数を呼び出す前に `createUser()` 関数を呼び出す場合、明示的な登録ワークフローの手順は以下のとおりです。

1. **ユーザがオンライン アプリケーションにログインします。**

   このユーザがシステムに存在している場合、システムによって検証されます。ユーザ名が有効でない場合、アプリケーションは適切なアクションを行う必要があります。

2. **アプリケーションは RiskFort の `createUser()` 関数を呼び出します。**

   この段階で、アプリケーションは、`riskfortissuanceAPI` の `createUser()` 関数に明示的な呼び出しを行う必要があります。この呼び出しで、ユーザ名、姓、組織、電子メールおよび個人保証メッセージ（PAM）などの関係するすべてのユーザの詳細を RiskFort に渡す必要があります。

3. **RiskFort はデータベースにユーザを作成します。**

   `createUser()` 呼び出しが成功した場合、RiskFort は RiskFort データベースにユーザレコードを作成します。これで、ユーザは RiskFort に登録されます。

4. **アプリケーションは、RiskFort に必要な情報を収集します。**

   この段階で、アプリケーションは `json.js` と呼ばれる RiskFort のユーティリティ スクリプトを使用してユーザのシステムから以下の情報を集める必要があります。この情報は、リスクの分析のために RiskFort によって使用されます。

   - オペレーティング システム、プラットフォーム、ブラウザ情報（ブラウザの言語、HTTP ヘッダ情報など）、ロケールおよび画面設定が含まれる**ユーザのシステム情報**。
   - Flash Shared Object （FSO）cookie またはブラウザ cookie などの DeviceID が含まれる**デバイス情報**。
   - IP アドレスおよび ISP 関連の情報が含まれる**場所情報**。
   - （**追加情報を使用する場合のオプション**）トランザクション量、ロケールおよび関連情報が含まれている可能性のある**追加の入力情報**。

5. **アプリケーションは RiskFort の `evaluateRisk()` を呼び出してリスク分析を行います。**

   この場合、リスク分析を実行する前にユーザを登録したため、RiskFort システムはユーザを「知っており」、`ALERT` アドバイスは生成されません。詳細については、「リスク評価ワークフロー」を参照してください。

6. **RiskFort はリスク分析を実行します。**

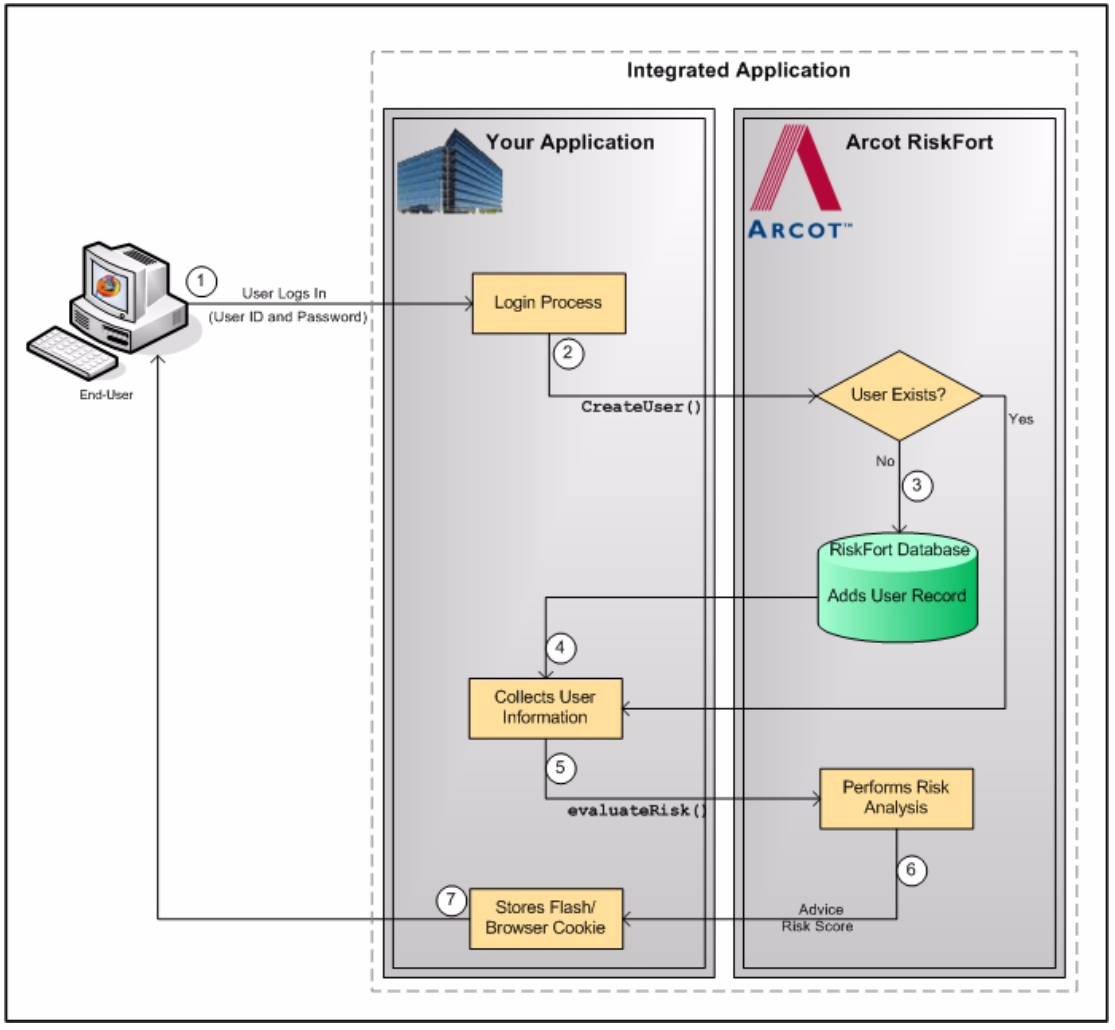
   RiskFort はリスク スコアおよびアドバイスを生成します。

7. **アプリケーションはエンド ユーザのシステムにデバイス ID を保存します。**

   ユーザのアプリケーションは、ユーザが現在トランザクションに使用しているデバイスで cookie （FSO またはブラウザ cookie）として `evaluateRisk()` によって返されたデバイス ID を保存する必要があります。

図 2-1 では、`evaluateRisk()` 呼び出しの前に `createUser()` を呼び出す場合の明示的な登録ワークフローを示します。

図 2-1 明示的な登録ワークフロー：evaluateRisk() の *前に* createUser() を呼び出す



シナリオ 2

`evaluateRisk()` 関数を呼び出した後に `createUser()` 関数を呼び出す場合、明示的な登録ワークフローの手順は以下のとおりです。

1. **ユーザがオンライン アプリケーションにログインします。**

   このユーザがシステムに存在している場合、システムによって検証されます。ユーザ名が有効でない場合、アプリケーションは適切なアクションを行う必要があります。

2. **アプリケーションは、RiskFort に必要な情報を収集します。**

   この段階で、アプリケーションは `json.js` と呼ばれる RiskFort のユーティリティ スクリプトを使用してユーザのシステムから以下の情報を集める必要があります。この情報は、リスクの分析のために RiskFort によって使用されます。

   - オペレーティング システム、プラットフォーム、ブラウザ情報（ブラウザの言語、HTTP ヘッダ情報など）、ロケールおよび画面設定が含まれる**ユーザのシステム情報**。
   - Flash Shared Object （FSO）cookie またはブラウザ cookie などの DeviceID が含まれる**デバイス情報**。
   - IP アドレスおよび ISP 関連の情報が含まれる**場所情報**。
   - （**追加情報を使用する場合のオプション**）トランザクション量、ロケールおよび関連情報が含まれている可能性のある**追加の入力情報**。

3. **アプリケーションは RiskFort の `evaluateRisk()` 関数を呼び出します。**

   この段階で、アプリケーションは `riskfortAPI` の `evaluateRisk()` 関数を呼び出す必要があります。この呼び出しでは、前の手順で収集したユーザおよびデバイス情報をすべて RiskFort に渡す必要があります。

4. **アプリケーションは RiskFort の `evaluateRisk()` を呼び出してリスク分析を行います。**

   RiskFort は、ユーザのリスク分析を実行し、アドバイスを生成します。この場合、ユーザが RiskFort システムにまだ「知られていない」ので、`ALERT` アドバイスが生成されます。

5. **アプリケーションは RiskFort の `createUser()` 関数を呼び出します。**

   生成される `ALERT` アドバイスに対して、アプリケーションは、`riskfortissuanceAPI` の `createUser()` 関数に明示的な呼び出しを行う必要があります。この呼び出しで、ユーザ名、姓、組織、電子メールおよび個人保証メッセージ（PAM）などの関係するすべてのユーザの詳細を RiskFort に渡す必要があります。

6. **RiskFort はデータベースにユーザを作成します。**

   `createUser()` 呼び出しが成功した場合、RiskFort は RiskFort データベースにユーザ レコードを作成します。これで、ユーザは RiskFort に登録されます。

7. **アプリケーションは RiskFort の `evaluateRisk()` 関数を再度呼び出します。**

   この段階で、アプリケーションは `riskfortAPI` の `evaluateRisk()` 関数を再度呼び出す必要があります。この呼び出しでは、手順 2 で収集したユーザおよびデバイス情報をすべて RiskFort に渡す必要があります。

8. **RiskFort は、ユーザのリスク分析を実行します。**

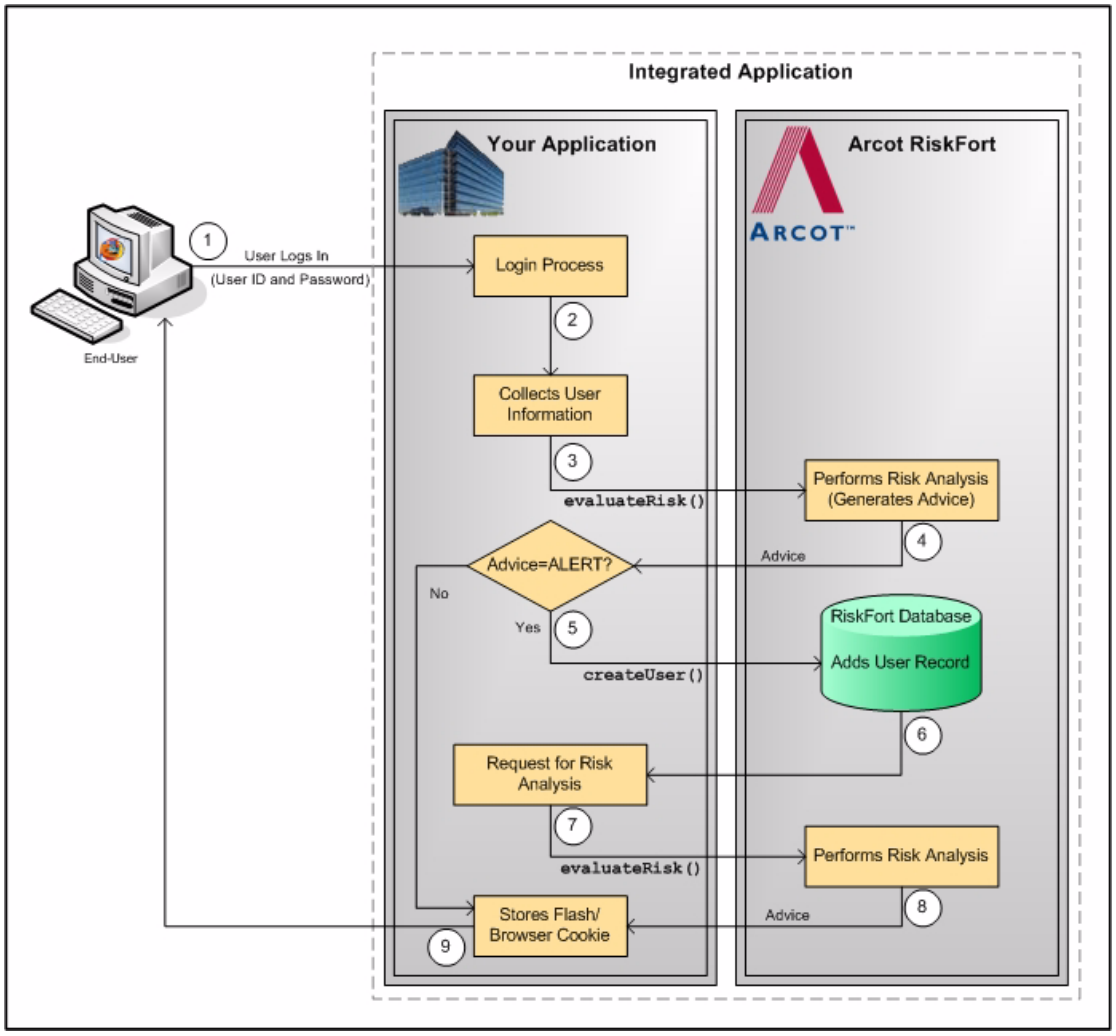
   この場合、RiskFort はルールを実行し、リスク スコアおよびアドバイスを生成します。

9. **アプリケーションはエンド ユーザのシステムにデバイス ID を保存します。**

   ユーザのアプリケーションは、ユーザが現在トランザクションに使用しているデバイスで cookie （FSO またはブラウザ cookie）として `evaluateRisk()` によって返されたデバイス ID を保存する必要があります。

図 2-2 では、`evaluateRisk()` 呼び出しの前に `createUser()` を呼び出す場合の明示的な登録ワークフローを示します。

図 2-2 明示的な登録ワークフロー：evaluateRisk() の *前に* createUser() を呼び出す



暗黙的な登録

暗黙的な登録の場合には、アプリケーション コードから RiskFort の `createUser()` 関数を明示的に呼び出して RiskFort データベースにユーザを作成する必要はありません。代わりに、RiskFort は「不明なユーザ」に `ALERT` アドバイスを生成するときに、自動的に関数を呼び出してユーザを登録します。

この登録を機能させるには、Administration Console のその他のルールの設定ページで最初に［**User Creation Mode**］オプションを［**Implicit**］に設定することが重要です。

暗黙的な登録ワークフローの手順は次のとおりです。

1. **ユーザがオンライン アプリケーションにログインします。**

   このユーザがシステムに存在している場合、システムによって検証されます。ユーザ名が有効でない場合、アプリケーションは適切なアクションを行う必要があります。

2. **アプリケーションは、RiskFort に必要な情報を収集します。**

   この段階で、アプリケーションは `json.js` と呼ばれる RiskFort のユーティリティ スクリプトを使用してユーザのシステムから以下の情報を集める必要があります。この情報は、リスクの分析のために RiskFort によって使用されます。

   - オペレーティング システム、プラットフォーム、ブラウザ情報（ブラウザの言語、HTTP ヘッダ情報など）、ロケールおよび画面設定が含まれる**ユーザのシステム情報**。
   - Flash Shared Object （FSO）cookie またはブラウザ cookie などの DeviceID が含まれる**デバイス情報**。
   - IP アドレスおよび ISP 関連の情報が含まれる**場所情報**。
   - （**追加情報を使用する場合のオプション**）トランザクション量、ロケールおよび関連情報が含まれている可能性のある**追加の入力情報**。

3. **アプリケーションは RiskFort の `evaluateRisk()` 関数を呼び出します。**

   この段階で、アプリケーションは `riskfortAPI` の `evaluateRisk()` 関数を呼び出す必要があります。この呼び出しでは、前の手順で収集したユーザおよびデバイス情報をすべて RiskFort に渡す必要があります。

4. **RiskFort は、ユーザのリスク分析を実行します。**

   この場合、ユーザが RiskFort システムにまだ「知られていない」ので、デフォルトの `ALERT` アドバイスが生成されます。

5. **RiskFort はデータベースにユーザを作成します。**

   生成される `ALERT` アドバイスに対して、RiskFort は、`createUser()` 関数を使用して RiskFort データベースにユーザ レコードを作成します。これで、ユーザは RiskFort に登録されます。

6. **アプリケーションは RiskFort の `evaluateRisk()` 関数を再度呼び出します。**

この段階で、アプリケーションは `riskfortAPI` の `evaluateRisk()` 関数を再度呼び出す必要があります。この呼び出しでは、手順 2 で収集したユーザおよびデバイス情報をすべて RiskFort に渡す必要があります。

7. **RiskFort は、ユーザのリスク分析を実行します。**

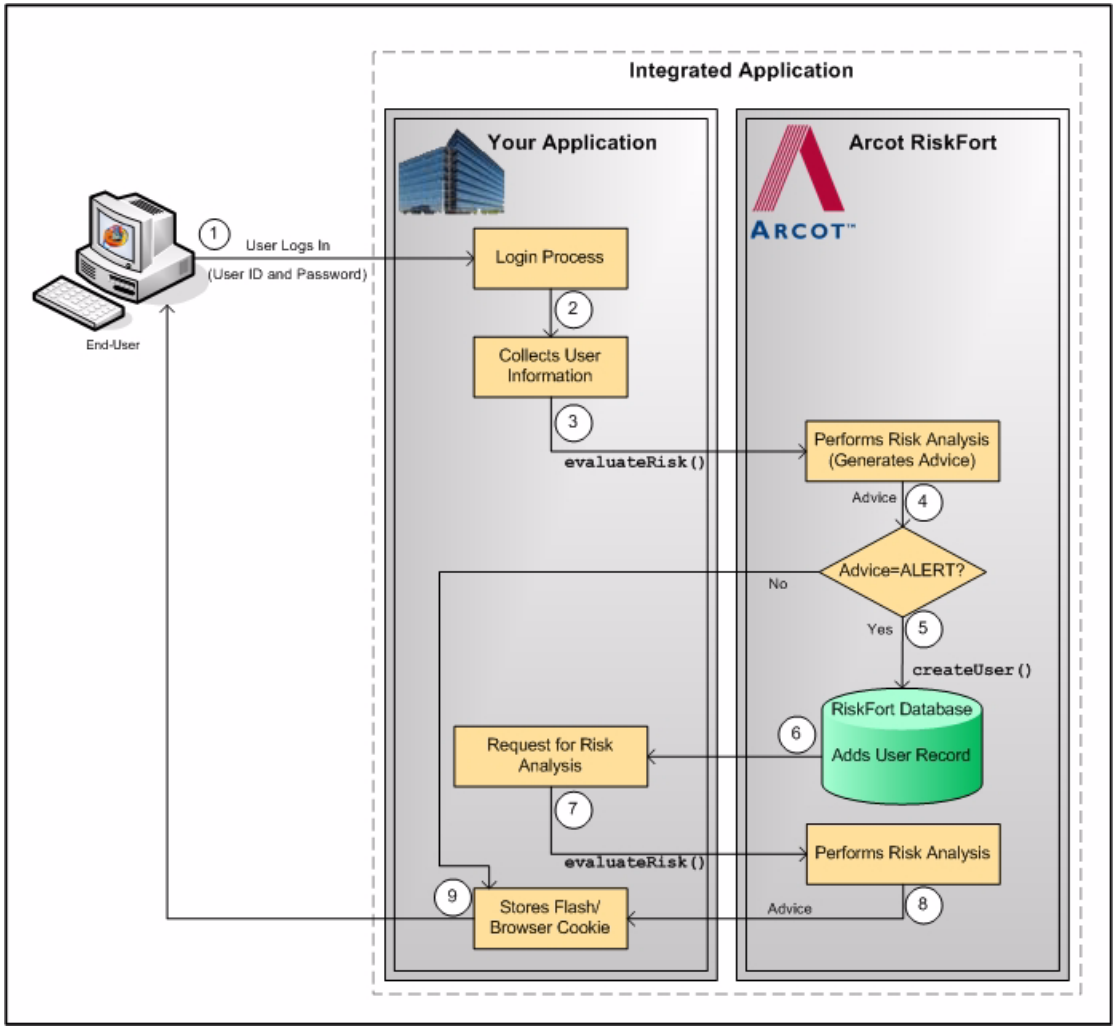
   この場合、RiskFort はルールを実行し、リスク スコアおよびアドバイスを生成します。

8. **アプリケーションはエンド ユーザのシステムにデバイス ID を保存します。**

   ユーザが作成された後、アプリケーションはユーザが現在のトランザクションに使用しているデバイスの cookie として `evaluateRisk()` によって返された DeviceID（FSO またはブラウザ cookie）を格納する必要があります。

図 2-3 に、RiskFort が自動的にユーザを作成する際の暗黙的な登録ワークフローを示します。

図 2-3 暗黙的な登録ワークフロー



リスク評価ワークフロー

リスク評価ワークフローにより、オンライン アプリケーションは、受信したユーザ リクエストに潜在的なリスクがあるかどうかを特定できます。

- リスクが低い場合、ユーザはオンライン アプリケーションへのアクセスを許可されます。

- リスクが高い場合、ユーザはシステムへのアクセスを拒否されます。
- トランザクションが疑わしいとしてタグ付けされた場合、このワークフローはアプリケーションに対して追加の（2 次）認証をユーザにチャレンジして ID を証明するように求めます。

RiskFort のリスク分析機能は、ユーザがオンライン アプリケーションにログインする前、またはユーザがログインに成功した後のいずれでも実装できます。RiskFort の `evaluateRisk()` 関数を呼び出すときに応じて、以下のワークフローが可能です。

- ログイン前のリスク評価ワークフロー
- ログイン後のリスク評価ワークフロー

ログイン前のリスク評価ワークフロー

ユーザがオンライン アプリケーションにアクセスする場合、このワークフローの実装によりユーザがログインする前であっても、ユーザの潜在的なリスクを評価できます。このワークフローでは、リスク評価の基準として、デバイス ID と場所情報（IP アドレス、Device ID、MFP など）に関連する入力情報、およびユーザ固有の情報を必要としないルールのみを使用します。

ユーザがオンライン アプリケーションにログインする前に、RiskFort のリスク分析機能を呼び出す場合、リスク評価ワークフローは以下のとおりです。

1. **ユーザはオンライン アプリケーションにアクセスします。**

   ユーザがオンライン アプリケーションにアクセスする場合、ユーザがログインする前であっても、ユーザの潜在的なリスクを評価できます。

2. **アプリケーションは、RiskFort に必要な情報を収集します。**

   この段階で、アプリケーションは `json.js` と呼ばれる RiskFort のユーティリティ スクリプトを使用してユーザのシステムから以下の情報を集める必要があります。この情報は、リスクの分析のために RiskFort によって使用されます。

   - オペレーティング システム、プラットフォーム、ブラウザ情報（ブラウザの言語、HTTP ヘッダ情報など）、ロケールおよび画面設定が含まれる**ユーザのシステム情報**。
   - Flash Shared Object （FSO）cookie またはブラウザ cookie などの DeviceID が含まれる**デバイス情報**。
   - IP アドレスおよび ISP 関連の情報が含まれる**場所情報**。
   - （**追加情報を使用する場合のオプション**）トランザクション量、ロケールおよび関連情報が含まれている可能性のある**追加の入力情報**。

3. **アプリケーションは RiskFort の `evaluateRisk()` 関数を呼び出します。**

   この段階で、アプリケーションは `riskfortAPI` の `evaluateRisk()` 関数を呼び出す必要があります。この呼び出しでは、前の手順で収集した情報をすべて RiskFort に渡す必要があります。

4. **RiskFort は、ユーザのリスク分析を実行します。**

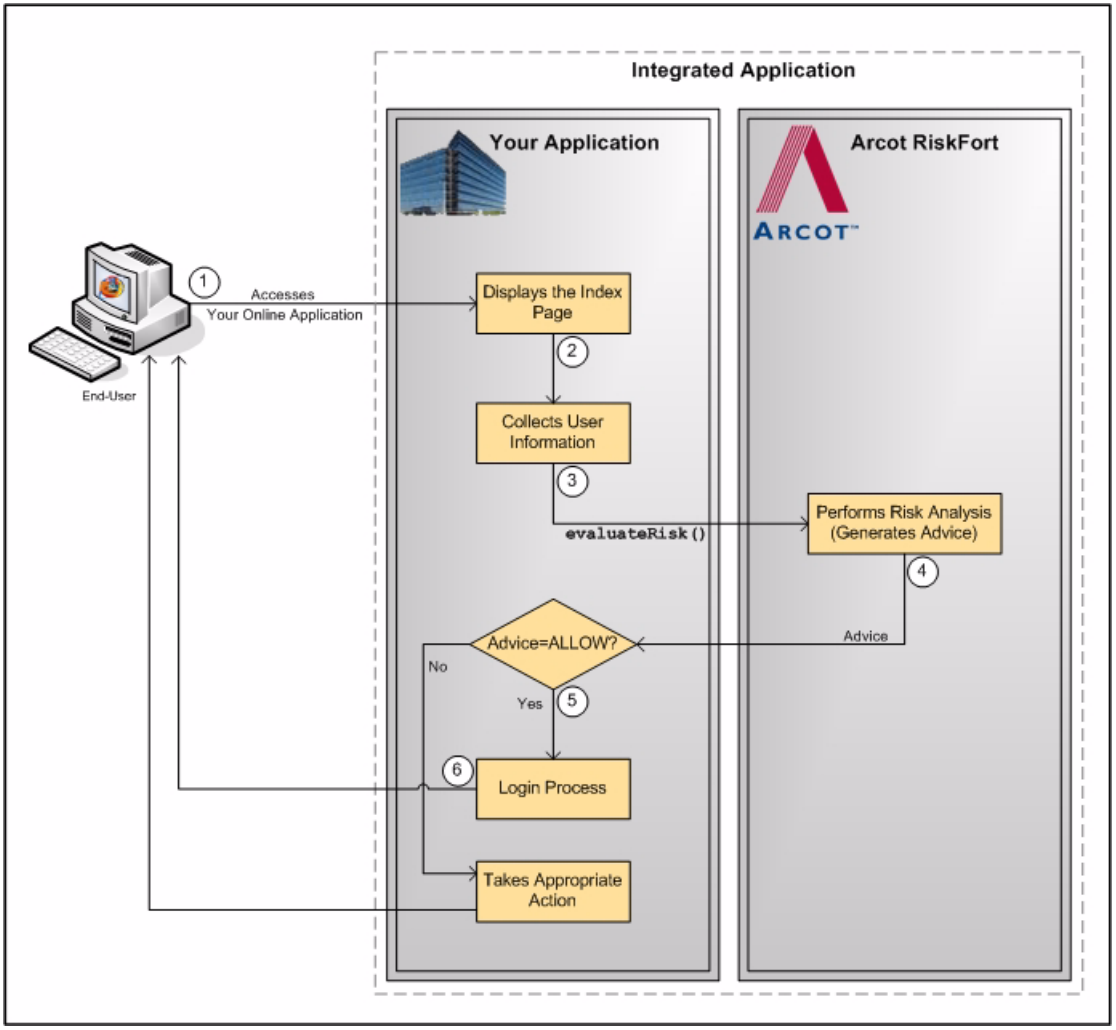
   RiskFort は、渡されたユーザの入力情報および設定されたルールに基づいて適切なリスク スコアおよびアドバイスを生成します。

5. **アプリケーションはユーザを検証します。**

   RiskFort のアドバイスに基づいて、アプリケーションは、ユーザのログイン プロセスの続行を許可するか、またはシステムへのアクセスを拒否することができます。

図 2-4 では、ログイン前のリスク評価ワークフローを示します。

図 2-4 ログイン前のリスク評価ワークフロー



ログイン後のリスク評価ワークフロー

ユーザがオンライン アプリケーションにアクセスする場合、最初にユーザをログインさせてから、このワークフローを実装することにより、ユーザの潜在的なリスクを包括的に評価できます。このワークフローは、デバイス ID 情報ならびにネットワーク情報、ユーザ情報およびトランザクション情報（実装されている場合）などの多数の要因を使用してユーザを評価します。

その後、リスク スコアおよびそれに続くアドバイスに基づいて、デバイス情報は RiskFort データベース内で更新されます。

- `ALLOW` の場合には、ユーザとデバイスの関連付け情報が更新されます。
- `ALERT` と `DENY` の場合には、ユーザとデバイスの関連付け情報がまったく更新*されません*。
- `INCREASEAUTH` の場合には、ユーザとデバイスの関連付け情報は更新されますが、ユーザの関連付け情報は追加の認証（2-24 ページの「2 次認証ワークフロー」）の結果が成功した場合にのみ作成されます。

ユーザのオンライン アプリケーションへのログインを認証した後で、RiskFort のリスク分析機能を呼び出す場合、リスク評価ワークフローは以下のとおりです。

1. **ユーザがオンライン アプリケーションにログインします。**

   このユーザがシステムに存在している場合、システムによって検証されます。ユーザが有効でない場合、アプリケーションは適切なアクションを行う必要があります。

2. **アプリケーションは、RiskFort に必要な情報を収集します。**

   この段階で、アプリケーションは `json.js` と呼ばれる RiskFort のユーティリティ スクリプトを使用してユーザのシステムから以下の情報を集める必要があります。この情報は、リスクの分析のために RiskFort によって使用されます。

   - オペレーティング システム、プラットフォーム、ブラウザ情報（ブラウザの言語、HTTP ヘッダ情報など）、ロケールおよび画面設定が含まれる**ユーザのシステム情報**。
   - Flash Shared Object （FSO）cookie またはブラウザ cookie などの DeviceID が含まれる**デバイス情報**。
   - IP アドレスおよび ISP 関連の情報が含まれる**場所情報**。
   - （**追加情報を使用する場合のオプション**）トランザクション量、ロケールおよび関連情報が含まれている可能性のある**追加の入力情報**。

3. **アプリケーションは RiskFort の `evaluateRisk()` 関数を呼び出します。**

   この段階で、アプリケーションは `riskfortAPI` の `evaluateRisk()` 関数を呼び出す必要があります。この呼び出しでは、前の手順で収集したユーザおよびデバイス情報をすべて RiskFort に渡す必要があります。

4. **RiskFort は、ユーザのリスク分析を実行します。**

   RiskFort は入力情報および設定されたルールを使用して、リスクを評価します。実行されたルールおよび情報が一致したかどうかの結果に基づいて、RiskFort は以下のアドバイスを生成します。

- ALERT、ユーザの情報が RiskFort データベースに存在しない場合。
- ALLOW、情報一致の信頼度が高い場合。
- DENY、情報一致の信頼度が低い場合。
- INCREASEAUTH、入力情報が疑わしい場合。

アドバイスが INCREASEAUTH である場合、続行する方法の詳細については、「2 次認証ワークフロー」を参照してください。

5. **アプリケーションは RiskFort のアドバイスを使用して、適切なアクションを行います。**

   evaluateRisk() 呼び出しの結果に基づいて、アプリケーションは、ユーザにトランザクションの続行を許可するか、保護されているリソースへのアクセスを拒否するか、または 2 次認証を実行します。

   詳細については、2-24 ページの「2 次認証ワークフロー」を参照してください。

6. **アプリケーションは RiskFort の `postEvaluate()` 関数を呼び出します。**

   この段階で、アプリケーションは riskfortAPI の postEvaluate() 関数を呼び出す必要があります。evaluateRisk() 呼び出しによって生成された出力情報に基づいて、この呼び出しは、RiskFort が最終アドバイスを生成し、かつデバイスおよび関連付け情報を更新できるようにします。

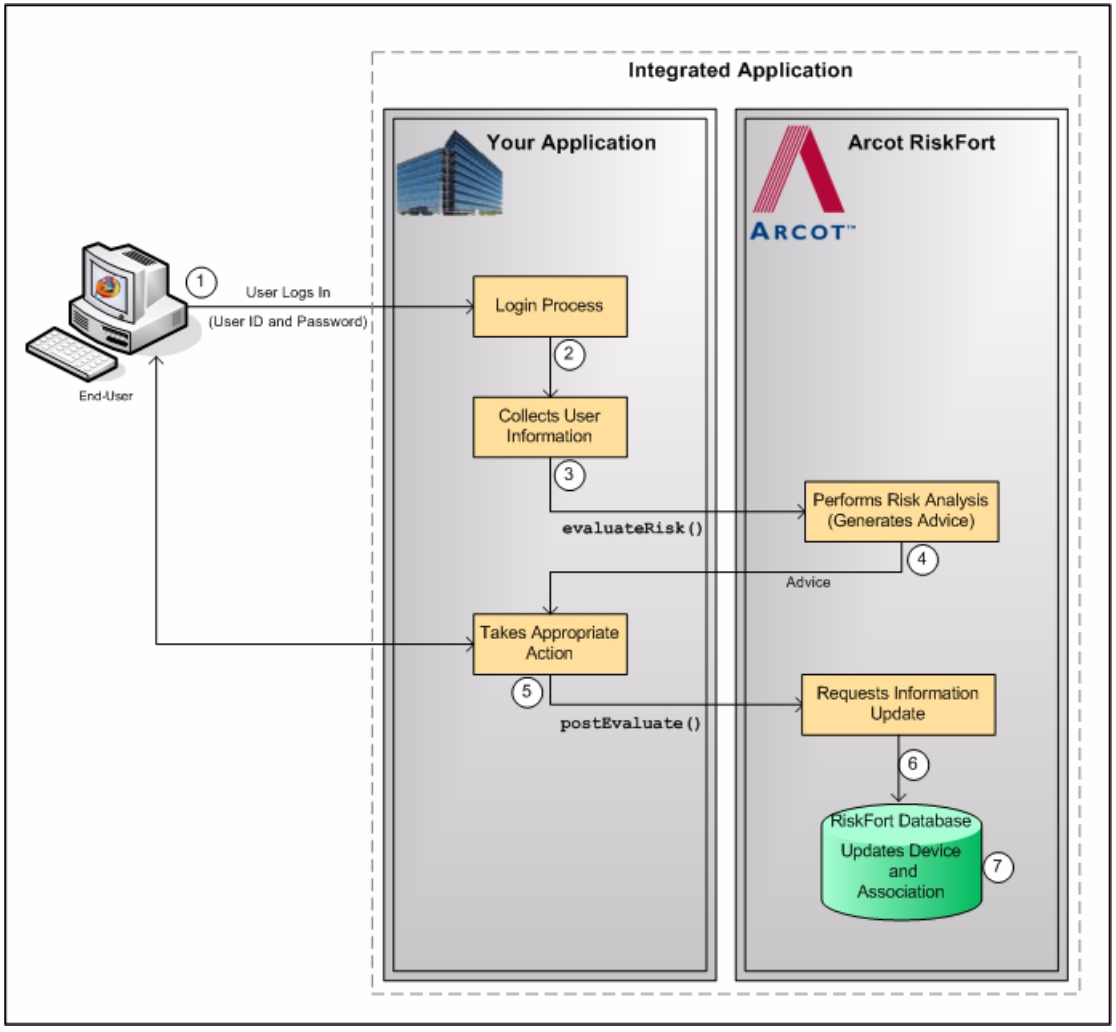
   この呼び出しでは、evaluateRisk() 呼び出しからのリスク スコアおよびアドバイス、2 次認証の結果（前の手順のアドバイスが INCREASEAUTH の場合）および任意の関連付けの名前（ユーザが指定している場合）を渡す必要があります。

7. **RiskFort はデバイスおよび関連付け情報を更新します。**

   入力データで変更が検出された場合、RiskFort は RiskFort データベースのデータおよび関連付け情報を更新します。

図 2-5 では、ログイン後のリスク評価ワークフローを示します。

図 2-5 ログイン後のリスク評価ワークフロー



2 次認証ワークフロー

RiskFort が INCREASEAUTH アドバイスを生成した場合、RiskFort は 2 次認証のためにアプリケーションにコントロールを一時的に移します。この場合、アプリケーションは、追加の認証を実行するためのメカニズムを実装する必要があります。たとえば、アプリケーションは、ユーザに業界標準の秘密の質問（母親の旧姓や生年月日など）を表示したり、ユーザに電話による帯域外認証を受けてもらうことができます。

ユーザの認証が成功したかどうかを特定したら、RiskFort に結果を転送する必要があります。RiskFort はこのフィードバックを使用して、最終アドバイスを生成したり、デバイス情報を更新したり、関連付け情報を作成したり、フィードバックを格納して今後のトランザクションのリスク分析に使用します。

2 次認証の場合のリスク評価ワークフローを以下に示します。

1. **ユーザがオンライン アプリケーションにログインします。**

   このユーザがシステムに存在している場合、システムによって検証されます。ユーザが有効でない場合、アプリケーションは適切なアクションを行う必要があります。

2. **アプリケーションは、RiskFort に必要な情報を収集します。**

   この段階で、アプリケーションは `json.js` と呼ばれる RiskFort のユーティリティ スクリプトを使用してユーザのシステムから以下の情報を集める必要があります。この情報は、リスクの分析のために RiskFort によって使用されます。

   - オペレーティング システム、プラットフォーム、ブラウザ情報（ブラウザの言語、HTTP ヘッダ情報など）、ロケールおよび画面設定が含まれる**ユーザのシステム情報**。
   - Flash Shared Object （FSO）cookie またはブラウザ cookie などの DeviceID が含まれる**デバイス情報**。
   - IP アドレスおよび ISP 関連の情報が含まれる**場所情報**。
   - （**追加情報を使用する場合のオプション**）トランザクション量、ロケールおよび関連情報が含まれている可能性のある**追加の入力情報**。

3. **アプリケーションは RiskFort の `evaluateRisk()` 関数を呼び出します。**

   この段階で、アプリケーションは `riskfortAPI` の `evaluateRisk()` 関数を呼び出す必要があります。この呼び出しでは、前の手順で収集したユーザおよびデバイス情報をすべて RiskFort に渡す必要があります。

4. **RiskFort は、ユーザのリスク分析を実行します。**

   RiskFort はトランザクションを疑わしいとしてフラグを立てた場合、`INCREASEAUTH` アドバイスを生成します。これは、ユーザをさらに認証するために追加のクレデンシャルが必要であることを示唆します。

5. **アプリケーションは 2 次認証を実行します。**

   2 次認証メカニズムに基づいて、アプリケーションは、ユーザに適切なページを表示します。たとえば、ユーザに以下のようなプロンプトを表示できます。

   - アプリケーションに登録したときに選択した秘密の質問に答える。

- ワンタイム パスワード（OTP）認証を実行する。
- 電話認証を実行する。

ユーザの入力情報を受信した後で、アプリケーションは追加の認証の結果を特定します。

6. **アプリケーションは RiskFort の `postEvaluate()` 関数を呼び出して、RiskFort に 2 次認証の結果を転送します。**

   この段階では、ユーザが 2 次認証に失敗したか、クリアしたかの事実に関係なく、アプリケーションは RiskFort に結果を渡す必要があります。この情報は、RiskFort が最新の正確なユーザ履歴を構築するのに役立ちます。

   これを行うには、アプリケーションは `riskfortAPI` の `postEvaluate()` 関数を呼び出す必要があります。この呼び出しでは、`evaluateRisk()` 呼び出しからリスク スコアおよびアドバイス、2 次認証の結果および任意の関連付けの名前（ユーザが指定した場合）を渡す必要があります。

7. **RiskFort は最終アドバイスを生成します。**

   2 次認証に関するアプリケーションのフィードバックを使用することにより、RiskFort は最終アドバイスを生成します。

8. **RiskFort はデバイス情報を更新し、関連付け情報を作成します。**

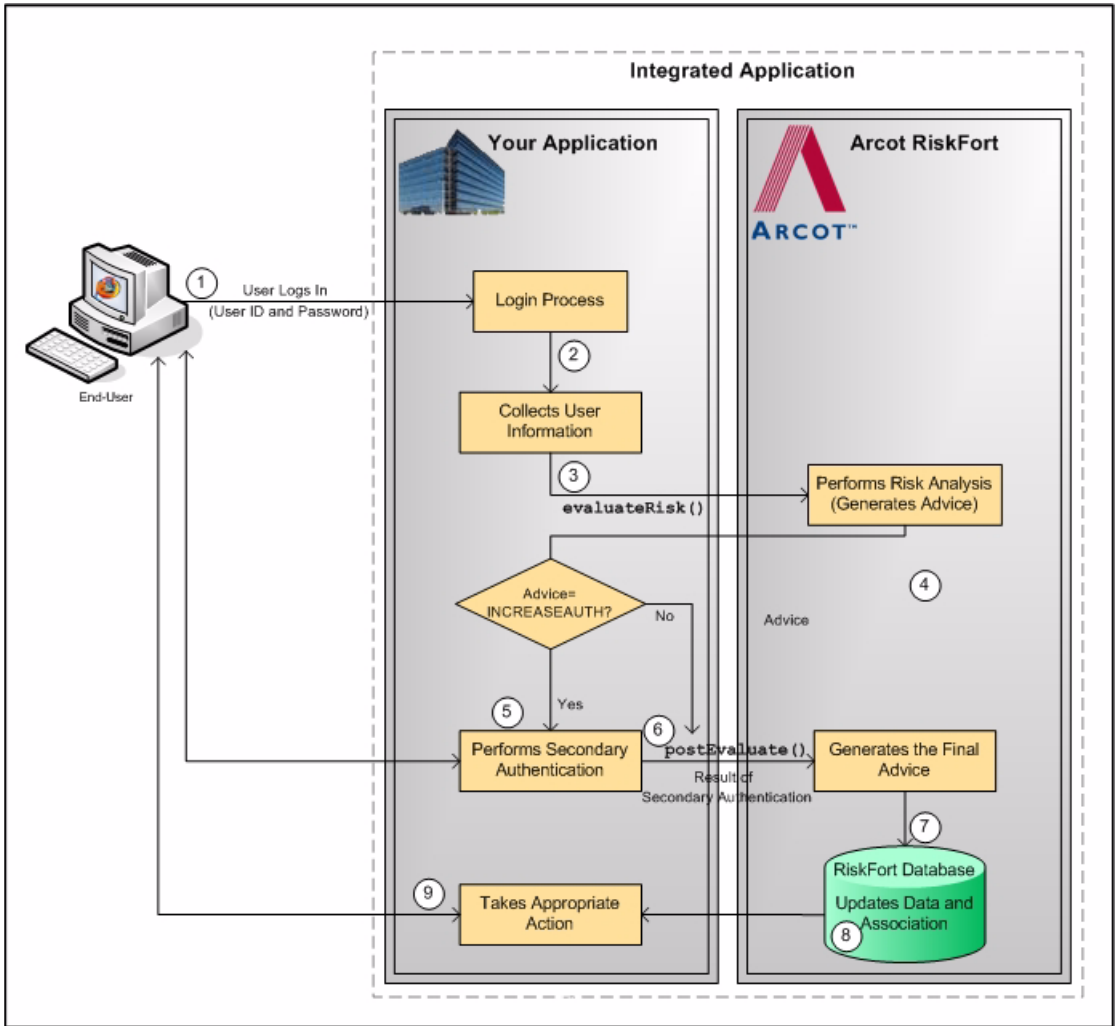
   `postEvaluate()` 呼び出しの結果に基づいて、RiskFort はデバイス属性を更新し、RiskFort データベースに関連付け情報を作成します。

9. **アプリケーションは適切なアクションを行います。**

   `postEvaluate()` 呼び出しの結果に基づいて、アプリケーションは、ユーザにトランザクションの続行を許可するか、保護されているリソースへのアクセスを拒否します。

図 2-6 に、2 次認証のリスク評価ワークフローを示します。

図 2-6 2 次認証のリスク評価ワークフロー



ワークフロー サマリ

表 2-1 では、RiskFort が提供するワークフローの概要について説明します。

注： 2 次認証ワークフローを除いて、これらのワークフローはすべて「バックグラウンド」で実装され、ユーザ エクスペリエンスに変更はありません。

表 2-1. RiskFort ワークフローの概要

ワークフロー	ワークフローのサブタイプ	説明	依存ワークフロー
登録	**明示的**	**ケース 1**：`evaluateRisk()` **の前に** `createUser()` を呼び出したときに、RiskFort データベースにユーザを作成します。 この場合、エンド ユーザが `ALERT` アドバイスを取得することはありません。	• ログイン後のリスク評価
		ケース 2：`evaluateRisk()` **の後で** `createUser()` を呼び出したときに、RiskFort データベースにユーザを作成します。	• ログイン後のリスク評価
	暗黙的	RiskFort は、**自動的に** RiskFort データベースにユーザを作成するため、`createUser()` 呼び出しを行う必要はありません。	• ログイン後のリスク評価
リスク評価	**ログイン前**	ユーザがオンライン アプリケーション システムにログインする前に、トランザクションのリスクを分析します。	None
	ログイン後	ユーザがオンライン アプリケーション システムにログインした後で、トランザクションのリスクを分析します。 また、ユーザ情報およびデバイス関連付け情報も更新します。	• 登録 • 2 次認証
	2 次認証の場合	RiskFort が `INCREASEAUTH` を推奨した後で、アプリケーションが 2 次認証を実行した場合、最終アドバイスを提供します。 また、ユーザ情報およびデバイス関連付け情報も更新します。	• ログイン後のリスク評価

第 3 章
事前準備

Issuance API または Risk Evaluation API を使用する前に、関連する JAR ファイルを CLASSPATH に追加する必要があります。アプリケーションでプロパティ ファイルを使用している場合、そのプロパティ ファイルも CLASSPATH に追加する必要があります。必要な JAR ファイルまたはプロパティ ファイルを CLASSPATH に追加したら、次に API を初期化する必要があります。

この章では、以下のトピックについて説明します。

Issuance API を統合する前に

このセクションでは、以下のトピックについて説明します。

Issuance JAR ファイルの CLASSPATH への追加

API を使用するには、CLASSPATH 変数に適切な Issuance JAR ファイルを追加する必要があります。以下の手順を実行します。

1. 必要に応じて、以下の表にリストされている Issuance JAR ファイルを <*APPLICATION_CONTEXT*> の適切なディレクトリ（lib など）にコピーする必要があります。

Windows の場合

表 3-1. CLASSPATH に追加する Issuance JAR ファイル（Windows の場合）

プライマリ JAR ファイル	arcot_core.jar arcot-riskfort-issuance.jar	<*install_location*>\Arcot Systems**sdk****java****lib****arcot**\
その他の JAR ファイル	bcprov-jdk14-139.jar commons-beanutils-1.7.0.jar commons-collections-3.1.jar commons-httpclient-3.1.jar commons-lang-2.0.jar commons-logging-1.0.4.jar commons-pool-1.4.jar dom4j-1.6.1.jar jaxen-1.1-beta-8.jar jdom-1.0.jar json-lib-0.7.1.jar log4j-1.2.9.jar servlet-api-2.4.jar oro-2.0.8.jar xalan-2.7.0.jar xercesImpl-2.6.2.jar xml-apis-1.0.b2.jar xmlParserAPIs-2.6.2.jar xom-1.1.jar	<install_location>\Arcot Systems**sdk****java****lib****external**\

UNIX ベースのプラットフォームの場合

表 3-2. CLASSPATH に追加する Issuance JAR ファイル（UNIX プラットフォームの場合）

プライマリ JAR ファイル	arcot_core.jar arcot-riskfort-issuance.jar	*<install_location>*/arcot/**sdk/java/lib/arcot/**
その他の JAR ファイル	bcprov-jdk14-131.jar commons-beanutils.jar commons-collections-3.1.jar commons-digester.jar commons-lang-2.0.jar commons-logging.jar commons-pool.jar log4j-1.2.9.jar servlet.jar sqljdbc.jar	*<install_location>*/arcot/**sdk/java/lib/external/**

2. JAR ファイルのコピー先ディレクトリの親ディレクトリ（*<APPLICATION_CONTEXT>*）を CLASSPATH 環境変数に追加します。

プロパティ ファイルの CLASSPATH への追加

アプリケーションでプロパティ ファイルを使用している場合は、riskfort.issuance.properties ファイルを CLASSPATH に追加する必要があります。これを行うには、以下の手順を実行します。

1. 次の場所から、properties ディレクトリを *<APPLICATION_CONTEXT>* の適切なディレクトリ（classes など）にコピーします。

   - **Windows の場合**

     <install_location>\Arcot Systems**sdk\java**

   - **UNIX プラットフォームの場合**

     <install_location>/arcot/**sdk/java/**

2. properties ディレクトリの親ディレクトリを CLASSPATH 環境変数に追加します。

Issuance API の初期化

com.arcot.riskfortissuanceAPI パッケージの UserRepManager クラスを使用して、Issuance API を初期化します。初期化プロセスではすべての接続プールが作成され、データベース プールが作成され、ロガーが初期化されます。

注： API を初期化した後に、設定の変更を適用することは *できません*。設定を変更するには、API を再度初期化する必要があります。

以下のサブセクションで説明するように、UserRepManager クラスには Issuance API を初期化するための 2 つの方法があります。

方法 1

プロパティ ファイルを使用した API の初期化

initialize(java.lang.String propertyLocation) メソッドは、入力のプロパティ ファイルにリストされたパラメータを使用して Issuance API を初期化します。NULL が渡されると、パラメータは riskfort.issuance.properties ファイルから読み取られます。このファイルは CLASSPATH の properties フォルダにあります。

プロパティ ファイルのフィールドおよびフォーマットは、以下のようにする必要があります。

```
//RiskFort サーバの IP アドレス
HOST.1=

//RiskFort サーバのポート番号
PORT.1=

// 接続のタイプ 使用可能な値は TLS および TCP です。
TRANSPORT_TYPE=

// 以下は TRANSPORT_TYPE を TLS に設定した場合に必要です。ファイルは
//PEM 形式である必要があります。ファイルの完全なパスを指定します。
```

```
CA_CERT_FILE=
```

表 3-3 に、initialize() メソッドの詳細を示します。

表 3-3. Issuance プロパティ ファイルを使用した API の初期化

説明	入力パラメータ	出力値
指定されたプロパティ ファイルを使用して Issuance API を初期化します。	• propertyLocation プロパティ ファイルの絶対パス。	API が正常に初期化されなかった場合は、UserRepositoryException がスローされます。

方法 2

マップを使用した API の初期化

UserRepManager クラスの initialize(java.util.Map properties) メソッドは、入力されたマップに基づいて Issuance API を初期化します。

表 3-4 に、initialize() メソッドの詳細を示します。

表 3-4. マップを使用した API の初期化

説明	入力値	出力値
指定されたマップを使用して Issuance API を初期化します。	• map 設定情報を指定するキーと値のペア。サポートされているキーは以下のとおりです。 •HOST.1 RiskFort サーバを使用可能なホストの IP アドレス。 •PORT.1 RiskFort サーバを使用可能なポート。デフォルト値は 7680 です。 •TRANSPORT_TYPE 接続のタイプ。選択可能な値は SSL と TCP です。 •CA_CERT_FILE （TRANSPORT_TYPE を SSL に設定した場合にのみ必要）サーバの CA 証明書ファイル用のパス。このファイルは PEM 形式である必要があります。ファイルの完全なパスを指定します。	API が正常に初期化されなかった場合は、UserRepositoryException がスローされます。

追加入力の準備

Issuance API では、必須入力に加えて name-value のペア形式の追加入力も受け入れます。この入力には、ロケール、呼び出し元アプリケーションの詳細、または他のユーザ関連の詳細などの情報を追加することができます。このような name-value の追加のカスタム パラメータを使用すると、各トランザクションからの入力をリアルタイムでキャプチャすることができます。また、これらのパラメータは RiskFort データベースで正常にユーザを作成するために、展開されたアドオン ルールによって処理されます。

RiskFort の com.arcot.riskfortissuanceAPI.AdditionalInputs パッケージでは AdditionalInputs クラスを提供しています。このクラスを使用して、使用する追加情報を設定できます。Issuance の AdditionalInputs クラスによってサポートされている事前定義済みの追加入力パラメータには、以下のものが含まれます。

- AR_RF_LOCALE_ID

  呼び出し元アプリケーションにメッセージを返す場合に RiskFort が使用するロケールを指定します。

- AR_RF_CALLER_ID

  呼び出し元アプリケーションでのトランザクション識別子を指定します。これは、エンド ツー エンドのトランザクションを追跡する場合に役立ちます。

ヒント： AdditionalInputs クラスには*任意*のパラメータを追加できます。また、このパラメータをアドオン ルールまたは評価コールアウトに渡すことができます。

カスタムの Issuance パラメータを実装する方法

1. AdditionalInputs() メソッドを使用して、AdditionalInputs クラスを実装するオブジェクトを取得します。
2. put() メソッドを使用して、返されるオブジェクトに必要な入力を設定します。メソッドの構文は以下のとおりです。

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

   指定した *name* の追加入力が*存在しない*場合、追加入力が作成されます。それ以外の場合は、新しい *value* で上書きされます。*name* と *value* のどちらにも長い文字列は使用しないことをお勧めします。

注： *name* および *value* のパラメータには = および改行文字（\n）を*使用しないでください*。*name* または *value* にこれらのいずれかの文字が含まれた場合の API 動作は未定義です。

Risk Evaluation API を統合する前に

このセクションでは、以下のトピックについて説明します。

- Risk Evaluation JAR ファイルの CLASSPATH への追加
- プロパティ ファイルの CLASSPATH への追加
- Risk Evaluation API の初期化
- 追加入力の準備

Risk Evaluation JAR ファイルの CLASSPATH への追加

API を使用するには、CLASSPATH 変数に適切な Risk Evaluation JAR ファイルを追加する必要があります。以下の手順を実行します。

1. 必要に応じて、以下の表にリストされている Risk Evaluation JAR ファイルを *<APPLICATION_CONTEXT>* の適切なディレクトリ（lib など）にコピーする必要があります。

Windows の場合

表 3-5. CLASSPATH に追加する Risk Evaluation JAR ファイル（Windows の場合）

プライマリ JAR ファイル	arcot_core.jar arcot-riskfort-evaluaterisk.jar arcot-riskfort-mfp.jar	*<install_location>*\Arcot Systems**sdk****java****lib****arcot**\
その他の JAR ファイル	bcprov-jdk14-131.jar commons-beanutils.jar commons-collections-3.1.jar commons-digester.jar commons-lang-2.0.jar commons-logging.jar commons-pool.jar log4j-1.2.9.jar servlet.jar sqljdbc.jar	*<install_location>*\Arcot Systems**sdk****java****lib****external**\

UNIX ベースのプラットフォームの場合

表 3-6. CLASSPATH に追加する Risk Evaluation JAR ファイル（UNIX の場合）

プライマリ JAR ファイル	arcot_core.jar arcot-riskfort-evaluaterisk.jar arcot-riskfort-mfp.jar	<*install_location*>/arcot/**sdk**/**java**/**lib**/**arcot**/
その他の JAR ファイル	bcprov-jdk14-131.jar commons-beanutils.jar commons-collections-3.1.jar commons-digester.jar commons-lang-2.0.jar commons-logging.jar commons-pool.jar log4j-1.2.9.jar servlet.jar sqljdbc.jar	<*install_location*>/arcot/**sdk**/**java**/**lib**/**external**/

2. JAR ファイルのコピー先ディレクトリの親ディレクトリ（<*APPLICATION_CONTEXT*>）を CLASSPATH 環境変数に追加します。

プロパティ ファイルの CLASSPATH への追加

アプリケーションでプロパティ ファイルを使用している場合は、riskfort.risk-evaluation.properties ファイルを CLASSPATH に追加する必要があります。これを行うには、以下の手順を実行します。

1. 次の場所から、properties ディレクトリを <*APPLICATION_CONTEXT*> の適切なディレクトリ（classes など）にコピーします。

   - **Windows の場合**

     <*install_location*>\Arcot Systems**sdk****java**\

   - **UNIX プラットフォームの場合**

     <*install_location*>/arcot/**sdk**/**java**/

2. properties ディレクトリの親ディレクトリを CLASSPATH 環境変数に追加します。

Risk Evaluation API の初期化

com.arcot.riskfortAPI パッケージの RiskFactory クラスを使用して、Risk Evaluation API を初期化します。初期化プロセスではすべての接続プールが作成され、データベース プールが作成され、ロガーが初期化されます。初期化されると、呼び出し元のアプリケーションに適切なオブジェクトが返されます。

注 : API を初期化した後に、設定の変更を適用することは *できません*。設定を変更するには、API を再度初期化する必要があります。

以下のサブセクションで説明するように、RiskFactory クラスには Risk Evaluation API を初期化するための 2 つの方法があります。

方法 1

プロパティ ファイルを使用した API の初期化

initialize(java.lang.String propertyLocation) メソッドは、入力のプロパティ ファイルにリストされたパラメータを使用して Risk Evaluation API を初期化します。NULL が渡されると、パラメータは riskfort.risk-evaluation.properties ファイルから読み取られます。このファイルは CLASSPATH の properties フォルダにあります。

このプロパティ ファイルのフィールドおよびフォーマットは、以下のようにする必要があります。

```
//RiskFort サーバの IP アドレス
HOST.1=

//RiskFort サーバのポート番号
PORT.1=

// 接続のタイプ 使用可能な値は TLS および TCP です。
TRANSPORT_TYPE=

// 以下は TRANSPORT_TYPE を TLS に設定した場合に必要です。ファイルは
//PEM 形式である必要があります。ファイルの完全なパスを指定します。
CA_CERT_FILE=
```

表 3-7 に、initialize() メソッドの詳細を示します。

表 3-7. Risk Evaluation プロパティ ファイルを使用した API の初期化

説明	入力パラメータ	出力値
指定されたプロパティ ファイルを使用して Risk Evaluation API を初期化します。	• propertyLocation プロパティ ファイルの絶対パス。	API が正常に初期化されなかった場合は、UserRepositoryException がスローされます。

方法 2

マップを使用した API の初期化

RiskFactory クラスの initialize(java.util.Map properties) メソッドは、入力されたマップに基づいて Risk Evaluation API を初期化します。表 3-8 に、initialize() メソッドの詳細を示します。

表 3-8. マップを使用した API の初期化

説明	入力値	出力値
入力されたマップを使用して Risk Evaluation API を初期化します。	• map 設定情報を指定するキーと値のペア。サポートされているキーは以下のとおりです。 •HOST.1 RiskFort サーバを使用可能なホストの IP アドレス。 •PORT.1 RiskFort サーバを使用可能なポート。デフォルト値は 7680 です。 •TRANSPORT_TYPE 接続のタイプ。選択可能な値は SSL と TCP です。 •CA_CERT_FILE （TRANSPORT_TYPE を SSL に設定した場合にのみ必要）サーバの CA 証明書ファイル用のパス。このファイルは PEM 形式である必要があります。ファイルの完全なパスを指定します。	API が正常に初期化されなかった場合は、UserRepositoryException がスローされます。

追加入力の準備

Risk Evaluation API では、必須入力に加えて name-value のペア形式の追加入力も受け入れます。この入力には、ロケール、呼び出し元アプリケーションの詳細、またはリスク評価に使用できる他のトランザクション関連の詳細などの情報を追加することができます。このような name-value の追加のカスタム パラメータを使用すると、各トランザクションからの入力をリアルタイムでキャプチャすることができます。また、これらのパラメータはリスク評価を正常に行うために、展開されたアドオン ルールによって処理されます。

RiskFort の com.arcot.riskfortissuanceAPI.AdditionalInputs パッケージでは AdditionalInputs クラスを提供しています。このクラスを使用して、使用する追加情報を設定できます。Risk Evaluation の AdditionalInputs クラスによってサポートされている事前定義済みの追加入力パラメータには、以下のものが含まれます。

- AR_RF_LOCALE_ID

  呼び出し元アプリケーションにメッセージを返す場合に RiskFort が使用するロケールを指定します。

- AR_RF_CALLER_ID

  呼び出し元アプリケーションでのトランザクション識別子を指定します。これは、エンド ツー エンドのトランザクションを追跡する場合に役立ちます。

ヒント： AdditionalInputs クラスには*任意*のパラメータを追加できます。また、このパラメータをアドオン ルールまたは評価コールアウトに渡すことができます。

カスタムのリスク評価パラメータを実装する方法

1. AdditionalInputs() メソッドを使用して、AdditionalInputs クラスを実装するオブジェクトを取得します。
2. put() メソッドを使用して、返されるオブジェクトに必要な入力を設定します。メソッドの構文は以下のとおりです。

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

   指定した *name* の追加入力が存在しない場合、追加入力が作成されます。それ以外の場合は、新しい *value* で上書きされます。*name* と *value* のどちらにも長い文字列は使用しないことをお勧めします。

注： *name* および *value* のパラメータには = および改行文字（\n）を*使用しないでください*。*name* または *value* にこれらのいずれかの文字が含まれた場合の API 動作は未定義です。

第 4 章
ユーザの管理

RiskFort に新しいユーザを作成する操作を行うのは、新規ユーザを RiskFort データベースに追加する場合の 1 回限りです。通常、アプリケーションの既存のユーザが初めて RiskFort にアクセスする場合に、ユーザを追加します。

Issuance Java API はプログラム可能なインターフェースを提供します。これを使用して Java クライアント（Java サーブレットや JSP ページなど）は発行に関連するリクエストを RiskFort サーバに送信します。この API は、RiskFort サーバに送信するリクエスト メッセージを作成し、レスポンスを受信し、リターン レスポンスとしてそれをパッケージ化してアプリケーションで読み取れるようにします。

この章では、Java API を使用して RiskFort データベースにユーザを作成する方法、およびそれによって実装されるメソッドとインターフェースの概要について説明します。次に、ユーザの詳細の読み取りおよび更新に使用されるこの API 内のインターフェースとメソッドについて説明します。ユーザ関連の各タスクの説明（ユーザ情報の作成、読み取り、または更新）の後に、タスクをテストするために使用できるサンプル コード スニペットを示します。この章では以下のトピックについて説明します。

ユーザの作成

RiskFort データベースにユーザを作成するには、`User` と `UserRepositoryService` のインターフェースを使用する必要があります。

ユーザの作成方法

1. `RiskFactory` クラスを使用して、Issuance API が初期化されていることを確認します。

   詳細については、3-32 ページの「Issuance API の初期化」を参照してください。

2. 必要に応じて、`com.arcot.riskfortissuanceAPI.AdditionalInputs` パッケージ内に `AdditionalInputs` クラスを使用することによって、トランザクションに追加の入力を準備します。

   詳細については、3-34 ページの「追加入力の準備」を参照してください。

3. `UserManager.getNewUserObject(String userName, String groupName)` メソッドを使用し、`User` インターフェースを実装するオブジェクトを取得します。

   このメソッドは、ユーザ名と指定のグループが設定された `User` オブジェクトを返します。

4. `user.setProperty()` メソッドを使用して、返されたオブジェクトの必要なプロパティを設定します。

5. `UserRepManager.getGrpUserRepService()` メソッドを使用して、`UserRepositoryService` インターフェースを実装するオブジェクトを取得します。

   このメソッドは、`UserRepositoryService` インターフェースのインスタンスを返し、次に指定されたグループの内部リポジトリ実装を返します。

6. 最後に、ユーザを作成する `UserRepositoryService` インターフェースの `create()` メソッドを呼び出します。

   このメソッドは、`UserRepResult` クラスのインスタンスを返します。これは応答コード（`FAILURE`、`INVALID` または `SUCCESS`）、オプションのメッセージ、および RiskFort サーバ側で生成されたトランザクション識別子を指定します。

エラーの処理

いずれかの Issuance API メソッドの実行中に発生したエラーはすべて、`UserRepositoryException` 例外としてスローされます。`UserRepositoryException` オブジェクトには、発生したエラーに関連付けられたエラー コードと（RiskFort サーバ側で生成された）トランザクション識別子を含む `getCode` と `getTransactionId()` の各メンバが含まれています。

ユーザ作成のためのサンプル コード

注： 詳細な実用コードのサンプルについては、付録 B の「ユーザ操作のためのサンプル コード」を参照してください。

以下のサンプル コード スニペットを使用して、RiskFort Database にユーザを作成するためのコードを実装する方法を理解してください。

```
public void createUser(User user, AdditionalInput additionalInput) throws
UserRepositoryException
{
    //Gets the implementation user repository service.
    UserRepositoryService srv = UserRepManager.getGrpUserRepService();

    //Get the implementation of User interface.
    User user = UserManager.getNewUserObject(userName, orgName);

    //set user properties
    user.setProperty(User.USER_PROPERTY_FIRSTNAME, firstName);
    user.setProperty(User.USER_PROPERTY_LASTNAME, lastName);
    user.setProperty(User.USER_PROPERTY_EMAILADDR, email);
    user.setPAM(pam);
    srv.create(user, additionalInput);
 }
```

ユーザ詳細の読み取り

RiskFort データベースのユーザの詳細を読み取るには、`User` と `UserRepositoryService` のインターフェースを使用する必要があります。

ユーザ詳細を読み取る方法

1. RiskFactory クラスを使用して、Issuance API が初期化されていることを確認します。

   詳細については、3-32 ページの「Issuance API の初期化」を参照してください。

2. 必要に応じて、com.arcot.riskfortissuanceAPI.AdditionalInputs パッケージ内に AdditionalInputs クラスを使用することによって、トランザクションに追加の入力を準備します。

   詳細については、3-34 ページの「追加入力の準備」を参照してください。

3. UserRepManager.getGrpUserRepService() メソッドを使用して、UserRepositoryService インターフェースを実装するオブジェクトを取得します。

   このメソッドは、UserRepositoryService インターフェースのインスタンスを返し、次に指定されたグループの内部リポジトリ実装を返します。

4. 最後に、User インターフェースの read() メソッドを呼び出してユーザ詳細を読み取ります。

   このメソッドは完全に読み込まれた User オブジェクトを返し、操作が FAILURE または SUCCESS かどうか指定します。

エラーの処理

いずれかの Issuance API メソッドの実行中に発生したエラーはすべて、UserRepositoryException 例外としてスローされます。UserRepositoryException オブジェクトには、発生したエラーに関連付けられたエラー コードと（RiskFort サーバ側で生成された）トランザクション識別子を含む getCode と getTransactionId() の各メンバが含まれています。

ユーザの詳細を読み取るためのサンプル コード

以下のサンプル コード スニペットをアプリケーション コードに挿入して、RiskFort データベースからユーザの詳細を読み取ることができます。

```
private User read (String userName, String orgName) throws
UserRepositoryException
{
    UserRepositoryService srv = UserRepManager.getGrpUserRepService();
    User user = srv.read(userName, orgName, additionalInput);
    return user;
}
```

ユーザの詳細の更新

RiskFort データベースのユーザ情報を更新するには、`User` と `UserRepositoryService` のインターフェースおよび `UserManager` クラスを使用する必要があります。

ユーザの詳細を更新する方法

1. `RiskFactory` クラスを使用して、Issuance API が初期化されていることを確認します。

   詳細については、3-32 ページの「Issuance API の初期化」を参照してください。

2. 必要に応じて、`com.arcot.riskfortissuanceAPI.AdditionalInputs` パッケージ内に `AdditionalInputs` クラスを使用することによって、トランザクションに追加の入力を準備します。

   詳細については、3-34 ページの「追加入力の準備」を参照してください。

3. `UserRepManager.getGrpUserRepService()` メソッドを使用して、`UserRepositoryService` インターフェースを実装するオブジェクトを取得します。

   このメソッドは、`UserRepositoryService` インターフェースのインスタンスを返し、次に指定されたグループの内部リポジトリ実装を返します。

4. `UserManager.getUserDetails(userName, groupName)` メソッドを使用し、`User` インターフェースを実装するオブジェクトを取得します。

   このメソッドは、ユーザ名と指定のグループが設定された `User` オブジェクトを返します。

5. `user.setProperty()` メソッドを使用して、返されたオブジェクトの必要なプロパティを確認して設定します。

6. 最後に、`UserRepositoryService` インターフェースの `update()` メソッドを呼び出してユーザ詳細を更新します。

   このメソッドは、`UserRepResult` クラスのインスタンスを返します。これは応答コード（`FAILURE`、`INVALID` または `SUCCESS`）、オプションのメッセージ、および RiskFort サーバ側で生成されたトランザクション識別子を指定します。

エラーの処理

いずれかの Issuance API メソッドの実行中に発生したエラーはすべて、UserRepositoryException 例外としてスローされます。UserRepositoryException オブジェクトには、発生したエラーに関連付けられたエラー コードと（RiskFort サーバ側で生成された）トランザクション識別子を含む getCode と getTransactionId() の各メンバが含まれています。

ユーザの詳細を更新するためのサンプル コード

以下のサンプル コード スニペットをアプリケーション コードに挿入して、RiskFort データベース内のユーザの詳細を更新することができます。

```
private String updateUser(User user, AdditionalInput additionalInput)
{
    UserRepositoryService srv = UserRepManager.getGrpUserRepService();
    String errorCode = null;
    User user;
    try {
              user = UserManager.read(userName, orgName, additionalInput);
          }
    catch (UserRepositoryException e1)
        {
              errorCode = e1.getCode();
              return errorCode;
        }
    if (fname != null)
              user.setProperty(User.USER_PROPERTY_FIRSTNAME, fname);
    if (lname != null)
              user.setProperty(User.USER_PROPERTY_LASTNAME, lname);
    if (email != null)
              user.setProperty(User.USER_PROPERTY_EMAILADDR, email);
    if (pam != null)
              user.setPAM(pam);


    try
```

```
        {
              srv.update(user, additionalInput);
         }
    catch (UserRepositoryException e) {
              errorCode = e.getCode();
         }
    return errorCode;
}
```

第 5 章
デバイス ID とマシン FingerPrint の収集

RiskFort では、パラメータの 1 つとしてユーザとデバイス（デスクトップ コンピュータ、ラップトップ、およびノート）の組み合わせ情報を使用し、ログインの試行やトランザクションに関連するリスクを判断します。そのため、エンド ユーザのオンライン上での同一性の検証が課題となります。また、RiskFort では、この目的で（他の入力情報に加えて）デバイス ID および MFP 技術（第 2 章の「RiskFort ワークフローについて」で説明）も使用します。RiskFort はこれらの技術を使用してユーザ プロファイルを作成し、ユーザには意識させずに正確な結果を提供します。これにはユーザがすでに持っているハードウェアが使用され、エンド ユーザ エクスペリエンスが著しく変更されることはありません。

この章では、デバイス ID を取得して設定し、エンド ユーザのデバイスから MFP データを収集して、それを RiskFort へ渡す方法について詳しく説明します。この章では、以下のトピックについて説明します。

デバイス ID と MFP の基礎知識

このセクションでは、デバイス ID と MFP の基礎知識について説明します。

デバイス ID

デバイス ID とは、デバイス識別子の文字列です。RiskFort はこのデバイス ID を生成し、cookie としてエンド ユーザのシステムに保存して、エンド ユーザがオンライン アプリケーションにログインしたりトランザクションを実行する場合に使用するデバイスを識別して、追跡します。この情報には暗号化された形式が使用されます。

RiskFort のデバイス ID は以下の形式で格納できます。

- **Flash cookie**：.sol または .ssl（通信に SSL が使用されている場合）の拡張子で保存される Flash Shared Object（FSO）。通常 Windows では、この cookie は以下のようなユーザ プロファイルの Flash Player ディレクトリにあります。

  <*user_profile*>\Application Data\Macromedia\FlashPlayer**#SharedObjects**\

  前述のパスでは、通常、<*user_profile*> は C:\Documents and Settings\<user_name>\ を表します。

- **ブラウザ cookie**：HTTP cookie。拡張子と保存場所はエンド ユーザが使用しているブラウザにより異なります。通常 Windows では、この cookie は以下の場所にあります。

  - **Microsoft IE**

    C:\Documents and Settings\<*user_name*>\Local Settings**Temporary Internet Files**\

  - **Mozilla Firefox**

    C:\Documents and Settings\<*user_name*>\Application Data\Mozilla\Firefox\Profiles\<*value*>.default**cookies.sqlite**

  - **Safari**

    C:\Documents and Settings\<*user_name*>\Application Data\Apple Computer\Safari**cookies.plist**

Flash cookie を推奨する理由

ほとんどのブラウザ cookie は、有効期限が設定されているか、アクティビティがなくなって一定の期間が経過すると自動的に削除されます。また、ブラウザ cookie は、ユーザによって誤って削除されたり、クリアされる可能性があります。このような場合、RiskFort は INCREASEAUTH アドバイスを生成します。その結果、デバイス情報が RiskFort データベースに存在するにも関わらず、ユーザは 2 次認証を実行する必要があります。

ブラウザ cookie とは異なり、Flash cookie には有効期限がなく、手動で削除されるまで保持されます。また、一般的なユーザはブラウザ インターフェースからこれらの FSO を簡単に見つけて削除することはできないため、ユーザが誤って Flash cookie を削除してしまう可能性はブラウザ cookie よりはるかに低くなります。

さらに、Arcot が提供する Flash ソリューションを使用すると、ブラウザ cookie よりも確実にドメイン間攻撃に対応できます。

MFP

マシン FingerPrint （デバイス フィンガープリント、または PC フィンガープリントとも呼ばれる）は、ブラウザ、オペレーティング システム、その他の属性など、ユーザのシステム情報の収集に使用されるデバイス識別方法です。RiskFort はこの情報を分析して、リアル タイムでデバイスのリスク プロファイルを生成します。

RiskFort がエンド ユーザのデバイスから収集する MFP 属性には、以下のものが含まれます。

- オペレーティング システム名およびバージョン
- ブラウザ情報（名前、メジャー バージョン、マイナー バージョン、JavaScript バージョン、HTTP ヘッダ、ユーザ エージェントなど）
- 画面設定（高さ、幅、色深度など）
- システム情報（タイム ゾーン、言語、システム ロケールなど）

RiskFort は、エンド ユーザによるトランザクションごとに、データベースに格納されている対応する MFP と、受信した MFP を照合します。この照合率（%）が Administration Console のその他のルールの設定ページで指定した値よりも小さい場合、そのトランザクションは危険であると考えられます。

必要なファイル

エンド ユーザのデバイスのデバイス ID と MFP 情報を収集するには、表 5-1 にリスト表示されたファイルが必要です。このファイルは、RiskFort をインストールすると使用可能になります。

表 5-1. デバイス ID およびマシン FingerPrint （MFP）情報の収集に必要なファイル

場所	ファイル名	適用可能な cookie タイプ	説明
Windows の場合 <*install_location*>\Arcot Systems**sdk****javascript**\ **Solaris の場合** <*install_location*>/arcot/**sdk**/**javascript**/	rfutil.js	Flash	このファイルは、デバイス ID の Flash cookie を取得および設定するための JavaScript 関数を提供します。
		ブラウザ	このファイルは、HTTP cookie を get および set するための JavaScript 関数を提供します。

表 5-1. デバイス ID およびマシン FingerPrint （MFP）情報の収集に必要なファイル

場所	ファイル名	適用可能な cookie タイプ	説明
Windows の場合 <*install_location*>\Arcot Systems**sdk****javascript**\ **Solaris の場合** <*install_location*>/arcot/**sdk**/**javascript**/	mfp_json.js	Flash ブラウザ	このファイルには、エンド ユーザのデバイスから MFP 関連の情報を収集し、すべてのマシン FingerPrint の値を持つエンコードされた 1 つの文字列を生成するための JavaScript 関数が含まれます。 アプリケーションは、このファイルを追加して、クライアント側で jsonSignature() メソッドを呼び出す必要があります。アプリケーション側では buildDeviceSignature() メソッドを呼び出し、それを RiskFort が理解できるフォーマットに変換する必要があります。 evaluateRisk() メソッドを呼び出す*前*に、これを行う必要があります。 このメソッドは、インデックス ページまたはログイン ページなど、オンライン アプリケーションがリスク評価を行う必要があるページから呼び出す必要があります。

表 5-1. デバイス ID およびマシン FingerPrint （MFP）情報の収集に必要なファイル

場所	ファイル名	適用可能な cookie タイプ	説明
Windows の場合 *<install_location>*\Arcot Systems**sdk****flash**\ **Solaris の場合** *<install_location>*/arcot/**sdk**/**flash**/	rfdevice.swf	Flash	このファイルには、Flash cookie を管理するためのコードが含まれます。また、このファイルによって cookie へのドメイン間アクセスがサポートされ、あるドメインのアプリケーションのページが、別のドメインで設定された cookie にアクセスできるようになります。 **重要：**すべてのオンライン アプリケーション ページで、同一で完全な URL を使用して rfdevice.swf を参照する必要があります。
Windows の場合 *<install_location>*\Arcot Systems**sdk****flash**\ **Solaris の場合** *<install_location>*/arcot/**sdk**/**flash**/	crossdomain.txt	Flash	このファイルには、Flash cookie にアクセスできるドメインのリストを指定します （デフォルトでは、RiskFort の Flash ムービーを提供しているドメインのサブドメインのみが、Flash cookie にアクセスできます）。 crossdomain.txt のドメイン エントリのフォーマットは以下のとおりです。 &domainName=<pipe-separated_domain_list>& **重要：**このファイルは rfdevice.swf と同じ場所に存在する必要があります。

デバイス ID の設定

この章の最初に説明したように、デバイス ID は、現在のトランザクションでエンド ユーザが使用しているデバイスを検証するための重要な側面です。そのため、ユーザ コンピュータに Flash cookie またはブラウザ（HTTP）cookie を設定する必要があります。このセクションでは、エンド ユーザ デバイスに設定される、必要な cookie タイプの設定方法について説明します。

- Flash cookie の設定
- HTTP cookie の設定

Flash cookie の設定

Flash cookie をエンド ユーザのコンピュータに設定するには、以下のファイルが必要です。

- `rfutil.js`
- `rfdevice.swf`
- `crossdomain.txt`

Flash cookie を設定するには、以下のタスクを実行します。

1. Flash cookie を `get` または `set` するアプリケーション ページに、`rfutil.js` を追加します。
   a. `rfutil.js` を適切な Web アプリケーション フォルダ（`$APP_SERVER_HOME/`**`javascript`**`/` など）にコピーします。このフォルダは、`rfutil.js` ファイルを追加しようとしているページが存在する場所に対して相対的に表します。
   b. 以下の JavaScript コードをアプリケーションの該当する Web ページに追加します。

```
<script type="text/javascript" src="location_to_rfutil.js/"></script>
```

   前述のコード スニペットで、*`location_to_rfutil.js`* は `rfutil.js` への相対パスに置き換えます。

2. rfdevice.swf と crossdomain.txt を適切な Web アプリケーション フォルダ（*$APP_SERVER_HOME*/**JavaScript**/ など）にコピーします。

重要： crossdomain.txt ファイルは rfdevice.swf と同じ場所内にある*必要があります。*

3. Flash cookie が複数のドメインからアクセスされる場合は、以下のフォーマットで crossdomain.txt にドメインのリストを追加します。

```
&domainName=<pipe-separated_domain_list>&
```

たとえば、自分のサイトとパートナーのサイトからのページを集約した Web サイトの場合、エントリは以下のようになります。

```
&domainName=*.my-bank.com|*.my-partner.com&
```

4. すべてのアプリケーション ページで、同一の絶対 URL を使用して rfdevice.swf を参照していることを確認します。

   たとえば、rfdevice.swf が *login.my-bank.com* から配信される場合、両方のサイト（*login.my-bank.com* および *online.my-partner.com*）には *login.my-bank.com* にある rfdevice.swf が含まれている必要があります。

HTTP cookie の設定

HTTP cookie をエンド ユーザ コンピュータに設定するには、HTTP cookie を取得または設定するアプリケーション ページに rfutil.js を追加する必要があります。

以下の手順を実行します。

1. rfutil.js を適切な Web アプリケーション フォルダにコピーします。このフォルダは、rfutil.js ファイルを追加しようとしているページが存在する場所に対して相対的に表します。
2. 以下の JavaScript コードをアプリケーションの該当する Web ページに追加します。

```
<script type="text/javascript"
src="<location_to_rfutil/>.js"></script>
```

前述のコード スニペットで、<location_to_rfutil>.js は rfutil.js への相対パスに置き換えます。

MFP の作成とデバイス ID の収集

MFP およびデバイス ID の収集機能を実装するには、リスク評価が必要なイベントを含むアプリケーションの各ページに、対応するコード スニペットを実装する必要があります。たとえば、ログイン イベントのリスクを評価する場合、アプリケーションでは、必要な JavaScript ファイルとコード スニペットをログイン ページに実装する必要があります。同様に、ログイン前のイベントについては、ユーザがオンライン アプリケーションの最初のページにアクセスするときに、このセクションで説明する手順がトリガされる必要があります。

エンド ユーザのデバイスから MFP を作成し、デバイス ID を収集する手順を以下に示します。

- 手順 1：JavaScript ファイルの追加
- 手順 2：MFP の収集
- 手順 3：MFP の再フォーマット
- 手順 4：IP アドレスの収集
- 手順 5：デバイス ID の収集

これらの手順は、evaluateRisk() メソッドを呼び出す前に、ログイン プロセスで表示するページ数に応じて、オンライン アプリケーションの 1 つのページに実装するか、複数のページにわたって実装できます。

手順 1：JavaScript ファイルの追加

ログイン ページまたはインデックス ページ（index.jsp または login.jsp など）などの適切な Web ページを変更して、エンド ユーザのコンピュータから MFP 関連の情報を収集したり、デバイス ID （cookie）を収集できるようにする必要があります。

注： RiskFort がエンド ユーザのシステムにデバイス ID を設定する時期と方法の詳細については、2-9 ページの「登録ワークフロー」を参照してください。

スクリプト コードを実装する方法

1. 必要なアプリケーション ページに、以下のように mfp_json.js JavaScript コードを追加します。

```
<script type="text/javascript" src="javascript/mfp_json.js"></script>
```

前述のコード スニペットでは、mfp_json.js が index.jsp を含むフォルダに対して相対的なフォルダに存在すると考えます。

2. エンド ユーザのシステムに設定されたデバイス ID を取得するために、必要なアプリケーション ページに以下のような rfutil.js JavaScript コードを追加します。

```
<script type="text/javascript" src="javascript/rfutil.js"></script>
```

3. 以下の場所から適切な Web アプリケーション フォルダ（*$APP_SERVER_HOME*/**javascript**/）に mfp_json.js ファイルをコピーします。
   - **Windows の場合**

     <install_location>\Arcot Systems**sdk****javascript**\
   - **UNIX プラットフォームの場合**

     <install_location>/arcot/**sdk**/**javascript**/
4. 以下の場所から適切な Web アプリケーション フォルダ（*$APP_SERVER_HOME*/**javascript**/）に rfutil.js ファイルをコピーします。
   - **Windows の場合**

     <install_location>\Arcot Systems**sdk****javascript**\
   - **UNIX プラットフォームの場合**

     $ARCOT_HOME/arcot/**sdk**/**javascript**/

手順 2：MFP の収集

MFP を収集するために以下を実装します。

注： これらの手順をさらに深く理解するには、5-63 ページの「サンプル コードのリファレンス」のコードを参照してください。

1. MFP 関連の処理を行う前に、HTML コードに以下のコードを追加します。

```
   <body onload="init()">


  <form name="CollectMFPToEvaluate" method="POST">
     <input type="hidden" name="MFP" length=1056>
     <input type="hidden" name="IpAddress">
     <input type="hidden" name="CallerID">
     <input type="hidden" name="DeviceID">
     <input type="button" value="Login"
onClick="callfuncAfterInitFinished();">
     </form>


  </body>
  </html>
```

2. `jsonSignature()` メソッドを呼び出します。これは、手順 1：JavaScript ファイルの追加で追加した `mfp_json.js` JavaScript ファイルの一部です。

```
   document.CollectMFPToEvaluate.MFP.value = jsonSignature();
```

このメソッドは、エンド ユーザのデバイスから MFP 関連の情報を収集して、JSON 文字列を返します。

サンプル アプリケーションのリファレンス

`ArRFMFPCollectionServlet.java` を参照することもできます。これは RiskFort サンプル アプリケーションに含まれています。このファイルでは、MFP およびその他の必要な情報を収集したり、セッションでこれらのパラメータを設定する方法について例を示しています。このファイルは、サンプル アプリケーションを展開した後に、以下の場所から入手可能です。

<RISKFORT_SAMPLEAPP_HOME>\WEB-INF\classes\com\arcot**riskfort****sampleapp****servlets**\

手順 3：MFP の再フォーマット

オンライン アプリケーションから受信した MFP を RiskFort サーバに送信できるようにするには、ここで再フォーマットする必要があります。これは、前の手順（手順 2：MFP の収集）で収集した署名がブラウザ固有であり、エンド ユーザが使用しているブラウザによって異なるためです。そのため、この MFP を RiskFort サーバが理解して処理できる RiskFort のネイティブ文字列に変換する必要があります。

`com.arcot.riskfortAPI.DeviceContext` API の以下のメソッドを使用して、収集された MFP を JSON 文字列に再フォーマットします。

```
buildDeviceSignature(signatureJSON, httpHeaders, extraInfo)
```

サンプル アプリケーションのリファレンス

`ArRFEvaluateHelper.java` を参照することもできます。これは RiskFort サンプル アプリケーションに含まれています。このファイルでは `buildDeviceSignature()` メソッドを例として、処理に必要な API を呼び出す方法を示します。このファイルは以下の場所にあります。

<RISKFORT_SAMPLEAPP_HOME>\WEB-INF\classes\com\arcot**riskfort****sampleapp****helpers**\

手順 4：IP アドレスの収集

RiskFort では、エンド ユーザ デバイスの IP アドレスを収集するメカニズムを提供していません。そのため、これを実行するための独自のロジックを実装する必要があります。

推奨事項については、5-67 ページの「IP アドレスの収集」を参照してください。

手順 5：デバイス ID の収集

デバイス ID （cookie）をエンド ユーザのシステムに設定したら（5-54 ページの「デバイス ID の設定」で説明）、今度は MFP と共に cookie を `get` していることを確認する必要があります。ただし、この収集メカニズムは、ユーザのシステムに設定された cookie のタイプにより異なります。

Flash cookie の場合

Flash cookie を使用している場合は、以下の手順を実行します。

1. Web ページの <body> エレメントの最初のメソッドが `init()` メソッドであることを確認します。

```
<body onload="init();">
```

2. `init()` 関数が定義されていることを確認します。たとえば、以下のようなコード スニペットを使用できます。

```
function init()
 {
   var so = new
SWFObject("<%=request.getContextPath()%>/flash/rfdevice.swf",
             "cookiemanager", "0", "0", "6", "#ffff00");
   so.addParam("allowScriptAccess", "always");
   so.write("flashcontent");

   var divid = document.getElementById("flashsetting");
   divid.style.display = 'block';
 }
```

注： すでに `init()` 関数を使用している場合は、関数定義に必要なコードのみをコピーします。

3. ページの［**Login**］（または［**Submit**］）ボタンをクリックしたら、以下のコード スニペットが呼び出されることを確認します。

```
    <input type="button" value="Login"
onClick="callfuncAfterInitFinished();">
```

4. `callfuncAfterInitFinished()` 関数が定義されていることを確認します。この関数では、Flash ムービーが完全にロードされるまで、コード内の次のステートメントを待機させる必要があります。そうしないと、Flash cookie に対する後続の読み取り操作または書き込み操作が失敗する場合があります。

   たとえば、以下のようなコード スニペットを使用できます。

```
function callfuncAfterInitFinished()
  {
    /*
     ムービーのロードが完了すると、arcotIsInitDone 変数は
     1 に設定されます。arcotMaxRetries は、チェックを実行できる
     最大回数です。
     これは rfutil.js に定義されています。
    */
    if (arcotIsInitDone != 1 && arcotMovieLoadCheckCnt <
arcotMaxRetries)
      {
         arcotMovieLoadCheckCnt++;
         /*
           ARCOT_RETRY_PERIOD_MSEC は、チェックが完了した後に再び
           実行されるまでの間隔です。これは rfutil.js に定義されています。
         */
         setTimeout("callfuncAfterInitFinished()",
ARCOT_RETRY_PERIOD_MSEC);
      }
    //Flash ムービーが完全にロードされている場合。
    else if (arcotIsInitDone == 1)
      {
         // 必ず login() メソッドの本体から collectSystemInfo() 関数を
         // 呼び出す必要があります。
         login();
      }
```

```
    // 適当な時間内にムービーのロードが行われなかった場合。
    else
      {
         // 必ず login() メソッドの本体から collectSystemInfo() 関数を
         // 呼び出す必要があります。
         login();
      }
  }
```

HTTP cookie の場合

ブラウザの（HTTP）cookie の場合には、Flash cookie の場合のように、`init()` メソッドを呼び出す必要はありません。ただし、以下の内容を確認する必要があります。

1. ページの［**Login**］（または［**Submit**］）ボタンをクリックしたら、以下のコード スニペットが呼び出されることを確認します。

```
    <input type="button" value="Login"
onClick="collectSystemInfo();">
```

2. `collectSystemInfo()` 関数が定義されていることを確認します。たとえば、以下のようなコード スニペットを使用できます。

```
function collectSystemInfo()
{
  // 要件に応じて cookie タイプを設定します。
    {<code for cookie type>}


  //jsonSignature() はマシン FingerPrint を収集します。
     document.CollectMFPToEvaluate.MFP.value = jsonSignature();


     document.CollectMFPToEvaluate.submit();
  }
```

必要に応じて、MFP を作成してデバイス ID を収集したら、それを入力として `evaluateRisk()` メソッドに渡す必要があります。詳細については、6-69 ページの「リスク評価の実行」を参照してください。

サンプル コードのリファレンス

以下のサンプル コードは、RiskFort の MFP およびデバイス ID 収集メカニズムを実装するために必要な変更内容を示しています。このファイルは、1 つのファイル（**`index.jsp`**）に収集するロジックを示しています。ただし、表示するページ数に応じて、`evaluateRisk()` メソッドを呼ぶ*前*に別のページに適切なコード スニペットを実装できます。

```
---------------------------------------------------------------------
<%@ 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">


//mfp_json.js はクライアント マシンから MFP を収集します。
<script type="text/javascript"
src="<%=request.getContextPath()%>/javascript/mfp_json.js"></script>


//rfutil.js は HTTP/Flash cookie および Flash 関数を設定および取得します。
<script type="text/javascript"
src="<%=request.getContextPath()%>/javascript/rfutil.js"></script>


<script language="javascript">



  function init()
```

```
  {

    var so = new SWFObject("<%=request.getContextPath()%>/flash/rfdevice.swf",
"cookiemanager", "0", "0", "6", "#ffff00");

    so.addParam("allowScriptAccess", "always");
    so.write("flashcontent");

    var divid = document.getElementById("flashsetting");

    // 最初は非表示
    divid.style.display = 'block';
  }


  function callfuncAfterInitFinished()
  {

    /*
     ムービーのロードが完了すると、arcotIsInitDone 変数は 1 に設定されます。
     arcotMaxRetries はチェックを実行できる最大回数です。
     これは rfutil.js に定義されています。
    */

    if (arcotIsInitDone != 1 && arcotMovieLoadCheckCnt < arcotMaxRetries)
      {
         arcotMovieLoadCheckCnt++;

         /*
           ARCOT_RETRY_PERIOD_MSEC は、チェックが完了した後に再び実行されるまでの
           間隔です。これは rfutil.js に定義されています。
         */
```

```
        setTimeout("callfuncAfterInitFinished()", ARCOT_RETRY_PERIOD_MSEC);

      }

    //Flash ムービーが完全にロードされている場合。
    else if (arcotIsInitDone == 1)
      {
         login();
      }

    // 適当な時間内にムービーのロードが行われなかった場合。
    else
      {
         login();
      }
  }


  function collectSystemInfo()
  {
        // 必要に応じて変数の値を設定します。
        var cookietype = "flashcookie";

        if(cookietype == "flashcookie")
          {
             document.CollectMFPToEvaluate.DeviceID.value =
getFlashCookie"CookieName");
          }
          else{
             document.CollectMFPToEvaluate.DeviceID.value =
getBrowserCookie("CookieName");
          }
```

```
        //jsonSignature() はマシン FingerPrint を収集します。
        document.CollectMFPToEvaluate.MFP.value = jsonSignature();

        document.CollectMFPToEvaluate.submit();
  }

  function login()

  {

   collectSystemInfo();

 // 送信用のフォームを処理します。
    //........
    //........
    //........
}

</script>
</head>
<!--
ユーザ情報を取得するための Arcot のフォーム。常に、ユーザのログイン用のフォームを処理する前にこれを実行します。
-->

<body onload="init()">

  <form name="CollectMFPToEvaluate" method="POST">
     <input type="hidden" name="MFP" length=1056>
     <input type="hidden" name="IpAddress">
```

```
     <input type="hidden" name="CallerID">
     <input type="hidden" name="DeviceID">
     <input type="button" value="Login" onClick="callfuncAfterInitFinished();">
     </form>

</body>
</html>
```

IP アドレスの収集

オンライン アプリケーションにアクセスするエンド ユーザは、自宅からアクセスする場合も、会社のネットワークからアクセスする場合もあります。後者に分類されるユーザの場合、ユーザがプロキシ サーバの背後に「隠れている」可能性があります。そのため、プロキシの背後からオンライン アプリケーションにアクセスしているエンド ユーザの IP アドレスを収集する方法は、自宅から直接アクセスするユーザとは異なります。

エンド ユーザがアプリケーションに直接アクセスしている場合

エンド ユーザがアプリケーションに直接アクセスしている場合は、JSP で `HttpServletRequest` インターフェースの `getRemoteAddr()` メソッドを使用できます。このメソッドは、リクエストを送信したクライアントの IP アドレスを含む文字列を返します。

第 6 章
リスク評価の実行

ユーザがオンライン アプリケーションにアクセスすると、アプリケーションはリスクを分析するために RiskFort にリクエストを転送します。RiskFort では、ユーザが初めてアクセスするのか、または RiskFort システムにすでに登録されているかに関わらず、すべてのユーザについてリスクを評価することができます。

Risk Evaluation Java API はプログラム可能なインターフェースを提供します。これを使用して Java クライアント（Java サーブレットおよび JSP ページなど）はリスク評価関連のリクエストを RiskFort サーバに送信します。この API は、RiskFort サーバに送信するリクエスト メッセージを作成し、レスポンスを受信し、それを戻り値構造としてパッケージ化してクライアントが読み取れるようにします。

この章では、Java API を使用してリスク評価を実行する方法、および API により実装されるメソッドおよびインターフェースの概要について説明します。次に、API のインターフェースおよびメソッドについて説明します。これを使用して RiskFort データベースで関連付けのリスト表示および削除を実行できます。リスク評価関連および関連付け管理タスクの説明の後にサンプル コード スニペットが続きます。これをコードで使用すると、タスクを実行できます。この章では以下のトピックについて説明します。

- リスクの評価および事後評価の実行
- 関連付けの管理

リスクの評価および事後評価の実行

トランザクションに関連付けられたリスクを評価し、それに続く事後評価を実行するには、`RiskXActionAPI` インターフェース（`com.arcot.riskfortAPI` パッケージで）を使用する必要があります。このインターフェースは RiskFort サーバのリスク評価機能に接続するクライアント側インターフェースを表し、リスク評価のためのさまざまなワークフローでサポートされている API を提供します。

トランザクションのリスクを評価して事後評価タスクを実行する方法

1. `RiskFactory` クラスを使用することによって Risk Evaluation API が初期化されていることを確認します。

詳細については、3-38 ページの「Risk Evaluation API の初期化」を参照してください。

2. 必要に応じて、`com.arcot.riskfortAPI.AdditionalInputs` パッケージ内に `AdditionalInputs` クラスを使用することによって、トランザクションに追加の入力を準備します。

   詳細については、3-40 ページの「追加入力の準備」を参照してください。

3. `RiskFactory.getRiskXActionAPI()` 静的メソッドを使用して `RiskXActionAPI` インターフェースを実装するオブジェクトを取得します。

   このメソッドは、`RiskXActionAPI` インターフェースを実装する `RiskFactory` クラスの使用可能なインスタンスを返します。

4. `RiskXActionAPI.evaluateRisk()` メソッドを使用して `RiskAssessment` クラスのオブジェクトを取得します。

   このメソッドには以下の入力パラメータが必要です。

   a. 呼び出し全体の追跡に使用される `CallerID` 文字列変数を定義して設定します。

   b. `DeviceContext()` メソッドを使用して `DeviceContext` クラスのオブジェクトを取得し、`DeviceContext.buildDeviceSignature()` メソッドを使用して JSON シグネチャを構築します。

   c. 構築した JSON シグネチャを呼び出します（第 5 章の「デバイス ID とマシン FingerPrint の収集」で詳述）。

   d. `LocationContext()` メソッドを使用して `LocationContext` クラスのオブジェクトを取得し、`LocationContext.setIpAddress()` メソッドを使用して、返されたオブジェクトに必要なプロパティを設定します。

   e. `UserContext()` メソッドを使用して `UserContext` クラスのオブジェクトを取得し、`UserContext.setUserId()` メソッドを使用して、返されたオブジェクトに必要なプロパティを設定します。

   f. `TransactionContext()` メソッドを使用して `TransactionContext` クラスのオブジェクトを取得し、必要に応じてこのクラス（`setAction()` および `setChannel()`）のメソッドを使用して、返されたオブジェクトに必要なプロパティを設定します。

   g. 追加の情報を使用している場合は、`AdditionalInputs()` メソッドを使用して `AdditionalInputs` クラスのオブジェクトを取得し、`AdditionalInputs.put()` メソッドを使用して、返されたオブジェクトに必要なプロパティを設定します。

      これらの追加入力は、名前と値のペアの形式です。例：

```
MerchantID=id;MerchantCountry=country;MerchantName=name
```

5. `RiskAssessment.getRiskAdvice()` メソッドを使用して `RiskAssessment` `RiskAdvice` オブジェクトの結果を取得します。

重要 : アドバイスが `INCREASEAUTH` である場合、アプリケーションは 2 次認証を実行し、`PostEvaluate()` メソッドの使用によりこの認証の結果を RiskFort へ渡す必要があります。

6. 最後に、`RiskXActionAPI.postEvaluate()` メソッドを使用してトランザクションの最終結果を特定し、`PostEvaluateResponse` オブジェクトを返します。

   何らかの操作を行った場合は、2 次認証の結果をメソッドに渡す必要があります。

7. `PostEvaluateResponse.isAllowAdvised()` メソッドを使用して `PostEvaluateResponse` オブジェクトの結果を問い合わせて、トランザクションの続行を許可するかどうか決定することができます。

エラーの処理

いずれかの Risk Evaluation API メソッドの実行中に発生したエラーはすべて、`RiskException` 例外としてスローされます。`RiskException` オブジェクトには、発生したエラーに関連付けられたエラー コードと（RiskFort サーバ側で生成された）トランザクション識別子を含む `getErrorCode` と `getTransactionId` の各メンバが含まれています。

リスク評価および事後評価のためのサンプル コード

注 : 詳細な実用コードのサンプルについては、“ リスク評価および事後評価のためのサンプル コード ” in Appendix B を参照してください。

以下のサンプル コード スニペットを使用して、アプリケーション コードに RiskFort のリスク評価および事後評価機能を実装する方法を理解してください。

```
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\":
\"20070312\",\"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\":[{\"nam
e\":\"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},\"e
xtra\":{\"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();
            }
}
```

関連付けの管理

RiskFort は、ユーザがアプリケーションにアクセスするために使用するデバイスにユーザを自動的に関連付ける（またはバインドする）ことにより、ユーザをシステムで有効なユーザであると一意に識別します。これを RiskFort の用語では関連付け（またはデバイス バインディング）と呼びます。バインドされないユーザは、認証のチャレンジが行われる可能性が高くなります。

RiskFort では、ユーザを複数のデバイスにバインドできます。たとえば、ユーザは、仕事用と自宅用のコンピュータからアプリケーションにアクセスできます。同様に、単一のデバイスを複数のユーザにバインドできます。たとえば、家族は 1 台のコンピュータを使用してアプリケーションにアクセスできます。

重要 : ユーザがインターネット カフェやキオスク システムのような公に共有されるデバイスとの関連付けを作成しないように、適切な措置を講じることを推奨します。

関連付けの管理には次のものが含まれます。

- 関連付けのリスト表示
- 関連付けの削除

関連付けのリスト表示

指定されたユーザの保存されている関連付けをすべてリスト表示する方法

1. `RiskFactory` クラスを使用することによって Risk Evaluation API が初期化されていることを確認します。

   詳細については、3-32 ページの「Issuance API の初期化」を参照してください。

2. 必要に応じて、`com.arcot.riskfortAPI.AdditionalInputs` パッケージ内に `AdditionalInputs` クラスを使用することによって、トランザクションに追加の入力を準備します。

   詳細については、3-40 ページの「追加入力の準備」を参照してください。

3. `RiskFactory.getRiskXActionAPI()` 静的メソッドを使用して `RiskXActionAPI` インターフェースを実装するオブジェクトを取得します。

   このメソッドは、`RiskXActionAPI` インターフェースを実装する `RiskFactory` クラスの使用可能なインスタンスを返します。

4. 呼び出し全体の追跡に使用される `CallerID` 文字列変数を定義して設定します。

5. `UserContext()` メソッドを使用して `UserContext` クラスのオブジェクトを取得します。

6. 返されたオブジェクトに必要なプロパティを設定します。

   たとえば、`UserContext.setUserId()` メソッドを呼び出すことによりユーザ ID を設定できます。

7. ListAssociationResponse オブジェクトを返す RiskXActionAPI.listAssociations() メソッドを呼び出します。

   ListAssociationResponse.getAllAssociations() メソッドは、指定されたユーザのすべての既知の関連付けの配列を返します。

エラーの処理

いずれかの Risk Evaluation API メソッドの実行中に発生したエラーはすべて、RiskException 例外としてスローされます。RiskException オブジェクトには、発生したエラーに関連付けられたエラー コードと（RiskFort サーバ側で生成された）トランザクション識別子を含む getErrorCode と getTransactionId の各メンバが含まれています。

関連付けをリスト表示するためのサンプル コード

以下のコード スニペットをアプリケーション コードに挿入することにより、既存の関連付けをすべてリスト表示することができます。

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

関連付けの削除

ユーザに対して、特定のユーザとデバイスとの関連付けを削除する方法

1. RiskFactory クラスを使用することによって Risk Evaluation API が初期化されていることを確認します。

   詳細については、3-38 ページの「Risk Evaluation API の初期化」を参照してください。

2. 必要に応じて、com.arcot.riskfortAPI.AdditionalInputs パッケージ内に AdditionalInputs クラスを使用することによって、トランザクションに追加の入力を準備します。

   詳細については、3-40 ページの「追加入力の準備」を参照してください。

3. RiskFactory.getRiskXActionAPI() 静的メソッドを使用して RiskXActionAPI インターフェースを実装するオブジェクトを取得します。

このメソッドは、RiskXActionAPI インターフェースを実装する RiskFactory クラスの使用可能なインスタンスを返します。

4. 呼び出し全体の追跡に使用される CallerID 文字列変数を定義して設定します。
5. UserContext() メソッドを使用して UserContext クラスのオブジェクトを取得します。
6. 返されたオブジェクトに必要なプロパティを設定します。

   たとえば、UserContext.setUserId() メソッドを呼び出すことによりユーザ ID を設定できます。
7. RiskXActionAPI.listAssociations() メソッドによって返された ListAssociationResponse 配列を使用して削除する関連付けの名前を取得します。
8. DeleteAssociationResponse オブジェクトを返す RiskXActionAPI.deleteAssociation() メソッドを呼び出します。

   DeleteAssociationResponse.DeleteAssociationResponse() メソッドは、指定したユーザの関連付けを削除します。

エラーの処理

いずれかの Risk Evaluation API メソッドの実行中に発生したエラーはすべて、RiskException 例外としてスローされます。RiskException オブジェクトには、発生したエラーに関連付けられたエラー コードと（RiskFort サーバ側で生成された）トランザクション識別子を含む getErrorCode と getTransactionId の各メンバが含まれています。

関連付けを削除するためのサンプル コード

以下のコード スニペットをアプリケーション コードに挿入することにより、既存の関連付けを削除することができます。

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

付録 A
追加の SDK 設定

この付録では、以下のトピックについて説明します。

- 複数の RiskFort サーバ インスタンスの設定
- SDK と RiskFort サーバ間の SSL 通信のセットアップ

複数の RiskFort サーバ インスタンスの設定

複数の RiskFort サーバ インスタンスと通信する Java SDK を設定するには、プロパティ ファイルを編集する必要があります。

デフォルトでは、これらのファイルは 1 つの RiskFort サーバ インスタンスを設定するエントリを提供します。これらのエントリには 1 とマークされており、1 つのインスタンスのみが設定されることを示します。設定するインスタンスの数に応じて、これらのエントリを複製し、インスタンス数に応じたインスタンス番号を付加する必要があります。

複数のサーバ インスタンスを設定する方法

1. 設定している SDK に応じて、以下のフォルダで利用可能なそれぞれのプロパティ ファイルを開きます。
   - **Windows の場合**

     `<install_location>\Arcot Systems\sdk\java\properties\`
   - **UNIX ベースのプラットフォームの場合**

     `<install_location>/arcot/sdk/java/properties/`
2. `transport.<n>` パラメータの値を必要な通信モードに設定します。デフォルトでは [`TCP`] に設定されています。

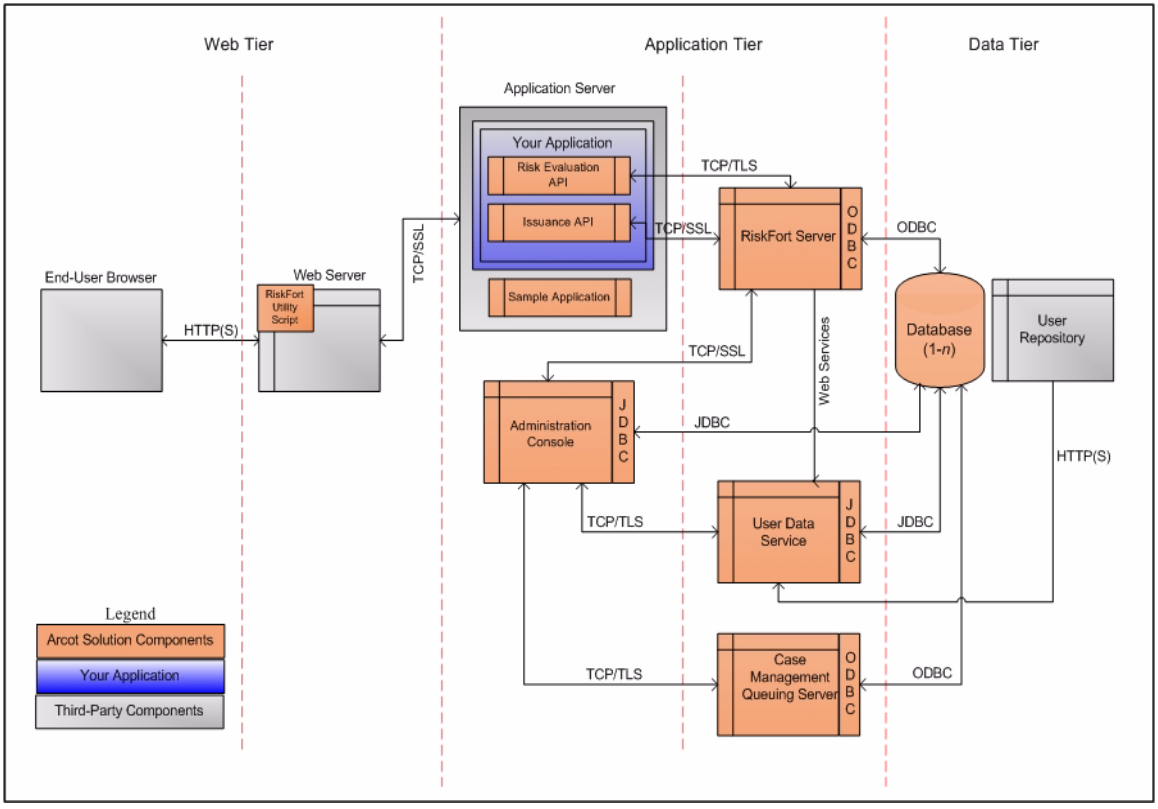
   通信モードを変更する場合は、A-80 ページの「SDK と RiskFort サーバ間の SSL 通信のセットアップ」を参照してください。
3. `host.<n>` パラメータの値を必要なサーバ インスタンスのホスト名または IP アドレスに設定します。

4. port.<n> パラメータの値を **RiskFort Native** または **Transaction Web Service** プロトコルがリスンしているポート番号に設定します。
5. 変更を保存して、ファイルを閉じます。

SDK と RiskFort サーバ間の SSL 通信のセットアップ

RiskFort では、RiskFort サーバと SDK の間で TCP ベースの通信をサポートするだけではなく、これらのコンポーネント間の通信を保護する Secure Socket Layer （SSL）をサポートしています。図 A-1 に示されているように、RiskFort にはサーバ側証明書を使用する一方向の Secure Socket Layer （SSL）を設定することも、サーバと SDK の間でサーバ側およびクライアント側証明書を使用する双方向 SSL を設定することも可能です。

図 A-1 通信モード



一方向の SSL 通信

トランザクション中に交換されるデータの整合性および機密性を確保するため、RiskFort サーバは、以下との間で一方向 SSL 通信をサポートします。

- RiskFort SDK
- サンプル アプリケーション
- `arrrfadmin` ツール

各トランザクションについて、RiskFort サーバは Risk Evaluation API および Issuance API に署名された証明書を提示して検証を受けます。その後、RiskFort API は、RiskFort サーバの ID を確認して、認証を完了します。

双方向 SSL 通信

トランザクション中に交換されるデータの整合性および機密性を確保するため、RiskFort サーバは、以下との間で（一方向 SSL 通信に加えて）双方向 SSL 通信をサポートします。

- **評価コールアウト**

注： ビジネス要件に基づいて、RiskFort によって独自のカスタム評価ルールを書き込むことができます。このカスタム ルールは評価コールアウトと呼ばれます。

詳細については、「Arcot RiskFort 2.2.6 管理ガイド」を参照してください。

- **スコアリング コールアウト**

注： RiskFort では、スコアリング コールアウトと呼ばれる独自のカスタム スコアリング ロジックを書き込むこともできます。

詳細については、「Arcot RiskFort 2.2.6 管理ガイド」を参照してください。

評価コールアウトまたはスコアリング コールアウトを使用している場合、コールアウトは RiskFort サーバに署名された証明書を提示する必要があります。RiskFort サーバはコールアウトの ID を確認し、次に署名された証明書をコールアウトに提示して検証を受けます。コールアウトが RiskFort サーバの ID を正常に確認したら、認証は完了します。

SSL 通信の設定

RiskFort サーバと Java SDK の間の通信を保護する方法

1. Java SDK と RiskFort サーバの間の SSL 通信を有効にする前に、以下を実行する必要があります。
   a. 信頼された証明機関からのデジタル証明書を取得する。
   b. HTTPS 対応のサーバ ポートにアプリケーションを公開する。
2. Master Administrator （MA）としてログインします。
3. RiskFort Administration Console の［**Protocol Configuration**］ページを使用して、以下の手順に従います。
   a. **RiskFort ネイティブ プロトコル**のトランスポート セキュリティ モードを［SSL］に設定する。
   b. **クライアント ストア**にトラステッド CA のルート証明書を指定する。
   c. 必要な**証明書チェーン**および**秘密キー**を参照する。
   d. 変更内容を保存します。
4. 次のディレクトリに移動します。
   - **Windows の場合**

     `<install_location>\Arcot Systems\sdk\java\properties\`
   - **UNIX ベースのプラットフォームの場合**

     `<install_location>/arcot/sdk/java/properties/`
5. 設定している SDK に応じて、以下のうち必要なプロパティ ファイルを開きます。
   - `riskfort.risk-evaluation.properties`
   - `riskfort.issuance.properties`
6. ファイルに以下のパラメータを指定します。
   - `TRANSPORT_TYPE`
   - `CA_CERT_FILE`

7. 変更を保存して、ファイルを閉じます。
8. RiskFort サーバを再起動します。
9. アプリケーション サーバを再起動します。

関連文書： RiskFort のさまざまなコンポーネント間で SSL を設定する方法の詳細については、「Arcot RiskFort 2.2.6 インストールおよび展開ガイド」の付録 F、「SSL の設定」を参照してください。

付録 B
サンプル コード

この付録では、以下の RiskFort 機能をテストするために実行できるサンプル コードを提供します。

- ユーザ操作（ユーザ操作のためのサンプル コード）
- リスク評価および事後評価（リスク評価および事後評価のためのサンプル コード）

ユーザ操作のためのサンプル コード

以下のコード サンプルを使用して RiskFort データベースでユーザ関連の操作（作成、読み取り、および更新）をテストすることができます。

```
/* RiskFort Issuance API にインポートするパッケージ */

import java.util.Map;
import java.util.HashMap;

import com.arcot.riskfortissuanceAPI.User;
import com.arcot.riskfortissuanceAPI.UserManager;
import com.arcot.riskfortissuanceAPI.UserRepManager;
import com.arcot.riskfortissuanceAPI.UserRepResult;
import com.arcot.riskfortissuanceAPI.UserRepositoryException;
import com.arcot.riskfortissuanceAPI.UserRepositoryService;
import com.arcot.riskfortissuanceAPI.AdditionalInputs;

public class RiskFort_Issuance {

  // この例の値は、サンプルで使用するためにハード コーディングされたものです。
public static void main(String[] args) {
```

```
    /*
    initialize :
      入力プロパティ ファイルから API オブジェクトを初期化します。
      initialize() はアプリケーションのスタートアップ時に一度だけ呼び出す必要があります。

      以下にプロパティ ファイルのフィールドおよびフォーマットを示します。
        HOST.1=RiskFort サーバの IP アドレス。
        PORT.1=RiskFort サーバのポート番号。
        TRANSPORT_TYPE = 接続タイプ。使用可能な値は TLS または TCP です。
        CA_CERT_FILE = TRANSPORT_TYPE = TLS の場合に必要な CA 証明書ファイル。サーバ
CA 証明書（PEM 形式）のファイル パス。

    public static synchronized void initialize(java.lang.String
propertyLocation)
            throws UserRepositoryException

    パラメータ :
      propertyLocation - クラス パスに対してパラメータとして渡すロケーションを示します。
     <code>null</code> が渡された場合は、デフォルトのロケーションである
properties/riskfort_issuance.properties が使用されます。
    戻り値 :

    スローする値 :
      UserRepositoryException
    */

    UserRepositoryService urs;

    // 入力プロパティ ファイルから API オブジェクトを初期化するためのサンプル コード。
    String propertiesFileLocation= "/properties/riskfort_issuance.properties";

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

```
      // 入力プロパティ ファイルから API オブジェクトを初期化します。
      UserRepManager.initialize(propertiesFileLocation);

      // User Repository Service の実装を取得します。
      urs = UserRepManager.getGrpUserRepService();

      System.out.println("RiskFort Issuance API initialized.");
     catch (UserRepositoryException e) {

      /* UserRepositoryException オブジェクトに対して以下のメソッドを使用し、エラー コードおよびエラー メッセージを以下のように取得できます。
     * 文字列コード = e.getCode();
     * 文字列メッセージ = e.getMessage();
     */
      System.out.println("Exception during initialize.");
      System.out.println("Error code: " + e.getCode());
      System.out.println("Error message: " + e.getMessage());

      /* 以下のエラー コードが API によって返されます。*/

      /* ERROR_CONF_API
      * 考えられる理由：
      *   設定ファイルのパラメータが不足しているか、正しく設定されていません。
      * 考えられるアクション：
      *   設定ファイルのすべてのパラメータを正しく設定する必要があります。
      */
    }


    /*
    create：
      システム内にユーザを作成します。
```

```
    public UserRepResult create(User user, AdditionalInputs additionalInputs)
          throws UserRepositoryException;


    パラメータ :
      user - システムに作成されるユーザ オブジェクトを示します。
      additionalInputs - 別の操作で必要になる可能性がある追加の入力。これは今後の使用の為に保持されています。
    戻り値 :
      UserRepResult - サーバで生成されたレスポンス コードおよびトランザクション識別子が含まれています。
    スローする値 :
      UserRepositoryException
    */


    // システム内にユーザを作成するためのサンプル コード。
    String userName = "TestUser";
    String orgName = "DEFAULTORG";
    String firstName = "Test";
    String lastName = "User";
    String email = "TestUser@abcd.com";
    String pam = "My PAM";


    Map additionalInputs = new HashMap();


    String extName1 = "AR_RF_CALLER_ID";
    String extValue1 = "test_caller";
    if(extName1 != null && extName1 != "" )
      additionalInputs.put(extName1, extValue1);


    // RiskFort サーバに送信される入力ユーザ情報を保持するオブジェクトを作成して初期化します。
    User user = null;
```

```
    // RiskFort サーバからのレスポンスを保持するオブジェクトを作成して初期化します。
    UserRepResult userRepRes = null;

    System.out.println("The following credentials are used to create a user.");
    System.out.println("Username: " + userName);
    System.out.println("Organization name: " + orgName);
    System.out.println("First name: " + firstName);
    System.out.println("Last name: " + lastName);
    System.out.println("Email: " + email);
    System.out.println("PAM: " + pam);

    try {
      // 実装ユーザ リポジトリ サービスを取得します。
      urs = UserRepManager.getGrpUserRepService();

      // ユーザ インターフェースの実装を取得します。
      user = UserManager.getNewUserObject(userName, orgName);

      // ユーザ プロパティを設定します。
      user.setProperty(User.USER_PROPERTY_FIRSTNAME, firstName);
      user.setProperty(User.USER_PROPERTY_LASTNAME, lastName);
      user.setProperty(User.USER_PROPERTY_EMAILADDR, email);
      user.setPAM(pam);

      System.out.println("Calling create.");
      // API を呼び出してユーザを作成し、userRepRes オブジェクトに RiskFort サーバのレスポンスを設定します。
      userRepRes = urs.create(user, additionalInputs);

      System.out.println("create succeeded.");

    } catch (UserRepositoryException e) {
```

```
      /* UserRepositoryException オブジェクトに対して以下のメソッドを使用し、エラー コードおよびエラー メッセージを以下のように取得できます。
      * 文字列コード = e.getCode();
      * 文字列メッセージ = e.getMessage();
      */
      System.out.println("Exception in 'create'");
      System.out.println("Error code: " + e.getCode());
      System.out.println("Error message: " + e.getMessage());

      /* 以下のエラー コードが API によって返されます。*/

      /* INVALID_EMAIL
      * 考えられる理由：
      *   ユーザ オブジェクトに設定された電子メールは API には無効です。
      * 考えられるアクション：
      *   適切なアクションを実行して電子メール ID を修正します。
      *   例：エンド ユーザに、有効な電子メール ID を指定してそれを API に渡すように要請します。
      */

      /* INVALID_ORG_NAME
      * 考えられる理由：
      *   ユーザ オブジェクトに設定されている組織名が API では無効です。
      * 考えられるアクション：
      *   ユーザ オブジェクトに有効な組織名を設定して再度 API を呼び出します。
      *
      */

      /* INVALID_USER_OBJECT
      * 考えられる理由：
      *   入力として API に渡されたユーザ オブジェクトは無効です。
      * 考えられるアクション：
      *   API に渡されたユーザ オブジェクトの構造が有効であるか確認します。
```

```
        *    必要な個所を修正して再度 API を呼び出します。
        */


        /* MANDATORY_ATTRIB_EXCEEDS_SIZE
        * 考えられる理由：
        *    API への呼び出しで必須入力属性の 1 つのサイズが超過しています。
        * 考えられるアクション：
        *    適切なアクションを実行して属性のサイズを制限します。
        *    UserRepositoryException クラスの getMessage() メソッドは、サイズが超過している属性を示します。
        */


        /* MANDATORY_ATTRIB_INVALID_CHAR
        * 考えられる理由：
        *    API への呼び出しで必須入力属性の 1 つに無効な文字が含まれています。
        * 考えられるアクション：
        *    適切なアクションを実行して属性を有効にします。
        *    UserRepositoryException クラスの getMessage() メソッドは、無効な文字が含まれている属性を示します。
        */


        /* MANDATORY_ATTRIB_MISSING
        * 考えられる理由：
        *    API への呼び出しで必須入力属性の 1 つが不足しています。
        * 考えられるアクション：
        *    不足している入力属性を正しく設定します。
        *    UserRepositoryException クラスの getMessage() メソッドは、不足している必須属性を示します。
        */


        /* OPTIONAL_ATTRIB_EXCEEDS_SIZE
        * 考えられる理由：
        *    API への呼び出しで任意の入力属性の 1 つのサイズが超過しています。
```

```
        * 考えられるアクション：
        *    適切なアクションを実行して属性のサイズを制限します。
        *    UserRepositoryException クラスの getMessage() メソッドは、サイズが超過している属性を示します。
        */


        /* OPTIONAL_ATTRIB_INVALID_CHAR
        * 考えられる理由：
        *    API への呼び出しで任意の入力属性の 1 つに無効な文字が含まれています。
        * 考えられるアクション：
        *    適切なアクションを実行して属性を有効にします。
        *    UserRepositoryException クラスの getMessage() メソッドは、無効な文字が含まれている属性を示します。
        */


        /* SERVER_INTERNAL_ERROR
        * 考えられる理由：
        *    トランザクションの処理中にサーバに問題が発生しました。
        * 考えられるアクション：
        *    サーバ ログで詳細を確認します。
        *    トランザクションの失敗を報告して再試行を要請します。
        */


        /* TCP_COMMUNICATION_ERROR
        * 考えられる理由：
        *    RiskFort サーバでの通信障害。
        * 考えられるアクション：
        *    トランザクションの失敗を報告して再試行を要請します。
        */


        /* UNKNOWN_ERROR
        * 考えられる理由：
        *    RiskFort サーバからのレスポンスが壊れています。
```

```
        * 考えられるアクション：
        *    トランザクションの失敗を報告して再試行を要請します。
        */

        /* USER_ALREADY_EXIST
        * 考えられる理由：
        *    API への入力に指定されたユーザはすでに存在します。
        * 考えられるアクション：
        *    適切なアクションを実行します。
        *    例： 存在しないユーザ名をこのユーザに指定するか、エンド ユーザにそれを実行するように要請します。
        */

        /* USERNAME_MISSING
        * 考えられる理由：
        *    ユーザ名が API への入力ユーザ オブジェクトに設定されていません。
        * 考えられるアクション：
        *    正確なユーザ名属性を API の入力ユーザ オブジェクトに設定します。
        */
    }
    /*

    update：
      システム内のユーザを更新します。
    public UserRepResult update(User user, AdditionalInputs additionalInputs)
      throws UserRepositoryException;

    パラメータ：
      user - システムで情報を更新する必要のあるユーザ オブジェクトを示します。
      additionalInputs - 別の操作で必要になる可能性がある追加の入力。これは今後の使用の為に保持されています。
    戻り値：
```

```
     UserRepResult - サーバで生成されたレスポンス コードおよびトランザクション識別子が含まれています。
    スローする値 :
      UserRepositoryException
    */

    // システムのユーザを更新するためのサンプル コード。
    // RiskFort サーバに送信される入力ユーザ情報を保持するオブジェクトを初期化します。
    user = null;
    // RiskFort サーバからのレスポンスを保持するオブジェクトを初期化します。
    userRepRes = null;

    // 指定されたユーザに対して更新される新しい pam。
    pam = "New changed PAM";
    System.out.println("PAM changed to: " + pam);
    try {
      // 実装ユーザ リポジトリ サービスを取得します。
      urs = UserRepManager.getGrpUserRepService();

      // ユーザ インターフェースの実装を取得します。
      user = UserManager.getUserDetails(userName, orgName);

      // 更新されるユーザ プロパティを設定します。
      if ( firstName != null )
      user.setProperty(User.USER_PROPERTY_FIRSTNAME, firstName);
      if ( lastName != null )
      user.setProperty(User.USER_PROPERTY_LASTNAME, lastName);
      if ( email != null)
      user.setProperty(User.USER_PROPERTY_EMAILADDR, email);
      if ( pam != null)
      user.setPAM(pam);

      System.out.println("Calling update.");
```

```
      // API を呼び出してユーザを更新し、userRepRes オブジェクトに RiskFort サーバのレスポンスを設定します。
      userRepRes = urs.update(user, additionalInputs);
      System.out.println("update succeeded.");


    } catch (UserRepositoryException e) {


      /* UserRepositoryException オブジェクトに対して以下のメソッドを使用し、エラー コードおよびエラー メッセージを以下のように取得できます。
      * 文字列コード = e.getCode();
      * 文字列メッセージ = e.getMessage();
      */
      System.out.println("Exception in 'update'");
      System.out.println("Error code: " + e.getCode());
      System.out.println("Error message: " + e.getMessage());


      /* 以下のエラー コードが API によって返されます。*/


      /* INVALID_EMAIL
      * 考えられる理由：
      *   ユーザ オブジェクトに設定された電子メールは API には無効です。
      * 考えられるアクション：
      *   適切なアクションを実行して電子メール ID を修正します。
      *   例：有効な電子メール ID を指定して API に渡すようにエンド ユーザに要請します。
      */


      /* INVALID_ORG_NAME
      * 考えられる理由：
      *   ユーザ オブジェクトに設定されている組織名が API では無効です。
      * 考えられるアクション：
      *   ユーザ オブジェクトに有効な組織名を設定して再度 API を呼び出します。
      *
      */
```

```
        /* INVALID_USER_OBJECT
        * 考えられる理由：
        *    入力として API に渡されたユーザ オブジェクトは無効です。
        * 考えられるアクション：
        *    API に渡されたユーザ オブジェクトの構造が有効であるか確認します。
        *    必要な個所を修正して再度 API を呼び出します。
        */

        /* MANDATORY_ATTRIB_EXCEEDS_SIZE
        * 考えられる理由：
        *    API への呼び出しで必須入力属性の 1 つのサイズが超過しています。
        * 考えられるアクション：
        *    適切なアクションを実行して属性のサイズを制限します。
        *    UserRepositoryException クラスの getMessage() メソッドは、サイズが超過している属性を示します。
        */

        /* MANDATORY_ATTRIB_INVALID_CHAR
        * 考えられる理由：
        *    API への呼び出しで必須入力属性の 1 つに無効な文字が含まれています。
        * 考えられるアクション：
        *    適切なアクションを実行して属性を有効にします。
        *    UserRepositoryException クラスの getMessage() メソッドは、無効な文字が含まれている属性を示します。
        */

        /* MANDATORY_ATTRIB_MISSING
        * 考えられる理由：
        *    API への呼び出しで必須入力属性の 1 つが不足しています。
        * 考えられるアクション：
        *    不足している入力属性を正しく設定します。
```

```
      *   UserRepositoryException クラスの getMessage() メソッドは、不足している必須属
性を示します。
      */


      /* OPTIONAL_ATTRIB_EXCEEDS_SIZE
      * 考えられる理由：
      *   API への呼び出しで任意の入力属性の 1 つのサイズが超過しています。
      * 考えられるアクション：
      *   適切なアクションを実行して属性のサイズを制限します。
      *   UserRepositoryException クラスの getMessage() メソッドは、サイズが超過してい
る属性を示します。
      */


      /* OPTIONAL_ATTRIB_INVALID_CHAR
      * 考えられる理由：
      *   API への呼び出しで任意の入力属性の 1 つに無効な文字が含まれています。
      * 考えられるアクション：
      *   適切なアクションを実行して属性を有効にします。
      *   UserRepositoryException クラスの getMessage() メソッドは、無効な文字が含まれ
ている属性を示します。
      */


      /* SERVER_INTERNAL_ERROR
      * 考えられる理由：
      *   トランザクションの処理中にサーバに問題が発生しました。
      * 考えられるアクション：
      *   サーバ ログで詳細を確認します。
      *   トランザクションの失敗を報告して再試行を要請します。
      */


      /* TCP_COMMUNICATION_ERROR
      * 考えられる理由：
      *   RiskFort サーバでの通信障害。
```

```
      * 考えられるアクション：
      *    トランザクションの失敗を報告して再試行を要請します。
      */

      /* UNKNOWN_ERROR
      * 考えられる理由：
      *    RiskFort サーバからのレスポンスが壊れています。
      * 考えられるアクション：
      *    トランザクションの失敗を報告して再試行を要請します。
      */

      /* USER_NOT_FOUND
      * 考えられる理由：
      *    API への入力に指定されたユーザは存在しません。
      * 考えられるアクション：
      *    適切なアクションを実行します。
      *    例： 渡したユーザ名が正しいことを確認します。
      */

      /* USERNAME_MISSING
      * 考えられる理由：
      *    ユーザ名が API への入力ユーザ オブジェクトに設定されていません。
      * 考えられるアクション：
      *    正確なユーザ名属性を API の入力ユーザ オブジェクトに設定します。
      */
    }

    /*
    read:
      システムからユーザを読み取ります。これは、完全に読み込まれたユーザ オブジェクトを返します。
    public User read(java.lang.String userName, java.lang.String orgName,
AdditionalInputs additionalInputs)
```

```
        throws UserRepositoryException;


    パラメータ :
      userName - システムから詳細を読み取る必要のあるユーザのユーザ名を示します。
      orgName - システムから詳細を読み取る必要のあるユーザが所属する組織を示します。
      additionalInputs - 別の操作で必要になる可能性がある追加の入力。これは今後の使用の為に保持されています。
    戻り値 :
      User - システムから情報が読み取られたユーザ オブジェクトを示します。
    スローする値 :
      UserRepositoryException
    */



    // システムからユーザ詳細を読み取るためのサンプル コード。


    // RiskFort サーバからのレスポンスを保留するオブジェクトを初期化します。
    user = null;


    try {
      // 実装ユーザ リポジトリ サービスを取得します。
      urs = UserRepManager.getGrpUserRepService();


      System.out.println("Calling read with username = " + userName + " and organization name = " + orgName);


      // API を呼び出してシステムからユーザ詳細を読み取ります。この API は、値がすべて設定されたユーザ オブジェクトを返します。
      user  = urs.read(userName, orgName, additionalInputs);


      System.out.println("User details read are as follows: ");
      System.out.println("Username: " + userName);
      System.out.println("Organization name: " + orgName);
```

```
        System.out.println("First name: " + firstName);
        System.out.println("Last name: " + lastName);
        System.out.println("Email: " + email);
        System.out.println("PAM: " + pam);

    } catch (UserRepositoryException e) {

        /* UserRepositoryException オブジェクトに対して以下のメソッドを使用し、エラー コードおよびエラー メッセージを以下のように取得できます。
        * 文字列コード = e.getCode();
        * 文字列メッセージ = e.getMessage();
        */
        System.out.println("Exception in 'read'");
        System.out.println("Error code: " + e.getCode());
        System.out.println("Error message: " + e.getMessage());

        /* 以下のエラー コードが API によって返されます。*/

        /* INVALID_ORG_NAME
        * 考えられる理由：
        *   ユーザ オブジェクトに設定されている組織名が API では無効です。
        * 考えられるアクション：
        *   ユーザ オブジェクトに有効な組織名を設定して再度 API を呼び出します。
        */

        /* MANDATORY_ATTRIB_EXCEEDS_SIZE
        * 考えられる理由：
        *   API への呼び出しで必須入力属性の 1 つのサイズが超過しています。
        * 考えられるアクション：
        *   適切なアクションを実行して属性のサイズを制限します。
        *   UserRepositoryException クラスの getMessage() メソッドは、サイズが超過している属性を示します。
        */
```

```
        /* MANDATORY_ATTRIB_INVALID_CHAR
        * 考えられる理由：
        *   API への呼び出しで必須入力属性の 1 つに無効な文字が含まれています。
        * 考えられるアクション：
        *   適切なアクションを実行して属性を有効にします。
        *   UserRepositoryException クラスの getMessage() メソッドは、無効な文字が含まれている属性を示します。
        */

        /* MANDATORY_ATTRIB_MISSING
        * 考えられる理由：
        *   API への呼び出しで必須入力属性の 1 つが不足しています。
        * 考えられるアクション：
        *   不足している入力属性を正しく設定します。
        *   UserRepositoryException クラスの getMessage() メソッドは、不足している必須属性を示します。
        */

        /* SERVER_INTERNAL_ERROR
        * 考えられる理由：
        *   トランザクションの処理中にサーバに問題が発生しました。
        * 考えられるアクション：
        *   サーバ ログで詳細を確認します。
        *   トランザクションの失敗を報告して再試行を要請します。
        */

        /* TCP_COMMUNICATION_ERROR
        * 考えられる理由：
        *   RiskFort サーバでの通信障害。
        * 考えられるアクション：
        *   トランザクションの失敗を報告して再試行を要請します。
        */
```

```
        /* UNKNOWN_ERROR
        * 考えられる理由：
        *    RiskFort サーバからのレスポンスが壊れています。
        * 考えられるアクション：
        *    トランザクションの失敗を報告して再試行を要請します。
        */

        /* USER_NOT_FOUND
        * 考えられる理由：
        *    API への入力に指定されたユーザは存在しません。
        * 考えられるアクション：
        *    適切なアクションを実行します。
        *    例： 渡したユーザ名が正しいことを確認します。
        */

        /* USERNAME_MISSING
        * 考えられる理由：
        *    ユーザ名が API への入力ユーザ オブジェクトに設定されていません。
        * 考えられるアクション：
        *    正確なユーザ名属性を API の入力ユーザ オブジェクトに設定します。
        */
    }
  }
}
```

Windows でコンパイルする方法

このテスト プログラムをコンパイルするには、RiskFort_Issuance.java と言う名前のファイルに保存します。arcot-riskfort-issuance.jar および関連する jar ファイルの arcot_core.jar、bcprov-jdk14-131.jar、commons-beanutils.jar、commons-collections-3.1.jar、commons-digester.jar、commons-lang-2.0.jar、

commons-logging.jar、commons-pool.jar、dom4j-1.6.1.jar、json-lib-0.7.1.jar、log4j-1.2.9.jar、および servlet.jar が JAVA コンパイラの CLASSPATH にあることを確認します。次に、JAVA コンパイラを実行します。

arcot-riskfort-issuance.jar ファイルは通常、RiskFort インストール ディレクトリの sdk\java\lib**arcot** ディレクトリにあります。RiskFort_Issuance.java ファイルが \Program Files\ARCOT SYSTEMS\sdk**\java** に保存されている場合は、以下のコマンドを使用します。JAVA ファイルがこのディレクトリにない場合は、CLASSPATH の sdk\java**lib** ディレクトリにフルパス名を指定します。

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

```
> javac -classpath
".;lib\arcot\arcot-riskfort-issuance.jar;lib\arcot\arcot_core.jar;lib\ex
ternal\bcprov-jdk14-131.jar;lib\external\commons-beanutils.jar;lib\exter
nal\commons-collections-3.1.jar;lib\external\commons-digester.jar;lib\ex
ternal\commons-lang-2.0.jar;lib\external\commons-logging.jar;lib\externa
l\commons-pool.jar;lib\external\dom4j-1.6.1.jar;lib\external\json-lib-0.
7.1.jar;lib\external\log4j-1.2.9.jar;lib\external\servlet.jar;%CLASSPATH
%" RiskFort_Issuance.java
```

これにより、JAVA ファイルと同じディレクトリに出力ファイル RiskFort_Issuance.class が作成されます。

UNIX プラットフォームでコンパイルする方法

このテスト プログラムをコンパイルするには、RiskFort_Issuance.java と言う名前のファイルに保存します。arcot-riskfort-issuance.jar および関連する jar ファイルの arcot_core.jar、bcprov-jdk14-131.jar、commons-beanutils.jar、commons-collections-3.1.jar、commons-digester.jar、commons-lang-2.0.jar、commons-logging.jar、commons-pool.jar、dom4j-1.6.1.jar、json-lib-0.7.1.jar、log4j-1.2.9.jar、および servlet.jar が JAVA コンパイラの CLASSPATH にあることを確認します。次に、JAVA コンパイラを実行します。

arcot-riskfort-issuance.jar ファイルは通常、RiskFort インストール ディレクトリの sdk/java/lib/**arcot** ディレクトリにあります。RiskFort_Issuance.java ファイルが /opt/arcot/sdk/**java** に保存されている場合は、以下のコマンドを使用します。JAVA ファイルがこのディレクトリにない場合は、CLASSPATH の sdk/java/**lib** ディレクトリにフルパス名を指定します。

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

```
> javac -classpath
".:./lib/arcot/arcot-riskfort-issuance.jar:./lib/arcot/arcot_core.jar:./
lib/external/bcprov-jdk14-131.jar:./lib/external/commons-beanutils.jar:.
/lib/external/commons-collections-3.1.jar:./lib/external/commons-digeste
r.jar:./lib/external/commons-lang-2.0.jar:./lib/external/commons-logging
.jar:./lib/external/commons-pool.jar:./lib/external/dom4j-1.6.1.jar:./li
b/external/json-lib-0.7.1.jar:./lib/external/log4j-1.2.9.jar:./lib/exter
nal/servlet.jar:$CLASSPATH" RiskFort_Issuance.java
```

これにより、JAVA ファイルと同じディレクトリに出力ファイル RiskFort_Issuance.class が作成されます。

Windows で実行する方法

テストを実行する前に、Arcot RiskFort Service がインストールされて開始されている必要があります。テストを実行するには、SDK ライブラリがパス内にあり、arcot-riskfort-issuance.jar が CLASSPATH 内にあることを確認してから、以下に示されているように JAVA コマンドを実行します。

テストを実行するには、以下のコマンドを使用します。

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

```
> java -classpath
".;lib\arcot\arcot-riskfort-issuance.jar;lib\arcot\arcot_core.jar;lib\ex
ternal\bcprov-jdk14-131.jar;lib\external\commons-beanutils.jar;lib\exter
nal\commons-collections-3.1.jar;lib\external\commons-digester.jar;lib\ex
ternal\commons-lang-2.0.jar;lib\external\commons-logging.jar;lib\externa
l\commons-pool.jar;lib\external\dom4j-1.6.1.jar;lib\external\json-lib-0.
7.1.jar;lib\external\log4j-1.2.9.jar;lib\external\servlet.jar;%CLASSPATH
%" RiskFort_Issuance
```

UNIX プラットフォームで実行する方法

テストを実行する前に、Arcot RiskFort Service がインストールされて開始されている必要があります。テストを実行するには、SDK ライブラリがパス内にあり、arcot-riskfort-issuance.jar が CLASSPATH 内にあることを確認してから、以下に示されているように JAVA コマンドを実行します。

テストを実行するには、以下のコマンドを使用します。

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

```
> java -classpath
".:./lib/arcot/arcot-riskfort-issuance.jar:./lib/arcot/arcot_core.jar:./
lib/external/bcprov-jdk14-131.jar:./lib/external/commons-beanutils.jar:.
```

```
/lib/external/commons-collections-3.1.jar:./lib/external/commons-digeste
r.jar:./lib/external/commons-lang-2.0.jar:./lib/external/commons-logging
.jar:./lib/external/commons-pool.jar:./lib/external/dom4j-1.6.1.jar:./li
b/external/json-lib-0.7.1.jar:./lib/external/log4j-1.2.9.jar:./lib/exter
nal/servlet.jar:$CLASSPATH" RiskFort_Issuance
```

出力

```
Initializing RiskFort Issuance API using
/properties/riskfort.issuance.properties
RiskFort Issuance API initialized.
The following credentials are used to create a user.
Username: TestUser
organization name = DEFAULTORG
First name: Test
Last name: User
Email: TestUser@abcd.com
PAM: My PAM
Calling create.
create succeeded.
PAM changed to: New changed PAM
Calling update.
update succeeded.
Calling read with username = TestUser and organization name = DEFAULTORG
User details read are as follows:
Username: TestUser
organization name = DEFAULTORG
First name: Test
Last name: User
Email: TestUser@abcd.com
PAM: New changed PAM
```

リスク評価および事後評価のためのサンプル コード

以下のサンプル コード スニペットをアプリケーション コードに挿入して、RiskFort のリスク評価および事後評価の機能をテストすることができます。

```
/* RiskFort Transaction API にインポートするパッケージ */

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.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 {

// この例の値は、サンプルで使用するためにハード コーディングされたものです。
public static void main(String[] args) {
  /*
  initialize:
    入力プロパティ ファイルから API オブジェクトを初期化します。
    initialize() はアプリケーションのスタートアップ時に一度だけ呼び出す必要があります。
```

```
    以下にプロパティ ファイルのフィールドおよびフォーマットを示します。
      HOST.1=RiskFort サーバの IP アドレス
      PORT.1=RiskFort サーバのポート番号
      TRANSPORT_TYPE = 接続タイプ。使用可能な値は TLS または TCP です。
      CA_CERT_FILE = TRANSPORT_TYPE = TLS の場合に必要な CA 証明書ファイル。サーバ CA
証明書（PEM 形式）のファイル パス。

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

  パラメータ：
      propertyLocation - クラス パスに対してパラメータとして渡すロケーションを示します。
      <code>null</code> が渡された場合は、デフォルトのロケーションである
properties/riskfort.risk-evaluation.properties が使用されます。
  戻り値：

  スローする値：
    RiskException - 何らかの理由でリクエストが失敗した場合。
    IOException
  */



  // 入力プロパティ ファイルから API オブジェクトを初期化するためのサンプル コード。
  // RiskXActionAPI オブジェクトを作成します。
  RiskXActionAPI api = null;

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

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

    // 入力プロパティ ファイルから RiskXActionAPI オブジェクトを初期化します。
```

```
      RiskFactory.initialize(propertiesFileLocation);
      // 以前に初期化されている RiskXActionAPI オブジェクトを取得します。
      api = RiskFactory.getRiskXActionAPI();

      System.out.println("RiskFort API initialized.");
  catch (IOException e) {
    // 適切なアクションを実行します。

  } catch (RiskException e) {

    /* UserRepositoryException オブジェクトに対して以下のメソッドを使用し、エラー コードおよびエラー メッセージを以下のように取得できます。
    * 文字列コード = e.getErrorCode();
    * 文字列メッセージ = e.getMessage();
    */
    System.out.println("Exception during initialize.");
    System.out.println("Error code: " + e.getErrorCode());
    System.out.println("Error message: " + e.getMessage());

    /* 以下のエラー コードが API によって返されます。*/

    /* ERRCODE_INVALID_INPUT
    * これには、2 つの理由が考えられます。
    * 1. 考えられる理由：
    *    プロパティ ファイルに指定されたトランスポートのタイプが無効です。
    *    考えられるアクション：
    *     「riskfort.risk-evaluation.properties」ファイルに有効なトランスポートのタイプを指定します。
    * 2. 考えられる理由：
    *    プロパティ ファイルで TRANSPORT_TYPE=TLS の場合、TLS に使用されたプライマリ キー ファイルが見つかりませんでした。
    *    考えられるアクション：
    *     プライマリ キー ファイルへのパスが正確であることを確認します。
```

```
    */

    /* ERRCODE_RISKXACTIONAPI_ALREADY_INITIALIZED
    * 考えられる理由：
    *     初期化している API はすでに初期化されています。
    * 考えられるアクション：
    *    RiskXActionAPI オブジェクトを取得してトランザクションを続行します。
    */

    /* ERROR_CONF_INVALID_POOL
    * 考えられる理由：
    *  RiskFort サーバにライブ接続のプールを作成できません。
    * 考えられるアクション：
    *
    */

    /* ERROR_CONF_NOT_AVAILABLE
    * これには、2 つの理由が考えられます。
    * 1.考えられる理由：
    *     プロパティ ファイルを読み取ることができませんでした。
    *     考えられるアクション：
    *      プロパティ ファイルへのパスが正確であることを確認します。
    * 2.考えられる理由：
    *     サーバ証明書のルート CA が無効です。
    *     考えられるアクション：
    *      有効なサーバ証明書を取得します。
    */
  }



  /*
  evaluateRisk：
```

```
    トランザクションに関連付けられたリスクを評価し、適宜アドバイスを返します。また、新しい
DeviceId も提供します。

    呼び出しアプリケーションによって行われるアクションを以下に示します。

      1. 出力 DeviceId をユーザのマシンに何らかの形式で保存する必要があります。最も一般的な
方法は、それを HTTP cookie として保存することです。ただし、HTTP cookie として保存すると、
マシン上のすべての cookie が削除される場合に一緒に削除されてしまう危険があります。

      2. ユーザのマシンから DeviceId を取得し、それを setDeviceID を使用して設定します。

      3. RiskAdvice が INCREASEAUTH の場合は、2 回目認証を実行し、2 回目の認証の結果を
PostEvaluate を使用して RiskFort に渡します。


  public RiskAssessment evaluateRisk(java.lang.String callerId,

        DeviceContext deviceContext,

        LocationContext locationContext,

        UserContext userContext,

        TransactionContext transactionContext,

        AdditionalInputs additionalInputs)

     throws RiskException


  パラメータ :

    callerId - アプリケーション自体のトラッキング目的で API を呼び出すアプリケーションに
よって決定される識別子。

    deviceContext - デバイス コンテキストの情報。

    locationContext - ロケーション コンテキストの情報（IP アドレス）。

    userContext - ユーザ コンテキスト情報。

    transactionContext - トランザクション コンテキスト情報。

    additionalInputs - 別の操作で必要になる可能性がある追加の入力。

  戻り値 :

    RiskAssessment - RiskAdvice、ユーザのマシンに配置する必要のある新しい DeviceId、
RiskScore およびその他のトランザクションの関連情報が含まれています。

  スローする値 :

    RiskException - 何らかの理由でリクエストが失敗した場合。

  */


  // トランザクションに関連付けられたリスクを評価するためのサンプル コード。
```

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

  // リスク評価に使用されるコンテキストを構築します。
  String callerId = "MyApplicationTrackingId"; // string used by the calling
application for tracking across calls.

  // 入力ユーザ関連情報。
  UserContext userContext = new UserContext();

  // ユーザの一意の識別子。たとえば、銀行の場合はユーザの銀行口座番号です。
userContext.setUserId("TestUser");
  userContext.setOrg("DEFAULTORG");
  System.out.println("Username: " + userContext.getUserID());
  System.out.println("Organization Name:" + userContext.getOrg());

  // 入力デバイス関連情報
  DeviceContext deviceContext = new DeviceContext();

  // JSON 署名は mfp_json.js から取得されます。
  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\":
\"20070312\",\"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\":[{\"nam
e\":\"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},\"e
xtra\":{\"javascript_ver\":\"1.6\",\"timezone\":-330}}";
```

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


  // デバイス ID を設定します。
  String idType = "HTTP_COOKIE";
  /* evaluateRisk が初めて呼び出される場合、デバイスは RiskFort サーバによって認識されないため、deviceId=null です。
   * RiskFort サーバはユーザのマシンの cookie に deviceId を設定します。これは、その後のトランザクションの間に RiskFort サーバに渡されます。
  */
  文字列 deviceId = null;
  deviceContext.setDeviceID(idType, deviceId);


  /* 各トランザクションでは、deviceId または aggregatorID の両方ではなく、いずれかを設定する必要があります。


deviceContext.setAggregatorID("LcPywTghrtyed6KDuRcMbWiFFTYR2oFThfdDOtBKqKcdEXsH9dFIFfrr/dsfdud");


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


  // 入力ロケーション関連情報。
  LocationContext locationContext = new LocationContext();


  InetAddress ipAddress = null;
  // ユーザのマシンの IP アドレス。これは一般的に HTTP ヘッダから抽出されます。
  try {
      ipAddress = InetAddress.getByName("10.150.10.150");
  } catch (UnknownHostException e) {
      // 適切なアクションを実行します。
  }
```

```
  locationContext.setIpAddress(ipAddress);
  System.out.println("Ip address: " + locationContext.getIPAddress());

  // 入力トランザクション関連情報。
  TransactionContext transactionContext = new TransactionContext();

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

  /* 各トランザクションに対し、トランザクション コンテキストに拡張可能エレメントを設定するか、
または追加の入力を設定する必要があります。

transactionContext.setExtensibleElements("MerchantID=id;MerchantCountry=countr
y;MerchantName=name");
  */

  Map additionalInputs = new HashMap();

  String extName1 = "MerchantID";
  String extValue1 = "id";
  String extName2 = "MerchantCountry";
  String extValue2 = "country";
  String extName3 = "MerchantName";
  String extValue3 = "name";
  if(extName1 != null && extName1 != "" )
    additionalInputs.put(extName1, extValue1);
  if(extName2 != null && extName2 != "" )
    additionalInputs.put(extName2, extValue2);
  if(extName3 != null && extName3 != "" )
    additionalInputs.put(extName3, extValue3);

  try {
```

```
    System.out.println("evaluateRisk called.");
    // API を呼び出してトランザクションに関連付けられたリスクを評価します。
    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 (RiskException e) {
    /* UserRepositoryException オブジェクトに対して以下のメソッドを使用し、エラー コード
およびエラー メッセージを以下のように取得できます。
    * 文字列コード = e.getErrorCode();
    * 文字列メッセージ = e.getMessage();
    */
    System.out.println("Exception in 'evaluateRisk'.");
    System.out.println("Error code: " + e.getErrorCode());
    System.out.println("Error message: " + e.getMessage());


    /* 以下のエラー コードが API によって返されます。*/


    /* ERRCODE_INVALID_PACKET_FROM_SERVER
    * 考えられる理由：
    *   サーバから受信したパケットのタイプが無効です。
    * 考えられるアクション：
    *   トランザクションの失敗を報告して再試行を要請します。
    */


    /* ERRCODE_PARSING_DATA
    * 考えられる理由：
    *   サーバから xml を解析する際にエラーが発生しました。
    * 考えられるアクション：
    *   トランザクションの失敗を報告して再試行を要請します。
    */
```

```
}


/*
postEvaluate :
   evaluateRisk の出力と、呼び出しアプリケーションによって実行された 2 回目の認証（存在する場合）に基づいて、トランザクションに対する最終的な決定を行います。
   また、必要に応じて RiskFort システムの情報を更新します。


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



パラメータ :
   callerId - アプリケーション自体のトラッキング目的で API を呼び出すアプリケーションによって決定される識別子。
   riskAssessment - evaluateRisk からの出力
   secondaryAuthenticationStatus - 2 回目の認証の結果。
        2 回目の認証が成功した場合は、「true」を渡し、それ以外の場合は「false」を渡します。
        evaluateRisk が INCREASEAUTH 以外のアドバイスを戻した場合（2 回目の認証を要求されなかった場合）は「false」を渡します。


   associationName - トランザクションを実行したマシンに関連付ける名前としてユーザが選択した値。
        ユーザは共有マシンに対して関連付けを選択しないようにする必要があります。その場合は、「null」を渡すことができます。
   additionalInputs - 別の操作で必要になる可能性がある追加の入力。これは今後の使用の為に保持されています。
戻り値 :
```

```
    PostEvaluateResponse - このトランザクションの続行を許可するかどうかを示します。
isAllowAdvised() を使用して確認することができます。トランザクションを許可する必要がある場合
は「true」を返し、拒否する必要がある場合は「false」を返します。


  スローする値：

    RiskException - 何らかの理由でリクエストが失敗した場合。

  */


  // evaluateRisk の出力および呼び出しアプリケーションによって実行された 2 回目の認証に基づ
いて最終的な決定を行うサンプル コード。

  // ここで入力として渡された RiskAssessment オブジェクトは evaluateRisk() への呼び出し
によって返されたオブジェクトです。


  PostEvaluateResponse postEvalResponse = null;


  String associationName; // トランザクションが実行されているマシンに関連付けられた名前


  boolean secondaryAuthenticationStatus; // 呼び出しアプリケーションによって実行されて
いる可能性のある 2 回目の認証の結果。


  // postEvaluate 呼び出しで使用されるコンテキストを構築します。

  associationName = "testAssociationName";

  secondaryAuthenticationStatus = true;

  try {

    System.out.println("Calling postEvaluate with Secondary Authentication
Status = " + secondaryAuthenticationStatus );

    System.out.println("Association name passed: " + associationName);


    // API を呼び出して evaluateRisk および 2 回目の認証に基づいた最終的な決定を行います。

    postEvalResponse = api.postEvaluate(callerId, riskAssessment,
secondaryAuthenticationStatus, associationName, additionalInputs);


    System.out.println("postEvaluate succeeded.");

  } catch (RiskException e) {
```

```
    /* UserRepositoryException オブジェクトに対して以下のメソッドを使用し、エラー コード
およびエラー メッセージを以下のように取得できます。
    * 文字列コード = e.getErrorCode();
    * 文字列メッセージ = e.getMessage();
    */
    System.out.println("Exception in 'postEvaluate'.");
    System.out.println("Error code: " + e.getErrorCode());
    System.out.println("Error message: " + e.getMessage());

    /* 以下のエラー コードが API によって返されます。*/

    /* ERRCODE_INVALID_PACKET_FROM_SERVER
    * 考えられる理由：
    *   サーバから受信したパケットのタイプが無効です。
    * 考えられるアクション：
    *   トランザクションの失敗を報告して再試行を要請します。
    */

    /* ERRCODE_PARSING_DATA
    * 考えられる理由：
    *   サーバからの xml を解析する際にエラーが発生しました。
    * 考えられるアクション：
    *   トランザクションの失敗を報告して再試行を要請します。
    */
  }
  System.out.println("Risk Evaluation done.");
  }
}
```

Windows でコンパイルする方法

このテスト プログラムをコンパイルするには、Assess_Risk.java と言う名前のファイルに保存します。arcot-riskfort-issuance.jar および関連する jar ファイルの arcot_core.jar、arcot-riskfort-mfp.jar、bcprov-jdk14-131.jar、

commons-beanutils.jar、commons-collections-3.1.jar、commons-digester.jar、commons-lang-2.0.jar、commons-logging.jar、commons-pool.jar、dom4j-1.6.1.jar、json-lib-0.7.1.jar、log4j-1.2.9.jar、および servlet.jar が JAVA コンパイラの CLASSPATH にあることを確認します。次に、JAVA コンパイラを実行します。

arcot-riskfort-issuance.jar ファイルは通常、RiskFort インストール ディレクトリの sdk\java\lib**arcot** ディレクトリにあります。Assess_Risk.java ファイルが \Program Files\Arcot Systems\sdk**java** に保存されている場合は、以下のコマンドを使用します。JAVA ファイルがこのディレクトリにない場合は、CLASSPATH の sdk\java**lib** ディレクトリにフルパス名を指定します。

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

```
> javac -classpath
".;lib\arcot\arcot-riskfort-issuance.jar;lib\arcot\arcot_core.jar;lib\ar
cot\arcot-riskfort-mfp.jar;lib\external\bcprov-jdk14-131.jar;lib\externa
l\commons-beanutils.jar;lib\external\commons-collections-3.1.jar;lib\ext
ernal\commons-digester.jar;lib\external\commons-lang-2.0.jar;lib\externa
l\commons-logging.jar;lib\external\commons-pool.jar;lib\external\dom4j-1
.6.1.jar;lib\external\json-lib-0.7.1.jar;lib\external\log4j-1.2.9.jar;li
b\external\servlet.jar;%CLASSPATH%" Assess_Risk.java
```

これにより、JAVA ファイルと同じディレクトリに出力ファイル Assess_Risk.class が作成されます。

UNIX プラットフォームでコンパイルする方法

このテスト プログラムをコンパイルするには、Assess_Risk.java と言う名前のファイルに保存します。arcot-riskfort-issuance.jar および関連する jar ファイルの arcot_core.jar、arcot-riskfort-mfp.jar、bcprov-jdk14-131.jar、commons-beanutils.jar、commons-collections-3.1.jar、commons-digester.jar、commons-lang-2.0.jar、commons-logging.jar、commons-pool.jar、dom4j-1.6.1.jar、json-lib-0.7.1.jar、log4j-1.2.9.jar、および servlet.jar が JAVA コンパイラの CLASSPATH にあることを確認します。次に、JAVA コンパイラを実行します。

arcot-riskfort-issuance.jar ファイルは通常、RiskFort インストール ディレクトリの sdk/java/lib/**arcot** ディレクトリにあります。Assess_Risk.java ファイルが /opt/arcot/sdk/**java** に保存されている場合は、以下のコマンドを使用します。JAVA ファイルがこのディレクトリにない場合は、CLASSPATH の sdk/java/**lib** ディレクトリにフルパス名を指定します。

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

```
> javac -classpath
".:./lib/arcot/arcot-riskfort-issuance.jar:./lib/arcot/arcot_core.jar:./
lib/arcot/arcot-riskfort-mfp.jar:./lib/external/bcprov-jdk14-131.jar:./l
ib/external/commons-beanutils.jar:./lib/external/commons-collections-3.1
.jar:./lib/external/commons-digester.jar:./lib/external/commons-lang-2.0
.jar:./lib/external/commons-logging.jar:./lib/external/commons-pool.jar:
./lib/external/dom4j-1.6.1.jar:./lib/external/json-lib-0.7.1.jar:./lib/e
xternal/log4j-1.2.9.jar:./lib/external/servlet.jar:$CLASSPATH"
Assess_Risk.java
```

これにより、JAVA ファイルと同じディレクトリに出力ファイル `Assess_Risk.class` が作成されます。

Windows で実行する方法

テストを実行する前に、Arcot RiskFort Service がインストールされて開始されている必要があります。テストを実行するには、SDK ライブラリがパス内にあり、`arcot-riskfort-issuance.jar` が `CLASSPATH` 内にあることを確認してから、以下に示されているように JAVA コマンドを実行します。

テストを実行するには、以下のコマンドを使用します。

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


> java -classpath
".;lib\arcot\arcot-riskfort-issuance.jar;lib\arcot\arcot_core.jar;lib\ar
cot\arcot-riskfort-mfp.jar;lib\external\bcprov-jdk14-131.jar;lib\externa
l\commons-beanutils.jar;lib\external\commons-collections-3.1.jar;lib\ext
ernal\commons-digester.jar;lib\external\commons-lang-2.0.jar;lib\externa
l\commons-logging.jar;lib\external\commons-pool.jar;lib\external\dom4j-1
.6.1.jar;lib\external\json-lib-0.7.1.jar;lib\external\log4j-1.2.9.jar;li
b\external\servlet.jar;%CLASSPATH%" Assess_Risk
```

UNIX プラットフォームで実行する方法

テストを実行する前に、Arcot RiskFort Service がインストールされて開始されている必要があります。テストを実行するには、SDK ライブラリがパス内にあり、`arcot-riskfort-issuance.jar` が `CLASSPATH` 内にあることを確認してから、以下に示されているように JAVA コマンドを実行します。

テストを実行するには、以下のコマンドを使用します。

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

```
> java -classpath
".:./lib/arcot/arcot-riskfort-issuance.jar:./lib/arcot/arcot_core.jar:./
lib/arcot/arcot-riskfort-mfp.jar:./lib/external/bcprov-jdk14-131.jar:./l
ib/external/commons-beanutils.jar:./lib/external/commons-collections-3.1
.jar:./lib/external/commons-digester.jar:./lib/external/commons-lang-2.0
.jar:./lib/external/commons-logging.jar:./lib/external/commons-pool.jar:
./lib/external/dom4j-1.6.1.jar:./lib/external/json-lib-0.7.1.jar:./lib/e
xternal/log4j-1.2.9.jar:./lib/external/servlet.jar:$CLASSPATH"
Assess_Risk
```

出力

```
Initializing RiskFort API using /properties/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:
{"DEVICESIG":{"VERSION":"1.0","OS_BROWSER":{"browser_ver":"1.5.0.11","os":"Win
dows NT
5.0","browser":"Netscape","javascript_ver":"1.6"},"SCREEN":{"colorDepth":"32",
"availWidth":"1024","width":"1024","height":"768","availHeight":"690"},"HTTP_H
EADER":{"user-agent":"Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US;
rv:1.8.0.11) Gecko/20070312
Firefox/1.5.0.11"},"OPTIONAL":{"SOFTWARE":[{"name":"Adobe Acrobat
Plugin","version":"7.00"},{"name":"Macromedia
Director","version":"10.1"},{"name":"Windows Media Player Plug-in Dynamic Link
Library"},{"name":"Macromedia Shockwave Flash","version":"9.0"},{"name":"Java
Virtual
Machine","version":"1.6.0"}]},"USER_PREF":{"sys_lang":"en-us","timezone":"-330
","user_lang":"en-us"}}}

Ip address: /10.150.10.150

evaluateRisk called.

evaluateRisk succeeded.

Device Id set on the user's machine:
EGPbuaASigkCN0qLiZePTTbpD4SZIQWS5CNlbEHVhrDVR1nd0lsni9cc/vpvtppg

Calling postEvaluate with Secondary Authentication Status = true

Association name passed: testAssociationName

postEvaluate succeeded.

Risk Evaluation done.
```

付録 C
Java API 関連資料

RiskFort SDK は、オンライン アプリケーションに RiskFort オブジェクトとプログラムによって統合する方法を提供する一連の Java クラスおよびメソッドを構成します。RiskFort Java SDK は、以下のコンポーネントで構成されています。

- Risk Evaluation Java クラス
- Issuance Java クラス
- 関連する Java クラスおよびメソッドに対する Javadoc 情報

注： RiskFort に同梱されているサンプル アプリケーションは、Java クラスおよびメソッドの使用法を実例で示します。RiskFort サンプル アプリケーションの詳細については、付録 G の 「RiskFort サンプル アプリケーション」を参照してください。

Javadoc HTML ドキュメントへのアクセス

RiskFort SDK で提供された Javadoc 情報を本書および他の Java 参考資料と共に使用して、RiskFort のリスク評価および発行サービスを新規または既存の Java アプリケーションに追加できます。

既存の RiskFort アプリケーションを更新する場合は、変更する前に非推奨の Java API についてリリース ノートおよび Javadoc HTML ドキュメントを参照する必要があります。

完全なディレクトリ ツリーで構成されている Javadoc HTML 情報を開発システムの別の場所にコピーし、次に、RiskFort SDK およびランタイム コンポーネントをアンインストールできます。あるいは、RiskFort パッケージから直接 Javadoc にアクセスすることもできます。

Javadoc のインストール場所については、C-122 ページの表 C-1 および C-127 ページの表 C-3 を参照してください。

Risk Evaluation API

表 C-1 に、Risk Evaluation SDK コンポーネントの一部としてインストールされるファイルをリスト表示します。これらのファイルの基本の場所は次のとおりです。

Windows

`<install_location>\Arcot Systems\`

UNIX プラットフォーム

`<install_location>/arcot/`

表 C-1. Risk Evaluation API ファイル

場所	ファイル名	説明
docs**riskfort**\ (Windows) docs/**riskfort**/ (UNIX プラットフォーム)	Arcot-RiskFort-2.2.6-risk-evaluation-sdk-javadocs.zip または Arcot-RiskFort-2.2.6-risk-evaluation-sdk-javadocs.tar.gz	Risk Evaluation Java API と共に提供されている Java クラスおよびメソッドの Javadoc HTML ドキュメント。

表 C-1. Risk Evaluation API ファイル

場所	ファイル名	説明
samples**java**\ (Windows) samples/**java**/ (UNIX プラットフォーム)	riskfort-2.2.6-sample-application.war	Risk Evaluation および Issuance API の両方の使用法を実例で示すデモンストレーション。 サンプル アプリケーションを実行し使用する方法の詳細については、付録 G の「RiskFort サンプル アプリケーション」を参照してください。
	riskfort-2.2.6-sample-callouts.war	コールアウト機能の使用法を実例で示すデモンストレーション。 **関連文書：** サンプル コールアウトを実行し使用する方法の詳細については、「Arcot RiskFort *2.2.6* 管理ガイド」の第 9 章「サンプル コールアウトでの作業」を参照してください。
sdk\java\lib**arcot**\ (Windows) sdk/java/lib/**arcot**/ (UNIX プラットフォーム)	arcot_core.jar	Arcot 製品の構築に使用される共有コンポーネント、ツールキットおよびサービスのセットが含まれている専用 Java Archive（JAR）ファイル。
	arcot-pool.jar	RiskFort リソースパックとユーザ データ サービス（UDS）サーバ間の接続プールに必要なクラスとメソッドが含まれている Java Archive （JAR）ファイル。
	arcot-riskfort-evaluaterisk.jar	Risk Evaluation API に関連付けられたクラスとメソッドが含まれている Java Archive（JAR）ファイル。
	arcot-riskfort-mfp.jar	Device ID およびマシン FingerPrint（MFP）コレクションに関連付けられたクラスとメソッドが含まれている Java Archive （JAR）ファイル。

表 C-1. Risk Evaluation API ファイル

場所	ファイル名	説明
sdk\java**properties**\ (Windows) sdk/java/**properties**/ (UNIX プラットフォーム)	log4j.properties.risk-evaluation	ロギング動作を指定するために Risk Evaluation API に関連付けられたクラスとメソッドによって使用されるプロパティ ファイル。
	riskfort.risk-evaluation.properties	RiskFort サーバ情報を読み込むために Risk Evaluation API に関連付けられたクラスとメソッドによって使用されるプロパティ ファイル。

Risk Evaluation API で使用するサードパーティ ソフトウェア

Risk Evaluation API は、表 C-2 にリスト表示されているサードパーティ ソフトウェアも使用します。

- **Windows**

  <*install_location*>\Arcot Systems**sdk**\java\lib**external**\

- **UNIX プラットフォーム**

  <*install_location*>/arcot/**sdk**/java/lib/**external**/

表 C-2. Risk Evaluation API で使用するサードパーティ ファイル

ファイル名	説明
bcprov-jdk14-139.jar	暗号化の操作をサポートするための Bouncy Castle API。 *http://www.bouncycastle.org/latest_releases.html*
commons-beanutils-1.7.0.jar	Java java.lang.reflect および java.beans パッケージの周囲の Apache ラッパー。 *http://commons.apache.org/beanutils/*
commons-collections-3.1.jar	JDK クラスの新しいインターフェース、実装およびユーティリティをサポートするための Apache パッケージ。 *http://commons.apache.org/collections/*
commons-httpclient-3.1.jar	クライアント側認証、HTTP 状態管理および HTTP 接続管理をサポートするための Apache パッケージ。 *http://hc.apache.org/httpclient-3.x/index.html*

表 C-2. Risk Evaluation API で使用するサードパーティ ファイル

ファイル名	説明
commons-lang-2.0.jar	文字列処理方式、基本的な数値法、オブジェクト反射、作成とシリアル化、およびシステム プロパティのサポートを提供する Apache Java ユーティリティ パッケージ。 *http://commons.apache.org/lang/*
commons-logging-1.0.4.jar	異なるロギングの実装をサポートする Apache のパッケージ。 *http://commons.apache.org/logging/*
commons-pool-1.4.jar	オブジェクト プールの実装をサポートするための Apache パッケージ。 *http://commons.apache.org/pool/*
dom4j-1.6.1.jar	統合 XPath を使用して XML を処理し、DOM、SAX、JAXP および Java 2 コレクションをサポートするためのオープンソースの XML フレームワーク。 *http://www.dom4j.org/*
jaxen-1.1-beta-8.jar	DOM、XOM、dom4j および JDOM を含めて別のオブジェクトモデルと共に使用できる汎用 Java XPath エンジン。 *http://jaxen.codehaus.org/*
jdom-1.0.jar	Java コードから XML データにアクセスし、操作し、出力するための Java ベースのソリューション。 *http://www.jdom.org/*
json-lib-0.7.1.jar	Beans、マップ、コレクション、Java 配列および XML を JSON に変換して再度 Beans に戻すための Java ライブラリ。 *http://www.json.org/java/*
log4j-1.2.9.jar	アプリケーションのランタイム ロギング動作を制御するための Apache パッケージ。 *http://logging.apache.org/log4j/1.2/index.html*
servlet-api-2.4.jar	すべてのサーブレットが実装する必要のあるメソッドを定義するための Java パッケージ。 *http://java.sun.com/products/servlet/2.2/javadoc/javax/servlet/package-summary.html*
oro-2.0.8.jar	Perl5 互換正規表現、AWK のような正規表現、glob 表現、ならびに代入、分割、およびファイル名フィルタリングを実行するためのユーティリティ クラスを提供する文書処理 Java クラス。 *http://jakarta.apache.org/oro/*

表 C-2. Risk Evaluation API で使用するサードパーティ ファイル

ファイル名	説明
xalan-2.7.0.jar	XML ドキュメントを HTML、テキスト、または他の XML ドキュメント タイプに変換する XSLT プロセッサ。 *http://xml.apache.org/xalan-j/*
xercesImpl-2.6.2.jar	XML パーサ コンポーネントおよび設定を構築するためのフレームワーク。 *http://xerces.apache.org/xerces2-j/*
xml-apis-1.0.b2.jar	Java Management Extensions （JMX）互換の Model MBeans を作成するフレームワーク。 *http://commons.apache.org/modeler/*
xmlParserAPIs-2.6.2.jar	XML パーサ コンポーネントおよび設定を構築するためのフレームワーク。 *http://xml.apache.org/commons/*
xom-1.1.jar	Java で XML を処理するための XML オブジェクト モデル。 *http://www.xom.nu/*

Issuance API

表 C-3 に、発行 SDK コンポーネントの一部としてインストールされるファイルをリスト表示します。これらのファイルの基本の場所は次のとおりです。

- **Windows**

  `<install_location>\Arcot Systems\`

- **UNIX プラットフォーム**

  `<install_location>/arcot/`

表 C-3. Issuance API ファイル

場所	ファイル名	説明
docs**riskfort**\ (Windows) docs/**riskfort**/ (UNIX プラットフォーム)	Arcot-RiskFort-2.2.6-issuance-sdk-javadocs.zip または Arcot-RiskFort-2.2.6-issuance-sdk-javadocs.tar.gz	Issuance Java API と共に提供されている Java クラスおよびメソッドに関する Javadoc HTML ドキュメント。
samples**java**\ (Windows) samples/**java**/ (UNIX プラットフォーム)	riskfort-2.2.6-sample-application.war	Risk Evaluation および Issuance API の両方の使用法を実例で示すデモンストレーション。 サンプル アプリケーションを実行し使用する方法の詳細については、付録 G の「RiskFort サンプル アプリケーション」を参照してください。
sdk\java\lib**arcot**\ (Windows) sdk/java/lib/**arcot**/ (UNIX プラットフォーム)	arcot_core.jar	Arcot 製品の構築に使用される共有コンポーネント、ツールキットおよびサービスのセットが含まれている専用 Java Archive（JAR）ファイル。
	arcot-pool.jar	RiskFort リソースパックとユーザ データ サービス（UDS）サーバ間の接続プールに必要なクラスとメソッドが含まれている Java Archive （JAR）ファイル。
	arcot-riskfort-issuance.jar	Issuance API に関連付けられたクラスとメソッドが含まれている Java Archive（JAR）ファイル。
sdk\java**properties**\ (Windows) sdk/java/**properties**/ (UNIX プラットフォーム)	log4j.properties.riskfort-issuance	ロギング動作を指定するために Issuance API に関連付けられたクラスとメソッドによって使用されるプロパティ ファイル。
	riskfort.issuance.properties	RiskFort サーバ情報を読み込むために Issuance API に関連付けられたクラスとメソッドによって使用されるプロパティ ファイル。

Issuance API で使用するサードパーティ ソフトウェア

Issuance API は、以下に示すソフトウェアを除き、表 C-2 にリスト表示されているすべてのサードパーティ ソフトウェアも使用します。

- json-lib-0.7.1.jar
- servlet-api-2.4.jar

付録 D
例外コードおよびエラー コード

この付録では、RiskFort SDK によってスローされるすべての例外およびエラー コードの一覧を表示します。

SDK 例外

SDK エラー コードは次のように分類されます。

Risk Evaluation 例外

関連する操作が失敗した場合、RiskFort Risk Evaluation API は `RiskException` をスローします。`RiskException` クラスの `getErrorCode()` 関数はエラーのタイプに対応する文字列値を返します。可能な値を表 D-1 にリスト表示します。

表 D-1. Risk Evaluation エラー コード

エラー コード	値	説明
ERRCODE_INVALID_INPUT	ERRCODE_INVALID_INPUT	RiskFort サーバが受信した入力データが有効ではありません。 **注：** このエラーは頻繁にスローされません。
ERRCODE_INVALID_PACKET_FROM_SERVER	ERROR_INVALID_PACKET_FROM_SERVER	Server パケットが不正な形式であるか、破損しています。 **注：** このエラーは頻繁にスローされます。

表 D-1. Risk Evaluation エラー コード

エラー コード	値	説明
ERRCODE_MISSING_PARAMETER_IN_XML	ERRCODE_MISSING_PARAMETER_IN_XML	1 つ以上の XML パラメータが不完全であるか、不足しています。 注：このエラーは頻繁にスローされます。
ERRCODE_PARSING_DATA	ERROR_PARSING_DATA	RiskFort サーバは指定されたデータを解析できませんでした。
ERRCODE_RISKXACTIONAPI_ALREADY_INITIALIZED	ERRCODE_RISKXACTIONAPI_ALREADY_INITIALIZED	RiskXActionAPI インターフェースはすでに初期化されています。
ERRCODE_RISKXACTIONAPI_NOT_INITIALIZED	ERRCODE_RISKXACTIONAPI_NOT_INITIALIZED	RiskXActionAPI インターフェースはまだ初期化されていません。
ERRCODE_TRANSACTIONID_NULL	ERROR_TRASACTIONID_NULL	TransactionID の値が null です。
ERROR_CONF_INVALID_POOL	0135211004	RiskFort サーバへの接続を確立できませんでした。
ERROR_CONF_NOT_AVAILABLE	0135211005	API の設定に使用されるプロパティの設定キーをプロパティ ファイルから正しく読み取ることができませんでした。

Issuance 例外

関連する操作が失敗した場合、RiskFort ユーザ管理 API は UserRepositoryException をスローします。UserRepositoryException クラスの getCode() 関数はエラーのタイプに対応する文字列値を返します。可能な値を表 D-2 にリスト表示します。

表 D-2. Issuance エラー コード

エラー コード	値	説明
ERROR_CONF_API	14	API の設定に使用されるプロパティの設定キーをプロパティ ファイルから正しく読み取ることができませんでした。
INVALID_EMAIL	12	指定された電子メールは有効ではありません。
INVALID_ORG_NAME	1	指定された組織名は有効ではありません。
INVALID_USER_OBJECT	5	指定された User オブジェクトは無効です。

表 D-2. Issuance エラー コード

エラー コード	値	説明
MANDATORY_ATTRIB_EXCEEDS_SIZE	8	指定された必須属性は許可される最大サイズを超えています。
MANDATORY_ATTRIB_INVALID_CHAR	9	指定された必須属性には無効な文字が含まれています。
MANDATORY_ATTRIB_MISSING	7	必須属性が不足しています。
OPTIONAL_ATTRIB_EXCEEDS_SIZE	10	指定された任意属性は許可される最大サイズを超えています。
OPTIONAL_ATTRIB_INVALID_CHAR	11	指定された任意属性には無効な文字が含まれています。
SERVER_INTERNAL_ERROR	3	内部サーバ エラーのためにエラーが発生しました。
TCP_COMMUNICATION_ERROR	15	TCP ベースの通信エラーが発生しました。
UNKNOWN_ERROR	13	不明なエラーが発生しました。
USER_ALREADY_EXIST	2	指定されたユーザは、すでに RiskFort データベースに存在します。
USER_NOT_FOUND	4	指定されたユーザは、RiskFort データベースにありません。
USERNAME_MISSING	6	User オブジェクトのユーザ名がありません。

エラー コード

表 D-3 には、応答コード、理由コード、考えられる失敗の原因、および解決方法（該当する場合）をリスト表示します。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
0	0	操作は成功しました。	なし
1000	2002	内部エラーが発生しました。	**考えられる原因**： 予期しない内部エラーです。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
1050	0	この操作に使用されたパラメータのうちの 1 つの値が無効です。	**考えられる原因**： API に渡されたパラメータの値が無効です。たとえば、ユーザ ステータスで許可された値が 0 と 1 で、この値に 5 を設定した場合、このエラーが表示されます。 **解決方法**： パラメータに有効な値を指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
	2050	この操作に使用されたパラメータのうちの 1 つの値が空です。	**考えられる原因**： API に渡されたパラメータが空です。 **解決方法**： パラメータに空でない値を指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
1050	2051	この操作に使用されたパラメータのうちの 1 つの長さが許可された最大値を超えました。 **注：** この長さは、パラメータの長さ（たとえばパスワードの長さ）を参照します。	**考えられる原因**： API に渡されたパラメータの長さが上限値を超えています。 **解決方法**： 長さが許可された最大値以下になるパラメータを指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
	2052	操作で使用されたパラメータのうちの 1 つの長さが許可された最小値未満です。	**考えられる原因**： API に渡されたパラメータの長さが最小値未満です。 **解決方法**： 許可された最小値以上の長さの値をパラメータに指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
	2053	この操作に使用されたパラメータのうちの 1 つの値が許可された最大値を超えました。 **注：** この値は、パラメータの値を参照します。	**考えられる原因**： API に渡されたパラメータの値が許可された最大値を超えています。 **解決方法**： パラメータの値が許可された最大値以下になるような値をパラメータに指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
1050	2054	操作で使用されたパラメータのうちの 1 つの値が許可された最小値未満です。	**考えられる原因**： API に渡されたパラメータの値が許可された最小値未満です。 **解決方法**： 許可された最小値以上の値をパラメータに指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
	2055	この操作に使用されたパラメータのうちの 1 つの値が無効です。	**考えられる原因**： API に渡されたパラメータの値が無効です。たとえば、ユーザ ステータスで許可された値が 0 と 1 で、この値に 5 を設定した場合、このエラーが表示されます。 **解決方法**： パラメータに有効な値を指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
	2056	この操作に使用されたパラメータのうちの 1 つの値に無効な文字が含まれています。	**考えられる原因**： ParameterKey によって指定されたパラメータに無効な文字が含まれています。 **解決方法**： ParameterKey によって指定されたパラメータに有効な文字を指定します。
	2057	操作に使用されたパラメータのうちの 1 つは形式要件を満たしていません。	**考えられる原因**： ParameterKey によって指定されたパラメータには無効な形式があります。 **解決方法**： ParameterKey によって指定されたパラメータに有効な形式を指定します。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
1050	2061	この操作に使用されたパラメータのうちの 1 つの値が許可されていません。	**考えられる原因**： ParameterKey によって指定されたパラメータには無効な形式があります。 **解決方法**： ParameterKey によって指定されたパラメータに有効な形式を指定します。
	8104	指定されたコールアウト URL は有効ではありません。	**考えられる原因**： 指定された URL が正しくありません。 **解決方法**： 有効な URL を指定します。
	8105	指定された期間が有効ではありません。	**考えられる原因**： ［Start Date］が［End Date］よりも大きい値です。 ［Start Date］または［End Date］に指定された値は過去の日付です。 **解決方法**： ［Start Date］は［End Date］より小さくする必要があります。また、これらの日付は（例外ユーザのための期間であるため）、現在または未来の日付である必要があります。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7501	0	データベースでの現在の操作に失敗しました。	**考えられる原因**： データベースが実行されていません。 **解決方法**： データベースを起動します。 **考えられる原因**： サーバとデータベースの間の接続が完了していません。 **解決方法**： サーバとデータベースの間の接続を再確立します。 **考えられる原因**： 内部エラーが発生したため、操作が失敗しました。 **解決方法**： 詳細についてはデータベース ログを確認し、それらのログに基づいて適切なアクションが実行されていることを確認します。
7502		予期しない内部エラーのために例外が発生しました。	**考えられる原因**： 予期しないサーバ動作のために内部エラーが発生しました。 **解決方法**： 最も可能性の高い原因はサーバまたはデータベースの障害です。詳細についてはサーバ トランザクションおよびデータベース ログを確認し、サーバ ログに基づいて適切なアクションが実行されていることを確認します。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7503	0	時間を取得できませんでした。	**考えられる原因**： データベース設定が arcotcommon.ini に正しく設定されていません。 **解決方法**： ファイルのデータベースに関連するパラメータを確認して修正します。
7601		この操作に使用されたパラメータのうちの 1 つの値が存在しません。	**考えられる原因**： API に渡されたパラメータの値が存在しません。 **解決方法**：パラメータに有効な値を指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7602		新しいルール セットに指定された名前はすでに存在します。	**考えられる原因**： API に渡されたパラメータの値はすでに存在します。 **解決方法**： パラメータに有効な値を指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7603		指定された名前の Active ルール セットは見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7604		指定された名前の Proposed ルール セットは見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7605	0	ルール セットに指定された名前は存在しません。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。 サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7606	8101	タイプが共通の データが別のルール セットを参照しています。そのため、追加の操作は許可されません。	**考えられる原因**： ルールまたはルール セットが別のルール セットを参照しています。そのため、データを別のルール セットに追加できません。
7607		タイプが共通の データが別のルール セットを参照しています。そのため、値を設定する操作は許可されません。	**考えられる原因**： ルールまたはルール セットが別のルール セットを参照しています。そのため、データを別のルール セットに設定できません。
7608		タイプが共通の データが別のルール セットを参照しています。そのため、削除の操作は許可されません。	**考えられる原因**： ルールまたはルール セットが別のルール セットを参照しています。そのため、データを別のルール セットから削除できません。
7609		タイプが共通の データが別のルール セットを参照しています。そのため、更新の操作は許可されません。	**考えられる原因**： ルールまたはルール セットが別のルール セットを参照しています。そのため、データを更新できません。
7610		タイプが共通の データが別のルール セットを参照しています。そのため、値のローテーションは許可されません。	**考えられる原因**： ルールまたはルール セットが別のルール セットを参照しています。そのため、データ値をローテーションすることはできません。
7611		タイプが共通のパラメータが別のルール セットを参照しています。そのため、値を設定する操作は許可されません。	**考えられる原因**： ルールまたはルール セットが別のルール セットを参照しています。そのため、データを別のルール セットに設定できません。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7612	0	指定された拒否国について Active または Proposed のデータが見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が ARRFNEGATIVECOUNTRYLIST テーブルに見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7613 7617		指定された拒否 IP アドレスについて Active または Proposed のデータが見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が ARRFUNTRUSTEDIPLIST テーブルに見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7614 7616		指定のトラステッド IP アドレスについて Active または Proposed のデータが見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が ARRFTRUSTEDIPLIST テーブルに見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7615		指定されたルールについて Active または Proposed のデータが見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7618	0	指定されたスコアリング設定について Active または Proposed のデータが見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7619		指定された Execution 設定について Active または Proposed のデータが見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7620		操作に使用された設定状態パラメータの値は無効です。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7621		指定されたパラメータおよびルールに対するそれらの値が一致しないため、ルール セットは作成できません。	**考えられる原因**： API に渡されたパラメータおよび対応する値が無効です。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7622 7625	0	指定された otherOrgName パラメータおよび otherConfigName パラメータについて Active ルール セットが見つかりません。	**考えられる原因**： API に渡された otherOrgName パラメータと otherConfigName パラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7623 7626		otherOrgName パラメータおよび otherConfigName パラメータの指定されたデータについて Active ルール セットが見つかりません。	**考えられる原因**： API に渡された otherOrgName パラメータと otherConfigName パラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7624		循環依存が作成されるため、更新は許可されません。	**考えられる原因**： ルールまたはルール セットが別のルール セットを参照しています。
7627	8102	対応する Proposed データが見つからなかったため、指定された国を削除できません。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7628	8102	対応する Proposed データが見つからなかったため、指定された IP 範囲を削除できません。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7629		対応する Proposed データが見つからなかったため、トラステッド アグリゲータの指定 IP 範囲を削除できません。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7630		対応する Proposed データが見つからなかったため、トラステッド アグリゲータの指定 IP 範囲を削除できません。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7631	0	指定された IP 範囲は、拒否 IP アドレス リストに存在しません。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7632	0	指定のトラステッド IP アドレスで、トラステッド アグリゲータの Active または Proposed データが見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が ARRFTRUSTEDIPLIST テーブルに見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7633	0	指定のトラステッド アグリゲータの Proposed データが見つかりませんでした。	**考えられる原因**： API に渡されたパラメータの値が ARRFTRUSTEDIPLIST テーブルに見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7634	0	指定のトラステッド アグリゲータは、トラステッド IP アドレス リストに存在しません。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7635	0	指定のトラステッド アグリゲータはトラステッド IP アドレス リストにすでに存在するため、追加できません。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7636	0	指定されたアグリゲータ ID は、内部サーバ エラーのために生成できませんでした。	**考えられる原因**： アグリゲータ ID の生成がサーバの内部エラーのために失敗しました。 **解決方法**： 最も可能性の高い原因はデータベースの障害です。詳細についてはサーバ トランザクション ログを確認し、サーバ ログに基づいて適切なアクションが実行されていることを確認します。
7637	0	暗号化キーがデータベースに見つかりませんでした。	**考えられる原因**： 侵入キーが存在しません。 **解決方法**： 使用しているキーが正しいことを確認します。
7638	0	サポートされたポート タイプをトークン化することはできませんでした。	考えられる原因： サーバ ホストまたはポート、あるいはその両方が正しく設定されていない可能性があります。 **解決方法**： 正しいホストおよびポート番号を指定します。 **考えられる原因**： サーバが実行されていない可能性があります。 **解決方法**： サーバを起動します。 **考えられる原因**： SSL が設定されている場合、証明書が正しく設定されていない可能性があります。 **解決方法**： TLS 証明書を正しく設定します。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7639	0	指定された SSL Trust Store は存在しません。	**考えられる原因**： API に渡されたパラメータの値が見つかりませんでした。 **解決方法**： パラメータに有効な値を指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。 **考えられる原因**： 提供されたトラスト ストア名は有効ではありません。 **解決方法**： 有効なトラスト ストア名を指定する必要があります。 **考えられる原因**： 指定された組織名は有効ではありません。 **解決方法**： 有効な組織名を指定する必要があります。
7640	0	指定されたスコア範囲は見つかりませんでした。	**考えられる原因**： スコア範囲またはアドバイスが ARRFADVICECONFIG および ARRFADVICECODE のテーブル内に見つかりませんでした。RiskFort データベース スクリプトは正しく実行されませんでした。 **解決方法**： この問題を解決するには、Arcot プロフェッショナル サービス（ps@arcot.com）にお問い合わせください。
7641	0	スコアリング コールアウトに設定されている実行優先度が最高ではありません。	**考えられる原因**： Web サービス呼び出しに設定されている優先度は、RiskFort データベースでは既存の最高実行優先度よりも高くなっています。 **解決方法**： Web サービス呼び出しで優先度を正しく設定します。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7642	0	スコアに設定されている実行優先度が最高ではありません。	**考えられる原因**： Web サービス呼び出しに設定されている優先度は、RiskFort データベースでは既存の最高実行優先度よりも高くなっています。 **解決方法**： Web サービス呼び出しで優先度を正しく設定します。
7643	8103	必要なアドオン ルールの詳細が指定されていませんでした。そのため、ルールは追加されませんでした。	**考えられる原因**： アドオン ルールの詳細が Web サービス呼び出しで正しく指定されていませんでした。 **解決方法**： Web サービス呼び出しの入力パラメータを正しく設定します。
7644	0	指定されたルール タイプは有効ではありません。	**考えられる原因**： 指定されたルール タイプは ARRFADDONRULETYPE テーブルにないか、または無効なルール タイプのいずれかです。 **解決方法**： 有効なルール タイプを指定する必要があります。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7645	0	データの循環依存が作成されるため、アドオン ルールは追加されませんでした。	**考えられる原因**： 別のルールを参照するアドオン ルールを作成すると、追加したデータに循環依存が作成されます。したがって、この操作は許可されていません。
7646		パラメータの循環依存が作成されるため、アドオン ルールは追加されませんでした。	**考えられる原因**： 別のルールを参照するアドオン ルールを作成すると、追加したパラメータに循環依存が作成されます。したがって、この操作は許可されていません。
7647		指定したルールがルール セットで使用できません。	**考えられる原因**： API に渡されたルールの値は指定されたルールセットに見つかりませんでした。 **解決方法**： パラメータに有効なルールを指定します。サポートされているパラメータ値については、付録 E の 「入力データの検証」を参照してください。
7648		パラメータを取得する際にエラーが発生しました。	
7649		他のルールがこのルールに依存しています。そのため、このルールは削除できません。	**考えられる原因**： 別のルール セット内のルールが現在このルールを参照しています。そのため、このルールは削除できません。
7650		TypeName 入力は呼び出しで指定されていません。	**考えられる原因**： TypeName は Web サービス呼び出しで指定されていません。 **解決方法**： 入力データの TypeName に正しい入力値を指定します。
7651		指定された TypeName はすでに存在しています。	**考えられる原因**： 入力データの TypeName は、システムにすでにあります。 **解決方法**： 入力データに別の TypeName を指定します。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7652	0	この更新で循環冗長が作成されます。そのため、更新は許可されません。いくつかの（表示される）ルール セットが移行されました。	**考えられる原因**： *すべての必要なルール セットの移行の後に循環依存が作成されます。その結果、少数のルール セットのみが移行されました。 **解決方法**： まだ移行されていないルール セットを変更します。その結果、循環依存が除去されます。
7653		この更新で循環冗長が作成されます。このため、この更新は許可されていません。移行されたルール セットはありません。	**考えられる原因**： *すべての必要なルール セットの移行の後に循環依存が作成されます。 **解決方法**： 必要なルール セットを変更して循環依存が除去されるようにします。
7654		運用環境に移行する間にデータベース操作に失敗しました。いくつかの（表示される）ルール セットが移行されました。	**考えられる原因**： ルール セットの移行中に循環依存が見つかりました。そのため、少数のルール セットのみが移行されました。失敗したルール セットを含めて、移行されていないルール セットがあります。 **解決方法**： 詳細については RiskFort ログを確認し、それらのログに基づいて適切なアクションが実行されていることを確認します。 失敗したルール セットの循環依存を除去し、それを移行します。 また、失敗したルール セットが原因で移行されなかった残りのルール セットを移行します。
7655		値を大文字に変換できませんでした。	**考えられる原因**： 指定された入力データは大文字に変換できません。 **解決方法**： 入力データを確認します。

表 D-3. 応答コードと理由コード

応答コード	理由コード	説明	考えられる失敗の原因
7656	0	ユーザ プロファイル情報を取得できませんでした。	**考えられる原因**： UDS が起動および稼働していません。 **解決方法**： UDS が正しく展開されていて、起動および稼働していることを確認します。これを行う方法の詳細については、「Arcot RiskFort2.2.6 インストールおよび展開ガイド」を参照してください。 **考えられる原因**： ユーザが RiskFort システムに存在しません。 **解決方法**： ユーザを RiskFort システムに追加します。
7657		場所および接続情報がデータベース内に見つかりませんでした。	**考えられる原因**： Quova データは RiskFort データベースにアップロードされていません。 **解決方法**： `arrfupload.exe` ツールを使用して Quova データを RiskFort データベースにアップロードします。この詳細については、「*Arcot RiskFort2.2.6 管理ガイド*」を参照してください。
7658		例外ユーザがシステムに存在しません。	**考えられる原因**： 指定されたユーザは例外ユーザではありません。 **解決方法**： ユーザの正しい詳細が指定されていることを確認するか、または続行する前にユーザを例外ユーザ リストに追加します。

付録 E
入力データの検証

システムが無効なデータを処理しないことを保証し、ビジネス ルールを実行し、ユーザ入力が内部構造およびスキーマと互換性をもつことを保証するために、RiskFort サーバは API から受け取るデータを検証します。表 E-1 では、RiskFort サーバが入力データを検証する場合に使用する条件について説明します。

注： 以下の表で言及する属性の長さとは、文字数に相当します。属性 ID は、Java API では `paramName` として参照されます。

表 E-1. 属性の妥当性チェック

属性	属性 ID	妥当性チェック基準
ユーザ名	USER_NAME	ユーザ名は空ではない。
		長さは 1 ～ 256 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
組織名	ORG_NAME	組織名は空ではない。
		長さは 1 ～ 64 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
組織の表示名	DISPLAY_ORG_NAME	組織の表示名は空ではない。
		長さは 1 ～ 1024 文字である。
		無効な文字を含まない。
開始時刻	START_TIME	空ではない。
終了時刻	END_TIME	空ではない。
期間	DURATION	過去の日付は指定できない。
作成時間	CREATE_TIME	空ではない。
		現在の時刻以前である。
最終更新時刻	LAST_MODIFIED_TIME	空ではない。
		現在の時刻以前である。

表 E-1. 属性の妥当性チェック

属性	属性 ID	妥当性チェック基準
設定名	CONFIG_NAME	設定名は空ではない。
		長さは 1 ～ 64 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
チャネル名	CHANNEL_NAME	チャネル名は空ではない。
		長さは 1 ～ 64 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
設定状態	CONFIG_STATE	長さは 1 ～ 5 文字である。
AdminWS の設定状態	CONFIG_STATE_WS	長さは 2 文字以下。0 文字も使用可能。
国名	COUNTRY_NAME	国名は空ではない。
		長さは 0 ～ 50 文字である。
		無効な文字（ASCII 0 ～ 31）を含まない。
国コード	COUNTRY_CODE	国コードは空ではない。
		長さは 1 ～ 2 文字である。
		数字、アルファベット、アンダースコア、およびドットを使用できる。
開始 IP	START_IP	開始 IP アドレスは空ではない。
		長さは 0 ～ 4294967295 文字である。
		IP アドレスのフォーマットに従う。
終了 IP	END_IP	終了 IP アドレスは空ではない。
		長さは 0 ～ 4294967295 文字である。
		IP アドレスのフォーマットに従う。
マスク	MASK	マスクは空ではない。
		長さは 0 ～ 4294967295 文字である。
		IP アドレスのフォーマットに従う。
開始 IP	START_IP_STR	開始 IP アドレスは空ではない。
		長さは 7 ～ 15 文字である。
		IP アドレスのフォーマットに従う。

表 E-1. 属性の妥当性チェック

属性	属性 ID	妥当性チェック基準
終了 IP	END_IP_STR	終了 IP アドレスは空ではない。
		長さは 7 ～ 15 文字である。
		IP アドレスのフォーマットに従う。
マスク	MASK_STR	マスクは空ではない。
		長さは 7 ～ 15 文字である。
		IP アドレスのフォーマットに従う。
信頼できない IP タイプ	UNTRUSTED_IP_TYPE	長さは 1 ～ 65535 文字である。
開始 IP フィルタ	START_IP_FILTER	開始 IP フィルタは空ではない。
		長さは 7 ～ 15 文字である。
		IP アドレスのフォーマットに従う。
ソース IP フィルタ	SOURCE_IP_FILTER	ソース IP フィルタは空ではない。
		長さは 7 ～ 15 文字である。
		IP アドレスのフォーマットに従う。
ルール名	RULE_NAME	ルール名は空ではない。
		長さは 1 ～ 128 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
ルールの短縮名	RULE_MNEMONIC	ルールの短縮名は空ではない。
		長さは 1 ～ 128 文字である。
		無効な文字を含まない。ただし、数字、アルファベット、アンダースコア（_）、およびハイフン（-）は使用可能。
ルールの記述名	RULE_DESCR_NAME	ルールの記述名は空ではない。
		長さは 0 ～ 256 文字である。
		無効な文字を含まない。ただし、ASCII 0 ～ 31 は使用可能。
ルール タイプ	RULE_TYPE	ルール タイプは空ではない。
		長さは 1 ～ 128 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。

表 E-1. 属性の妥当性チェック

属性	属性 ID	妥当性チェック基準
ルールのライブラリ名	LIB_NAME	ルールのライブラリ名は空ではない。
		長さは 1 ～ 128 文字である。
		無効な文字を含まない。ただし、数字、アルファベット、アンダースコア（_）、およびハイフン（-）は使用可能。
パラメータ名	RULE_PARAM_NAME	パラメータ名は空ではない。
		長さは 1 ～ 64 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
パラメータ値	RULE_PARAM_VALUE_STR	パラメータ値は空ではない。
		長さは 1 ～ 512 文字である。
		無効な文字（ASCII 0 ～ 31）を含まない。
パラメータ値	RULE_PARAM_VALUE_BIN	パラメータ値は空ではない。
パラメータ タイプ	RULE_PARAM_TYPE	長さは 1 ～ 4 文字である。
説明	DESCRIPTION	説明は空ではない。
		長さは 1 ～ 256 文字である。
		無効な文字（ASCII 0 ～ 31）を含まない。
アグリゲータ名	AGGREGATOR_NAME	長さは 1 ～ 64 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
アグリゲータ ID	AGGREGATOR_ID	--
スコア	SCORE	値は 1 ～ 100 である。
スコアリング優先度	SCORING_PRIORITY	値は 1 ～ 2147483647 である。
実行優先度	EXEC_PRIORITY	値は 0 ～ 100000 である。
実行の有効化	EXECUTIONENABLED	値は 0 または 1 である必要がある。
スコアリングの有効化	SCORINGENABLED	値は 0 または 1 である必要がある。
他の組織名	OTHERORGNAME	長さは 1 ～ 64 文字である。
		無効な文字を含まない。
他の設定名	OTHERCONFIGNAME	長さは 1 ～ 64 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。

表 E-1. 属性の妥当性チェック

属性	属性 ID	妥当性チェック基準
共有タイプ	SHARINGTYPE	値は 0 ～ 3 である。
コールアウト タイプ	CALLOUT_TYPE	値は 0 ～ 2 である。
コールアウト URL	CALLOUT_URL	URL は空ではない。
		長さは 0 ～ 150 文字である。
		無効な文字を含まない。ただし、アルファベット、数字、および + / \ \ # $ % & - _ : . は使用可能。
コールアウトのタイムアウト	CALLOUT_TIMEOUT	値は 0 ～ 1000000 である。
インスタンス名	INSTANCE_NAME	長さは 0 ～ 32 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
プロトコル モジュール名	PROTOCOL_MODULE_NAME	長さは 0 ～ 128 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
クライアント SSL トラスト ストア名	CLIENT_SSL_TRUST_STORE_NAME	長さは 0 ～ 64 文字である。
		無効な文字を含まない。ただし、ASCII 32 ～ 127 は使用可能。
接続タイムアウト	CONNECTION_TIMEOUT	値は 0 ～ 1000000 である。
クライアント証明書	CLIENT_CERT	--
クライアント キー	CLIENT_KEY	--
サーバ ルート CA 証明書	SERVER_ROOT_CA_CERT	--
サーバの秘密キー	SERVER_PRIVATE_KEY	--
読み取りタイムアウト	READ_TIMEOUT	--
最小接続数	MIN_CONNECTION	--
最大接続数	MAX_CONNECTION	--
証明書の完全識別名	CERT_SUBJECT	--
発行者名	ISSUER_NAME	--
サーバ SSL 認証	SERVER_AUTH_SSL	--
クライアント SSL 認証	CLIENT_AUTH_SSL	--

表 E-1. 属性の妥当性チェック

属性	属性 ID	妥当性チェック基準
アクション	TRANS_ACTION	アクションは空ではない。
		長さは 1 ～ 32 文字である。
		無効な文字（ASCII 0 ～ 31）を含まない。
関連名	ASSOC_NAME	関連名は空ではない。
		長さは 1 ～ 32 文字である。
		無効な文字（ASCII 0 ～ 31）を含まない。

付録 F
RiskFort ロギング

RiskFort サーバとアプリケーションの間の通信を効率的に管理するには、サーバおよび他のコンポーネントのアクティビティとパフォーマンスに加えて発生した可能性のある問題に関する情報を取得することが必要です。

この付録では、RiskFort によってサポートされている各種ログ ファイル、これらのファイルに表示される重大度レベルおよびこれらのログ ファイルの形式について説明します。この章には、以下のトピックがあります。

ログ ファイルについて

RiskFort のログ ファイルは次のように分類できます。

- インストール ログ ファイルログ ファイル 4<1
- トランザクション ログ ファイル
- UDS ログ ファイル
- Administration Console のログ ファイル

ロギング関連のパラメータ（[arcot/riskfort/logger] の下に指定）はすべて、riskfortserver.ini ファイルによって制御されます。これらのロギング設定オプションには次のものが含まれます。

- [**Specifying log file name and path**]：RiskFort によってログ ファイルを書き込み、バックアップ ログ ファイルを格納するためのディレクトリを指定できます。診断ログ記録ディレクトリを指定すると、管理者はシステムとネットワーク リソースを管理することができます。

- ［**Log file size**］: ログ ファイルに含めることができる最大バイト数。ログ ファイルがこのサイズに到達すると、指定された名前を持つ新規ファイルが作成され、古いファイルはバックアップ ディレクトリに移動されます。
- ［**Using log file archiving**］: RiskFort コンポーネントが診断メッセージを実行し生成すると共に、ログ ファイルのサイズは増加します。ログ ファイルのサイズが増加し続けることをユーザが許容する場合、管理者はログ ファイルを手動で監視し駆除する必要があります。RiskFort では、収集されて保存されるログ ファイル データの量を制限する設定オプションを指定できます。RiskFort では、診断ログ記録ファイルのサイズを制御する、設定オプションを指定することができます。これにより、ログ ファイルの最大サイズを決定することができます。最大サイズに到達すると、新しいログ情報が保存される前に、古いログ情報がバックアップ ファイルに移動されます。
- ［**Setting logging levels**］: RiskFort では、ロギング レベルを設定することもできます。ロギング レベルを設定することによって、診断ログ ファイルに保存されるメッセージの数を縮小できます。たとえば、システムがクリティカルなメッセージだけを記録し保存するように、ロギング レベルを設定することができます。サポートされているログ レベルの詳細については、F-163 ページの「サポートされている重大度レベル」を参照してください。
- ［**Specifying time zone information**］: RiskFort では、ログ記録された情報のタイムスタンプにローカル タイムゾーンを使用するか、または GMT を使用することができます。

インストール ログ ファイルログ ファイル 4<1

RiskFort をインストールする際に、インストーラは `Arcot_RiskFort_InstallLog.log` ファイルで実行されるアクションをすべて記録します。このファイルの情報は、RiskFort インストールが正常に完了しなかった場合、問題のソースを識別するのに非常に役立ちます。

このファイルのデフォルトの場所は以下のとおりです。

Windows の場合

`<install_location>\Arcot Systems\logs\`

UNIX ベースの場合

`<install_location>/arcot/logs/`

トランザクション ログ ファイル

トランザクション ログは以下で構成されます。

- RiskFort サーバ ログ
- ケース管理サーバのログ ファイル

RiskFort サーバ ログ

RiskFort では、arcotriskfort.log ファイルにサーバによって処理されたすべてのリクエストおよび関連するアクションを記録します。このファイルのデフォルトの場所は以下のとおりです。

Windows の場合

<*install_location*>\Arcot Systems**logs**\

UNIX ベースの場合

<*install_location*>/arcot/**logs**/

注： RiskFort ロガーを使用してアプリケーションのログを設定することはできません。これらのログには、アプリケーションをホストしているサードパーティ アプリケーション サーバ（Apache Tomcat や IBM Websphere など）が使用するツールを使用することにより、アクセスできます。

ロギング関連のパラメータ（[arcot/riskfort/logger] 下に指定）はすべて、riskfortserver.ini ファイルによって制御されます。これは ARCOT_HOME の conf フォルダにあります。

ケース管理サーバのログ ファイル

ケース管理サーバ モジュールを展開して開始した場合、そのすべてのアクションおよび処理されたリクエストの詳細は arcotriskfortcasemgmtserver.log ファイルに記録されます。このファイルのデフォルトの場所は以下のとおりです。

Windows の場合

<*install_location*>\Arcot Systems**logs**\

UNIX ベースの場合

<install_location>/arcot/**logs**/

注： RiskFort ロガーを使用してアプリケーションのログを設定することはできません。これらのログには、アプリケーションをホストしているサードパーティ アプリケーション サーバ（Apache Tomcat や IBM Websphere など）が使用するツールを使用することにより、アクセスできます。

ログ記録関連のパラメータ（[arcot/riskfortcasemgmtserver/logger] 下に指定）はすべて、riskfortcasemgmtserver.ini ファイルによって制御されます。これは ARCOT_HOME の conf フォルダにあります。

UDS ログ ファイル

ユーザ データ サービス（UDS）情報およびアクションはすべて arcotuds.log ファイル内に記録されます。この情報には、以下のものが含まれます。

- UDS データベースの接続情報
- UDS データベースの設定情報
- UDS インスタンス情報、およびこのインスタンスによって実行されたアクション

このファイル内の情報は、Administration Console が UDS インスタンスに接続できなかった場合、その問題のソースを識別するのに非常に役立ちます。このファイルのデフォルトの場所は以下のとおりです。

Windows の場合

<install_location>\Arcot Systems**logs**\

UNIX ベースの場合

<install_location>/arcot/**logs**/

このファイルのロギングを制御するパラメータは udsserver.ini ファイルを使用することによって設定できます。これは ARCOT_HOME の conf フォルダにあります。

ロギング レベル、ログ ファイル名およびパス、ファイルの最大サイズ（バイト）およびアーカイブする情報に加えて、log4j.appender.debuglog.layout.ConversionPattern に対して適切な値を指定することにより、UDS のロギング パターンのレイアウトを制御することもできます。

このファイルに使用されているデフォルトの形式の詳細については、F-163 ページの「UDS および Administration Console のログ ファイルの形式」を参照してください。

Administration Console のログ ファイル

Administration Console を展開して開始した場合、そのすべてのアクションおよび処理されたリクエストの詳細は arcotadmin.log ファイルに記録されます。この情報には、以下のものが含まれます。

- データベースの接続情報
- データベースの設定情報
- インスタンス情報、およびこのインスタンスによって実行されたアクション
- UDS 設定情報
- キャッシュ リフレッシュなど、Master Administrator によって指定されたその他の Administration Console の情報

このファイル内の情報は、Administration Console が起動しなかった場合、その問題のソースを識別するのに非常に役立ちます。このファイルのデフォルトの場所は以下のとおりです。

Windows の場合

<*install_location*>\Arcot Systems**logs**\

UNIX ベースの場合

<*install_location*>/arcot/**logs**/

このファイルのロギングを制御するパラメータは adminserver.ini ファイルを使用することによって設定できます。これは ARCOT_HOME の conf フォルダにあります。

ロギング レベル、ログ ファイル名およびパス、ログ ファイルの最大サイズ（バイト）およびログ ファイルのアーカイブ情報に加えて、log4j.appender.debuglog.layout.ConversionPattern に対して適切な値を指定することにより、コンソールのロギング パターンのレイアウトを制御することもできます。

このファイルに使用されているデフォルトの形式の詳細については、F-163 ページの「UDS および Administration Console のログ ファイルの形式」を参照してください。

RiskFort サーバおよびケース管理サーバのログ ファイルの形式

F-159 ページの「RiskFort サーバ ログ」で詳述しているように、表 F-1 は、RiskFort ロガー（arcotriskfort.log）のエントリの形式について説明します。

表 F-1. RiskFort ログ形式

列	説明
タイム スタンプ	エントリがログに記録された時間を、指定されたタイム ゾーンに変換した時間。 この情報のログの形式を以下に示します。 `www mmm dd HH:MM:SS.mis yy timezone` 前の形式で、 • `www` は曜日を表します。 • `mis` はミリ秒を表します。 • `timezone` は、`riskfortserver.ini` ファイルで指定したタイムゾーンを表します。
Log Level（または Severity）	ログに記録されたエントリの重大度レベル。 詳細については、「サポートされている重大度レベル」を参照してください。
Process ID（pid）	エントリをログに記録したプロセスの ID。
Thread ID （tid）	エントリをログに記録したスレッドの ID。
Transaction ID	エントリをログに記録したトランザクションの ID。
Module ID（非推奨）	エントリをログに記録した RiskFort モジュール（ルール エンジンやスコアリング エンジンなど）の ID。 注： これは非推奨のエントリで、`0` を常に表示します。
Message	フリーフロー形式でサーバによってログに記録されたメッセージ。 注： このメッセージの精度は、`riskfortserver.ini` に設定した Log Level によって異なります。

UDS および Administration Console のログ ファイルの形式

表 F-2 では、以下のロガーのエントリの形式について説明します。

- `arcotuds.log` （UDS ログ ファイル）
- `arcotadmin.log` （Administration Console のログ ファイル）

表 F-2. UDS および Administration Console のログ形式

列	関連付けられたパターン（ログ ファイルで）	説明
タイム スタンプ	`%d{yyyy-MM-dd hh:mm:ss,SSS z} :`	エントリがログに記録された時間。このエントリはアプリケーション サーバのタイム ゾーンを使用します。この情報のログの形式を以下に示します。 `yyyy-MM-dd hh:mm:ss,SSS z` ここで、 • `SSS` はミリ秒を表します。 • `z` はタイム ゾーンを表します。
Thread ID	`[%t] :`	エントリをログに記録したスレッドの ID。
Log Level （または Severity）	`%-5p:`	ログに記録されたエントリの重大度レベル。 詳細については、「サポートされている重大度レベル」を参照してください。
Logger Class	`%-5c{3}(%L) :`	ログ リクエストをしたロガーの名前。
Message	`%m%n:`	サーバによってログ ファイルにフリーフロー形式でログ記録されたメッセージ。 **注：** メッセージの精度は、ログ ファイルに設定した Log Level によって異なります。

UDS および Administration Console のログ ファイル内の `PatternLayout` パラメータをカスタマイズする方法については、以下の URL を参照してください。

http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html

サポートされている重大度レベル

ログ レベル（または重大度レベル）では、RiskFort ログに格納されている情報の詳細のレベルを指定できます。また、これを使用してログ ファイルの大きさを制御することもできます。

表 F-3 では、重大度の高いものから順に、すべてのログ ファイルに表示されるログ レベルについて説明します。

表 F-3. RiskFort ログ レベル（重大度の高いものから順に）

ログ レベル		説明
0	FATAL	RiskFort サービスの突然の終了を引き起こす可能性がある、深刻で修復できないエラーに対してこのログ レベルを使用します。
1	WARNING	まだ FATAL でない不適当なランタイム例外、潜在的に有害な状況および回復可能な問題に対してこのログ レベルを使用します。
2	INFO	ランタイム イベントについての情報をキャプチャする場合にこのログ レベルを使用します。 すなわち、この情報ではアプリケーションの進捗状況を強調します。これには、以下のような変化が含まれる場合があります。 • 開始、停止、再起動のようなサーバの状態。 • サーバのプロパティ。 • サービスの状態。 • サーバ上のプロセスの状態。
3	LOW DETAIL	デバッグ目的で詳細情報をログする場合にこのログ レベルを使用します。これにはプロセス追跡およびサーバ状態の変化が含まれる場合があります。

注： ログ レベルを指定すると、そのレベルよりも*高い*重要性を持つ他のすべてのメッセージも同様にレポートされます。たとえば、LogLevel が 3 に指定されている場合、FATAL、WARNING および INFO レベルのログ レベルを備えたメッセージもキャプチャされます。

各ログ レベルのサンプル エントリ

以下のサブセクションではいくつかの **RiskFort ログ ファイル**のサンプル エントリ（ログレベルに基づいた）を示します。

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
```

付録 G
RiskFort サンプル アプリケーション

この RiskFort のバージョンには、サンプル アプリケーションが同梱されています。これは、RiskFort Java API の使用方法およびアプリケーションを RiskFort に統合する方法を実例で示す簡易 Java プリミティブ（コード）の「テンプレート」として機能します。このように、サンプル アプリケーションは、RiskFort とアプリケーションの間の統合プロセスを標準化する役割を果たします。

この付録にあるセクションは以下のとおりです。

サンプル アプリケーションについて

使用している環境で RiskFort を展開するには、オンライン アプリケーションにその API を統合する必要があります。サンプル アプリケーションは、RiskFort Java API を使用して RiskFort の最も一般的な機能を例示します。この詳細については以下のセクションで説明します。

- G-174 ページの「リスク評価の実行」
- G-184 ページの「ユーザの作成」

重要 : サンプル アプリケーションは運用環境では使用 *できません*。サンプル アプリケーションをコード参照としてのみ使用することにより、独自の Web アプリケーションを構築することを強く推奨します。運用環境では、サンプル アプリケーションは RiskFort が正常にインストールされたかどうか、およびリスク評価操作を実行できるかどうかを確認するためにのみ使用できます。

サンプル アプリケーションのコンポーネント

サンプル アプリケーションは以下で構成されています。

- **サーブレット**：Web サーバに新しい機能を追加するために使用できるプラットフォームに依存しないサーバ側モジュール （サーブレットはアプレットと同じですが、アプレットと異なり、サーブレットにはユーザ インターフェースがありません。JSP によって提供されるユーザ インターフェースを使用することによって既存のビジネス ロジックにアクセスできます)。
- **ヘルパー**：そのヘルパーを利用するクラスの一部でない追加の機能を提供するクラス。その結果、ヘルパー クラスはコードの維持および再利用を可能にします。

  たとえば、クラスが数の値を表示する必要がある場合、それを何の表示形式も付けずにそのまま表示することも、（カンマ区切り形式などの）表示形式を付けてきれいに表示することもできます。フォーマットを代わりに実行するために、クラスは別のクラスを利用できます。それはヘルパー クラスと呼ばれます。
- **JSP**：サーバおよびプラットフォームに依存しない動的 Web コンテンツを作成して使用できるようにする Java Server Pages （JSP)。

これらのサンプル アプリケーション コンポーネントはモデル ビュー コントローラ（MVC）フレームワークに編成されています。これにより、アプリケーションのユーザ インターフェースは下層のビジネス ロジックから分離されています。その結果、アプリケーションのユーザ インターフェースとビジネス ルールのどちらか、または両方を、もう一方に影響することなく容易に変更できます。

図 G-1 では、サンプル アプリケーションのコンポーネント間のやり取り（ユーザがアプリケーションによって正常に認証された後）を示します。

図 G-1 サンプル アプリケーションのコンポーネント間のやり取り

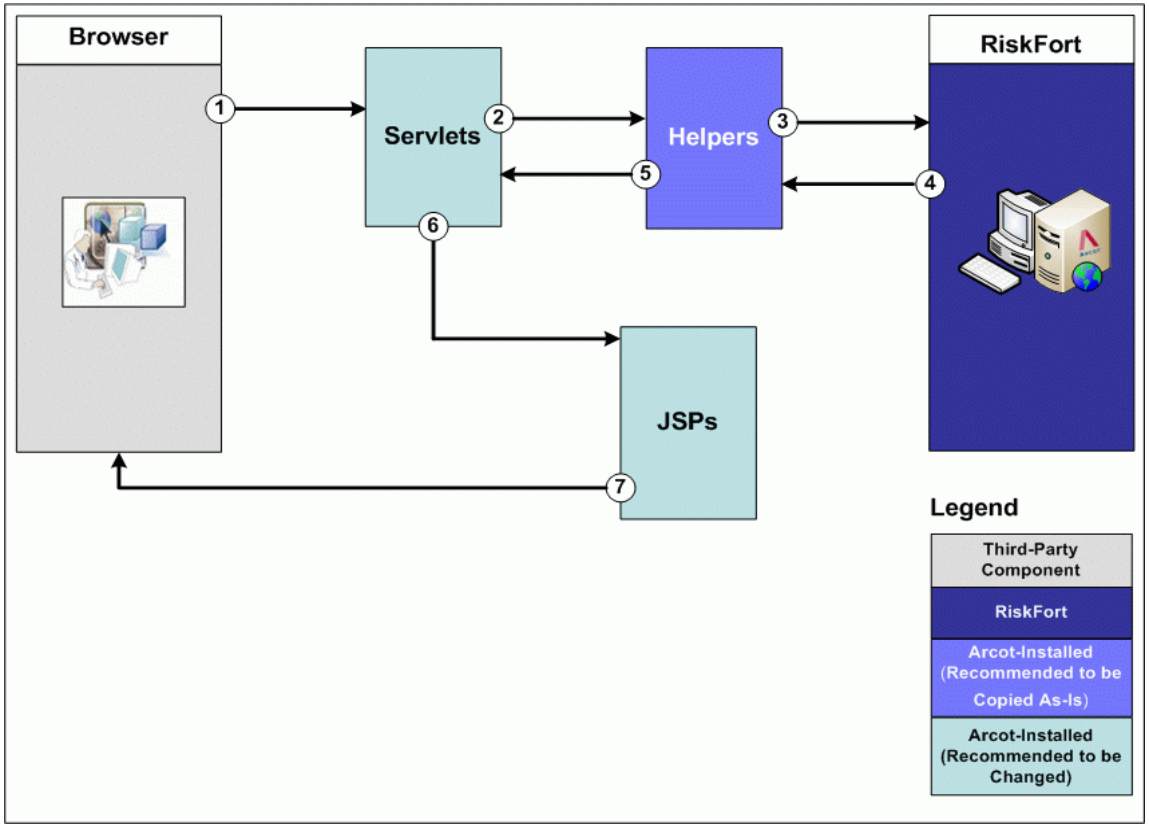


図 G-1 に示されているように、サンプル アプリケーションの各種コンポーネント間の情報フローは、以下のとおりです。

1. ユーザはブラウザ ウィンドウを使用して、トランザクションを開始します。

	注： この手順では、アプリケーションによって認証された後に、エンドユーザがすでに正常にログインしていると仮定します。

2. サーブレットはリクエストを処理し、対応するヘルパー関数を呼び出すことにより、必要な RiskFort SDK を呼び出すことができるようにします。
3. ヘルパー関数は、適切な API を呼び出して、前の手順でサーブレットによって転送された入力データを渡すことにより、RiskFort サーバと対話します。

4. RiskFort サーバは入力データを評価し、リスク スコアおよびアドバイスを返します。いずれの場合にも、関連するタイプのオブジェクトが返されます。

   評価が失敗した場合、例外が生成されます。

5. ヘルパー関数は、対応するサーブレットに対するレスポンスと共に RiskFort SDK によって作成されたオブジェクトを返します。

   失敗の場合には、ヘルパー関数が例外をすべてキャッチし、意味のあるエラー メッセージを表示します。

6. サーブレットは、ヘルパー関数からレスポンスを受信し、リクエスト属性に対応する値を設定し、それを JSP へ転送します。

7. JSP はサーブレットによって設定されたリクエスト属性を解析し、HTTP レスポンスを生成して、それをブラウザへ転送します。

   その後、このレスポンスはユーザに表示されます。

サンプル アプリケーションの推奨事項

この統合モデルでは、次の事項を強く推奨します。

- ヘルパー関数は変更を行わずに使用できます。
- サンプル アプリケーションで提供されている JSP をすべて独自のものに置換することにより、アプリケーションのユーザ インターフェースの外観を制御します。

注 : コードを変更する*前に* API ドキュメントをよく読みます。

- サンプル アプリケーションで提供されているサーブレットをすべて独自のコントローラ ロジックに置換します。
- サンプル アプリケーションを運用環境では使用しません。代わりに、サンプル アプリケーションをコード参照としてのみ使用することにより、独自の Web アプリケーションを構築することを強く推奨します。

サンプル アプリケーションのインストールおよび設定

RiskFort インストーラを実行する際に［**Complete**］オプションを選択すると、サンプル アプリケーションをインストールできます。このオプションを選択すると、他の RiskFort コンポーネントと共にサンプル アプリケーションもインストールされます。

カスタム インストールの一部として、サンプル アプリケーションをインストールするには、［Choose Install Set］画面で［**Risk Evaluation**］オプションを選択する必要があります。

関連文書： サンプル アプリケーションおよびその展開の前提条件に関する詳細情報については、「Arcot RiskFort *2.2.6* インストールおよび展開ガイド」を参照してください。

Windows の場合

RiskFort およびサンプル アプリケーションを Windows に正常にインストール（および後で設定）するには、インストールに使用するユーザ アカウントが `Administrators` グループに属している必要があります。そうでない場合、インストールがエラーなしで完了したとしても、DNS の作成と設定や RiskFort サービスの作成などのインストールにおけるいくつかの重要な手順が正常に完了していない可能性があります。

UNIX ベースのプラットフォームの場合

UNIX ベース プラットフォームへのインストールは、`root` アカウントに制限されていません。root 以外のユーザとしてインストールできます。

サンプル アプリケーションの設定

サンプル アプリケーションは以下のサブセクションで説明するシナリオのいずれにも展開できます。

サンプル アプリケーションと RiskFort を同じシステムに展開する

サンプル アプリケーションと RiskFort サーバが同じシステム上で実行されている場合、サンプル アプリケーションを設定する手順を実行する必要はありません。

ただし、RiskFort サーバがデフォルト（`7680`）以外のポートを使用する場合は、展開手順でポート番号だけを変更する必要があります。以下の手順に従ってください。

1. アプリケーション サーバに riskfort-2.2.6-sample-application.war が展開されていることを確認します。

関連文書： サンプル アプリケーションが正しく展開されていることを確認する方法は、「Arcot RiskFort *2.2.6* インストールおよび展開ガイド」を参照してください。

2. WAR ファイルが展開されている場所に移動して riskfort.risk-evaluation.properties ファイルを開きます。

注： 展開手順（および App_Home）は、使用しているアプリケーション サーバによって異なります。

詳細な手順については、アプリケーション サーバのドキュメントを参照してください。

Apache Tomcat で、riskfort.risk-evaluation.properties は、一般的に以下の場所にあります。

<*App_Home*>\webapps\riskfort-2.2.6-sample-application\WEB-INF\classes\properties**riskfort.risk-evaluation.properties**

3. HOST.1 の値が localhost であることを確認します。
4. PORT.1 の値を変更します。

PORT.1 は、RiskFort サーバが受信リクエストをリスンしているポートを表します。例：PORT.1=*7680*。

サンプル アプリケーションと RiskFort を異なるシステムに展開する

サンプル アプリケーションと RiskFort サーバが別々のシステムで実行されている場合、サンプル アプリケーションを設定するために以下の手順に従う必要があります。

1. アプリケーション サーバに riskfort-2.2.6-sample-application.war が展開されていることを確認します。

関連文書： サンプル アプリケーションが正しく展開されていることを確認する方法は、「Arcot RiskFort *2.2.6* インストールおよび展開ガイド」を参照してください。

2. WAR ファイルが展開されている場所に移動して riskfort.risk-evaluation.properties ファイルを開きます。

注： 展開手順（および App_Home）は、使用しているアプリケーション サーバによって異なります。

詳細な手順については、アプリケーション サーバのドキュメントを参照してください。

Apache Tomcat で、riskfort.risk-evaluation.properties は、一般的に以下の場所にあります。

<*App_Home*>\webapps\riskfort-2.2.6-sample-application\WEB-INF\classes\properties**riskfort.risk-evaluation.properties**

3. HOST.1 の値を変更します。

   HOST.1 は、RiskFort サーバのホスト名または IP アドレスを表します。例：HOST.1=10.150.1.111

4. PORT.1 の値を変更します。

   PORT.1 は、RiskFort サーバが受信リクエストをリスンしているポートを表します。例：PORT.1=7680。

5. **（RiskFort サーバが SSL モードで実行されていない場合のオプション）**この手順は、SSL でのサンプル アプリケーションと RiskFort サーバの間の通信を安全にする場合にのみ実行します。

   この場合は、TRANSPORT_TYPE の値を SSL に変更します。

   TRANSPORT_TYPE=SSL

6. サーバ CA 証明書の場所を CA_CERT_FILE=<*Server_CA_certificate_location*> に指定します。

重要：証明書は PEM 形式である*必要があります*。

   例：CA_CERT_FILE=C:\certs\riskfort_ca.pem

7. 変更を保存して、開いているファイルを閉じます。
8. サンプル アプリケーションが実行されているアプリケーション サーバを再起動します。

これらの手順を完了することにより、サンプル アプリケーションの展開も完了します。これで、以下のセクションで示すように、サポートされている各操作の API ワークフローを実行し、それを理解することができます。

リスク評価の実行

RiskFort サンプル アプリケーションでは、RiskFort リスク評価プロセスの特徴をいくつか実例で示します。ここでは以下の項目について説明します。

- 収集された情報に対するリスク評価の実行
- 収集されたユーザ情報の編集
- API ワークフローおよびリファレンス

収集された情報に対するリスク評価の実行

サンプル アプリケーションは、API を使用して、収集されたデータに対するリスク評価を行う方法を実例で示します。デモンストレーションを表示するには、以下の手順に従います。

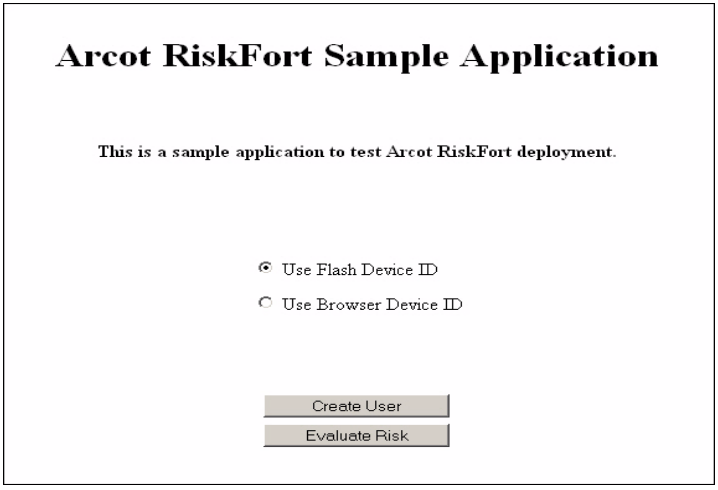
1. ブラウザ ウィンドウでサンプル アプリケーションの URL にアクセスします。通常の URL は次のとおりです。

   http://<app_server_host_name>:<port_number>/riskfort-2.2.6-sample-application/index.jsp

   前の URL では、<app_server_host_name> はサンプル アプリケーションが展開されているアプリケーション サーバのホスト名または IP アドレスを示し、<port_number> はサンプル アプリケーションが利用可能なポート番号を示します。

   図 G-2 に示されているように、RiskFort サンプル アプリケーションのランディング ページが表示されます。

図 G-2RiskFort サンプル アプリケーション：ランディング ページ

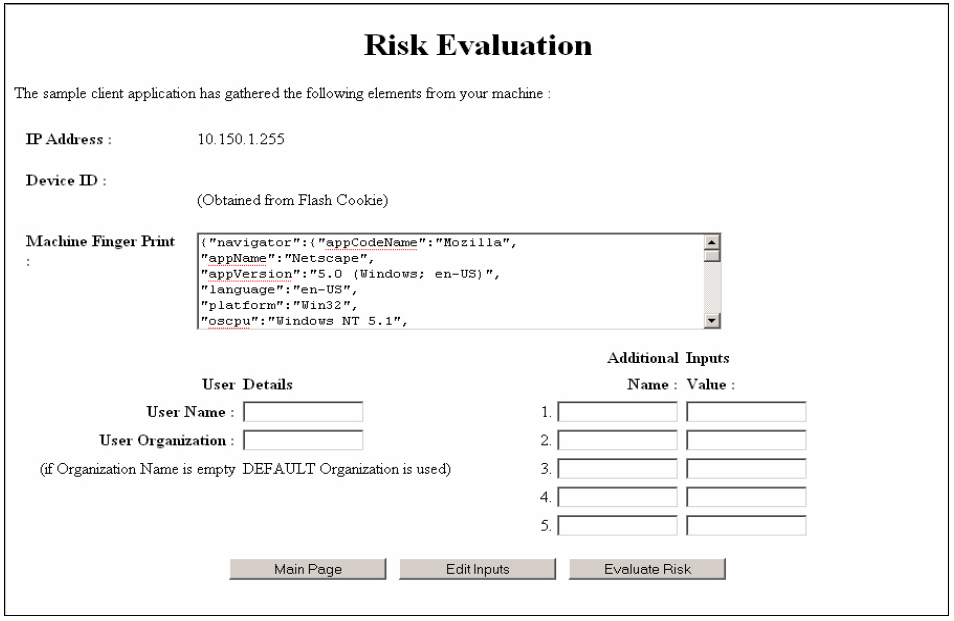


2. Device ID のストレージ設定に基づいて［**Use Flash Device ID**］または［**Use Browser Device ID**］オプションを選択します。

   RiskFort はリスク評価のパラメータとして、この格納された Device ID を使用します。

3. ［**Risk Evaluation**］をクリックして、マイ デフォルト プロファイル ページで［Risk Evaluation］を開きます。

図 G-3 リスク評価の入力ページ



4. ページで、以下を指定します。
   - ［**User Name**］フィールドにリスク評価の対象となるユーザの名前を指定します。

> **注 :** サンプル アプリケーションにまだ登録されていないユーザの名前を指定することもできます（G-174 ページの「リスク評価の実行」で詳述）。

   - （オプション）［**User Organization**］フィールドにユーザが所属する組織の名前を指定します。

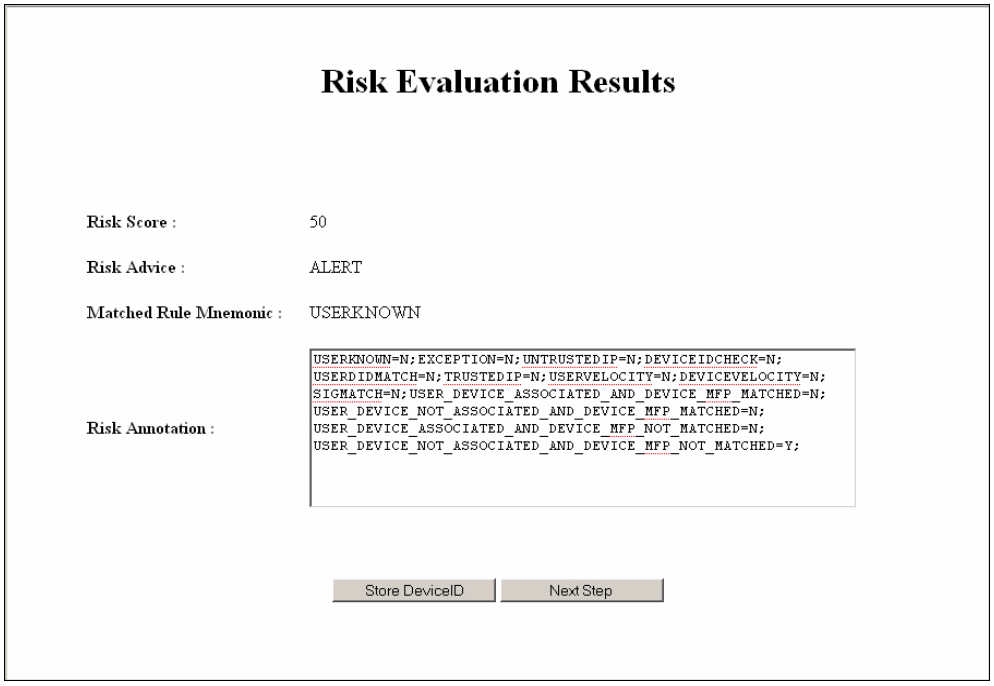
> **注 :** ［**User Organization**］フィールドを空にしておくと、値が自動的に DEFAULTORG に設定されます。

   - （オプション）ロケール情報やトランザクションの量などの情報を［**Additional Inputs**］の対応する［**Name**］および［**Value**］フィールドに指定します。

5. ［**Evaluate Risk**］をクリックして、指定されたユーザのリスク スコアおよびリスク アドバイスを生成します。

   リスク評価の結果ページが表示されます。図 G-4 に表示されているように、ユーザが RiskFort に登録されていない場合、アドバイスは通常、`ALERT` です。このアドバイスは、この付録の後の G-184 ページの「ユーザの作成」で説明されているように、RiskFort データベースにユーザを作成すると変わります。

図 G-4 リスク評価の結果



6. ［**Store DeviceID**］をクリックして、ユーザのローカル コンピュータにデバイス ID 情報（手順 2）の指定されたタイプを保存します。

   - 通常、Flash SharedObject （またはトークン）は以下の場所の適切なサブディレクトリに格納されます。

     - **Microsoft Windows：**

       ```
       C:\Documents and Settings\<user_profile>\Application
       Data\Macromedia\Flash Player\#SharedObjects\
       ```

     - **UNIX プラットフォーム：**

       ```
       /<user_home>/macromedia/flash/#SharedObjects/
       ```

- ブラウザ cookie を使用するように選択した場合、使用しているブラウザに基づいて、この cookie は以下の場所にあります。

  Windows Internet Explorer：

  - **Microsoft Windows：**

  ```
  C:\Documents and Settings\meeta\Local Settings\Temporary Internet Files\
  ```

  Mozilla Firefox：

  - **Microsoft Windows：**

  ```
  C:\Documents and Settings\<user_profile>\Application Data\Mozilla\Firefox\Profiles\<random_name>.default
  ```

  - **UNIX プラットフォーム：**

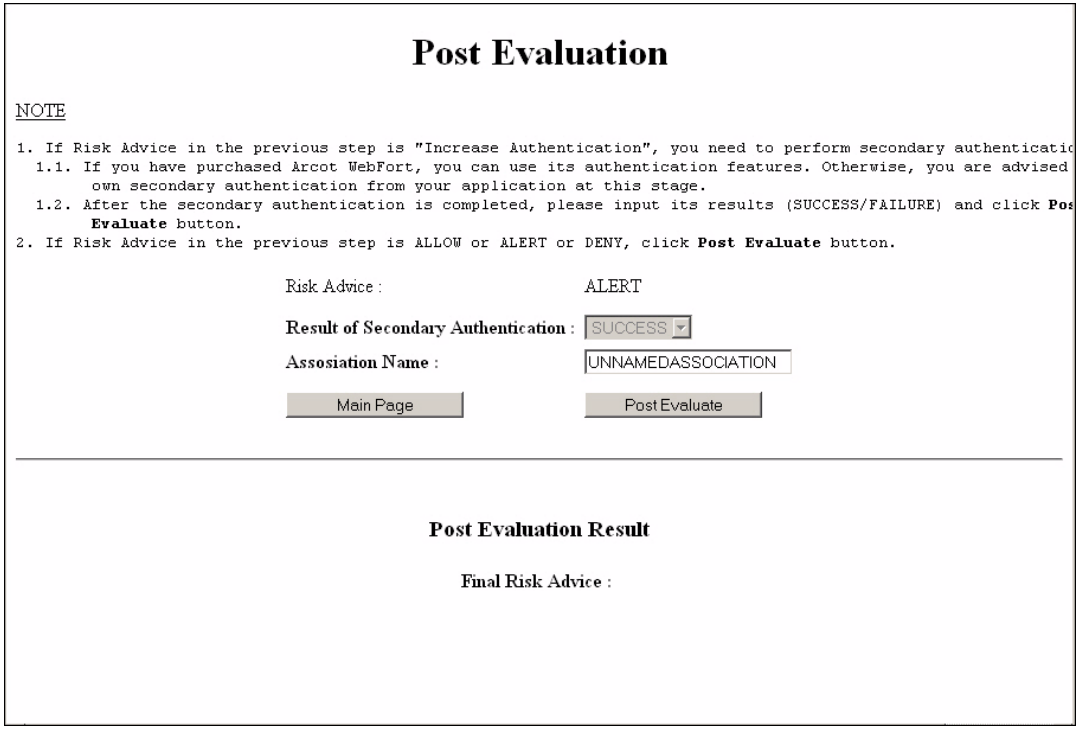
  ```
  /<user_home>/mozilla/firefox/profiles/<random_name>.default/
  ```

7. ［**Next Step**］をクリックして指定されたユーザの事後評価を実行します。

   事後評価ページ（図 G-5）が表示されます。

	**重要：**ページで示されているように、前のページのアドバイスが INCREASEAUTH の場合、2 次認証メカニズムを組み込んで、RiskFort の事後評価手順を実行する前にユーザが別の認証を実行することを強く推奨します。

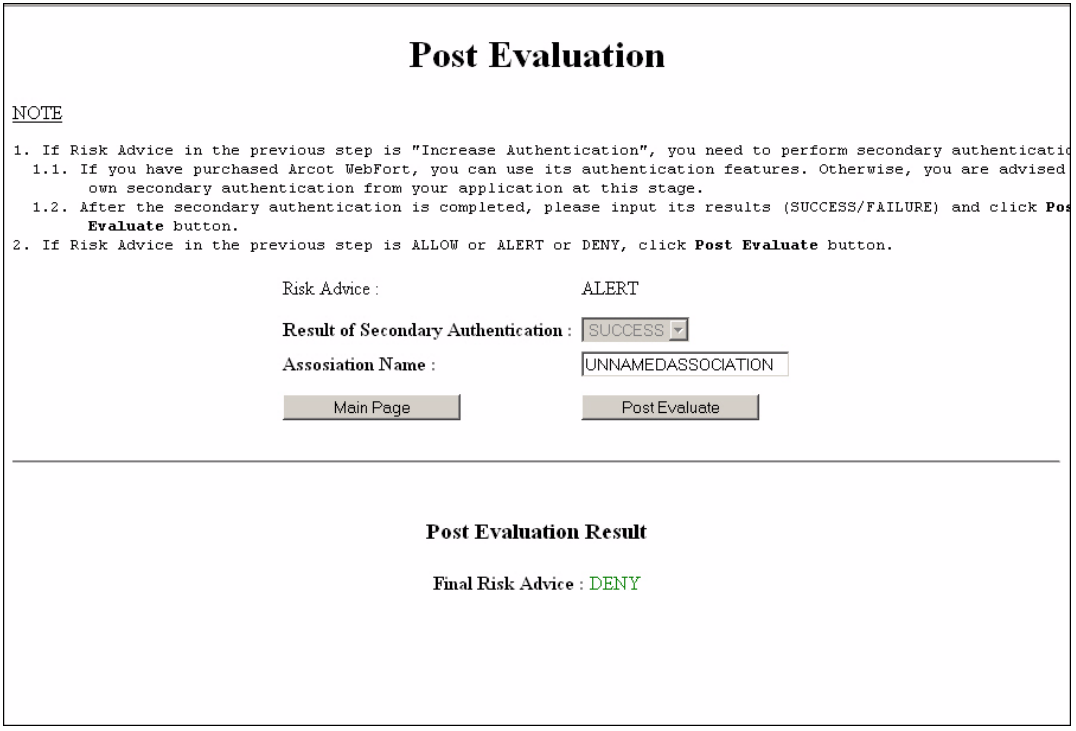
図 G-5 事後評価ページ



8. ［**Post Evaluate**］をクリックして、アプリケーションの最終的なリスク アドバイスを生成して表示します。

   最終的な結果がページ（図 G-6）の［Post Evaluation Results］フレームに表示されます。

図 G-6 事後評価の結果（ページの下部のフレーム）



収集されたユーザ情報の編集

サンプル アプリケーションでは、既存ユーザの収集されたデバイス ID 情報を編集することもできます。これによってユーザが実際に経験する可能性のある状況を想定できます。たとえば、登録されているユーザの IP アドレスが変更された場合、そのユーザは別の場所からログインしていると想定できます。

デモンストレーションの目的で、サンプル アプリケーションを使用して編集できるデータは次のとおりです。

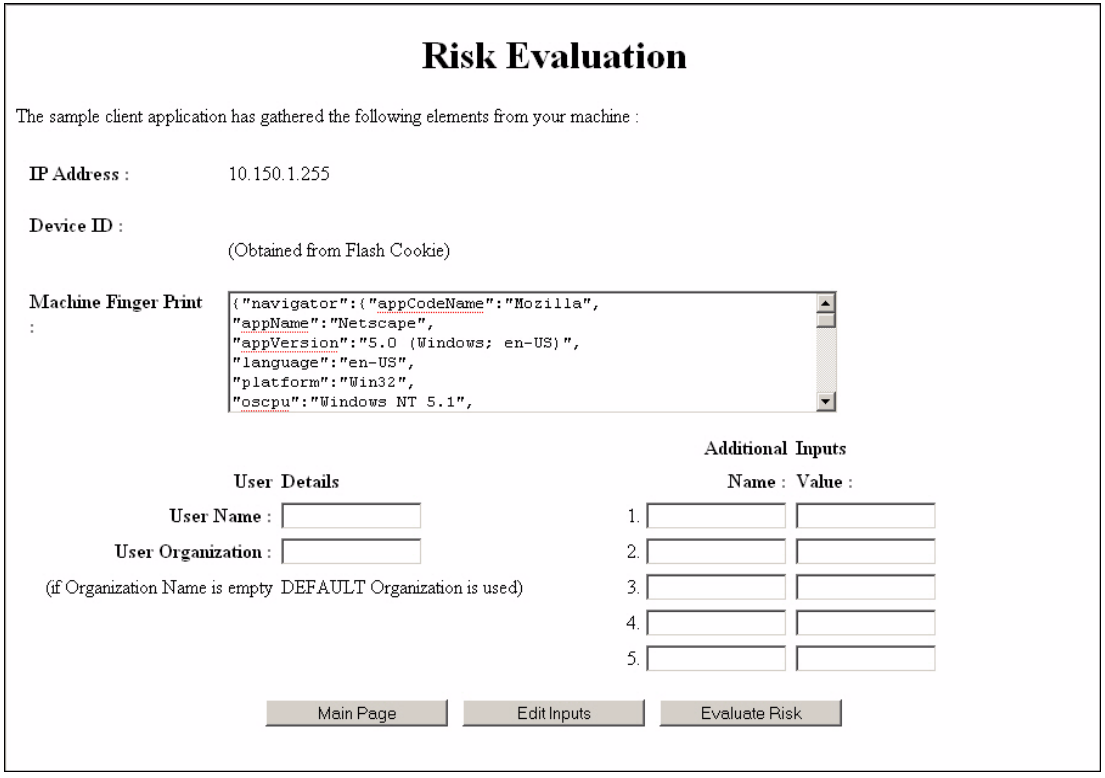
- IP アドレス
- デバイス ID

ユーザの情報が更新された後、G-179 ページの 手順 8 から G-177 ページの 手順 5 で説明されているように、リスク評価の手順を実行する必要があります。そのような場合、RiskFort は通常 `INCREASEAUTH` アドバイスを生成します。

サンプル アプリケーションを使用してユーザの情報を編集するには、以下のタスクを実行します。

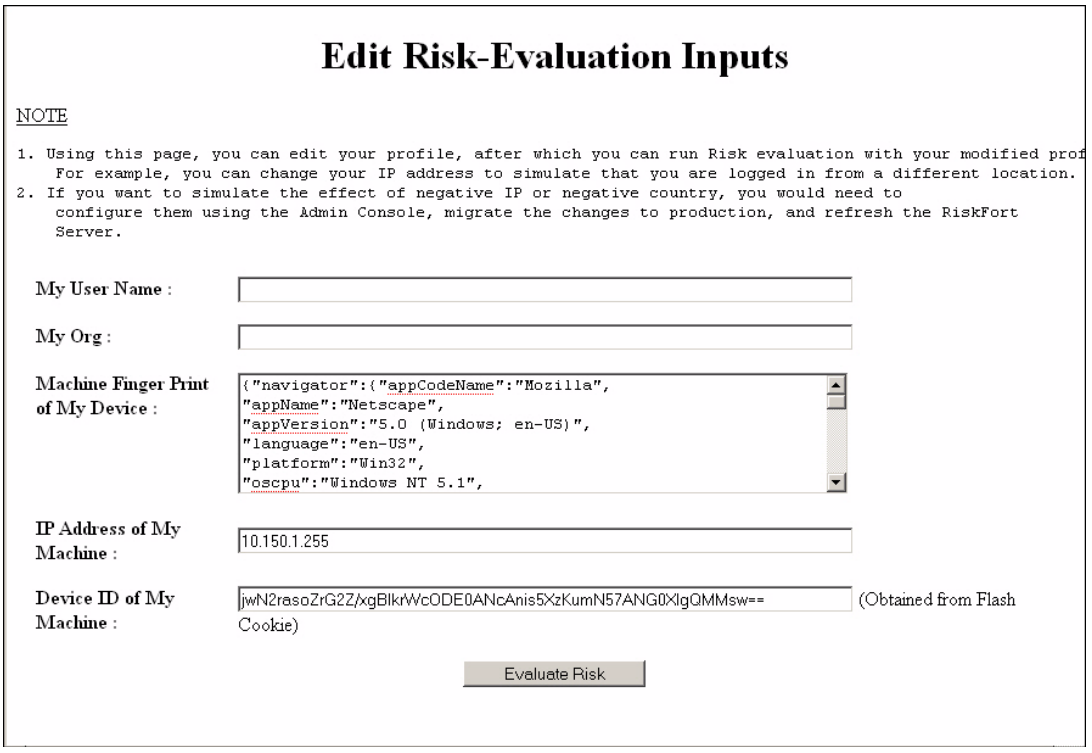
1. RiskFort サンプル アプリケーション ページ（図 G-2）で、［**Risk Evaluation**］をクリックして、リスク評価の入力ページ（図 G-7）を開きます。

図 G-7 リスク評価の入力ページ



2. ［**User Name**］を指定し、必要に応じて［**User organization**］を指定します。
3. ［**Edit Inputs**］をクリックして、リスク評価の入力データの編集ページ（図 G-8）を開きます。

図 G-8 リスク評価の入力データの編集ページ



4. ［**IP Address of My Machine**］フィールドの IP アドレス、または［**Device ID of My Machine**］フィールドのデバイス ID、または両方を変更します。
5. ［**Evaluate Risk**］をクリックして、指定されたユーザのリスク スコアおよびリスク アドバイスを生成します。

   リスク評価の結果ページ（図 G-4）が表示されます。
6. ［**Store DeviceID**］をクリックして、ユーザのローカル コンピュータにデバイス ID 情報の指定されたタイプを保存します。
7. ［**Next Step**］をクリックして指定されたユーザの事後評価を実行します。

   事後評価ページが表示されます。
8. （RiskFort がこの追加の認証の結果を受信するシナリオを想定するため）［**Result of Secondary Authentication**］フィールドに 2 次認証の結果を指定します。
9. ［**Post Evaluate**］をクリックして、アプリケーションの最終的なリスク アドバイスを生成して表示します。

最終的な結果がページの［Post Evaluation Results］フレームに表示されます。

API ワークフローおよびリファレンス

サンプル アプリケーションをリスク評価に使用する場合は、以下のようにします。

1. `com.arcot.riskfort.sampleapp.initialize` パッケージの `ArRFInitHandler.class` が呼び出されます。

   初期化が完了したら、RiskFort はリクエストを処理することができます。

2. RiskFort 関連の操作を担う `RiskFactory` が `com.arcot.riskfort.sampleapp.helpers` パッケージの `ArRFEvaluateHelper.class` によって呼び出されます。

   `ArRFEvaluateHelper.class` の `evaluateRisk()` メソッドが呼び出されると、`RiskXActionAPI` インターフェースが呼び出されます。

3. **リスク評価**を実行するには、アプリケーションのサーブレット、.jsp または他の呼び出しファイルは、`ArRFEvaluateHelper` クラスの `evaluateRisk()` メソッドを呼び出す必要があります。

   このメソッドはレスポンスとして `RiskAssessment` オブジェクトを返します。それには、`riskScore`、`riskAdvice`、`deviceID` およびアプリケーションの関連情報が含まれます。

	**重要 :** この段階で生成されたリスク スコアおよびアドバイスに基づいて、アプリケーションは、必要なアクションを実行するためのロジックを提供する必要があります。たとえば、CSR にトランザクション リクエストを転送したり、ユーザに追加の認証を実行するように強制するなどです。

4. 事後評価を実行するには、アプリケーションのサーブレット、.jsp または他の呼び出しファイルは、`ArRFEvaluateHelper` クラスの `postEvaluateHelper()` メソッドを呼び出す必要があります。

	**重要 :** `evaluateRisk()` メソッドが実行された後でのみ、`postEvaluateHelper()` メソッドを呼び出す必要があります。

このメソッドでは、入力データとして CallerID、RiskAssessment オブジェクト（evaluateRisk() 関数による戻り値）、2 次認証の結果（ある場合、secondaryAuthenticationStatus）、および関連名（オプション）を受け入れます。その後、アプリケーションに最終アドバイスが含まれた postEvaluateResponseObj オブジェクトを返します。

前のワークフローで必要なクラスおよびメソッドは、表 G-1 に詳述しています。

表 G-1. サンプル アプリケーションによるリスク評価に必要なメイン API

API	説明
クラス： ArRFEvaluateHelper	リスク評価および事後評価に必要なメソッドが含まれるヘルパー クラス。
ÉÅÉ\ÉbÉhÅF evaluateRisk()	リスク スコアおよびリスク アドバイスを生成するメソッド。
ÉÅÉ\ÉbÉhÅF postEvaluateHelper()	最終的なリスク アドバイスを生成するメソッド。このアドバイスはブール値です。True の場合、アドバイスは ALLOW ですが、False の場合、アドバイスはトランザクションを ALLOW に設定しません。 このメソッドを呼び出す必要があるのは、evaluateRisk() が実行された *後* でのみです。

ユーザの作成

RiskFort に新しいユーザを作成する操作を行うのは、新規ユーザを RiskFort データベースに追加する場合の 1 回限りです。通常、アプリケーションの既存のユーザが初めて RiskFort にアクセスする場合に、ユーザを追加します。

このセクションでは、サンプル アプリケーションでユーザの作成方法の実例を示し、同じアプリケーションの API ワークフローについても説明します。

- ユーザの作成
- 作成したユーザでのリスク評価の実行
- API ワークフローおよびリファレンス

ユーザの作成

サンプル アプリケーションでは、API を使用して RiskFort にユーザを作成する方法を実例で示します。デモンストレーションを表示するには、以下の手順に従います。

1. ブラウザ ウィンドウでサンプル アプリケーションの URL にアクセスします。通常の URL は次のとおりです。

   http://<app_server_host_name>:<port_number>/riskfort-2.2.6-sample-application/index.jsp

   前の URL では、<*app_server_host_name*> はサンプル アプリケーションが展開されているアプリケーション サーバのホスト名または IP アドレスを示し、<*port_number*> はサンプル アプリケーションが利用可能なポート番号を示します。

   RiskFort サンプル アプリケーション ページ（図 G-2）が表示されます。

2. ［**Create User**］をクリックして RiskFort データベースに新規ユーザを作成します。

	**注：** RiskFort にユーザを作成する場合は、どの cookie タイプにも依存しません。そのため、ユーザを作成するときにデバイス ID タイプ（前の図で表示したように、フラッシュまたはブラウザ）を指定する必要はありません。

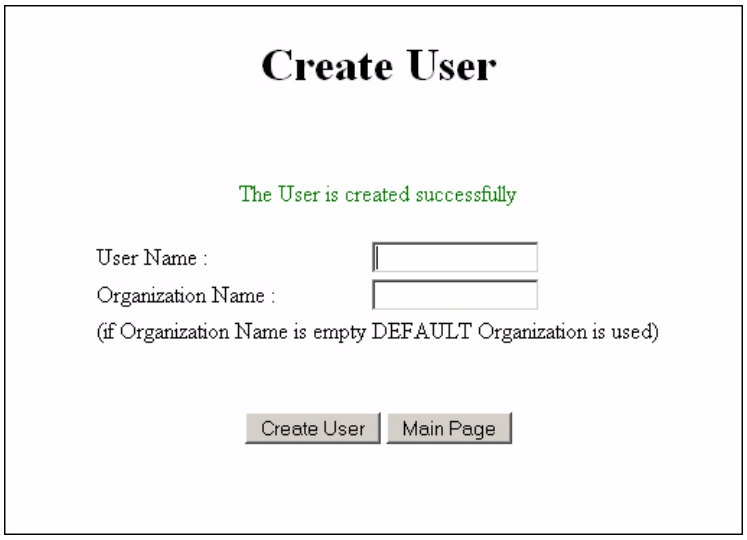
ユーザの作成ページ（図 G-9）が表示されます。

図 G-9 ユーザの作成ページ

3. ［**User Name**］と［**Organization Name**］を指定し、［**Create User**］をクリックします。

   指定されたユーザが正常に作成された場合は、図に示したような成功メッセージが表示されます。

図 G-10 ユーザの作成ページ：成功メッセージ

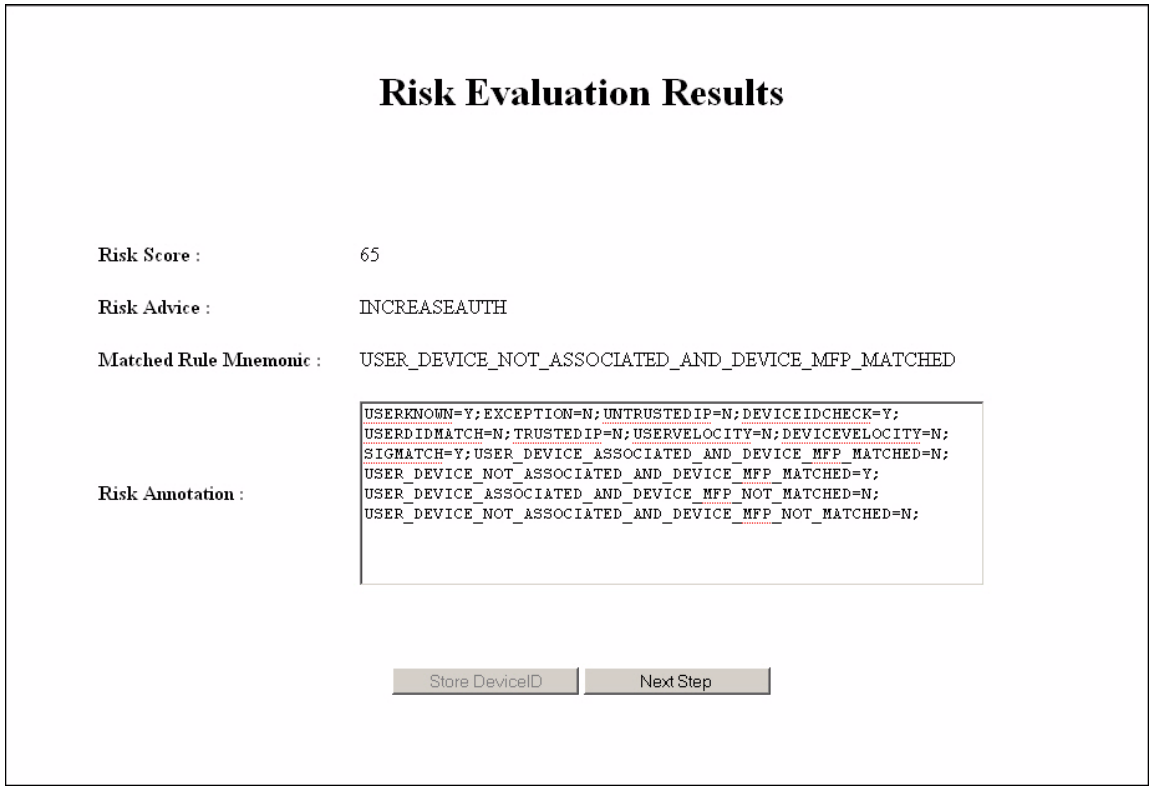


4. ［**Create User**］をクリックして別のユーザを作成するか、または［**Main Page**］をクリックしてサンプル アプリケーションの開始ページ（`index.html`）に戻ります。

作成したユーザでのリスク評価の実行

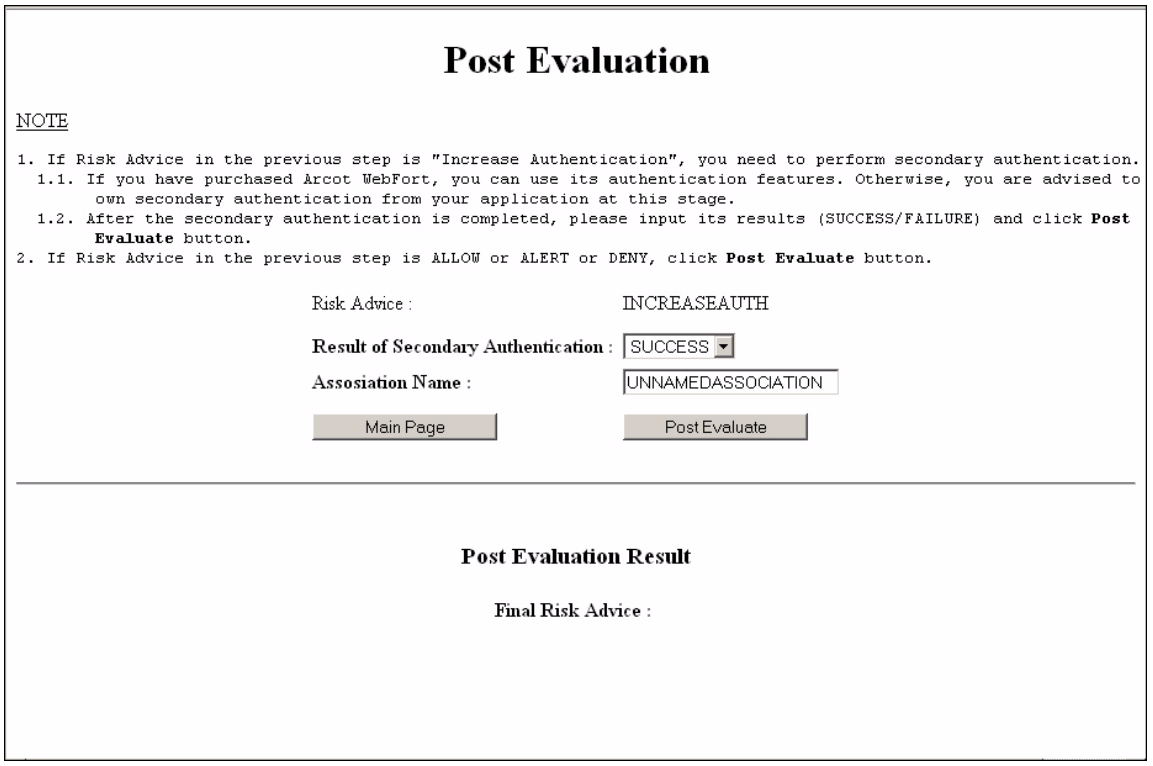
作成したユーザでリスク評価（G-174 ページの「収集された情報に対するリスク評価の実行」で説明したように、G-175 ページ の 手順 2 から G-177 ページ の手順 5）を実行すると、リスク評価の結果ページ（図 G-11）で、［**Risk Score**］および［**Risk Advice**］の値にそれぞれ `65` と `INCREASEAUTH` が表示されます。

図 G-11 リスク評価の結果：ユーザの作成後



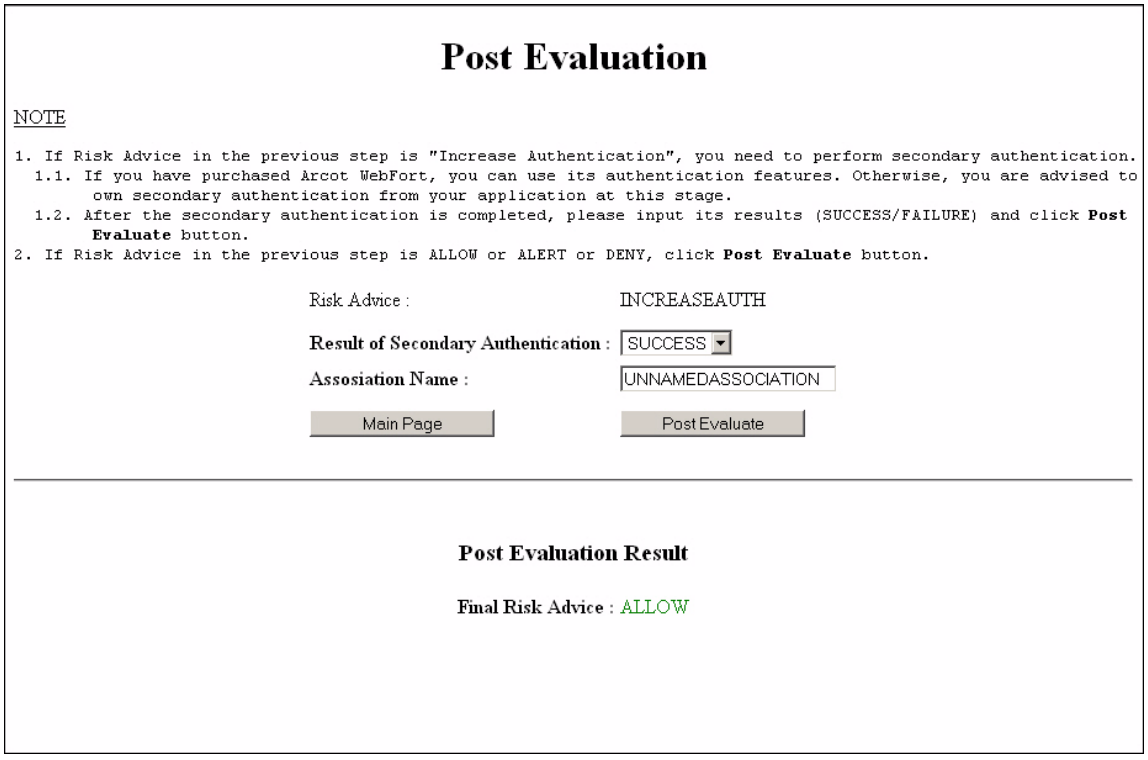
［**Next Step**］をクリックして事後評価を実行すると、以下のように事後評価ページ（図 G-12）が表示されます。

図 G-12 リスクの事後評価の結果：ユーザの作成後



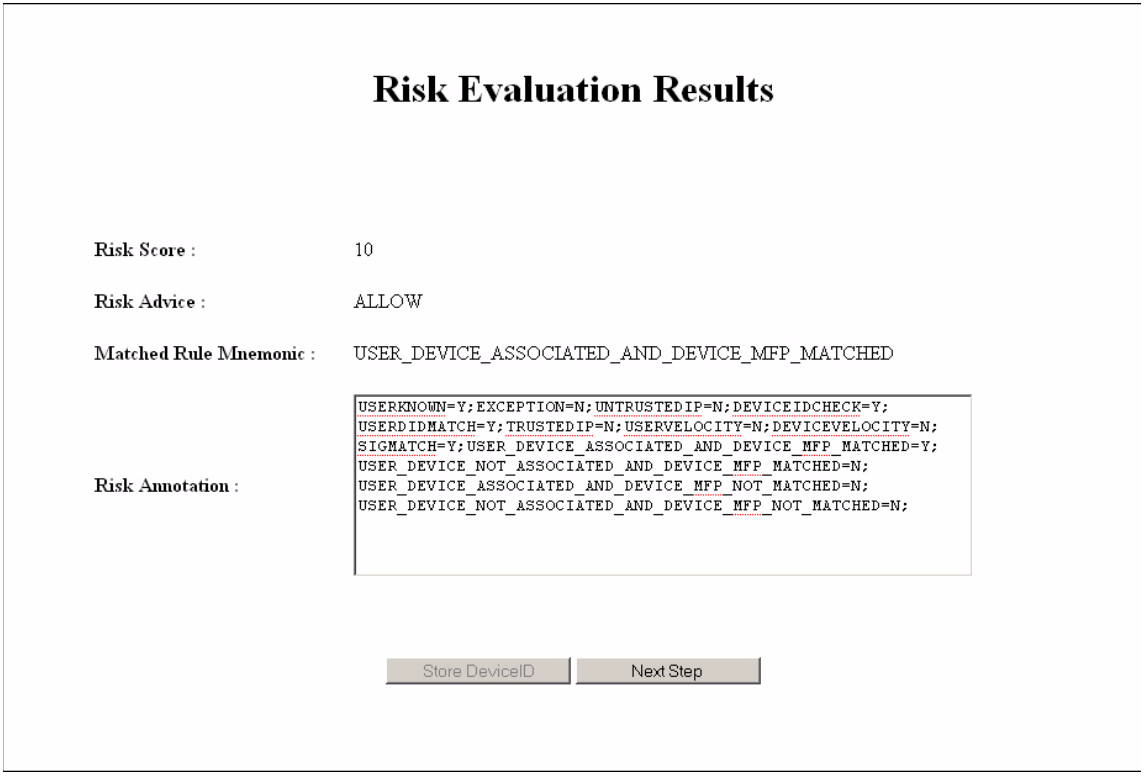
［**Result of Secondary Authentication**］フィールドで［SUCCESS］を選択し、［**Post Evaluate**］をクリックすると、図 G-13 に表示されているように、［Final Risk Advice］に［**ALLOW**］と表示されます。

図 G-13 リスクの事後評価の結果：ユーザの作成後



ユーザに対して **ALLOW** アドバイスが表示された後で、ユーザ IP またはデバイス ID 情報で変更を行わない限り、次のリスク評価結果では常に **ALLOW** （図 G-14）が表示されます。

図 G-14 リスクの事後評価の結果：ユーザの作成後



API ワークフローおよびリファレンス

サンプル アプリケーションのユーザの作成ページで、［**Create User**］ボタンをクリックすると、以下のようになります。

1. `com.arcot.riskfort.sampleapp.initialize` パッケージの `ArRFInitHandler.class` が呼び出されます。

   初期化が完了したら、RiskFort はリクエストを処理することができます。

2. ArRFCreateUser.jsp は ArRFCreateUserServlet.class サーブレットを呼び出します。このサーブレットは、次に、com.arcot.riskfort.sampleapp.helpers パッケージの ArRFUserIssuanceHelper.class ヘルパー クラスを呼び出します。

重要： createUser() 関数を呼び出す*前に* AuthProvisionFactory が初期化されたことを確認する必要があります。

   ArRFUserIssuanceHelper.class に表示されているように、アプリケーション サーブレット、.jsp または他の呼び出しファイルから呼び出す必要のある最も重要な関数は、createUser() メソッドです。

前のワークフローで必要なクラスおよびメソッドは、表 G-2 に詳述しています。

表 G-2. サンプル アプリケーションからユーザを作成するのに必要なメイン クラスおよびメソッド

API	説明
クラス： ArRFUserIssuanceHelper	指定されたユーザを作成するのに必要なメソッドが含まれるヘルパー クラス。
メソッド： createUser()	指定されたユーザを作成するメソッド。

付録 H
用語集

CSR（Customer Support Representatives、テクニカル サポート担当者）	セキュリティ システムのユーザに関連する日常業務を担当する管理者。 たとえば、管理者は、ユーザの登録支援、パスワードのリセット、登録レポートの生成を行うことができます。
Global Administrator	CSR（Customer Support Representatives、テクニカル サポート担当者）アカウントのセットアップおよびシステムの設定を担当する管理者。
Master Administrator	RiskFort 管理者の最高レベル。主な担当業務は、RiskFort を初期化し、Global Administrator アカウントを作成することです。
PKI（Public Key Infrastructure）	ネットワーク環境における公開キー暗号化法および証明書の使用を促進する標準およびサービス。
RiskFort	RiskFort には、指定されたトランザクションのリスクを評価するメカニズムが用意されています。
RiskFort ネイティブ プロトコル	RiskFort サーバ、そのコンポーネント、WebFort サーバ、および Administration Console の間の通信用の Arcot 専用プロトコル。
SSL（Secure Sockets Layer）	公衆ネットワーク上でのメッセージ伝送のセキュリティを管理するためのプロトコル。 このプロトコルは、TLS（Transport Layer Security）の前身です。
TCP（Transmission Control Protocol）	保証されたデータ伝送を行うためのインターネット プロトコル。暗号化されていないデータを送信します。
TLS（Transport Layer Security）	データ暗号化の使用により公衆ネットワークを介して通信をセキュリティ保護し認証するプロトコル。
アグリゲータ	複数の企業にまたがってユーザ情報を照合することによってアカウント集約サービスを提供するサードパーティ ベンダー。
アドオン ルール	RiskFort に付属する追加のリスク評価ルール。

クレデンシャル	ユーザ ID を証明するもの。デジタル クレデンシャルは、スマート カードまたは USB トークンなどのハードウェアまたはサーバ上に保存されている場合があります。そのクレデンシャルは認証時に検証されます。
コールアウト	外部で（RiskFort コンテキストの外で）実行するカスタム プログラム。
サーバ管理プロトコル	RiskFort サーバを起動およびシャットダウンするための Arcot 専用プロトコル。
スコアリング エンジン	個別の評価ルールからリスク スコアを収集し、スコアリングの優先順位順にそれらを処理する RiskFort サーバのコンポーネント。
スコアリング コールアウト	RiskFort のスコアリング エンジンによってスコアが付けられた後で動作し、最終リスク スコアを変更するカスタムのスコアリング ロジックを含むコールアウト。
スコアリング ルール	他のすべての設定済みルールの実行結果を受け取り、最終リスク スコアおよびリスク アドバイスを返す最後のルール。
ゾーン ホッピング	現実的な速度で移動可能な距離以上に隔たった場所からの、同一ユーザによる連続するトランザクション。
デジタル証明書	個人、コンピュータ システム、または組織の ID およびキーの所有権の証明となるデジタル ドキュメント。この認証方式は公開キー暗号化（PKI）法に基づいています。
デバイス頻度	指定された時間内に同一デバイスから発行されるトランザクションの数。
トラステッド IP アドレス	組織に信頼されているため今後のリスク評価から除外される IP アドレス。
トラステッド アグリゲータ	組織に信頼されているため今後のリスク評価から除外されるアグリゲータ。
ユーザ ID/ パスワード	登録時にユーザに発行されるクレデンシャルの 1 つ。
ユーザ頻度	指定された時間内に同一ユーザから発行されるトランザクションの数。
リスク アドバイス	トランザクションのリスクを評価した後に RiskFort によって呼び出し元のアプリケーションに示されるアクション（ALLOW、ALERT、DENY、INCREASEAUTH）。
リスク スコア	評価結果に応じて RiskFort から通知されるスコア。スコアは 0 ～ 100 です。数字が大きいほど、リスクも高くなります。

暗号化	内容を判読できないように情報にスクランブルをかける処理。
一方向 SSL	SSL セッションが確立される前に、クライアント アプリケーションが（サーバのデジタル証明書を受理することによって）サーバ アプリケーションの ID を検証します。
拒否 IP アドレス	過去において、アノニマイザ プロキシ、不正なトランザクション、または悪意のあるトランザクションの要求元であったことがわかっている IP アドレス。
拒否国	過去において、不正なトランザクションまたは悪意のあるトランザクションの要求元であったことがわかっている国。
公開キー	PKI で使用される 1 対のキーの一方。このキーは自由に配布され、証明書の一部として発行されます。 通常、公開キーの所有者に送信されたデータを暗号化するために使用されます。その後、公開キーの所有者は、対応する秘密キーを使用して、データを復号化します。
終端ルール	単独で全体的なリスク スコアを決定するルール。
証明書	「デジタル証明書」を参照してください。
双方向 SSL	SSL セッションが確立される前に、クライアント アプリケーションとサーバ アプリケーションの両方が（それぞれのデジタル証明書を提示することによって）互いの ID を確認します。
追加入力	リスク評価のためにアドオン ルールによって使用されるトランザクションに関連する追加のエレメント。
追加認証	現在のトランザクションが安全でないと RiskFort によって判断される場合に、RiskFort から与えられるリスク アドバイス。たとえば、ユーザが初めて大量のトランザクションを実行する場合が挙げられます。そのような場合、ユーザは、より強力な認証方式を使用して認証サーバに再認証されることを求められます。
認証	エンティティのログイン情報が本人のものであることを証明するプロセス。
認証トークン	トークンは、コンピュータ サービスの許可ユーザに与えられるオブジェクトで、認証の際に補助的に使用されます。
秘密キー	PKI で使用される 1 対のキーの一方。このキーは秘密に保持され、データの復号化または暗号化に使用できます。
非終端ルール	全体的なリスク スコアは、このルール単独では決定されません。他のルールも必要です。

評価コールアウト	すべての評価ルールの後で実行され、カスタムのリスク評価ロジックを含むコールアウト。
評価ルール	着信トランザクション データに適用される、事前に設定された RiskFort ロジック。
頻度チェック	「デバイス頻度」および「ユーザ頻度」を参照してください。
例外ユーザ	RiskFort に「登録」されており、指定された期間、リスク評価から除外されるユーザ。

索引