Arcot WebFort VAS®
Java 開発者ガイド
バージョン 6.2

455 West Maude Avenue, Sunnyvale, CA 94085

Arcot WebFort Java 開発者ガイド
バージョン 6.2
2010 年 5 月
部品番号：WF-0062-0DGJ-10

商標

特許

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

サードパーティ製ソフトウェア

Arcot WebFort および関連コンポーネントで使用されているサードパーティ製ソフトウェアの一覧については、「Arcot WebFort 6.2 インストールおよび展開ガイド」の付録「サードパーティ製ソフトウェアのライセンス」を参照してください。

目次

序文

このガイドでは、Arcot WebFort VAS で提供される強力で多用途の認証を使用する Web アプリケーションの開発方法について説明します。また、プログラム的に WebFort SDK と統合するために使用できる Java クラスおよびメソッドについて説明します。

WebFort API の詳細については、付属の**サンプル アプリケーション**に含まれている以下の JSP ファイルを参照してください。

- ユーザ操作
  - `ArWFCreateUser.jsp`：このページはユーザを作成するために使用します。
  - `ArWFUpdateUser.jsp`：このページはユーザ情報を更新するために使用します。
  - `ArWFFetchUser.jsp`：このページはユーザの詳細を取得するために使用します。
- クレデンシャル操作
  - `ArWFCreate<Credential>.jsp`：このページはクレデンシャルを作成するために使用します。
  - `ArWF<Credential>Authenticate.jsp`：このページはクレデンシャルを使ってユーザを認証するために使用します。
  - `ArWFUPPartialPwdAuthenticate.jsp`：このページはユーザの部分パスワードを使ってユーザを認証するために使用します。
  - `ArWFQnACallerSideAuthenticate.jsp`：このページはコール側検証によって QnA クレデンシャルを使ってユーザを認証するために使用します。
  - `ArWF<Credential>Synchronization.jsp`：このページはクライアント デバイス上の OTP と WebFort サーバを同期するために使用します。
  - `ArWFFetch<Credential>.jsp`：このページはクレデンシャルの詳細を取得するために使用します。
  - （ArcotID のみ）`ArWFSelectArcotIDClient.jsp`：このページは ArcotID Client を選択するために使用します。
  - （ArcotID のみ）`ArWFDownloadArcotID.jsp`：このページは ArcotID Client をダウンロードするために使用します。

注 : サンプル アプリケーションに記載されていない操作については、関連する Javadoc を参照してください。

対象読者

このガイドは、Arcot WebFort の API および関数を使用して WebFort の発行および認証機能を実装する必要がある Java アプリケーション プログラマを対象としています。

次の内容をよく理解している必要があります。

- Java プログラミング
- 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、HTTPS、および TCP/IP などのインターネット プロトコル
- SSL（Secure Sockets Layer）通信および関連する概念（公開および秘密キー交換、デジタル証明書およびデジタル署名、認証機関）
- SAML（Security Assertion Markup Language）の基本

本書の内容

このガイドは次のように構成されています。

- 第 1 章の「はじめに」では、WebFort Java SDK の概要とプログラミングを開始するための前提条件について説明します。
- 第 2 章の「WebFort ワークフローについて」では、WebFort SDK で実装できる一般的なワークフローについて説明します。
- 第 3 章の「SDK を使用する前に」では、`CLASSPATH` に追加する必要がある API 固有の JAR ファイルおよびプロパティ ファイルについて説明します。
- 第 4 章の「発行操作の実行」では、発行 API を使用してユーザとユーザのクレデンシャルを作成および管理する方法について説明します。

- 第 5 章の「ArcotID Client とアプリケーションの統合」では、ArcotID 認証に必要な ArcotID タイプ（Flash または署名済みの Java アプレット）を実装する方法について説明します。
- 第 6 章の「ユーザの認証」では、認証 API で提供される認証メカニズムのいずれかを使用して、ユーザを認証する方法について説明します。
- 第 7 章の「カスタム API の使用」では、カスタム クレデンシャルを作成し、それを使用して認証する方法について説明します。
- 付録 A の 「入力データの検証」では、SDK 入力パラメータの確認に使用する基準の一覧を示します。
- 付録 B の 「WebFort のログ」では、さまざまなログ ファイルの詳細と、デバッグする場合にサポートされるログ レベルを示します。
- 付録 C の 「追加設定」では、複数の WebFort サーバ インスタンス、および Java SDK と WebFort サーバ間の SSL をセットアップする方法について詳しく説明します。
- 付録 D の 「SDK の例外とエラー コード」では、発行 SDK と認証 SDK によって返される例外およびエラー コードの一覧を示します。
- 付録 E の 「WebFort サンプル アプリケーション」では、WebFort に付属のサンプル アプリケーションで実例が示されている WebFort のワークフローについて説明します。
- 付録 F の 「用語集」では、このガイドで使用されている主な用語の一覧を示します。

関連するドキュメント

その他の関連するドキュメントは以下のとおりです。

ドキュメント	説明
Arcot WebFort 6.2 インストールおよび展開ガイド	WebFort のインストールおよび設定に必要な情報について説明します。
Arcot WebFort 6.2 クイック インストール ガイド	WebFort をインストールする際に実行する必要があるタスクの概要について説明します。
Arcot WebFort 6.2 管理ガイド	WebFort の管理および設定に関する情報が記載されています。
ArcotID Client 6.0.2 リファレンス ガイド	クライアントで使用できる ArcotID Client タイプについて説明します。

表記規則と形式

ここでは、本書の表記規則、定型見出し、およびサポート窓口について説明します。

表記法

このマニュアルでは、以下の表記法を使用します。

斜体	強調、ガイド名
太字	ユーザ入力、GUI 画面テキスト
固定幅フォント	ファイル名およびディレクトリ名、拡張、コマンド プロンプト、CLI テキスト、コード
固定幅フォント 太字	パス内のターゲット ファイルまたはディレクトリ名
固定幅フォント 斜体	ユーザによって異なる場合があるファイル名またはディレクトリ名
リンク	ガイド内のリンク、URL リンク

形式

このマニュアルでは、次の形式で特別なメッセージを強調します。

注： 重要性が高い、または注目すべき情報を強調します。

ヒント：時間またはリソースを節約できる手順を強調します。

警告： このような注記を無視すると、誤作動や機器の障害につながるおそれがあります。

重要 : 操作を実行する前に知るべき情報です。

注意 : 想定される危険について注意を促します。

関連文書 : 参照すべきその他のガイドを紹介します。

サポートへのお問い合わせ

サポートが必要な場合は、次のアルコットのサポートにお問い合わせください。

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

この章では、WebFort Java SDK によって提供される API、および Java SDK を使用する前に行う必要がある確認作業について説明します。

WebFort Java SDK の概要

WebFort SDK（Software Development Kit）は、ユーザのアプリケーションと統合するための一連の API を含むプログラム インターフェースを提供します。SDK には以下の 2 種類があります。

認証 SDK

WebFort 認証 SDK は、WebFort でサポートされているクレデンシャルを認証するために使用できる API を提供します。

発行 SDK

WebFort 発行 SDK は、WebFort サーバと通信して、WebFort データベース内のユーザおよびクレデンシャル情報の作成、読み取り、更新を行います。発行 SDK を使用して、以下の操作を実行できます。

- ユーザの作成
- ユーザのクレデンシャルの作成
- ユーザ情報の取得
- ユーザ情報の更新
- クレデンシャルのライフサイクル管理操作（有効化、無効化、リセット、有効期限のリセット、削除など）の実行

WebFort SDK の機能

このセクションでは、認証 SDK と発行 SDK の主要機能について説明します。

- **SSL のサポート**

  SSL（Secure Socket Layer）を使用して、Java SDK と WebFort サーバ間の接続を保護できます。SDK と WebFort サーバ間に SSL を設定するには、WebFort のプロパティ ファイルを編集する必要があります。詳しい方法については、「SSL の設定」を参照してください。

- **フェイルオーバ**

  Java SDK はフェイルオーバ メカニズムをサポートします。WebFort サーバのインスタンスが操作不可能になった場合は、SDK が追加で設定されたインスタンスのいずれかに自動的に接続します。詳しい方法については、「複数の WebFort サーバ インスタンスの設定」を参照してください。

- **複数の SDK 初期化方法**

  プロパティ ファイルまたはマップを使用して認証 SDK または発行 SDK を初期化できます。詳しい方法については、「発行 SDK の初期化」または「認証 SDK の初期化」を参照してください。

- **1 つの関数を使った複数操作の処理**

  クレデンシャルのライフサイクル操作を異なる複数の認証情報に対して同時に実行できます。たとえば、1 つの `create()` 関数を使って ArcotID、質問と回答、およびワンタイム パスワード クレデンシャルを同時に作成できます。

- **追加パラメータのサポート**

  API は、必須の入力に加えて、名前 - 値ペアとして渡すことができる追加の入力を受け付けます。この入力には、ロケール、呼び出し側のアプリケーションの詳細、プロファイルなどの情報を含めることができます。

- **カスタム API のサポート**

  カスタム API では、WebFort のネイティブ認証方式のいずれかを並行してサポートすることにより、追加の認証方式をサポートできます。カスタム API の詳細については、第 7 章の「カスタム API の使用」を参照してください。

事前準備

WebFort API を使ってアプリケーションを WebFort と統合するためのコードを作成する前に、以下の点を確認してください。

- 必要なオペレーティング システム上に WebFort VAS® がインストールされ、実行されていること。

関連文書 : インストールと設定の詳細については、「Arcot WebFort 6.2 インストールおよび展開ガイド」を参照してください。

- WebFort Java SDK の使用に必要な正しいバージョンの JDK がインストールされていること。詳細については、「Arcot WebFort 6.2 インストールおよび展開ガイド」を参照してください。

第 2 章
WebFort ワークフローについて

WebFort では、認証 SDK および発行 SDK で構築可能なさまざまなワークフローを設計できます。ほとんどの場合、従来のオンライン操作を大幅に変えることなく、組織の要件に基づいたワークフローを設計できます。

注： この章で説明するタスクは、複数の方法でカスタマイズできます。ここでは一般的なワークフローの例で説明しますので、この章で説明した各手順に厳密に従う必要はありません。

この章では、サンプルのワークフローと以下の概要について説明します。

- 既存ユーザの移行
- ArcotID 認証ワークフロー
- ArcotID ローミング ダウンロード ワークフロー
- パスワードを忘れたときのワークフロー
- ワークフローのサマリ

既存ユーザの移行

WebFort では、容易に既存の認証方式から ArcotID 認証にユーザを移行できます　。

注： ディレクトリ サービス（LDAP）を使用している場合、WebFort を LDAP に接続し、WebFort でサポートされる属性に LDAP 属性をマッピングする必要があります。詳細については、「Arcot WebFort 6.2 管理ガイド」を参照してください。

- すべてのユーザの移行
- 選択したユーザの移行

すべてのユーザの移行

すべてのユーザを移行する一般的な手順は次のとおりです。

1. ユーザがアプリケーションにログインします。

   ユーザは既存の認証方式を使用して、アプリケーションにログインします。

2. アプリケーションは、クレデンシャルの作成に必要な情報をユーザから収集します。

   アプリケーションでは、適切なページをユーザに表示できます。たとえば、ArcotID のパスワードを設定するようにユーザに促したり、既存のパスワードを ArcotID パスワードとして設定したり、Q&A（質問と回答）をセカンダリ認証に使用する場合の質問と回答を収集したりできます。

3. アプリケーションは、`ArcotIDIssuance` クラスの `create()` メソッドを呼び出します。

   アプリケーションは、`ArcotIDIssuance` クラスの `create()` メソッドを呼び出して、ユーザの ArcotID を作成します。

4. WebFort は結果を返します。

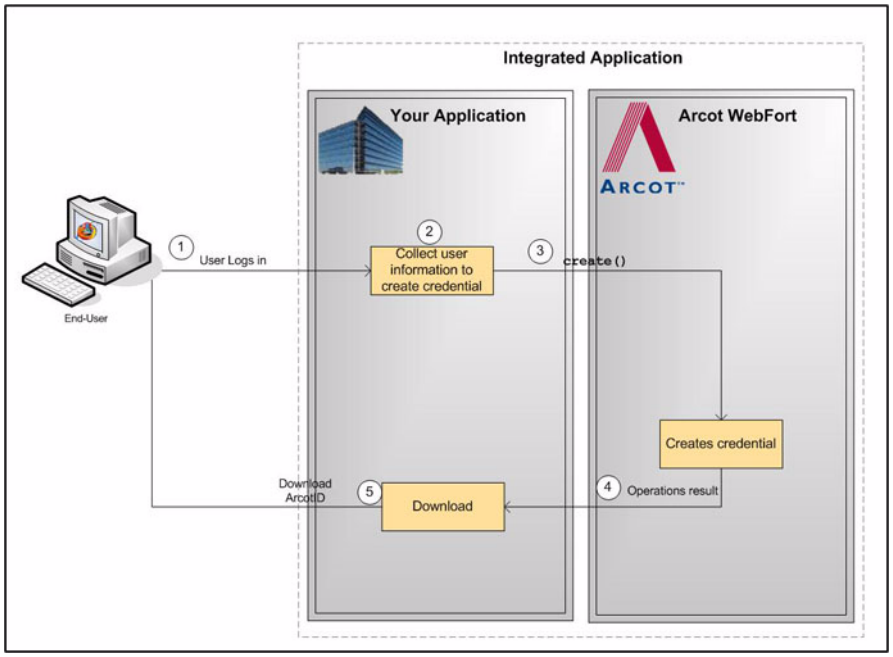
   WebFort は create 操作の結果をアプリケーションに通知します。

5. アプリケーションはユーザのシステムに ArcotID をダウンロードします。

   `create()` 関数が成功した場合、アプリケーションはユーザによる操作なしでエンドユーザのシステムに ArcotID をダウンロードします。

図 2-1 に、システム内のすべてのユーザを移行する場合のワークフローを示します。

図 2-1 すべてのユーザの移行



選択したユーザの移行

選択したユーザを移行する一般的な手順は次のとおりです。

1. ユーザがアプリケーションにログインします。

   ユーザは既存の認証方式を使用して、アプリケーションにログインします。

2. アプリケーションはユーザ ステータスを取得します。

   アプリケーションはユーザ情報を取得し、ユーザ アカウントが移行の対象としてマークされているかどうかを識別します。

3. アプリケーションはユーザをリダイレクトします。

   認証に成功すると、ユーザは移行ページにリダイレクトされます。

4. アプリケーションは、クレデンシャルの作成に必要な情報をユーザから収集します。

アプリケーションでは、適切なページをユーザに表示できます。たとえば、ArcotID のパスワードを設定するようにユーザに促したり、既存のパスワードを ArcotID パスワードとして設定したり、Q&A をセカンダリ認証に使用する場合の質問と回答を収集したりできます。

5. アプリケーションは、`ArcotIDIssuance` クラスの `create()` メソッドを呼び出します。

   アプリケーションは、`ArcotIDIssuance` クラスの `create()` メソッドを呼び出して、ユーザの ArcotID を作成します。

6. WebFort は結果を返します。

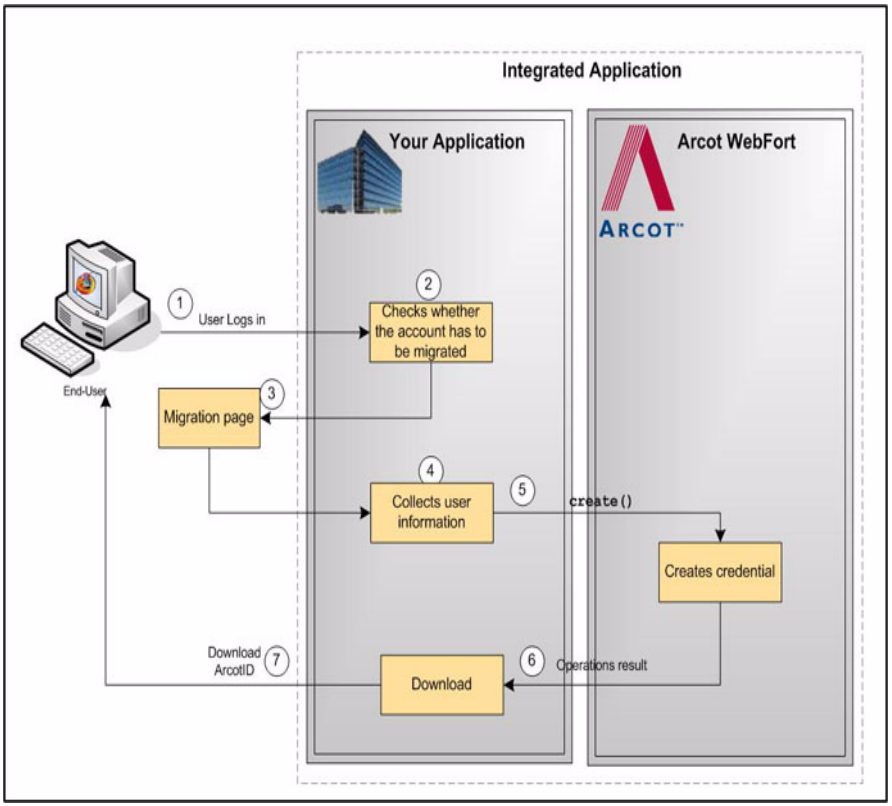
   WebFort は create 操作の結果をアプリケーションに通知します。

7. アプリケーションはユーザのシステムに ArcotID をダウンロードします。

   `create()` 関数が成功した場合、アプリケーションはユーザによる操作なしでエンドユーザのシステムに ArcotID をダウンロードします。

図 2-2 に、一括でユーザを ArcotID 認証に移行するためのワークフローを示します。

図 2-2 選択したユーザの移行



ArcotID 認証ワークフロー

認証時に認証ページでユーザがクレデンシャルを指定すると、クレデンシャルは最初に WebFort サーバで確認され、その後ユーザが認証されます。

ユーザ名とパスワード、Q&A、および OTP は単純なパスワード ベースの認証方式ですが、ArcotID はチャレンジ / レスポンス タイプの認証です。以下のワークフローに、ArcotID 認証の手順を示します。

注： その他のクレデンシャルを使用する場合に呼び出すメソッドの詳細については、第 6 章の「ユーザの認証」を参照してください。

1. アプリケーションは WebFort の `ArcotIDAuth.getChallenge()` 関数をコールします。

   アプリケーションは ArcotID Client をロードし、`ArcotIDAuth` インターフェースの `getChallenge()` 関数を明示的にコールします。この API の詳細については、6-74 ページの「ArcotID 認証」を参照してください。

2. ユーザはクレデンシャルを指定します。

   ユーザはログインに使用するユーザ名と ArcotID パスワードを指定します。

3. アプリケーションは ArcotID Client を呼び出します。

   ArcotID Client はチャレンジに署名します。

4. WebFort は署名されたチャレンジを確認します。

   アプリケーションは `verifySignedChallenge()` 関数を呼び出し、ArcotID パスワードで署名されたチャレンジを確認します。

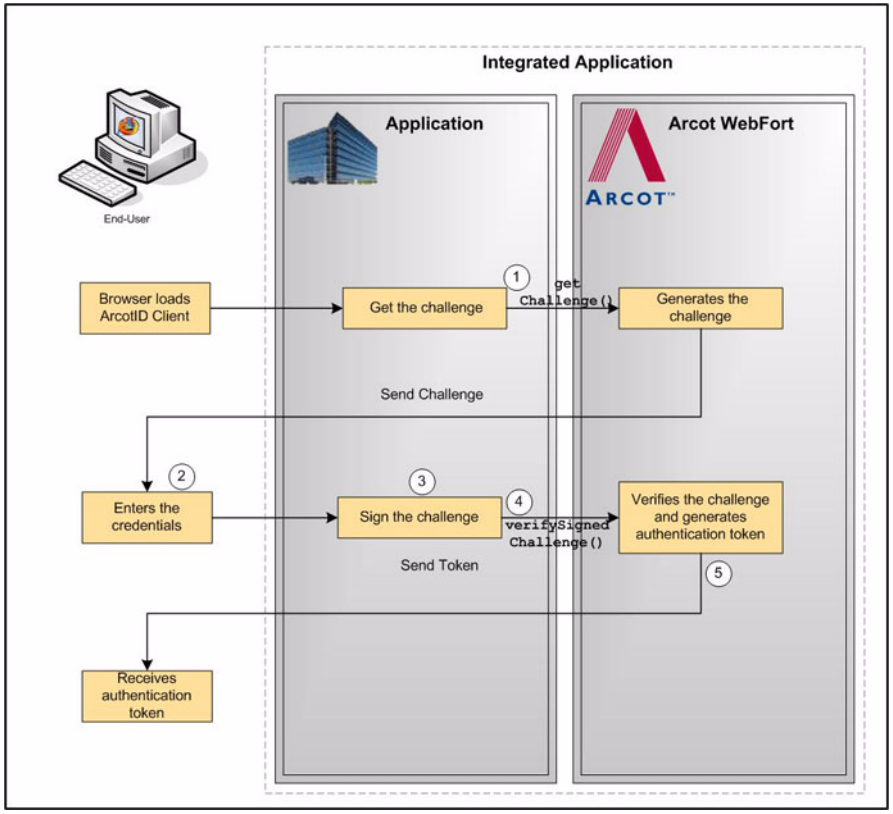
5. WebFort はユーザを認証します。

   `verifySignedChallenge()` コールが成功すると、認証トークンが生成され、ユーザが正常に認証されます。

   WebFort でサポートされるさまざまなトークンの詳細については、6-71 ページの「認証 SDK の初期化」を参照してください。

   図 2-3 に、ArcotID 認証プロセスのワークフローを示します。

図 2-3 ArcotID 認証



ArcotID ローミング ダウンロード ワークフロー

ArcotID 認証を実行するには、現在の認証セッションで使用されるユーザのシステムに、ユーザの ArcotID が存在している必要があります。ユーザが出張している場合や ArcotID が保存されているプライマリ システムへのアクセス権を持っていない場合は、WebFort サーバから ArcotID をダウンロードして認証を実行する必要があります。別のシステムに ArcotID をダウンロードするこのプロセスは、ローミング ダウンロードと呼ばれます。

ArcotID のローミング ダウンロード手順は次のとおりです。

1. オンライン アプリケーションにログインします。

   アプリケーションがユーザを認証します。

2. ユーザが ArcotID のダウンロードを選択します。

   ArcotID をダウンロードできる適切なページが表示されます。

3. WebFort はセカンダリ認証を実行します。

   使用しているセカンダリ認証メカニズムに基づき、適切なページがユーザに表示されます。たとえば、次の作業をユーザに促すことができます。

   - アプリケーションへの登録時にユーザが選択した、セキュリティの質問に答える。
   - 電子メール、SMS、またはその他のカスタマイズされた方法でユーザに送信される OTP（ワンタイム パスワード）を入力する。

4. アプリケーションは WebFort の `ArcotIDAuth.getArcotID()` 関数をコールします。

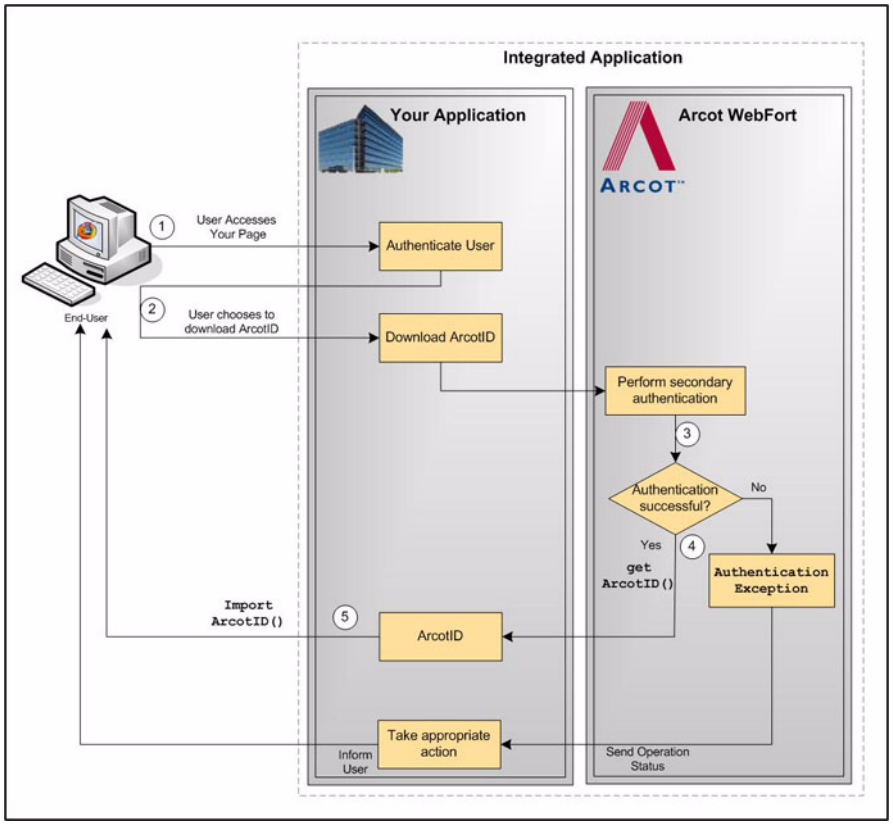
   セカンダリ認証に成功した場合、アプリケーションが次に実行するのは `ArcotIDAuth` インターフェースの `getArcotID()` 関数のコールのみです。このコールにより、対応する ArcotID（Base-64 エンコード形式）をアプリケーションにダウンロードします。

5. ユーザのシステムに ArcotID をダウンロードします。

   `ImportArcotID()` クライアント側 API を呼び出し、ユーザによる操作なしでエンドユーザのシステムに ArcotID をダウンロードします。

図 2-4 に、ArcotID のローミング ダウンロードのワークフローを示します。

図 2-4 ArcotID のローミング ダウンロード



パスワードを忘れたときのワークフロー

ユーザが ArcotID パスワードを忘れた場合、FYP （Forgot Your Password）ワークフローを使用してパスワードをリセットできます。

このとき、ユーザは登録時に設定した質問に回答するように促されます。または、登録時に選択したその他のカスタマイズされた方法を使用できます。

FYP ワークフローの一般的な手順は次のとおりです。

1. ユーザ名を指定します。

ログインするユーザ名を指定します。

2. ［FYP］リンクをクリックします。

   ユーザはパスワードを覚えていないため、［FYP］リンクをクリックします。

3. WebFort はセカンダリ認証を実行します。

   使用しているセカンダリ認証メカニズムに基づき、適切なページがユーザに表示されます。たとえば、次の作業をユーザに促すことができます。

   - アプリケーションへの登録時にユーザが選択した、セキュリティの質問に答える。
   - 電子メール、SMS、またはその他のカスタマイズされた方法でユーザに送信される OTP（ワンタイム パスワード）を入力する。

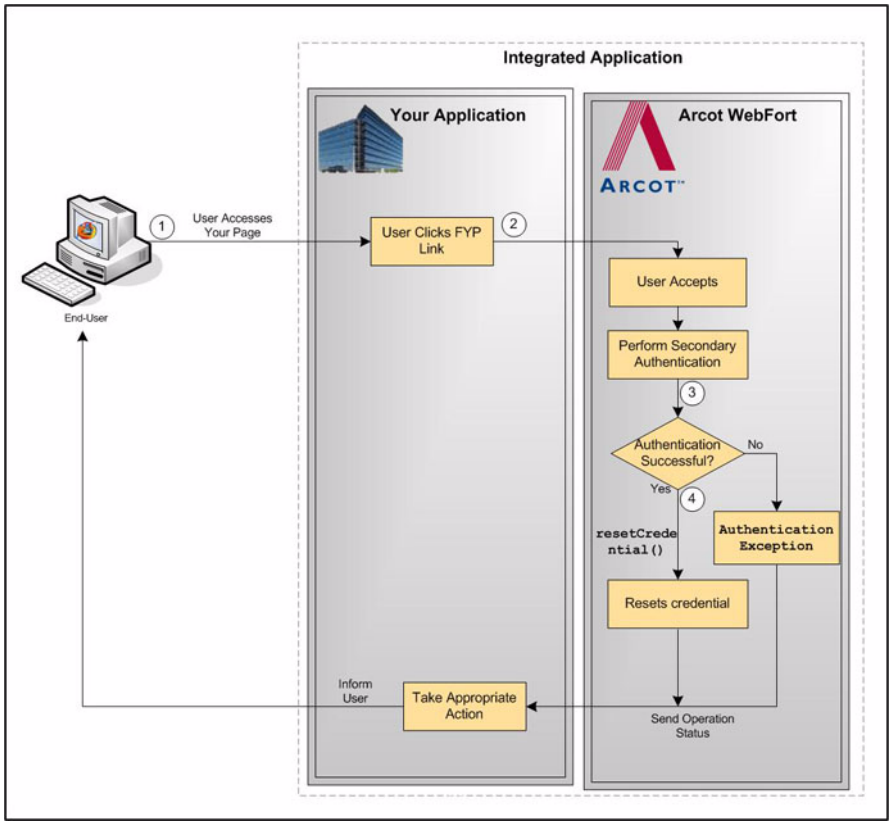
4. アプリケーションは `CredentialIssuance` インターフェースの WebFort の `resetCredential()` 関数をコールします。

   セカンダリ認証に成功した場合、アプリケーションは `CredentialIssuance` インターフェースの `resetCredential()` 関数を呼び出す必要があります。アプリケーションは、新しいパスワードの入力をユーザに促し、このパスワードを入力として `resetCredential()` 関数に渡します。

   クレデンシャルのリセットに使用する API の詳細については、「クレデンシャルのリセット」を参照してください。

図 2-5 に、パスワードを忘れたときのワークフローを示します。

図 2-5 FYP ワークフロー



ワークフローのサマリ

表 2-1 に、WebFort API を使用して実装できるワークフローの概要を示します。

表 2-1. WebFort ワークフローの概要

ワークフロー	説明	依存するワークフロー
登録	`UserIssuance()` クラスの `create()` 関数をコールすると、WebFort データベースに新しいユーザが作成されます。	なし
クレデンシャルの作成	ユーザのクレデンシャルを作成します。	• 登録
認証	ユーザが指定したクレデンシャルを使用してユーザを認証します。	• 登録 • クレデンシャルの作成
ArcotID ダウンロード	ユーザの ArcotID をシステムにダウンロードします。	• 登録 • クレデンシャルの作成
移行	ArcotID 認証にユーザを移行します。	なし
FYP	パスワードをリセットします。	• 登録 • クレデンシャルの作成

第 3 章
SDK を使用する前に

認証 SDK および発行 SDK は、オンライン アプリケーションをプログラム的に WebFort と統合するための API のセットを構成しています。WebFort Java SDK は、以下のコンポーネントで構成されています。

- Authentication Java クラス
- Issuance Java クラス
- 関連する Java クラスおよびメソッドの Javadoc

注： WebFort に付属しているサンプル アプリケーションで、Java クラスおよびメソッドの使用方法の実例が示されています。WebFort サンプル アプリケーションの詳細については、付録 E の 「WebFort サンプル アプリケーション」 を参照してください。

発行 SDK または認証 SDK を使用する前に、関連する JAR ファイルを `CLASSPATH` で指定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、`CLASSPATH` でそれらのファイルを指定する必要があります。

この章では、アプリケーションから発行および認証 API メソッドをコールするために実行する必要がある手順について説明します。

WebFort SDK Javadoc へのアクセス

認証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統合するには、WebFort SDK に付属の Javadoc を参照してください。

すでに WebFort に統合されているアプリケーションを更新する場合、廃止された Java API についてリリース ノートで確認してから、アプリケーションを変更する必要があります。

認証 SDK Javadoc（Arcot-WebFort-6.2-authentication-sdk-javadocs.zip）および発行 SDK Javadoc（Arcot-WebFort-6.2-issuance-sdk-javadocs.zip）は、以下の場所にあります。

Windows の場合

<*install_location*>\Arcot Systems\docs\webfort

UNIX プラットフォームの場合

<*install_location*>/arcot/docs/webfort

CLASSPATH への認証ファイルの追加

認証 API を使用するには、CLASSPATH に API の JAR ファイル（表 3-1 を参照）および webfort.authentication.properties ファイルを追加し、プロパティ ファイルが CLASSPATH/properties フォルダ内に存在している必要があります。

JAR ファイル

認証 SDK に必要な JAR ファイルは、以下の場所にあります。

Windows の場合

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

UNIX プラットフォームの場合

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

表 3-1. 認証 SDK の JAR

ファイル名	説明
arcot/arcot-pool.jar	commons プールを使用して接続プールを実装する場合に使用します。
arcot/arcot-webfort-common.jar	認証 SDK によって使用される共有コンポーネントのセットを含む、専用の JAR（Java Archive）ファイル。
arcot/arcot-webfort-authentication.jar	認証 API が含まれます。

表 3-1. 認証 SDK の JAR

ファイル名	説明
external/bcprov-jdk14-139.jar	暗号化機能に使用します。たとえば、サーバの PEM 証明書を読み取ります。
external/commons-httpclient-3.1.jar	HTTP ベースの通信に使用します。
external/commons-lang-2.4.jar	認証 SDK によって使用される文字列および例外処理のユーティリティが含まれています。
external/commons-pool-1.4.jar	接続プールに使用します。
external/log4j-1.2.9.jar	アプリケーションの動作をログ記録するランタイムを制御するための Apache パッケージです。

プロパティ ファイル

WebFort サーバの接続性を含む webfort.authentication.properties ファイルは、認証 SDK の初期化に使用されます。また、ユーザは init API（「認証 SDK の初期化」を参照）を使用して認証 API を初期化できます。これは入力パラメータとして 1 組の名前と値のペアを受け取ります。

注： また、webfort.authentication.properties からその他のファイルに WebFort サーバ接続性パラメータをコピーし、コピー先のファイルを使用して SDK を初期化できます。

プロパティ ファイルは以下の場所にあります。

Windows の場合

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

UNIX プラットフォームの場合

<install_location>/arcot/sdk/java/properties

注： 追加の WebFort サーバ インスタンスを設定するため、または SSL 対応のためにプロパティ ファイルを編集する方法については、付録 C の 「追加設定」を参照してください。

CLASSPATH への発行ファイルの追加

発行 API を使用するには、CLASSPATH に API の JAR ファイル（表 3-2 を参照）および webfort.issuance.properties ファイルを追加し、プロパティ ファイルが CLASSPATH/properties フォルダ内に存在している必要があります。

JAR ファイル

発行 SDK に必要な JAR ファイルは、以下の場所にあります。

Windows の場合

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

UNIX プラットフォームの場合

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

表 3-2. 発行 SDK の JAR

ファイル名	説明
arcot/arcot-pool.jar	commons プールを使用して接続プールを実装する場合に使用します。
arcot/arcot-webfort-common.jar	発行 SDK によって使用される共有コンポーネントのセットを含む、専用の JAR（Java Archive）ファイル。
arcot/arcot-webfort-issuance.jar	発行 API を含む JAR ファイル。

表 3-2. 発行 SDK の JAR

ファイル名	説明
• external/activation-1.1.jar • external/axiom-api-1.2.7.jar • external/axiom-impl-1.2.7.jar • external/axis2-adb-1.4.jar • external/axis2-java2wsdl-1.4.jar • external/axis2-kernel-1.4.jar • external/backport-util-concurrent-2.2.jar • external/bcprov-jdk14-139.jar • external/commons-codec-1.3.jar • external/commons-collections-3.1.jar • external/commons-httpclient-3.1.jar • external/commons-lang-2.4.jar • external/commons-logging-1.1.jar • external/commons-pool-1.4.jar • external/geronimo-jms_1.1_spec-1.1.jar • external/log4j-1.2.9.jar • external/neethi-2.0.jar • external/stax-api-1.0.1.jar • external/wsdl4j-1.6.2.jar • external/wstx-asl-3.2.0.jar • external/XmlSchema-1.2.jar	Axis JAR ファイルで使用されます。

プロパティ ファイル

WebFort サーバの接続性を含む webfort.issuance.properties ファイルは、発行 SDK の初期化に使用されます。また、ユーザは init API（4-23 ページの「発行 SDK の初期化」を参照）を使用して発行 API を初期化できます。これは入力パラメータとして 1 組の名前と値のペアを受け取ります。

注： また、webfort.issuance.properties からその他のファイルに WebFort サーバ接続性パラメータをコピーし、コピー先のファイルを使用して SDK を初期化できます。

プロパティ ファイルは以下の場所にあります。

Windows の場合

`<install_location>\Arcot Systems\sdk\java\properties`

UNIX プラットフォームの場合

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

注： 追加の WebFort サーバ インスタンスを設定するため、または SSL 対応のためにプロパティ ファイルを編集する方法については、付録 C の 「追加設定」を参照してください。

第 4 章
発行操作の実行

WebFort でユーザを認証するには、各ユーザのアカウントをデータベースに作成する必要があります。これは、1 回限りのプロセスです。ユーザを Arcot データベースに作成するか、LDAP に接続してユーザ情報を取得するように WebFort を設定できます。

重要 : WebFort の展開から LDAP のユーザ情報にアクセスする場合、ユーザの詳細情報を読み取る API のみを使用できます。

この章では、以下のようなユーザ操作やクレデンシャル操作で使用する API について説明します。

- 発行 SDK の初期化
- ユーザ操作
- クレデンシャル操作

発行 SDK の初期化

`com.arcot.webfort.issuance.api` パッケージの `Issuance` クラスを使用して発行 SDK を初期化します。初期化後、呼び出し元のアプリケーションに適切なメッセージが返されます。

`Issuance` クラスには、発行 SDK を初期化するメソッドが 2 つあります。

メソッド 1：マップを使用して SDK を初期化する

このメソッドは、指定したマップに基づいて発行アプリケーションを初期化します。表 4-1 で、この init() メソッドについて詳しく説明します。

表 4-1. マップを使用した SDK の初期化

説明	入力値	出力値
指定したマップを使用して発行 SDK を初期化します。	• map 設定情報を指定するキーと値のペア。以下のキーを指定できます。 1.issuance.host.1 WebFort サーバが稼働しているシステムの IP アドレス。 2.issuance.port.1 Transaction Web Services プロトコルがリスニングするポート。デフォルト値は 9744 です。 • locale API のロケール。デフォルト値は en_US に設定されています。	SDK が正常に初期化されない場合は例外を返します。

メソッド 2：プロパティ ファイルを使用して SDK を初期化する

このメソッドは、プロパティ ファイル内に指定されているパラメータを使用して発行 SDK を初期化します。NULL を渡した場合は、webfort.issuance.properties ファイルからパラメータが読み取られます。該当する設定パラメータが記述された別のファイル名を指定した場合は、そのファイルから読み取られます。

webfort.issuance.properties ファイル内のパラメータ（transport、host、port）は、マップの該当パラメータと同じです。表 4-2 で、この init() メソッドについて詳しく説明します。

表 4-2. プロパティ ファイルを使用した SDK の初期化

説明	入力値	出力値
プロパティ ファイルを使用して発行 SDK を初期化します。	• location プロパティ ファイルの絶対パス。 • locale API のロケール。デフォルト値は en_US に設定されています。	SDK が正常に初期化されない場合は例外を返します。

発行 SDK リソースの解放

Issuance クラスには、発行 SDK が使用するソケットなどのリソースを解放するためのメソッドも用意されています。

重要 : SDK を再初期化する場合は、事前にこのメソッドを呼び出す必要があります。

表 4-3 で、release() メソッドについて詳しく説明します。

表 4-3. API の解放

説明	入力値	出力値
発行 SDK を解放します。	API の locale（ロケール）。	API が正常に解放されない場合は例外を返します。

ユーザ操作

ここでは、以下のユーザ操作について説明します。

- ユーザ入力の準備
- 追加入力の準備
- ユーザの作成
- ユーザ アカウントの無効化
- ユーザの有効化
- ユーザの詳細情報の読み取り
- ユーザ詳細情報の更新
- ユーザ操作のまとめ

ユーザ入力の準備

WebFort サーバに送信する発行リクエストはすべて、WebFort サーバで処理できる形式で準備する必要があります。ユーザ操作に先立ってデータを準備するには、UserInput クラスを使用します。このクラスを使用して、ユーザ操作中に使用される情報を設定できます。

実行する操作に応じて、入力データの準備には以下の作業が該当します。

- 名の設定
- 姓の設定
- ミドル ネームの設定
- 電子メール ID の設定
- 電話番号の設定
- PAM の設定
- カスタム属性の追加

名の設定

ユーザの名を設定するには、UserInput クラスの setFirstName() メソッドを呼び出します。

姓の設定

ユーザの姓を設定するには、UserInput クラスの setLastName() メソッドを呼び出します。

ミドル ネームの設定

ユーザのミドル ネームを設定するには、UserInput クラスの setMiddleName() メソッドを呼び出します。

電子メール ID の設定

ユーザの電子メール ID を設定するには、UserInput クラスの setEmailId() メソッドを呼び出します。

電話番号の設定

ユーザの電話番号を設定するには、UserInput クラスの setTelephoneNumber() メソッドを呼び出します。

PAM の設定

WebFort データベースでユーザの PAM（Personal Assurance Message）を設定するには、UserInput クラスの setPAM() メソッドを呼び出します。

PAM はクライアントに対してサーバを検証するメッセージで、WebFort で保護されたリソースにユーザがアクセスしようとすると表示されます。

カスタム属性の追加

`setCustomAttribute()` メソッドを使用して、ユーザに関するカスタム属性を定義します。

これらの属性は、`UserInput` クラスで提供される標準属性に加えて使用されます。

注： WebFort サーバは、ユーザから送信される入力データをすべて検証します。この入力データの検証に WebFort サーバで使用する基準の詳細については、付録 A の 「入力データの検証」を参照してください。

追加入力の準備

以下の場合は、追加の情報入力に備える必要があります。

- コールアウトまたはプラグインを実装して、WebFort の標準のユーザ発行機能の強化を図る場合。
- 完全に新しい独自のユーザ発行方式を記述する場合。

上記いずれの場合も、WebFort サーバに送信する必要がある追加の情報を、名前 - 値のペアの形式で設定する必要があります。WebFort の `com.arcot.webfort.common.api` パッケージには、こうした補足情報を指定するための `AdditionalInput` クラスが用意されています。

`AdditionalInput` クラスで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を一部以下に紹介します。

- `AR_WF_LOCALE_ID`

  呼び出し元アプリケーションにメッセージを返す際に WebFort で使用するロケールを指定します。

- `AR_WF_CALLER_ID`

  トランザクションの追跡に役立ちます。セッション ID またはトランザクション ID を使用して呼び出し元情報を指定できます。

ユーザの作成

WebFort データベースでユーザを作成するには、以下の手順に従って `UserIssuance` インターフェースを使用する必要があります。

1. `UserInput` クラスを使用して、ユーザの情報を設定するメソッドを取得します。

2. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

3. `UserIssuance` インターフェースの `create()` メソッドを呼び出して、ユーザ アカウントを作成します。

   このメソッドは、ユーザおよびトランザクションの詳細情報を指定する、`UserResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

ユーザ アカウントの無効化

ユーザ アカウントは、指定された期間無効にすることができます。たとえば、組織の従業員が長期休暇を取った場合、該当ユーザのアカウントを無効にして、指定した期間中の不正アクセスを防止できます。アカウントが無効になっているユーザは、クレデンシャル操作を一切実行できません。

ユーザ アカウントを無効にする方法

1. `UserInput` クラスを使用して、ユーザの情報を設定するメソッドを取得します。
2. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

3. `UserIssuance` インターフェースの `disable()` メソッドを呼び出して、ユーザ アカウントを無効にします。

このメソッドは、ユーザおよびトランザクションの詳細情報を指定する、`UserResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

ユーザの有効化

`enable()` メソッドを使用して、以前に無効化したユーザ アカウントをアクティブにします。

注： 有効化できるのは、すでに無効化されているユーザ アカウントのみです。

ユーザを有効にする方法

1. `UserInput` クラスを使用して、ユーザの情報を設定するメソッドを取得します。
2. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
3. `UserIssuance` インターフェースの `enable()` メソッドを呼び出して、ユーザ アカウントを有効化します。

   このメソッドは、ユーザおよびトランザクションの詳細情報を指定する、`UserResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

ユーザの詳細情報の読み取り

WebFort データベースからユーザの詳細情報を取得するには、`UserIssuance` インターフェースを使用する必要があります。

ユーザの詳細情報を読み取る方法

1. `UserInput` クラスを使用して、ユーザの情報を設定するメソッドを取得します。
2. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
3. `UserIssuance` インターフェースの `fetch()` メソッドを呼び出して、ユーザの詳細情報を読み取ります。

   このメソッドは、ユーザおよびトランザクションの詳細情報を指定する、`UserResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

ユーザ詳細情報の更新

発行 API には、データベースでユーザ アカウントの詳細情報を変更する `update()` メソッドがあります。たとえば、ユーザの電子メール アドレスや連絡先番号といった詳細情報を変更できます。

注 : ユーザ名またはユーザが所属する組織の名前は変更 *できません*。

ユーザの詳細情報を更新する方法

1. `UserInput` クラスを使用して、ユーザの情報を設定するメソッドを取得します。
2. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
3. `UserIssuance` インターフェースの `update()` メソッドを呼び出して、ユーザの詳細情報を更新します。

   このメソッドは、ユーザおよびトランザクションの詳細情報を指定する、`UserResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

ユーザ操作のまとめ

表 4-4 に、この章で説明したすべてのユーザ操作の実行に必要な入力パラメータをまとめます。

表 4-4. ユーザ操作

操作	必要な入力	出力
作成 更新	• ユーザ名（userName） •（オプション）組織名（orgName） 注：組織名が未指定の場合、ユーザはデフォルトの組織に所属しているものとみなされます。 •（オプション）追加入力（AdditionalInput） •（オプション）Personal Assurance Message（PAM） •（オプション）ユーザ属性（firstName、lastName、middleName、telephoneNumber、emailId） •（オプション）定義されているカスタム属性	• UserResponse
無効化 有効化 取得	• ユーザ名（userName） •（オプション）組織名（orgName） 注：組織名が未指定の場合、ユーザはデフォルトの組織に所属しているものとみなされます。 •（オプション）追加入力（AdditionalInput）	

クレデンシャル操作

ここでは、発行 API でサポートされているクレデンシャル ライフサイクル操作について説明します。この章で説明する操作は、WebFort でサポートされているすべてのクレデンシャルに対して、以下の方法のいずれかで実行できます。

- **WebFort SDK の使用**

  このモードでは、クレデンシャルの管理操作をプログラムによって自動化できます。

- **WebFort Administration Console の使用**

  Administration Console は Web ベースのアプリケーションです。主として、ユーザからのリクエスト（クレデンシャルの無効化や有効化、クレデンシャルの有効期限のリセットなど）に対応する CSR（テクニカル サポート担当者）用のツールです。

関連文書： Administration Console の使用に関する詳細については、「Arcot WebFort 6.2 管理ガイド」を参照してください。

ここでは、以下のクレデンシャル ライフサイクル操作について説明します。

- 入力の準備
- ユーザのステータスの確認
- クレデンシャルの作成
- クレデンシャルの無効化
- クレデンシャルの有効化
- クレデンシャルのリセット
- クレデンシャル詳細情報の取得
- クレデンシャルの再発行
- クレデンシャルの有効期間のリセット
- カスタム属性のリセット
- 質問数の取得
- クレデンシャルの削除
- 無署名属性の設定
- 無署名属性の削除
- 出力の読み取り
- クレデンシャルのステータスの確認
- クレデンシャル操作のまとめ

注： この章で説明する操作はそれぞれ、別々のクレデンシャルに対して同時に実行できます。操作が単一のクレデンシャルに対して失敗した場合、他のクレデンシャルに対する操作も無効とみなされます。たとえば、ArcotID、Q&A、および OTP の作成時に、ArcotID と OTP は正常に作成されたものの Q&A を作成できなかった場合、3 つのクレデンシャルをすべて再作成する必要があります。

入力の準備

このインターフェースおよびサブインターフェースについては、以下の入力の準備が必要です。

- 共通の入力
- クレデンシャル固有の入力

共通の入力

`CredentialInput` インターフェースでは、すべてのクレデンシャル タイプに共通の設定を行うことができます。このインターフェースを使用して以下の情報が設定されます。

- 有効期間
- カスタム属性
- プロファイル名
- 無効期間

有効期間

クレデンシャルの作成時に、発行 API を使用してクレデンシャルの有効期間を設定できます。暦上の特定の日付を渡す場合は `CredentialInput` クラスの `setValidity()` メソッドを呼び出すか、または ArcotDate.Type クラスを使用して有効日を設定する場合は `setValidityEx()` クラスを使用します。

クレデンシャルの有効期間は、`create()` メソッドや `resetValidity()` メソッドへの入力となります。

カスタム属性

発行 API を使用して、各クレデンシャル タイプのカスタム属性を追加できます。この機能により、クレデンシャルに関する補足情報を維持できます。たとえば、ユーザに 6 つ以上のシステム上で ArcotID をダウンロードさせたくない場合に、その旨を指定した属性を作成できます。この属性は、`create()` メソッドや `resetNotes()` メソッドへの入力となります。

カスタム属性を追加するには、`CredentialInput` クラスの `setNote()` メソッドを呼び出します。

プロファイル名

通常、同じクレデンシャル情報一式を多数のユーザに適用できます。そのような場合、ユーザごとにクレデンシャルを個別に入力する手間を省くため、共通の情報をすべて設定したプロファイルを 1 つ作成し、このプロファイルを複数のユーザ間で共有できます。各プロファイルは一意のプロファイル名で識別されます。

発行 API により、クレデンシャルのプロファイル名を設定できます。プロファイル名を設定するには、`CredentialInput` クラスの `setProfileName()` メソッドを呼び出します。

注 : プロファイルが設定されていない場合は、クレデンシャル用のデフォルトプロファイルが使用されます。

無効期間

ユーザが休暇または長期休暇中は、該当ユーザのクレデンシャルをその期間に限り無効にし、所定の期間経過後に自動的に有効にすることができます。この機能によってクレデンシャルの有効化処理が合理化され、ユーザが User Administrator （UA）にリクエストを行う必要がなくなります。

発行 API を使用して、クレデンシャルの無効期間を設定できます。無効期間を設定するには、`CredentialInput` クラスの `setDisableStartTime()` メソッドおよび `setDisableEndTime()` メソッドを呼び出します。`setDisableStartTime()` メソッドおよび `setDisableEndTime()` メソッドは、無効期間の設定に ArcotDate.Type クラスを使用します。

ArcotDate.Type クラス

`ArcotDate.Type` クラスでは、以下の日付形式を使用して、有効期間や無効期間の開始日と終了日を設定できます。

- **現在の日付**

  WebFort サーバの現在の日付を使用して、有効期間または無効期間を設定します。

- **無期限**

  クレデンシャルが永久に有効で、期限切れにならないものと指定します。

- **相対日付**

  無効期間の開始日に対応する相対的な日付を使用します。たとえば、相対日付を 1 か月とした場合、無効期間の終了日は開始日の 1 か月後になります。

- **特定の日付**

  指定された日付を使用して有効期間または無効期間を設定します。

クレデンシャル固有の入力

`com.arcot.webfort.issuance.api` パッケージには、サポート対象の以下のクレデンシャルに固有の情報を設定するために使用できるインターフェースがあります。

- ArcotID の入力の準備
- Q&A の入力の準備
- ユーザ名 / パスワード認証の入力の準備
- OATH OTP の入力の準備

ArcotID の入力の準備

ArcotID に関する以下の入力は、`ArcotIDInput` クラスを使用して設定できます。

1. **無署名属性**

   ユーザの ArcotID を作成した後で ArcotID の属性を定義できます。これらの属性（名前 - 値のペア）は ArcotID の無署名の部分に設定されるため、無署名属性と呼ばれます。

   注： すでに存在する属性を追加した場合は、既存の属性が新しい値で上書きされます。

   無署名属性を設定する方法

   a. `ArcotIDInput` クラスを使用して、ArcotID の情報を設定するメソッドを取得します。

   b. `ArcotIDAttribute` クラスを使用して、ArcotID に設定する無署名属性を定義します。

   c. `ArcotIDInput` クラスの `setUnsignedAttributes` メソッドを呼び出します。

2. **パスワード**

   ArcotID 用のパスワードを設定するか、または現在の ArcotID パスワードを変更するには、`setPassword` メソッドを使用する必要があります。ArcotID パスワードを設定するには、以下の手順に従います。

a. `ArcotIDInput` クラスを使用して、ArcotID の情報を設定するメソッドを取得します。

b. `ArcotIDInpu` クラスの `setPassword` メソッドを呼び出します。

3. **ArcotID 属性**

`ArcotIDResponse` に ArcotID 属性を取得するには、`setFetchAttributeFlag()` フラグを有効にする必要があります。ArcotID 属性を取得するには、以下の手順に従います。

a. `ArcotIDInput` クラスを使用して、ArcotID の情報を設定するメソッドを取得します。

b. `ArcotIDInput` クラスの `setFetchAttribute` メソッドを呼び出します。

Q&A の入力の準備

`QnAInput` クラスを使用して、Q&A 認証用の質問と回答を設定する必要があります。質問と回答を追加するには、以下の手順に従います。

1. `QnAInput` クラスを使用して、Q&A の情報を設定するメソッドを取得します。
2. `QnAInput` クラスの `setQuestionAnswer` メソッドを呼び出します。

ユーザ名 / パスワード認証の入力の準備

ユーザ名 / パスワード認証用のパスワードは `UPInput` クラスを使用して設定します。パスワードを設定するには、以下の手順に従います。

1. `UPInput` クラスを使用して、ユーザ名 / パスワードの情報を設定するメソッドを取得します。
2. `UPInput` クラスの `setPassword` メソッドを呼び出します。

OATH OTP の入力の準備

OATH OTP の発行に使用するトークン ID を設定するには、`OATHOTPInput` クラスを使用する必要があります。トークン ID を設定するには、以下の手順に従います。

1. `OATHOTPInput` クラスを使用して、OTP のトークン ID を設定するメソッドを取得します。
2. `OATHOTPInput` クラスの `setTokenID` メソッドを呼び出します。

ArcotOTP の入力の準備

ArcotOTP の生成に使用されるパスワードを設定または変更するには、setPassword メソッドを使用する必要があります。ArcotID パスワードを設定するには、以下の手順に従います。

1. ArcotOTPInput クラスを使用して、ArcotID の情報を設定するメソッドを取得します。
2. ArcotOTPInput クラスの setPassword メソッドを呼び出します。

ユーザのステータスの確認

WebFort は、クレデンシャル操作の一部を実行する前にユーザ ステータス情報を使用します。データベース内のユーザ ステータスは ACTIVE または DISABLED のいずれかになります。

発行 SDK でこれらの確認を実行するには、Administration Console のクレデンシャル プロファイル設定ページで、該当オプションを有効にする必要があります。表 4-5 に、すべてのクレデンシャル操作と、操作の種類に応じて実行されるユーザ チェックをまとめます。

表 4-5. ユーザ ステータスの確認

状態 / 操作	ユーザの存在	ユーザ ステータス	ユーザ属性
作成	実行する	実行する	実行する
削除	実行しない	実行しない	実行しない
無効化	実行しない	実行しない	実行しない
有効化	実行する	実行する	実行しない
取得	実行しない	実行しない	実行しない
再発行	実行する	実行する	実行しない
リセット	実行する	実行する	実行しない
有効期間のリセット	実行する	実行する	実行しない
無署名属性の削除	実行しない	実行しない	実行しない
無署名属性の設定	実行しない	実行しない	実行しない

クレデンシャルの作成

com.arcot.webfort.issuance.api パッケージには、ユーザのクレデンシャルを作成するメソッドを含む CredentialIssuance インターフェースがあります。

クレデンシャルを作成する方法

1. 作成するクレデンシャルのタイプに応じた < *クレデンシャル名* >Input クラスを使用して、このクラスを実装するオブジェクトを取得します。

   必要な入力はクレデンシャルごとに異なります。たとえば、ユーザ名 / パスワード認証と ArcotID 認証にはパスワードが必要ですが、Q&A クレデンシャルの場合は質問と対応する回答が必要となります。

   注 : 各種クレデンシャルで必要となる入力の詳細については、「クレデンシャル操作のまとめ」を参照してください。

2. CredentialInput 抽象クラスを使用して、クレデンシャルの共通情報を設定するメソッドを取得します。
3. CredentialInputList クラスを呼び出して、各種クレデンシャルの入力クラスを渡します。
4. (**オプション**) コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、AdditionalInput に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

5. CredentialIssuance インターフェースの create() メソッドを呼び出して、クレデンシャルを作成します。

   このメソッドは、すべてのクレデンシャルおよびトランザクションの詳細情報を指定する、CredentialResponse インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。com.arcot.webfort.issuance.api.exception パッケージおよび com.arcot.webfort.common.api.exception パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

クレデンシャルの無効化

ユーザのクレデンシャルは、指定された期間無効にすることができます。たとえば、従業員が長期休暇を取る場合に、該当ユーザのクレデンシャルを無効にして、ユーザ不在時の不正アクセスを防止できます。

クレデンシャルを無効にする方法

1. 無効にするクレデンシャルのタイプに応じた < *クレデンシャル名* >Input クラスを使用して、このクラスを実装するオブジェクトを取得します。

   必要な入力はクレデンシャルごとに異なります。たとえば、ユーザ名 / パスワード認証と ArcotID 認証にはパスワードが必要ですが、Q&A クレデンシャルの場合は質問と対応する回答が必要となります。

   注 : 各種クレデンシャルで必要となる入力の詳細については、「クレデンシャル操作のまとめ」を参照してください。

2. CredentialInput 抽象クラスを使用して、クレデンシャルの共通情報を設定するメソッドを取得します。
3. CredentialInputList クラスを呼び出して、各種クレデンシャルの入力クラスを渡します。
4. (**オプション**) コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、AdditionalInput に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

5. CredentialIssuance インターフェースの disable() メソッドを呼び出して、クレデンシャルを無効にします。

   このメソッドは、すべてのクレデンシャルおよびトランザクションの詳細情報を指定する、CredentialResponse インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。com.arcot.webfort.issuance.api.exception パッケージおよび com.arcot.webfort.common.api.exception パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

クレデンシャルの有効化

`enable` メソッドを使用して、無効化またはロックされたユーザのクレデンシャルをアクティブにします。たとえば、ユーザが認証時に間違ったクレデンシャルを使用した場合や、設定されている最大試行回数を超過した場合に、クレデンシャルが無効化またはロックされることがあります。

クレデンシャルを有効にする方法

1. 有効にするクレデンシャルのタイプに応じた < *クレデンシャル名* >`Input` クラスを使用して、このクラスを実装するオブジェクトを取得します。

   必要な入力はクレデンシャルごとに異なります。たとえば、ユーザ名 / パスワード認証と ArcotID 認証にはパスワードが必要ですが、Q&A クレデンシャルの場合は質問と対応する回答が必要となります。

   注 : 各種クレデンシャルで必要となる入力の詳細については、「クレデンシャル操作のまとめ」を参照してください。

2. `CredentialInput` 抽象クラスを使用して、クレデンシャルの共通情報を設定するメソッドを取得します。
3. `CredentialInputList` クラスを呼び出して、各種クレデンシャルの入力クラスを渡します。
4. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

5. `CredentialIssuance` インターフェースの `enable()` メソッドを呼び出して、クレデンシャルを有効にします。

   このメソッドは、すべてのクレデンシャルおよびトランザクションの詳細情報を指定する、`CredentialResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

クレデンシャルのリセット

発行 API を使用して、クレデンシャルをリセットできます。たとえば、ArcotID パスワードまたは質問と回答をリセットできます。

クレデンシャルをリセットする方法

1. リセットするクレデンシャルのタイプに応じた < *クレデンシャル名* >`Input` クラスを使用して、このクラスを実装するオブジェクトを取得します。

   必要な入力はクレデンシャルごとに異なります。たとえば、ユーザ名 / パスワード認証と ArcotID 認証にはパスワードが必要ですが、Q&A クレデンシャルの場合は質問と対応する回答が必要となります。

   注 : 各種クレデンシャルで必要となる入力の詳細については、「クレデンシャル操作のまとめ」を参照してください。

2. `CredentialInput` 抽象クラスを使用して、クレデンシャルの共通情報を設定するメソッドを取得します。
3. `CredentialInputList` クラスを呼び出して、各種クレデンシャルの入力クラスを渡します。
4. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

5. `CredentialIssuance` インターフェースの `resetCredential()` メソッドを呼び出して、クレデンシャルをリセットします。

このメソッドは、すべてのクレデンシャルおよびトランザクションの詳細情報を指定する、`CredentialResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

クレデンシャル詳細情報の取得

ユーザのクレデンシャルの詳細情報を読み取るには、`fetch()` メソッドを実装する必要があります。

ユーザのクレデンシャル情報を読み取る方法

1. 詳細情報を取得する必要があるクレデンシャルのタイプに応じた < *クレデンシャル名* >`Input` クラスを使用して、このクラスを実装するオブジェクトを取得します。

   必要な入力はクレデンシャルごとに異なります。たとえば、ユーザ名 / パスワード認証と ArcotID 認証にはパスワードが必要ですが、Q&A クレデンシャルの場合は質問と対応する回答が必要となります。

   注 : 各種クレデンシャルで必要となる入力の詳細については、「クレデンシャル操作のまとめ」を参照してください。

2. `CredentialInput` 抽象クラスを使用して、クレデンシャルの共通情報を設定するメソッドを取得します。
3. `CredentialInputList` クラスを呼び出して、各種クレデンシャルの入力クラスを渡します。
4. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

5. `CredentialIssuance` インターフェースの `fetch()` メソッドを呼び出して、クレデンシャルの詳細情報を読み取ります。

   このメソッドは、すべてのクレデンシャルおよびトランザクションの詳細情報を指定する、`CredentialResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

クレデンシャルの再発行

発行 API を使用して、ユーザのクレデンシャルを再作成できます。クレデンシャルがユーザに対して再発行された場合、該当ユーザは以前のクレデンシャルを使用してログインすることはできません。

クレデンシャルを再発行する方法

1. 再発行するクレデンシャルのタイプに応じた < *クレデンシャル名* >`Input` クラスを使用して、このクラスを実装するオブジェクトを取得します。

   必要な入力はクレデンシャルごとに異なります。たとえば、ユーザ名 / パスワード認証と ArcotID 認証にはパスワードが必要ですが、Q&A クレデンシャルの場合は質問と対応する回答が必要となります。

   注 : 各種クレデンシャルで必要となる入力の詳細については、「クレデンシャル操作のまとめ」を参照してください。

2. `CredentialInput` 抽象クラスを使用して、クレデンシャルの共通情報を設定するメソッドを取得します。

3. `CredentialInputList` クラスを呼び出して、各種クレデンシャルの入力クラスを渡します。
4. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
5. `CredentialIssuance` インターフェースの `reissue()` メソッドを呼び出して、クレデンシャルを再作成します。

   このメソッドは、すべてのクレデンシャルおよびトランザクションの詳細情報を指定する、`CredentialResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

クレデンシャルの有効期間のリセット

発行済みのクレデンシャルは、作成時に指定された期間中有効です。`CredentialIssuance` インターフェースの `resetValidity()` メソッドを使用して、クレデンシャルの有効期限が切れる前に有効期間をリセットできます。このメソッドは、クレデンシャルの有効期間を延長または短縮するために使用されますが、パスワードやその他のクレデンシャル属性はリセット *しません*。

クレデンシャルの有効期間をリセットする方法

1. リセットするクレデンシャルのタイプに応じた < *クレデンシャル名* >`Input` クラスを使用して、このクラスを実装するオブジェクトを取得します。

必要な入力はクレデンシャルごとに異なります。たとえば、ユーザ名 / パスワード認証と ArcotID 認証にはパスワードが必要ですが、Q&A クレデンシャルの場合は質問と対応する回答が必要となります。

注： 各種クレデンシャルで必要となる入力の詳細については、「クレデンシャル操作のまとめ」を参照してください。

2. `CredentialInput` 抽象クラスを使用して、クレデンシャルの共通情報を設定するメソッドを取得します。
3. `CredentialInputList` クラスを呼び出して、各種クレデンシャルの入力クラスを渡します。
4. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
5. `CredentialIssuance` インターフェースの `resetValidity()` メソッドを呼び出して、クレデンシャルの有効期間をリセットします。

   このメソッドは、すべてのクレデンシャルおよびトランザクションの詳細情報を指定する、`CredentialResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

カスタム属性のリセット

クレデンシャルに関連付けられたカスタム属性をリセットできます。`CredentialIssuance` インターフェースの `resetNotes()` メソッドを使用して、クレデンシャルの属性をリセットできます。

クレデンシャルのカスタム属性をリセットする方法

1. 属性をリセットするクレデンシャルのタイプに応じた < *クレデンシャル名* >Input クラスを使用して、このクラスを実装するオブジェクトを取得します。

   必要な入力はクレデンシャルごとに異なります。たとえば、ユーザ名 / パスワード認証と ArcotID 認証にはパスワードが必要ですが、Q&A クレデンシャルの場合は質問と対応する回答が必要となります。

   注 : 各種クレデンシャルで必要となる入力の詳細については、「クレデンシャル操作のまとめ」を参照してください。

2. CredentialInput 抽象クラスを使用して、クレデンシャルの共通情報を設定するメソッドを取得します。
3. CredentialInputList クラスを呼び出して、各種クレデンシャルの入力クラスを渡します。
4. (**オプション**) コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、AdditionalInput に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

5. CredentialIssuance インターフェースの resetNotes() メソッドを呼び出して、カスタム属性をリセットします。

   このメソッドは、すべてのクレデンシャルおよびトランザクションの詳細情報を指定する、CredentialResponse インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。com.arcot.webfort.issuance.api.exception パッケージおよび com.arcot.webfort.common.api.exception パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

質問数の取得

Q&A 認証用にユーザが設定する必要がある質問の数は、組織ごとに異なる場合があります。ユーザが設定する必要がある質問の数を取得するには、以下の手順に従います。

1. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

2. `CredentialIssuance` インターフェースの `fetchQnAConfiguration()` メソッドを呼び出して、質問の数を取得します。

   このメソッドは、すべてのクレデンシャルおよびトランザクションの詳細情報を指定する、`CredentialResponse` インターフェースのインスタンスを返します。

クレデンシャルの削除

ユーザのクレデンシャルを削除する方法

1. 削除するクレデンシャルのタイプに応じた < *クレデンシャル名* >`Input` クラスを使用して、このクラスを実装するオブジェクトを取得します。

   必要な入力はクレデンシャルごとに異なります。たとえば、ユーザ名 / パスワード認証と ArcotID 認証にはパスワードが必要ですが、Q&A クレデンシャルの場合は質問と対応する回答が必要となります。

   注： 各種クレデンシャルで必要となる入力の詳細については、「クレデンシャル操作のまとめ」を参照してください。

2. `CredentialInput` 抽象クラスを使用して、クレデンシャルの共通情報を設定するメソッドを取得します。

3. `CredentialInputList` クラスを呼び出して、各種クレデンシャルの入力クラスを渡します。

4. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

5. `CredentialIssuance` インターフェースの `delete()` メソッドを呼び出して、クレデンシャルを削除します。

   このメソッドは、Q&A 設定の詳細情報を指定する、`ConfigurationResponse` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

無署名属性の設定

ユーザの ArcotID の無署名属性を設定するには、`setArcotIDUnsignedAttributes()` メソッドを実装する必要があります。

	**注：** この操作は ArcotID クレデンシャルにのみ適用されます。

1. `ArcotIDAttributes` クラスを使用して、ArcotID の無署名属性を設定します。
2. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

3. `CredentialIssuance` インターフェースの `setArcotIDUnsignedAttributes()` メソッドを呼び出して、ArcotID の無署名属性を設定します。

   このメソッドは、トランザクション ID、メッセージ、応答コード、および理由コードを指定する、`TransactionDetails` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

無署名属性の削除

ユーザの ArcotID の無署名属性を削除するには、`deleteArcotIDUnsignedAttributes()` メソッドを実装する必要があります。

注： この操作は ArcotID クレデンシャルにのみ適用されます。

ArcotID の無署名属性を削除するには、以下の手順に従います。

1. （**オプション**）コールアウト、プラグイン、または発行操作用の他のカスタム メソッドを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、`AdditionalInput` に情報を格納します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

2. `CredentialIssuance` インターフェースの `deleteArcotIDUnsignedAttributes()` メソッドを呼び出して、ArcotID の無署名属性を削除します。

   このメソッドは、トランザクション ID、メッセージ、応答コード、および理由コードを指定する、`TransactionDetails` インターフェースのインスタンスを返します。

エラー処理

発行 API のメソッドを実行中にエラーが発生した場合は、例外が返されます。`com.arcot.webfort.issuance.api.exception` パッケージおよび `com.arcot.webfort.common.api.exception` パッケージにこれらの例外が含まれます。例外クラスの詳細については、「発行例外」および「共通例外」を参照してください。

例外がスローされない場合は、処理は正常に終了しています。戻り値のオブジェクトを参照して結果を処理できます。エラーが不在だからといって、リクエストが正常に処理されたとは限りません。

出力の読み取り

表 4-6 に、クレデンシャルおよびユーザの詳細情報を取得するメソッドをまとめます。

注： 以下の表に記載されたメソッドの大半は、NULL を返す場合もあります。

表 4-6. 出力メソッド

メソッド	説明
共通の出力メソッド	
getCreateTime()	クレデンシャルが作成された時刻を取得します。
getDisableEndTime()	無効化されたクレデンシャルが有効化される時刻を取得します。
getDisableStartTime()	クレデンシャルが無効化される時刻を取得します。
getLastFailedAuthAttemptTime()	認証試行が前回失敗した時刻を取得します。
getLastSuccessAuthAttemptTime()	認証試行が前回正常終了した時刻を取得します。
getLastUpdatedTime()	クレデンシャルの最終更新時刻を取得します。
getNotes()	クレデンシャルに対して設定されているカスタム属性を取得します。
getNumberOfFailedAuthAttempts()	ユーザの認証試行の失敗総数を取得します。
getOrgName()	ユーザの所属組織名を取得します。
getProfileName()	クレデンシャルの作成に使用されたプロファイル名を取得します。
getProfileVersion()	プロファイルのバージョン番号を取得します。
getStatus()	クレデンシャルのステータスを取得します。
getUserName()	認証ユーザの名前を取得します。
getValidityEndTime()	クレデンシャルの有効期限が切れる日付を取得します。
getValidityStartTime()	クレデンシャルの発効日を取得します。
ArcotID の出力メソッド	
getUnsignedAttributes()	ユーザが設定した ArcotID の無署名属性を取得します。

表 4-6. 出力メソッド

メソッド	説明
Q&A の出力メソッド	
`getQuestions()`	ユーザに対して設定された質問を取得します。
ユーザ名 / パスワード認証の出力メソッド	
この認証方式で利用できる出力メソッドはありません。	
ワンタイム パスワードの出力メソッド	
`getOTP()`	ユーザのワンタイム パスワード（OTP）を取得します。
`getUsageCount()`	OTP を使用できる回数を取得します。
`OATH` ワンタイム パスワードの出力メソッド	
`getCounterOffSet()`	サーバ上の OATH OTP カウントを取得します。
`getLength()`	OATH OTP の長さを取得します。
`getTokenID()`	OATH OTP トークン ID を取得します。
`getType()`	ユーザに発行される OATH OTP のタイプ（HOTP または TOTP）を取得します。
`ArcotOTP` の出力メソッド	
`getCard()`	WebFort サーバによって生成される OTP カードを取得します。
`getCounterOffSet()`	サーバ上の ArcotOTP カウントを取得します。
`getType()`	ArcotOTP のタイプを取得します。
ユーザの出力メソッド	
`getCustomAttribute`	指定されたユーザ属性の値を取得します。
`getCustomAttributeMap()`	マップ内のユーザ属性を取得します。キーは属性名、値は対応する属性値です。
`getEmailId()`	ユーザの電子メール ID を取得します。
`getFirstName()`	ユーザの名を取得します。
`getLastName()`	ユーザの姓を取得します。
`getMiddleName()`	ユーザのミドル ネームを取得します。
`getOrgName()`	ユーザが所属する組織の名前を取得します。
`getPAM()`	ユーザの PAM（Personal Assurance Message ）を取得します。
`getStatus()`	ユーザの状態を取得します。
`getTelephoneNumber()`	ユーザの電話番号を取得します。
`getUserName()`	ユーザの名前を取得します。

表 4-6. 出力メソッド

メソッド	説明
カスタム クレデンシャルの出力メソッド	
getCreateTime()	カスタム クレデンシャルが作成された時刻を取得します。
getLastFailedAuthAttemptTime()	認証試行が前回失敗した時刻を取得します。
getLastSuccessAuthAttemptTime()	認証試行が前回正常終了した時刻を取得します。
getLastUpdatedTime()	カスタム クレデンシャルの最終更新時刻を取得します。
getNumberOfFailedAuthAttempts()	ユーザの認証試行の失敗総数を取得します。
getOrgName()	ユーザの所属組織名を取得します。
getStatus()	カスタム クレデンシャルのステータスを取得します。
getUserName()	認証ユーザの名前を取得します。
getValidityEndTime()	カスタム クレデンシャルの有効期限が切れる日付を取得します。
getValidityStartTime()	カスタム クレデンシャルの発効日を取得します。

クレデンシャルのステータスの確認

クレデンシャルのステータスを確認するには、com.arcot.webfort.common.api パッケージを使用します。表 4-7 に、さまざまなクレデンシャル ステータスをまとめます。

表 4-7. クレデンシャルのステータス

ステータス	説明
ACTIVE	クレデンシャルはアクティブで、認証に使用できます。
DISABLED	クレデンシャルは管理者によって無効化されています。
LOCKED	ユーザが認証に連続して失敗し、設定されている最大試行回数に達したため、クレデンシャルはロックされています。たとえば、最大試行回数が 3 に設定されている場合、3 回目の認証で間違ったクレデンシャルを使用すると、クレデンシャルがロックされます。
VERIFIED	ユーザが提示した OTP が WebFort サーバによって正常に認証されると、クレデンシャルが検証済みになります。 **注：** このステータスは OTP にのみ適用されます。
DELETED	ユーザのクレデンシャルがデータベースから削除されました。

状態遷移

表 4-8 に、クレデンシャルの状態と、状態間の遷移の可否をまとめます。

表 4-8. クレデンシャル状態の遷移

現在の状態 / 変更後の状態	Enabled	Locked	Disabled	Deleted	Verified
Enabled	可	可	可	否	可
Locked	可	可	否	否	否
Disabled	可	可	可	否	可
Deleted	可	可	可	可	可
Verified	可	否	否	否	否

クレデンシャルの操作および状態

表 4-9 は、すべてのクレデンシャル操作と、各操作を特定の状態のクレデンシャルに対して実行可能かどうかを示しています。クレデンシャルの状態が操作後に変化する場合、表には変化後のクレデンシャルの状態も記載されています。

注： 「*可能*」は、操作を実行できることを示しますが、クレデンシャルの状態は操作の後に変化しません。

表 4-9. クレデンシャルの操作および状態

状態 / 操作	Enabled	Locked	Disabled	Deleted	Verified（OTP *のみ*）
作成	不可	不可	不可	可能 -> Enabled	不可
有効化	可能 -> Enabled	可能 -> Enabled	可能 -> Enabled	不可	不可
無効化	可能 -> Disabled	可能 -> Disabled	可能 -> Disabled	不可	不可
取得	可能	可能	可能	可能	可能
リセット	可能 -> Enabled	可能 -> Enabled	可能 -> Enabled	不可	可能 -> Enabled

表 4-9. クレデンシャルの操作および状態

状態 操作	Enabled	Locked	Disabled	Deleted	Verified（OTP のみ）
有効期間のリセット	可能	可能	可能	不可	可能
再発行	可能 -> Enabled	可能 -> Enabled	可能 -> Enabled	不可	可能 -> Enabled
削除	可能 -> Deleted	可能 -> Deleted	可能 -> Deleted	可能 -> Deleted	可能 -> Deleted
無署名属性の削除（ArcotID のみ）	可能	可能	可能	不可	可能
無署名属性の設定（ArcotID のみ）	可能	可能	可能	不可	可能

クレデンシャル操作のまとめ

ここでは、各クレデンシャルのライフサイクル管理操作の実行に必要な入力パラメータ、および以下の操作で予期される出力について説明します。

- ArcotID 操作
- ユーザ名 / パスワード操作
- 質問と回答操作
- ワンタイム パスワード操作

ArcotID 操作

表 4-10 に、ArcotID 操作用の入出力情報をまとめます。

表 4-10. ArcotID 操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの作成 (create())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • ArcotID パスワード（password）。 • （オプション）ArcotID 属性（signedAttributes および unsignedAttributes）。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。渡さない場合は、組織のデフォルト プロファイルが使用されます。渡す場合は、指定したプロファイル名が組織で利用可能である必要があります。 • （オプション）アプリケーションの各クレデンシャルについて管理しているカスタム属性（noteName、noteValue）。	• CredentialOutput • TransactionDetails
クレデンシャルのリセット (resetCredentail())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。	• CredentialOutput • TransactionDetails
クレデンシャルの取得 (fetch())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	• CredentialOutput • TransactionDetails
クレデンシャルの再発行 (reissue())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • ArcotID パスワード（password）。 • （オプション）ArcotID 属性（signedAttributes および unsignedAttributes）。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。	• CredentialOutput • TransactionDetails

表 4-10. ArcotID 操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの有効期間のリセット (`resetValidity()`)	• （オプション）ユーザ名（`userName`）。 • （オプション）組織名（`orgName`）。 • （オプション）追加入力（`AdditionalInput`）。 • クレデンシャルの有効期間が終了する時刻（`validityEndTime`）。	• `CredentialOutput` • `TransactionDetails`
カスタム属性のリセット (`resetNotes()`)	• （オプション）ユーザ名（`userName`）。 • （オプション）組織名（`orgName`）。 • （オプション）追加入力（`AdditionalInput`）。 • アプリケーションで管理している属性（`noteName`、`noteValue`）。	• `CredentialOutput` • `TransactionDetails`
クレデンシャルの無効化 (`disable()`)	• （オプション）ユーザ名（`userName`）。 • （オプション）組織名（`orgName`）。 • （オプション）追加入力（`AdditionalInput`）。	• `CredentialOutput` • `TransactionDetails`
クレデンシャルの有効化 (`enable()`)		
クレデンシャルの削除 (`delete()`)		
ArcotID の無署名属性の設定 (`setArcotIDUnsignedAttributes()`)	• （オプション）ユーザ名（`userName`）。 • （オプション）組織名（`orgName`）。 • （オプション）追加入力（`AdditionalInput`）。 • ArcotID 属性（`signedAttributes` および `unsignedAttributes`）。	`TransactionDetails`
無署名属性の削除 (`deleteArcotIDUnsignedAttributes()`)	• （オプション）ユーザ名（`userName`）。 • （オプション）組織名（`orgName`）。 • （オプション）追加入力（`AdditionalInput`）。 • ArcotID の無署名属性の配列。	`TransactionDetails`

ユーザ名 / パスワード操作

表 4-11 に、ユーザ名 / パスワード操作用の入出力情報をまとめます。

表 4-11. ユーザ名 / パスワード操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの作成 (create())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • パスワード（password）。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。渡さない場合は、組織のデフォルト プロファイルが使用されます。渡す場合は、指定したプロファイル名が組織で利用可能である必要があります。 • （オプション）アプリケーションの各クレデンシャルについて管理している属性（noteName、noteValue）。	CredentialResponse
クレデンシャルのリセット (resetCredentail())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。	CredentialResponse
クレデンシャルの取得 (fetch())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルの再発行 (reissue())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • パスワード（password）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルの有効期間のリセット (resetValidity())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • クレデンシャルの有効期間が終了する時刻（validityEndTime）。	CredentialResponse
カスタム属性のリセット (resetNotes())	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • アプリケーションで管理している属性（noteName、noteValue）。	CredentialResponse

表 4-11. ユーザ名 / パスワード操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの無効化（disable()）	• （オプション）ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルの有効化（enable()）		
クレデンシャルの削除（delete()）		

質問と回答操作

表 4-12 に、Q&A 操作用の入出力情報をまとめます。

表 4-12. Q&A 操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの作成（create()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • 質問と回答（question および answer）のリスト。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。渡さない場合は、組織のデフォルト プロファイルが使用されます。渡す場合は、指定したプロファイル名が組織で利用可能である必要があります。 • （オプション）アプリケーションの各クレデンシャルについて管理している属性（noteName、noteValue）。	CredentialResponse
クレデンシャルのリセット（resetCredentail()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • 質問と回答（question および answer）のリスト。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。渡さない場合は、組織のデフォルト プロファイルが使用されます。渡す場合は、指定したプロファイル名が組織で利用可能である必要があります。	CredentialResponse
クレデンシャルの取得（fetch()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	• CredentialResponse • questions

表 4-12. Q&A 操作

操作 （使用する関数）	必要な入力	出力
クレデンシャル設定の取得（fetchQnAConfiguration()）	• （オプション）組織名（orgName）。 • （オプション）クレデンシャルのプロファイル名。渡さない場合は、組織のデフォルト プロファイルが使用されます。渡す場合は、指定したプロファイル名が組織で利用可能である必要があります。 • （オプション）追加入力（AdditionalInput）。	• ConfigurationResponse • プロファイルで設定されている質問
クレデンシャル設定の取得（fetchQnAConfiguration()）	• （オプション）組織名（orgName）。 • （オプション）クレデンシャルのプロファイル名。渡さない場合は、組織のデフォルト プロファイルが使用されます。渡す場合は、指定したプロファイル名が組織で利用可能である必要があります。 • （オプション）追加入力（AdditionalInput）。 •	• ConfigurationResponse • questions
クレデンシャルの再発行（reissue()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • 質問と回答（question および answer）のリスト。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルの有効期間のリセット（resetValidity()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • クレデンシャルの有効期間が終了する時刻（validityEndTime）。	CredentialResponse
カスタム属性のリセット（resetNotes()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • アプリケーションで管理している属性（noteName、noteValue）。	CredentialResponse
クレデンシャルの無効化（disable()） クレデンシャルの有効化（enable()） クレデンシャルの削除（delete()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse

ワンタイム パスワード操作

表 4-13 に、OTP 操作用の入出力情報をまとめます。

表 4-13. ワンタイム パスワード操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの作成（create()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。渡さない場合は、組織のデフォルト プロファイルが使用されます。渡す場合は、指定したプロファイル名が組織で利用可能である必要があります。 • （オプション）アプリケーションの各クレデンシャルについて管理している属性（noteName、noteValue）。	• CredentialResponse • Password
クレデンシャルの取得（fetch()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルのリセット（resetCredential()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	• CredentialResponse • Password
クレデンシャルの有効期間のリセット（resetValidity()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • クレデンシャルの有効期間が終了する時刻（validityEndTime）。	CredentialResponse
カスタム属性のリセット（resetNotes()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • アプリケーションで管理している属性（noteName、noteValue）。	CredentialResponse

表 4-13. ワンタイム パスワード操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの無効化 (disable())	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルの有効化 (enable())		
クレデンシャルの削除 (delete())		

OATH OTP 操作

表 4-14 に、OATH OTP 操作用の入出力情報をまとめます。

表 4-14. OATH OTP 操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの作成（create()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。渡さない場合は、組織のデフォルト プロファイルが使用されます。渡す場合は、指定したプロファイル名が組織で利用可能である必要があります。 • （オプション）アプリケーションの各クレデンシャルについて管理している属性 （noteName、noteValue）。	• CredentialResponse • Password
クレデンシャルの取得（fetch()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルのリセット (resetCredentail())	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	• CredentialResponse • Password

表 4-14. OATH OTP 操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの有効期間のリセット（resetValidity()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • クレデンシャルの有効期間が終了する時刻（validityEndTime）。	CredentialResponse
カスタム属性のリセット（resetNotes()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • アプリケーションで管理している属性（noteName、noteValue）。	CredentialResponse
クレデンシャルの無効化（disable()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルの有効化（enable()）		
クレデンシャルの削除（delete()）		

ArcotOTP 操作

表 4-15 に、ArcotOTP 操作用の入出力情報をまとめます。

表 4-15. ArcotOTP 操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの作成（create()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • （オプション）クレデンシャルのプロファイル名。渡さない場合は、組織のデフォルト プロファイルが使用されます。渡す場合は、指定したプロファイル名が組織で利用可能である必要があります。 • （オプション）アプリケーションの各クレデンシャルについて管理している属性 （noteName、noteValue）。	• CredentialResponse • Password
クレデンシャルの取得（fetch()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルのリセット（resetCredential()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	• CredentialResponse • Password
クレデンシャルの有効期間のリセット（resetValidity()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • クレデンシャルの有効期間が終了する時刻（validityEndTime）。	CredentialResponse
カスタム属性のリセット（resetNotes()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。 • アプリケーションで管理している属性（noteName、noteValue）。	CredentialResponse

表 4-15. ArcotOTP 操作

操作 （使用する関数）	必要な入力	出力
クレデンシャルの無効化 （disable()）	• ユーザ名（userName）。 • （オプション）組織名（orgName）。 • （オプション）追加入力（AdditionalInput）。	CredentialResponse
クレデンシャルの有効化（enable()）		
クレデンシャルの削除（delete()）		

第 5 章
ArcotID Client とアプリケーションの統合

ArcotID Client は、WebFort サーバが提供するチャレンジに署名するためにエンド ユーザが使用するソフトウェアです。ArcotID ベースの認証を実装予定の場合は、ArcotID Authentication API を呼び出す前に、ArcotID Client とアプリケーションを統合する必要があります。この章では、各種のクライアントについて説明し、これらのクライアントをアプリケーションと統合する方法を詳説します。また、ArcotID Client で提供される API も紹介します。この章では、以下の内容について説明します。

ArcotID Client の概要

ArcotID Client は、WebFort が発行したチャレンジにユーザ側で署名するために使用されますが、ユーザの ArcotID のダウンロードを円滑に進めるうえでも有用です。幅広いエンド ユーザ環境に対応できるよう、ArcotID Client は Flash クライアントおよび署名済みの Java アプレットの形態で提供されます。クライアントの種類によって、利便性や機能性の程度は異なります。設定に際して必要となるユーザ操作や管理権限も、選択したクライアントに応じて変化します。

Flash クライアント

この実装の ArcotID Client は、Adobe Flash Player（バージョン 9 以降）がインストールされた任意の Web ブラウザで動作します。

注： ArcotID の操作に ArcotID Flash クライアントを使用する場合は、Flash クライアントにサービスを提供するアプリケーションで HTTPS を有効にしておく必要があります。

署名済みの Java アプレット

この実装の ArcotID Client は、Sun Java Runtime Environment （JRE）がインストールされた任意の Web ブラウザで実行できます。

注： Arcot 署名済み Java アプレットを使用する場合は、署名済みアプレットの呼び出し前に、同アプレットの受け入れの確認を求めるセキュリティ メッセージがユーザに対して表示されます。

ArcotID Client ファイルのコピー

ArcotID Client はエンド ユーザ システム コンポーネントです。そのため、使用予定のクライアントの種類に応じて、アプリケーションが実行されるシステム上の適切な場所に、関連ファイルのパッケージを配置する必要があります。

ここでは、アプリケーションとパッケージ化する必要があるファイルについて説明します。

- Flash クライアントの場合
- Java 署名済みアプレットの場合

Flash クライアントの場合

Flash クライアントのパッケージには以下のファイルが含まれます。

- `arcotclient.js`

  ArcotID Flash クライアント API が含まれています。

- `ArcotIDClient.swf`

  ArcotID Flash Client の実装が含まれています。

Flash クライアントを設定する方法

1. アプリケーション ホーム内の適切なディレクトリに `arcotclient.js` ファイルと `ArcotIDClient.swf` ファイルをコピーします。
2. API の呼び出し元のアプリケーションの Web ページに以下の JavaScript コードを含めます。

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

上記のコード スニペットの location_to_arcotclient.js を arcotclient.js へのパスに置き換えます。

3. すべてのアプリケーション ページで、ArcotIDClient.swf が同じ URL で参照されるようにします。

Java 署名済みアプレットの場合

Java 署名済みアプレット クライアント パッケージには以下のファイルが含まれます。

- arcotclient.js

  Java 署名済みアプレット クライアント API が含まれています。

- ArcotApplet.jar（Sun JRE）

  Java 署名済みアプレット クライアントの実装が含まれています。

Java 署名済みアプレット クライアントを設定する方法

1. アプリケーション ホーム内の適切なディレクトリに arcotclient.js ファイルと ArcotApplet.jar ファイルをコピーします。
2. アプリケーションの関連 Web ページに以下の JavaScript コードを含めます。

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

上記のコード スニペットの location_to_arcotclient.js を arcotclient.js へのパスに置き換えます。

3. すべてのアプリケーション ページで、Java アプレットが同じ URL で参照されるようにします。

ArcotID Client API

ArcotID 認証を実装する場合、以下の処理を行うにはアプリケーションを ArcotID Client API を用いて統合する必要があります。

- ArcotID のダウンロード
- チャレンジの署名

ArcotID のダウンロード

アプリケーションから ArcotID をエンド ユーザ システムにダウンロードするには、`ImportArcotID()` 関数を使用する必要があります。この関数は、ダウンロードする必要がある ArcotID を Base64 でエンコードした文字列と、ストレージ モードを入力パラメータとして取ります。

ArcotID は、現在のセッションの間一時的にダウンロードするか、または完全にダウンロードできます。このストレージ モードは、ArcotID の格納用として選択された記憶媒体によって指定されます。ArcotID は、以下の媒体に格納できます。

- ハード ディスク
- ユニバーサル シリアル バス（USB）
- メモリ

ダウンロードされた ArcotID は、拡張子 `.aid` を付けて保存されます。ArcotID ファイルの名前は、ユーザ名、組織名、およびドメイン名のハッシュ値に基づいて決まります。

チャレンジの署名

クライアント API の `SignChallengeEx()` 関数を使用して、WebFort サーバから取得したチャレンジに署名する必要があります。

関連文書： API の詳細については、「*ArcotID Client 6.0.2 リファレンス ガイド*」を参照してください。

第 6 章
ユーザの認証

この章では、WebFort でサポートされている各種の認証方式で使用する API について説明します。この章では、以下の内容について説明します。

認証 SDK の初期化

`com.arcot.webfort.authentication.api` パッケージの `Authentication` クラスを使用して認証 SDK を初期化します。この初期化プロセスでは、データベース テーブルをすべてキャッシュし、データベース プールとロガーを作成します。初期化後、呼び出し元のアプリケーションに適切なメッセージが返されます。

注： API の初期化後は、設定の変更は一切適用 *できません*。設定の変更を有効にするためには、API を再初期化する必要があります。

`Authentication` クラスには、認証 SDK を初期化するメソッドが 2 つあります。

メソッド 1：マップを使用して SDK を初期化する

このメソッドは、指定したマップに基づいて認証 SDK を初期化します。表 6-1 で、この `init()` メソッドについて詳しく説明します。

表 6-1. マップを使用した SDK の初期化

説明	入力値	出力値
指定したマップを使用して認証 SDK を初期化します。	• `map` 設定情報を指定するキーと値のペア。以下のキーを指定できます。 1. `authentication.host.1` WebFort サーバが稼働しているシステムの IP アドレス。 2. `authentication.port.1` Authentication Native プロトコルがリスニングするポート。デフォルト値は 9742 です。 • `locale` API のロケール。デフォルト値は `en_US` に設定されています。	SDK が正常に初期化されない場合は例外を返します。

メソッド 2：プロパティ ファイルを使用して SDK を初期化する

このメソッドは、プロパティ ファイル内に指定されているパラメータを使用して認証 SDK を初期化します。`NULL` を渡した場合は、`webfort.authentication.properties` ファイルからパラメータが読み取られます。該当する設定パラメータが記述された別のファイル名を指定した場合は、そのファイルから読み取られます。

webfort.authentication.properties ファイル内のパラメータ（transport、host、port）は、マップの該当パラメータと同じです。表 6-2 で、この init() メソッドについて詳しく説明します。

表 6-2. プロパティ ファイルを使用した SDK の初期化

説明	入力値	出力値
プロパティ ファイルを使用して認証 SDK を初期化します。	• location プロパティ ファイルの絶対パス。 • locale API のロケール。デフォルト値は en_US に設定されています。	SDK が正常に初期化されない場合は例外を返します。

認証 API リソースの解放

Authentication クラスには、認証 SDK が使用するソケットなどのリソースを解放するためのメソッドも用意されています。

重要 : SDK を再初期化する場合は、事前にこのメソッドを呼び出す必要があります。

表 6-3 で、release() メソッドについて詳しく説明します。

表 6-3. API の解放

説明	入力値	出力値
認証 SDK を解放します。	API のロケール。	API が正常に解放されない場合は例外を返します。

追加入力の準備

以下の場合は、追加の情報入力に備える必要があります。

- コールアウトまたはプラグインを実装して、WebFort の標準の認証機能の強化を図る場合。
- 完全に新しい独自の認証方式を記述する場合。

上記いずれの場合も、WebFort サーバに送信する必要がある追加の情報を、名前 - 値のペアの形式で設定する必要があります。WebFort の com.arcot.webfort.common.api には、こうした補足情報を指定するための AdditionalInput クラスが用意されています。

AdditionalInput クラスで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を一部以下に紹介します。

- AR_WF_LOCALE_ID

  呼び出し元アプリケーションにメッセージを返す際に WebFort で使用するロケールを指定します。

- AR_WF_CALLER_ID

  トランザクションの追跡に役立ちます。セッション ID またはトランザクション ID を使用して呼び出し元情報を指定できます。

ArcotID 認証

ArcotID はチャレンジ / レスポンス認証方式であり、WebFort サーバがチャレンジを提供します。ArcotID Client が署名済みのチャレンジをアプリケーションを通じて WebFort サーバに送信します。ここでは、以下の内容について説明します。

1. ArcotID のダウンロード
2. ArcotID 認証

ArcotID 認証を正常に実施するため、第 5 章の「ArcotID Client とアプリケーションの統合」の説明に従って、ArcotID Client とアプリケーションを統合する必要があります。

注 : ArcotID のダウンロードと認証は複数の方法で実行できます。詳細については、第 2 章の「WebFort ワークフローについて」を参照してください。ここでは、これらの操作に使用される API に着目して解説します。

ArcotID のダウンロード

ArcotID 認証の実行に際しては、ユーザの ArcotID が、認証リクエストの送信元システムに存在することが前提となります。ArcotID が不在の場合は、システムにダウンロードする必要があります。そのような場合、ArcotID のダウンロード前に、ユーザは第 2 認証を実行する必要があります。

ArcotID をダウンロードする方法

1. （**オプション**）コールアウトまたはプラグインを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。

2. `ArcotIDAuth` インターフェースの `getArcotID()` メソッドを呼び出して、ユーザの ArcotID をアプリケーション側で取得します。

   このメソッドは、ユーザの ArcotID を格納した、`ArcotIDResponse` インターフェースのインスタンスを返します。

3. ユーザの ArcotID が HTML または JSP（Java Server Page）に設定されます。

4. クライアント サイドの `ImportArcotID()` API を呼び出して、アプリケーションからエンド ユーザ システムに ArcotID をダウンロードします。

ArcotID 認証

ArcotID 認証を実行する方法

1. （**オプション**）コールアウトまたはプラグインを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。

2. `ArcotIDAuth` インターフェースの `getChallenge()` メソッドを呼び出して、WebFort サーバからチャレンジを取得します。

   このメソッドは、トランザクションの詳細情報とサーバから取得したチャレンジを格納した、`ArcotIDChallengeResponse` のインスタンスを返します。

3. チャレンジが HTML ページを通じてエンド ユーザに送信されます。

4. クライアント サイドの ArcotID `SignChallengeEx()` メソッドを呼び出して、チャレンジに署名します。

   アプリケーションが ArcotID パスワードを収集し、その ArcotID パスワードを使用して ArcotID Client がチャレンジに署名します。

5. `ArcotIDAuth` インターフェースの `verifySignedChallenge()` メソッドを呼び出して、署名済みのチャレンジを検証します。必要に応じて、`AuthTokenType` クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、AuthResponse インターフェースのインスタンスを返します。

質問と回答認証

質問と回答（Q&A）認証メカニズムは、ArcotID のダウンロード または FYP（Forgot Your Password）認証の際の 2 次認証メカニズムとして使用するか、または独立した認証メカニズムとして使用できます。

この方式では、Q&A の作成段階でユーザが各自で質問と回答一式を設定するか、事前定義済みの質問をユーザに提示するようにアプリケーションを設定できます。設定可能な質問の最大数、ユーザに出題される質問の数、認証中に回収すべき正しい回答の最小数といったパラメータはすべて、WebFort Administration Console を使用して設定できます。

WebFort の呼び出し元検証と呼ばれる機能を使用して、ユーザから回答を収集し、回答内容を検証し、検証結果を WebFort サーバに送信できます。

呼び出し元検証機能を使用した Q&A 認証

以下の手順では、呼び出し元検証機能を有効にして Q&A 認証を実行する方法について説明します。

1. QnAAuth インターフェースの getQuestions() メソッドを呼び出して、WebFort サーバからユーザの質問と回答を取得します。

   注： QnAAuth インターフェースには getQuestions() メソッドが 2 つあります。ブール値の入力を取るメソッドを呼び出して（fetchAnswers）回答を取得する必要があります。

   このメソッドは、出題される質問、各質問の回答、トランザクション ID、メッセージ、応答コード、理由コードを格納した、QnAResponse インターフェースのインスタンスを返します。

2. ユーザの質問と回答を保持するオブジェクトを準備します。そのためには、AuthQnAInfo インターフェースの以下のメソッドを記載順に呼び出す必要があります。

   a. getNumberofQuestions

      ユーザに対して設定されている質問の数を確認するには、このメソッドを呼び出します。

b. `getQuestion`

ユーザに対して設定されている質問を取得するには、このメソッドを呼び出します。このメソッドで取得される質問の数は、`getNumberofQuestions()` メソッドによって返される数によって決まります。

c. WebFort サーバから取得した質問に対するユーザ側の回答を収集するためのロジックを実装します。

d. `answerQuestion`

注： `AuthQnAInfo` インターフェースには `answerQuestion()` メソッドが 2 つあります。入力の 1 つとして検証ステータスを取るメソッドを呼び出す必要があります。

このメソッドを呼び出して、アプリケーションによって収集される回答を設定します。

3. （**オプション**）コールアウトまたはプラグインを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。

4. `QnAAuth` インターフェースの `verifyAnswers()` メソッドを、手順 2 で作成された `AuthQnAInfo` オブジェクトを渡して呼び出し、ユーザによる回答内容を検証します。必要に応じて、`AuthTokenType` クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、`AuthResponse` インターフェースのインスタンスを返します。

サーバ検証を使用した Q&A 認証

以下の手順では、Q&A クレデンシャルを WebFort サーバで直接検証する Q&A 認証の実行方法について説明します。

1. `QnAAuth` インターフェースの `getQuestions()` メソッドを呼び出して、WebFort サーバからユーザの質問と回答を取得します。

注： `QnAAuth` インターフ ェースには `getQuestions()` メソッドが 2 つあります。質問のみを取得するメソッドを呼び出す必要があります。

このメソッドは、出題される質問、各質問の回答、トランザクション ID、メッセージ、応答コード、理由コードを格納した、`QnAResponse` インターフェースのインスタンスを返します。

2. ユーザの質問と回答を保持するオブジェクトを準備します。そのためには、`AuthQnAInfo` インターフェースの以下のメソッドを記載順に呼び出す必要があります。

   a. `getNumberofQuestions`

      ユーザに対して設定されている質問の数を確認するには、このメソッドを呼び出します。

   b. `getQuestion`

      ユーザに対して設定されている質問を取得するには、このメソッドを呼び出します。このメソッドで取得される質問の数は、`getNumberofQuestions()` メソッドによって返される数によって決まります。

   c. WebFort サーバから取得した質問に対するユーザ側の回答を収集するためのロジックを実装します。

   d. `answerQuestion`

注： `AuthQnAInfo` インターフェースには `answerQuestion()` メソッドが 2 つあります。入力の 1 つとして検証ステータスを*取らない*メソッドを呼び出す必要があります。

      このメソッドを呼び出して、アプリケーションによって収集される回答を設定します。

3. （**オプション**）コールアウトまたはプラグインを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。

4. `QnAAuth` インターフェースの `verifyAnswers()` メソッドを、手順 2 で作成された `AuthQnAInfo` オブジェクトを渡して呼び出し、ユーザによる回答内容を検証します。必要に応じて、`AuthTokenType` クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、`AuthResponse` インターフェースのインスタンスを返します。

ユーザ名 / パスワード認証

認証 API には、従来のユーザ名 / パスワード認証を実行する PasswordAuth インターフェースがあります。この認証メカニズムでは、ユーザが認証時にユーザ名と対応するパスワードを指定し、入力されたパスワードが検証されます。

WebFort は、部分パスワードによる認証に対応しています。この機能を有効にした場合、ユーザはパスワード中の異なる位置の文字を入力するように求められます。たとえば、パスワードが「casablanca!」である場合に、1、3、および 8 の位置の文字の入力を求められます。この場合は、「csn」となります。

完全なパスワード認証

標準的なパスワード認証を実行する方法

1. ユーザのパスワードを収集するためのロジックを実装します。
2. （**オプション**）コールアウトまたはプラグインを実装する場合は、AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。
3. PasswordAuth インターフェースの verifyPassword() メソッドを呼び出して、ユーザが提示したパスワードを検証します。必要に応じて、AuthTokenType クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、AuthResponse インターフェースのインスタンスを返します。

部分パスワード認証

標準的なパスワード認証を実行する方法

1. （**オプション**）コールアウトまたはプラグインを実装する場合は、AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。
2. PasswordAuth インターフェースの getPasswordChallenge() メソッドを呼び出して、WebFort サーバからチャレンジを取得します。チャレンジには、ユーザが答える必要があるパスワードの文字位置が含まれています。

3. ユーザのパスワードを収集するためのロジックを実装します。
4. PasswordAuth インターフェースの verifyPassword() メソッドを呼び出して、ユーザが提示したパスワードを検証します。必要に応じて、AuthTokenType クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、AuthResponse インターフェースのインスタンスを返します。

ワンタイム パスワード認証

ワンタイム パスワード（OTP）は、WebFort サーバが生成する数値文字列または英数字文字列です。WebFort は、あらかじめ設定された回数再利用できる OTP をサポートしています。この設定は Administration Console を使用して行います。OTP の寿命は、OTP が有効な期間の長さと、OTP の利用可能回数によって決まります。

OTP 認証を実行する方法

1. ユーザから OTP を収集するためのロジックを実装します。
2. （**オプション**）コールアウトまたはプラグインを実装する場合は、AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。
3. OTPAuth インターフェースの verifyOTP() メソッドを呼び出して、ユーザの OTP を検証します。必要に応じて、AuthTokenType クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、AuthResponse インターフェースのインスタンスを返します。

OATH ワンタイム パスワード認証

OATH 準拠の OTP を認証する方法

1. ユーザから OATH OTP を収集するためのロジックを実装します。
2. （**オプション**）コールアウトまたはプラグインを実装する場合は、AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

詳細については、「追加入力の準備」を参照してください。

3. `OATHAuth` インターフェースの `verifyOTP()` メソッドを呼び出して、ユーザの OTP を検証します。必要に応じて、`AuthTokenType` クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、`AuthResponse` インターフェースのインスタンスを返します。

OATH ワンタイム パスワードの同期

クライアントとサーバの OATH OTP を同期する方法

1. （**オプション**）コールアウトまたはプラグインを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。

2. `OATHAuth` インターフェースの `synchronizeOTP()` メソッドを呼び出して、サーバ OTP とクライアント OTP を同期します。必要に応じて、`AuthTokenType` クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、`AuthResponse` インターフェースのインスタンスを返します。

ArcotOTP 認証

ArcotOTP を認証する方法

1. ユーザから ArcotOTP を収集するためのロジックを実装します。
2. （**オプション**）コールアウトまたはプラグインを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。

3. `ArcotOTPAuth` インターフェースの `verifyOTP()` メソッドを呼び出して、ユーザの OTP を検証します。必要に応じて、`AuthTokenType` クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、AuthResponse インターフェースのインスタンスを返します。

ArcotOTP の同期

クライアントとサーバの ArcotOTP を同期する方法

1. （**オプション**）コールアウトまたはプラグインを実装する場合は、AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。

2. ArcotOTPAuth インターフェースの synchronizeOTP() メソッドを呼び出して、サーバの ArcotOTP とクライアントの ArcotOTP を同期します。必要に応じて、AuthTokenType クラスを使用して、認証が正常に終了した後でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、クレデンシャルの詳細情報、トークン情報を格納した、AuthResponse インターフェースのインスタンスを返します。

認証トークン

WebFort 認証 API は、エンド ユーザの認証が正常に終了した後で該当エンド ユーザに適切なトークンを提供します。このトークンが WebFort サーバに提示され、ユーザが認証されている点、保護されているリソースに該当ユーザがアクセス可能である点が証されます。

認証 API を使用して、認証後にトークンを返す必要があるかどうかを指定できます。また、認証後に返す必要のあるトークンのタイプも指定できます。AuthTokenType クラスは、戻すトークンのタイプとして、以下のタイプのトークンを指定できます。

- **ネイティブ トークン**

  認証の正常終了後に Arcot 独自の（またはネイティブの）トークンが必要な場合は、このタイプを指定します。このトークンは有効期限が切れるまでの間に複数回使用できます。

- **ワンタイム トークン**

  認証の正常終了後にワンタイム トークンが必要な場合は、このタイプを指定します。このトークンは有効期限が切れるまでの間に 1 回のみ使用できます。

- **SAML トークン**

  SAML（Security Assertion Markup Language）は、セキュリティ ドメイン間で交換される認証データの形式を指定するオープン標準です。WebFort によって発行されたネイティブ トークン、デフォルト トークン、ワンタイム トークンは WebFort サーバのみが解釈可能ですが、WebFort サーバが発行した SAML トークンは他の認証システムでも解釈できます。WebFort は、SAML のバージョン **1.1** および **2.0** をサポートしています。

  - SAML 1.1 トークン

    認証の正常終了後に SAML 1.1 トークンを必要とするカスタム（非 WebFort）認証メカニズムを使用している場合は、このタイプのトークンを指定します。

  - SAML 2.0 トークン

    認証の正常終了後に SAML 2.0 トークンを必要とするカスタム（非 WebFort）認証メカニズムを使用している場合は、このタイプのトークンを指定します。

- **デフォルト トークン**

  認証の正常終了後に、サーバで設定されたデフォルト トークンが必要となる場合は、このタイプのトークンを指定します。

- **カスタム**

  カスタム クレデンシャル認証を実行している場合は、このタイプのトークンを指定します。

認証トークンの検証

WebFort サーバは、ユーザに対して発行されるネイティブ トークンおよびワンタイム トークンのみを検証できます。認証トークンを SSO（Single Sign-On）で使用する場合は、これらのトークンを検証する必要があります。この場合、ユーザを一度のみ認証し、複数のリソースの利用を認証トークンを使用して許可します。

トークンが有効かどうかを検証する方法

1. （**オプション**）コールアウトまたはプラグインを実装する場合は、`AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。

2. `Authentication` クラスの `verifyAuthToken()` メソッドを呼び出して、ユーザのトークンを検証します。

このメソッドは、クレデンシャルおよびトランザクションの詳細情報を格納した、AuthTokenResponse インターフェースのインスタンスを返します。

PAM の取得

PAM（Personal Assurance Message）は、フィッシング サイトではなく組織の純正サイトにアクセスしていることをエンド ユーザに保証するセキュリティ機能です。

ユーザの PAM を取得する方法

1. （**オプション**）コールアウトまたはプラグインを実装する場合は、AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   詳細については、「追加入力の準備」を参照してください。

2. Authentication クラスの getPAM() メソッドを呼び出して、ユーザの PAM を取得します。

   このメソッドは、ユーザの詳細情報、PAM、トランザクションの詳細情報を格納した、PAMResponse インターフェースのインスタンスを返します。

認証操作のまとめ

表 6-4 に、この章で説明した認証操作の実行に必要な入力パラメータをまとめます。

表 6-4. 認証操作

操作	必要な入力	出力
ArcotID	• ユーザ名（userName） • （オプション）組織名（orgName） 注： 組織名が未指定の場合、ユーザはデフォルトの組織に所属しているものとみなされます。 • 署名済みチャレンジ（signedResponse） • （オプション）追加入力（AdditionalInput） • （オプション）認証トークン タイプ（AuthTokenType）	• AuthResponse
Q&A	• ユーザ名（userName） • （オプション）組織名（orgName） 注： 組織名が未指定の場合、ユーザはデフォルトの組織に所属しているものとみなされます。 • 認証用の質問と回答（qnaInfo） • （オプション）追加入力（AdditionalInput） • （オプション）認証トークン タイプ（AuthTokenType）	

表 6-4. 認証操作

操作	必要な入力	出力
ユーザ名 / パスワード	• **ユーザ名**（userName） • **（オプション）組織名**（orgName） **注：** 組織名が未指定の場合、ユーザはデフォルトの組織に所属しているものとみなされます。 • **認証用のユーザ パスワード**（password） • **（オプション）追加入力**（AdditionalInput） • **（オプション）認証トークン タイプ**（AuthTokenType）	• AuthResponse
OTP	• **ユーザ名**（userName） • **（オプション）組織名**（orgName） **注：** 組織名が未指定の場合、ユーザはデフォルトの組織に所属しているものとみなされます。 • **認証用のワンタイム パスワード**（otp） • **（オプション）追加入力**（AdditionalInput） • **（オプション）認証トークン タイプ**（AuthTokenType）	
OATH OTP	• **ユーザ名**（userName） • **（オプション）組織名**（orgName） **注：** 組織名が未指定の場合、ユーザはデフォルトの組織に所属しているものとみなされます。 • **認証用の** OATH OTP（otp） • **（オプション）追加入力**（AdditionalInput） • **（オプション）認証トークン タイプ**（AuthTokenType）	
ArcotOTP	• **ユーザ名**（userName） • **（オプション）組織名**（orgName） **注：** 組織名が未指定の場合、ユーザはデフォルトの組織に所属しているものとみなされます。 • **認証用の** ArcotOTP（otp） • **（オプション）追加入力**（AdditionalInput） • **（オプション）認証トークン タイプ**（AuthTokenType）	

第 7 章
カスタム API の使用

WebFort のカスタム API を使用して、新しいクレデンシャルを追加できます。これらの API を使用して新しいクレデンシャルを作成および管理し、認証に使用できます。カスタム API は以下のいずれかの目的で使用できます。

- 完全に新しい認証方式を追加する（証明書ベースやハードウェア ベースの方式など）。
- WebFort 認証方式と、WebFort 認証方式に基づく方式を一緒にサポートする。たとえば、WebFort OTP と同時に、WebFort OTP に基づく認証方式を使用できます。

注： WebFort でカスタム API をサポートし、この章で説明している操作を実行するには、これらの関数を提供するプラグインまたはコールアウトを設定する必要があります。

この章では、カスタム クレデンシャル用の発行 API および認証 API の両方について説明します。この章では、以下の内容について説明します。

- 発行操作
- 認証操作

発行操作

カスタム クレデンシャルについては以下の発行操作がサポートされています。

- クレデンシャルの作成
- クレデンシャルの無効化
- クレデンシャルの有効化
- クレデンシャルのリセット
- クレデンシャルの再発行 l
- クレデンシャルの有効期間のリセット
- クレデンシャル詳細情報の取得

- クレデンシャルの削除

クレデンシャルの作成

com.arcot.webfort.issuance.api パッケージには、クレデンシャルを作成するメソッドを含む CustomIssuance インターフェースがあります。

クレデンシャルを作成するには、以下の手順に従います。

1. クレデンシャル情報をカプセル化する CustomInput クラスを使用します。
2. AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
3. CustomIssuance インターフェースの create() メソッドを呼び出して、クレデンシャルを作成します。

   このメソッドは、クレデンシャルおよびトランザクションの詳細情報を指定する、CustomResponse インターフェースのインスタンスを返します。

クレデンシャルの無効化

クレデンシャルを無効にするには、以下の手順に従います。

1. クレデンシャル情報をカプセル化する CustomInput クラスを使用します。
2. AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
3. CustomIssuance インターフェースの disable() メソッドを呼び出して、クレデンシャルを無効にします。

   このメソッドは、クレデンシャルおよびトランザクションの詳細情報を指定する、CustomResponse インターフェースのインスタンスを返します。

クレデンシャルの有効化

クレデンシャルを有効にするには、以下の手順に従います。

1. クレデンシャル情報をカプセル化する CustomInput クラスを使用します。

2. `AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

3. `CustomIssuance` インターフェースの `enable()` メソッドを呼び出して、クレデンシャルを有効にします。

   このメソッドは、クレデンシャルおよびトランザクションの詳細情報を指定する、`CustomResponse` インターフェースのインスタンスを返します。

クレデンシャルのリセット

クレデンシャルをリセットするには、以下の手順に従います。

1. クレデンシャル情報をカプセル化する `CustomInput` クラスを使用します。
2. `AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

3. `CustomIssuance` インターフェースの `resetCredential()` メソッドを呼び出して、カスタム クレデンシャルをリセットします。

   このメソッドは、クレデンシャルおよびトランザクションの詳細情報を指定する、`CustomResponse` インターフェースのインスタンスを返します。

クレデンシャルの再発行

クレデンシャルを再発行するには、以下の手順に従います。

1. クレデンシャル情報をカプセル化する `CustomInput` クラスを使用します。
2. `AdditionalInput` クラスの `setAdditionalInput()` メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

3. `CustomIssuance` インターフェースの `reissue()` メソッドを呼び出して、カスタム クレデンシャルを再発行します。

   このメソッドは、クレデンシャルおよびトランザクションの詳細情報を指定する、`CustomResponse` インターフェースのインスタンスを返します。

クレデンシャルの有効期間のリセット

クレデンシャルの有効期間をリセットするには、以下の手順に従います。

1. クレデンシャル情報をカプセル化する CustomInput クラスを使用します。
2. AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
3. CustomIssuance インターフェースの resetValidity() メソッドを呼び出して、カスタム クレデンシャルの有効期間をリセットします。

   このメソッドは、クレデンシャルおよびトランザクションの詳細情報を指定する、CustomResponse インターフェースのインスタンスを返します。

クレデンシャル詳細情報の取得

クレデンシャルの詳細情報を取得するには、以下の手順に従います。

1. クレデンシャル情報をカプセル化する CustomInput クラスを使用します。
2. AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
3. CustomIssuance インターフェースの fetch() メソッドを呼び出して、カスタム クレデンシャルの詳細情報を取得します。

   このメソッドは、クレデンシャルおよびトランザクションの詳細情報を指定する、CustomResponse インターフェースのインスタンスを返します。

クレデンシャルの削除

クレデンシャルを削除するには、以下の手順に従います。

1. クレデンシャル情報をカプセル化する CustomInput クラスを使用します。
2. AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。

3. CustomIssuance インターフェースの delete() メソッドを呼び出して、カスタム クレデンシャルを削除します。

   このメソッドは、クレデンシャルおよびトランザクションの詳細情報を指定する、CustomResponse インターフェースのインスタンスを返します。

認証操作

com.arcot.webfort.authentication.api パッケージには、パスワード ベースおよびチャレンジ / レスポンス ベースの認証方式を認証するメソッドを含む CustomAuth インターフェースがあります。

パスワード ベース認証

パスワードベースの認証を実行する方法

1. ユーザ パスワードを収集する必要があります。
2. AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
3. CustomAuth インターフェースの verifyCredential() メソッドを呼び出して、ユーザが提示したパスワードを検証します。必要に応じて、AuthTokenType クラスを使用して、認証が正常に終了した*後*でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、ユーザの詳細情報、トークン情報を指定する、CustomAuthResponse インターフェースのインスタンスを返します。

チャレンジ / レスポンス ベース認証

チャレンジ / レスポンス ベースの認証を実行する方法

1. AdditionalInput クラスの setAdditionalInput() メソッドを呼び出して、このクラスを実装するオブジェクトを取得します。

   このクラスでは、補足情報を名前 - 値のペアとして設定します。詳細については、「追加入力の準備」を参照してください。
2. CustomAuth インターフェースの getChallenge() メソッドを呼び出して、チャレンジを受信します。

このメソッドは、トランザクションの詳細情報を指定してサーバからチャレンジを取得する、CustomChallengeResponse インターフェースのインスタンスを返します。

3. 使用するクレデンシャルのタイプに応じて認証を実行します。たとえば、チャレンジ / レスポンス認証の場合は、ユーザがチャレンジに署名します。または、質問と回答認証の場合は、ユーザが認証用の回答を提示します。
4. CustomAuth インターフェースの verifyCredential() メソッドを呼び出して、クレデンシャルを検証します。必要に応じて、AuthTokenType クラスを使用して、認証が正常に終了した*後*でユーザに返す必要があるトークンのタイプを指定できます。

   このメソッドは、トランザクションの詳細情報、ユーザの詳細情報、トークン情報を指定する、CustomAuthResponse インターフェースのインスタンスを返します。

付録 A
入力データの検証

システムによる無効なデータの処理を防ぎ、ビジネス ルールを適用し、ユーザ入力と内部構造およびスキーマとの互換性を確保するため、WebFort サーバは API から受信するデータを検証します。表 A-1 で、この入力データを検証する際に WebFort サーバが使用する基準について説明します。

注： 以下の表で説明する属性の長さは、文字列の長さに相当します。

注： 属性 ID は Java API では `paramName` と呼ばれます。

表 A-1. 属性の妥当性検査

属性	属性 ID	妥当性検査基準
User name	USER_NAME	ユーザ名が空ではない。
		ユーザ名の長さが 1 ～ 256 文字である。
User status	USER_STATUS	ユーザ ステータスがアクティブかどうかをチェックする。 • アクティブ状態 -> USER_STATUS_ACTIVE • 無効状態 -> USER_STATUS_DISABLED
Email ID	EMAIL_ID	電子メール ID が空ではない。
		電子メール ID の長さが 1 ～ 128 文字である。
		スペース以外の空白文字（ASCII 0 ～ 31）が含まれていない。
		以下の点をチェックする。 • @ はピリオド（.）の前にある。 • @ が 1 文字以上含まれている。 • ピリオド（.）が 1 文字以上含まれている。

表 A-1. 属性の妥当性検査

属性	属性 ID	妥当性検査基準
First name	FIRST_NAME	名が空ではない。
		名の長さが 1 ～ 32 文字である。
		スペース以外の空白文字（ASCII 0 ～ 31）が含まれていない。
Last name	LAST_NAME	姓が空ではない。
		姓の長さが 1 ～ 32 文字である。
		スペース以外の空白文字（ASCII 0 ～ 31）が含まれていない。
Personal Assurance Message	PAM	PAM が空ではない。
		PAM の長さが 1 ～ 128 文字である。
		スペース以外の空白文字（ASCII 0 ～ 31）が含まれていない。
Start time	START_TIME	有効な日付形式かどうかをチェックする。
End time	END_TIME	有効な日付形式かどうかをチェックする。
Organization name	ORG_NAME	組織名が空ではない。
		組織名の長さが 1 ～ 64 文字である。
		スペース以外の空白文字（ASCII 0 ～ 31）が含まれていない。
Locale name	LOCALE_NAME	ロケール名が空ではない。
		ロケール名がロケールの ISO のセットに従っているかどうかをチェックする。
Authentication user password	AUTH_USER_PASSWORD	ユーザ パスワードが空ではない。
		ユーザ パスワードの長さが 1 ～ 64 文字である。
		ユーザ パスワードを一連の文字列と照合してチェックする。
		スペース以外の空白文字（ASCII 0 ～ 31）が含まれていない。
Password maximum length	PWD_MAX_LENGTH	パスワード最大文字数の最小値 > 4 文字。
		パスワード最大文字数の最大値 < 64 文字。
Password minimum length	PWD_MIN_LENGTH	パスワード最小文字数の最小値 > 4 文字。
		パスワード最小文字数の最大値 < 64 文字。

表 A-1. 属性の妥当性検査

属性	属性 ID	妥当性検査基準
Question	AUTH_QUESTIONS	質問が空ではない
		質問の長さが 1 ～ 64 文字である。
		スペース以外の空白文字（ASCII 0 ～ 31）が含まれていない。
回答	AUTH_ANSWERS	回答が空ではない。
		回答の長さが 1 ～ 64 文字である。
		スペース以外の空白文字（ASCII 0 ～ 31）が含まれていない。
Duplicate Question and answers	DUPLICATE_QUESTION_AND_ANSWER	質問は重複していない。
		回答は重複していない。
		質問が回答と同じではない。
Token type	AUTH_TOKEN_TYPE	次のいずれかの値であることをチェックする。 • DEFAULT_TOKEN • NATIVE_TOKEN • OTP_TOKEN • SAML11_TOKEN • SAML20_TOKEN • NO_TOKEN
Password	Password	パスワードが空ではない。
		パスワードの長さが 1 ～ 64 文字である。
		スペース以外の空白文字（ASCII 0 ～ 31）が含まれていない。
OTP maximum length	OTP_MAX_LENGTH	OTP の長さが 4 ～ 64 文字である。

付録 B
WebFort のログ

WebFort サーバのログには、さまざまなタイプの操作に関するトランザクション データおよびフローの詳細が含まれています。これらのログを使用して、誤った設定、誤ったトランザクション データ、その他のエラーなど、さまざまなタイプの問題をデバッグできます。この付録では、WebFort サーバのスタートアップ ログおよびトランザクション ログを理解する方法について説明します。

サーバのスタートアップ中など、操作によってはエラー メッセージが表示されない場合があります。エラー メッセージが表示された場合でも、問題を特定するツールの 1 つとして、ログ ファイルを使用する必要がある可能性があります。

この付録では、以下のトピックについて説明します。

ログ ファイルについて

WebFort では、ログの場所およびログ ファイルのディレクトリ情報に加えて、ログ パラメータも変更できます。これらのファイルで変更できる一般的なログ設定オプションは、次のとおりです。

- **Specifying log file name and path**：WebFort ではログ ファイルの書き込み先およびバックアップ ログ ファイルの保存先のディレクトリを指定できます。さらに、トレース ログ記録を有効にするかどうかを制御できます。
- **Log file size**：ログ ファイルに保存できる最大バイト数。ログ ファイルがこのサイズに達すると、指定した名前で新規ファイルが作成され、古いファイルがバックアップ ディレクトリに移動されます。
- **Using log file archiving**：WebFort 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できます。この設定により、ログ ファイルの最大サイズを指定できます。最大サイズに達すると、古いログ情報がバックアップ ファイルに移動され、その後新しいログ情報が保存されます。

- **Setting logging levels**：WebFort ではログ レベルも設定できます。ログ レベルを設定して、診断ログ ファイルに保存されるメッセージ数を削減できます。たとえば、クリティカルなメッセージのみをレポートおよび保存するように、ログ レベルを設定できます。サポートされるログ レベルの詳細については、「サポートされる重大度レベル」を参照してください。
- **Specifying time zone information**：WebFort ではログ記録された情報のタイム スタンプに、ローカル タイムゾーンまたは GMT を使用できます。

WebFort のログ ファイルは、スタートアップ ログとトランザクション ログに分類されます。ログ ファイルのデフォルトの場所は次のとおりです。

Windows の場合

<install_location>\Arcot Systems\logs\

UNIX ベースの場合

<install_location>/arcot/logs/

表 B-1 で、WebFort ログ ファイルについて説明します。

表 B-1. ログ ファイル

ログ ファイル タイプ	ログ ファイル名	説明
スタートアップ ログ	arcotwebfortstartup.log（スタートアップ ログ）	WebFort サーバを起動すると、このファイルにスタートアップ（またはブート）アクションがすべて記録されます。WebFort サービスが起動しない場合、問題の原因を特定するためにこのファイルの情報が役立ちます。

表 B-1. ログ ファイル

ログ ファイル タイプ	ログ ファイル名	説明
トランザクション ログ	arcotwebfort.log（サーバ ログ）	WebFort は、サーバで処理されるすべてのリクエストを、arcotwebfort.log ファイルに記録します。このファイルのログを制御するパラメータは、Administration Console で設定できます。設定するには、インスタンス管理画面で目的のインスタンスをクリックし、インスタンス固有の設定サブ画面にアクセスする必要があります。
	arcotwebfortstas.log（統計ログ）	WebFort では、ログ記録の統計を取るためにこのファイルを使用します。
	arcotwebfort.log（トレース ログ）	WebFort ではトレース ログも作成されます。このログには、フローの詳細が含まれます。トレース ログは、arcotwebfort.log ファイルに記録されます。トレース メッセージのエントリは「TRACE:」で始まります。

WebFort ログ ファイルの形式

表 B-2 で、WebFort ロガー内のエントリの形式について説明します。

表 B-2. WebFort のログ フォーマット

列	説明
タイム スタンプ	エントリがログに記録された時刻は、設定したタイムゾーンに変換されます。この情報のログの形式は次のとおりです。 mm/dd/yy HH: MM: SS.mis mis はミリ秒を表します。
ログ レベル（LEVEL）（重大度）	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、「サポートされる重大度レベル」を参照してください。

表 B-2. WebFort のログ フォーマット

列	説明
プロトコル名 (PROTOCOLNAME)	トランザクションに使用されたプロトコル。以下の値が使用されます。 • AUTH_NATIVE • ADMIN_WS • ASSP_WS • RADIUS • SVRMGMT_WS • TXN_WS サーバが起動中、シャットダウン中、またはモニタリング モードのときはプロトコルは使用されないため、それぞれ次の値が表示されます。 • STARTUP • SHUTDOWN • MONITOR
スレッド ID (THREADID)	このエントリをログに記録したスレッドの ID。
トランザクション ID (000TXNID)	このエントリをログに記録したトランザクションの ID。
メッセージ	自由形式でログ ファイルに記録されるメッセージ。 **注：**メッセージの情報量は、ログ ファイルに設定した**ログ レベル**によって異なります。

サポートされる重大度レベル

ログ レベル（重大度レベル）を使用して、WebFort ログに保存される情報の詳細のレベルを指定できます。また、この設定により、ログ ファイルが増大する速度を制御できます。

表 B-3 で、ログ レベルについて重大度の降順で説明します。

表 B-3. WebFort ログ レベル（重大度の降順）

ログ レベル		説明
0	FATAL	WebFort サービスの突然の終了を引き起こす可能性がある、重大で回復不可能なエラーにはこのログ レベルを使用します。
1	WARNING	望まないランタイム例外、潜在的に有害な状況、および回復可能で FATAL（致命的）ではない問題にはこのログ レベルを使用します。

表 B-3. WebFort ログ レベル（重大度の降順）

ログ レベル		説明
2	INFO	ランタイム イベントに関する情報を取得する場合にこのログ レベルを使用します。 つまりアプリケーションの進行状況に重点を置きます。この情報には次の内容が含まれます。 • 起動、停止、再起動などのサーバ状態。 • 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。 • リクエストされたトランザクション。 • 関連する操作。 • トランザクションの結果。
3	DEBUG	デバッグ目的で詳細情報をログに記録する場合に、このログ レベルを使用します。これには、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、それよりも重要度が*高い*レベルのメッセージもレポートされます。たとえば、LogLevel に INFO を指定すると、ログ レベルが FATAL および WARNING のメッセージも取得されます。

関連文書： WebFort サーバ（arcotwebfort.log）では、これらの任意のレベルのログを設定でき、TRACE ログを有効にしてフローの詳細を取得することもできます。この詳細については、「Arcot WebFort 6.2 管理ガイド」の「WebFort サーバ ログ記録設定の管理」を参照してください。

以下のサブセクションでは、WebFort ログ ファイル内のサンプル エントリを（ログ レベルごとに）示します。

FATAL

```
09/08/13 14:32:51.889 FATAL STARTUP      00003820 00WFMAIN - Arcot WebFort
Authentication Service SHUTDOWN due to initialization failure!


09/08/13 14:33:58.154 FATAL STARTUP      00003612 00WFMAIN - [7]: Database
password could not be obtained from securestore.enc for [wf-61-st1-mssql-test]


09/08/13 14:33:58.154 FATAL STARTUP      00003612 00WFMAIN - Arcot WebFort
Authentication Service SHUTDOWN due to initialization failure!
```

WARNING

```
09/08/13 14:13:31.451 WARN  SVRMGMT_WS   00005808 00055503 -
ArDBConnection::GetDBDiagnosis: SQL State:01000, Native Code: 2746, ODBC code:
[Arcot Systems][ODBC SQL Server Driver][DBNETLIB]ConnectionWrite (send()).


09/08/13 14:13:31.451 WARN  SVRMGMT_WS   00005808 00055503 -
ArDBConnection::GetDBDiagnosis: SQL State:01000, Native Code: 2746, ODBC code:
[Arcot Systems][ODBC SQL Server Driver][DBNETLIB]ConnectionWrite (send()).SQL
State:08S01, Native Code: B, ODBC code: [Arcot Systems][ODBC SQL Server
Driver][DBNETLIB]General network error.Check your network documentation.


09/08/13 14:13:31.451 WARN  SVRMGMT_WS   00005808 00055503 -
ArDBPoolManagerImpl::isKnownFailure: Error state [08S01] is detected as known
failure(type:0)!


09/08/13 14:13:31.451 WARN  SVRMGMT_WS   00005808 00055503 -
ArDBPoolManagerImpl::reportQueryFailure: Query failure is detected as DBFO for
primary DSN [wf-61-st1-mssql] and context [70].Marking it bad.


09/08/13 14:13:31.451 WARN  MONITOR      00004984 0MONITOR -
ArDBPoolManagerImpl::recoverPool: [primary] pool [wf-61-st1-mssql] is marked
bad.Will try to clean and connect...


09/08/13 14:13:31.467 WARN  SVRMGMT_WS   00005808 00055503 - ArDBM::Caught
ArcotException in _DbOp!.err : [Arcot Exception,Error: All database pools are
inactive]
```

INFO

```
09/08/13 11:52:17.493 INFO  STARTUP      00003504 00WFMAIN -
-----------------------------------------------------------------

09/08/13 11:52:17.493 INFO  STARTUP      00003504 00WFMAIN - Listing :
[Server startup]

09/08/13 11:52:17.493 INFO  STARTUP      00003504 00WFMAIN -
-----------------------------------------------------------------

09/08/13 11:52:17.493 INFO  STARTUP      00003504 00WFMAIN - Timezone
information.....................: [09/08/13 11:52:17 India Standard
Time]

09/08/13 11:52:17.493 INFO  STARTUP      00003504 00WFMAIN -
ARCOT_HOME.............................: [D:\ArcotInstalls2\Arcot
Systems]

09/08/13 11:52:17.493 INFO  STARTUP      00003504 00WFMAIN - Server
ProcessID.........................: [4316]

09/08/13 11:52:17.493 INFO  STARTUP      00003504 00WFMAIN -
-----------------------------------------------------------------
```

DEBUG

```
10/03/25 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 -
ArDBPoolManagerImpl::getLockedDBConnection: [primary] DSN [webfort] is
active.Will get the connection from this

10/03/25 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 -
ArDBPoolManagerImpl::getLockedDBConnection: Returning DBPool [0112FD80]

10/03/25 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 - ArDBM::Number of
queries being executed [1]

10/03/25 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 - ArDBM::Found query
string for query-id : [SSL_TRUST_STORE_FETCH_ALL].

10/03/25 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 - ArDBM::Executing
Query[ArWFSSLTrustStoreQuery_FetchAll]

10/03/25 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 - Number of rows
fetched : 0
```

（WebFort サーバのみ）トレース ログ

```
10/03/25 15:23:38.515 DEBUG SVRMGMT_WS   00004396 00000596 - TRACE: Released
Cache read lock on [01129D98]

10/03/25 15:23:38.515 DEBUG SVRMGMT_WS   00004396 00000596 - TRACE:
CallTrace::Leaving : [ArDBPoolManagerImpl::selectAnActivePool].time : 0

10/03/25 15:23:38.515 DEBUG SVRMGMT_WS   00004396 00000596 - TRACE:
CallTrace::Entering : [ArDBPool::getLockedDBConnectionConst]

10/03/25 15:23:38.515 DEBUG SVRMGMT_WS   00004396 00000596 - TRACE:
ArDBPool::getLockedDBConnection [(primary)] : GotContext [1], [3] more
connections available

10/03/25 15:23:38.515 DEBUG SVRMGMT_WS   00004396 00000596 - TRACE:
CallTrace::Leaving : [ArDBPool::getLockedDBConnectionConst].time : 0
```

付録 C
追加設定

この付録では、以下の項目について説明します。

複数の WebFort サーバ インスタンスの設定

複数の WebFort サーバ インスタンスを使用できるように 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。このファイルには、1 つの WebFort サーバ インスタンスを設定するためのエントリがデフォルトで指定されています。これらのエントリの末尾には、1 つのサーバのみを設定することを示す 1 が追加されています。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、インスタンス番号を適宜追加する必要があります。

WebFort サーバ インスタンスを設定するには、以下の手順に従います。

1. 設定する SDK に応じて、以下のフォルダに格納された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

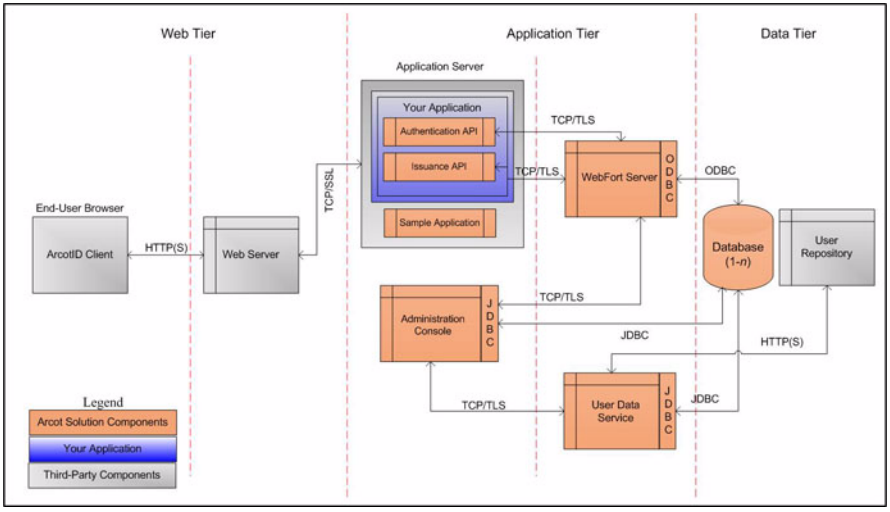
`<install_location>/arcot/sdk/java/properties`

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 に設定されます。通信モードを変更する場合は、「SSL の設定」を参照してください。
3. `host.<n>` パラメータの値を WebFort サーバのホスト名または IP アドレスに設定します。
4. `port.<n>` パラメータの値を Authentication Native プロトコルまたは Transaction Web Services プロトコルがリスニングしているポート番号に設定します。

SSL の設定

セッション中に交換されるデータの整合性と機密性を保証するため、WebFort は Java SDK と WebFort サーバ間の SSL（Secure Socket Layer）通信をサポートしています。デフォルトでは、すべてのコンポーネント間の通信モードに TCP（Transmission Control Protocol）が使用されます。

図 C-1WebFort の通信モード



Java SDK と WebFort サーバ間の SSL 通信を有効にする前に、以下の作業を行う必要があります。

1. 信頼できる認証局からデジタル証明書を入手します。
2. HTTPS 対応のサーバ ポートを介してアプリケーションを公開します。

Java SDK と WebFort サーバ間の SSL を設定する方法

1. 設定する SDK に応じて、以下のフォルダに格納された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. <*SDK name*>.transport.<*n*> パラメータの値を ssl に設定します。
3. <*SDK name*>.serverCACertPEMPath パラメータの値をサーバの CA 証明書ファイルのパスに設定します。このファイルのフォーマットは .PEM である必要があります。
4. <*SDK name*>.clientCertKeyP12Path パラメータの値をクライアントの証明書ファイルのパスに設定します。このファイルのフォーマットは .P12 である必要があります。
5. <*SDK name*>.clientCertKeyPassword パラメータの値を p12 ストアのパスワードに設定します。
6. ファイルを保存して閉じます。

付録 D
SDK の例外とエラー コード

この付録では、WebFort 6.2 SDK から返されるすべての例外とエラー コードについて説明します。以下のトピックがあります。

例外

WebFort の例外は、以下のように分類されます。

共通例外

`com.arcot.webfort.common.api.exception` パッケージは、WebFort サーバと SDK から返される例外を提供します。表 D-1 に、このパッケージの例外を示します。

表 D-1. `com.arcot.webfort.common.api.exception` **の共通例外クラス**

クラス	例外の発生元	説明
`CredentialNotFoundException`	サーバ	この例外は、ユーザが認証に使用しようとしたクレデンシャルが見つからなかった場合に返されます。
`InvalidParamException`	サーバ	この例外は、操作に使用したパラメータのいずれかに無効な値があった場合に返されます。
`InvalidSDKConfigurationException`	SDK	この例外は、API を初期化する API への入力として設定ファイルの絶対パスが指定されたが、そのファイルの読み取りができなかった場合に返されます。

表 D-1. `com.arcot.webfort.common.api.exception` **の共通例外クラス**

クラス	例外の発生元	説明
`SDKAlreadyInitializedException`	SDK	この例外は、SDK がすでに初期化されている場合に返されます。
`SDKException`	SDK	この例外は、クライアント側のすべての例外の基底クラスです。
`SDKInternalErrorException`	SDK	この例外は、以下の場合に発生します。 • リクエストが有効でない。 • SDK が接続を解放しなかった。 • SDK が未分類のエラーを生成した。
`SDKNotInitializedException`	SDK	この例外は、SDK を初期化する前に関数を使用した場合に返されます。
`ServerException` サーバ	サーバ	サーバ側のすべての例外の基底クラスです。
`ServerUnreachableException` SDK	SDK	この例外は、SDK が WebFort サーバに接続できなかった場合に返されます。
`TransactionException`	サーバ	この例外は、トランザクションの実行中に内部エラーが発生した場合に返されます。たとえば、UDS が実行されていない場合や、データベースが停止している場合などです。
`UserNotFoundException`	サーバ	この例外は、操作を実行しようとしたユーザが WebFort に登録されていない場合に返されます。

発行例外

`com.arcot.webfort.issuance.api.exception` パッケージは、ユーザとクレデンシャルのステータスに基づいて返される例外クラスを提供します。表 D-2 に、WebFort サーバによって返される発行例外を示します。

表 D-2. `com.arcot.webfort.issuance.api.exception` **の発行例外クラス**

クラス	説明
`CredentialAlreadyExistsException`	この例外は、ユーザがすでに持っているクレデンシャル タイプを作成しようとした場合に返されます。ユーザは同じタイプのクレデンシャルを複数持つことができません。
`UserAlreadyExistsException`	この例外は、すでに存在するユーザ名を使ってユーザを作成しようとした場合に返されます。

認証例外

com.arcot.webfort.authentication.api.exception パッケージは、ユーザ認証とクレデンシャルのステータスに基づいて返される例外クラスを提供します。表 D-3 に、WebFort サーバによって返される認証例外を示します。

表 D-3. com.arcot.webfort.authentication.api.exception **の認証例外クラス**

クラス	説明
AttemptsExhaustedException	この例外は、ユーザが不正なクレデンシャルを使って認証しようとした回数が許可された最大認証試行回数に達した場合に返されます。
CredReissuedException	この例外は、ユーザが認証に使用しようとしたクレデンシャルが再発行されていた場合に返されます。
InactiveAccountException	この例外は、ユーザが以下のいずれかの状態のクレデンシャルを使って認証しようとした場合に返されます。 • 無効 • ロック済み • 削除済み • 確認済み（OTP *のみ*）
InvalidCredException	この例外は、ユーザが指定したクレデンシャルが有効でない場合に返されます。

エラー コード

WebFort のエラー コードは、以下のように分類されます。

- SDK コード
- サーバ コード

SDK コード

表 D-4 に、SDK レスポンス コード、障害の原因、および解決方法（該当する場合）を示します。

表 D-4. SDK レスポンス コードと障害の原因

SDK レスポンス コード	説明	考えられる障害の原因
0	SDK はリクエストを正常に送信し、サーバからレスポンスを受信しました（またはその逆）。 **注：** これは、トランザクションが成功したことを示すものではありません。	該当なし
1	予期しない理由で SDK の内部エラーが発生しました。	**考えられる原因：** SDK による予期しない動作。
2 (`SDKNotInitializedException` によって返される)	SDK が正常に初期化されませんでした。	**考えられる原因：** 初期化せずに API をコールしたときに返されます。 **解決方法：** SDK を初期化する関数が正常に完了しているかどうかを確認します。
3 (`SDKAlreadyInitializedException` によって返される)	SDK はすでに初期化されています。	**考えられる原因：** ユーザはすでに初期化された SDK を初期化しようとしています。
4	初期化 API への入力として設定ファイルの絶対パスが指定されましたが、そのファイルを読み取ることができません。	**考えられる原因：** 設定ファイルのパスが正しくない可能性があります。 **解決方法：** 設定ファイルのパスを正しく指定します。 **考えられる原因：** 設定ファイルの読み取り権限が設定されていません。 **解決方法：** 設定ファイルの読み取り権限を設定します。

表 D-4. SDK レスポンス コードと障害の原因

SDK レスポンス コード	説明	考えられる障害の原因
5 (`ServerUnreachableException` によって返される)	SDK は設定されたサーバにリクエストを送信できません。	**考えられる原因**： サーバのホスト、ポート、また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**： 正しいホストとポート番号を設定します。 **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は、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**： SSL 証明書を正しく設定します。
6	出力構造体によって送信されたバッファのサイズが十分ではありません。	**考えられる原因**： 出力構造体で渡されたバッファのサイズが、データを格納するためには十分ではありません。 **解決方法**： すべてのデータを格納できる、十分なサイズのバッファを送信します。

表 D-4. SDK レスポンス コードと障害の原因

SDK レスポンス コード	説明	考えられる障害の原因
7（InvalidSDKConfigurationException によって返される）	SDK が正しく設定されていません。	**考えられる原因**： サーバのホスト、ポート、また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**： 正しいホストとポート番号を設定します。 **考えられる原因**： SSL が設定されている場合は、証明書が正しくない可能性があります。 **解決方法**： 有効なクライアント PKCS#12 ファイルとサーバ ルート CA 証明書を設定します。
999（SDKInternalErrorException によって返される）	内部エラー。	**考えられる原因**： 予期しない SDK の内部エラーです。

サーバ コード

表 D-5 に、レスポンス コード、理由コード、障害の原因、および解決方法（該当する場合）を示します。

表 D-5. レスポンス コードと理由コード

レスポンス コード	理由コード	説明	考えられる障害の原因
0	0	操作が正常に完了しました。	該当なし
	6100	認証に成功しましたが、クレデンシャルが猶予期間中です。	**対処法**： クレデンシャルの有効期限がすでに切れています。ユーザにクレデンシャルを再発行するように通知してください。
	6101	認証に成功しましたが、クレデンシャルが警告期間中です。	**対処法**： クレデンシャルの有効期限がもうすぐ切れます。ユーザにクレデンシャルを再発行するように通知してください。

表 D-5. レスポンス コードと理由コード

レスポンス コード	理由コード	説明	考えられる障害の原因
1000	0	内部エラー。	**考えられる原因**： 予期しない内部エラーです。
	2000	データベースが実行されていません。	**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間の接続を再度確立します。
1000	2001	設定が見つかりません。	**考えられる原因**： トランザクションの処理に必要な設定が見つかりません。 **解決方法**： サーバのトランザクション ログを詳しく調べて、必要な設定が作成され、割り当てられていることを確認します。 **考えられる原因**： トランザクションの処理に必要な設定が作成されていますが、サーバのキャッシュに存在しません。 **解決方法**： サーバのキャッシュをリフレッシュします。
	2002	トランザクション ID を生成できませんでした。	**考えられる原因**： サーバの内部エラーのためにトランザクション ID を生成できませんでした。 **解決方法**： 最も可能性の高い原因は、データベース障害です。サーバのトランザクション ログを詳しく調べ、サーバのログに基づいて適切な処置を行います。

表 D-5. レスポンス コードと理由コード

レスポンス コード	理由コード	説明	考えられる障害の原因
1050	2050	操作に使用されたいずれかのパラメータの値が空です。	**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 A の 「入力データの検証」を参照してください。
1050	2051	操作に使用されたいずれかのパラメータの長さが最大許容値を超えています。 **ヒント：**この長さは、パラメータの長さ（パスワード長など）を指しています。	**考えられる原因**： API に渡されたパラメータの長さが最大値を超えています。 **解決方法**： パラメータの長さが最大許容値を超えないように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 A の 「入力データの検証」を参照してください。
	2052	操作に使用されたいずれかのパラメータの長さが最小許容値を下回っています。	**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を下回っています。 **解決方法**： パラメータの長さが最小許容値以上になるように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 A の 「入力データの検証」を参照してください。
	2053	操作に使用されたいずれかのパラメータの値が最大許容値を超えています。 **ヒント：**この値は、パラメータの値（ArcotID の平文キー長など）を指しています。	**考えられる原因**： API に渡されたパラメータの値が最大許容値を超えています。 **解決方法**： パラメータの値が最大許容値を超えないように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 A の 「入力データの検証」を参照してください。

表 D-5. レスポンス コードと理由コード

レスポンス コード	理由コード	説明	考えられる障害の原因
1050	2054	操作に使用されたいずれかのパラメータの値が最小許容値を下回っています。	**考えられる原因**： API に渡されたパラメータの値が最小許容値を下回っています。 **解決方法**： パラメータの値が最小許容値以上になるように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 A の 「入力データの検証」を参照してください。
	2055	操作に使用されたいずれかのパラメータの値が無効です。	**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 ステータスとして許可された値は 0 と 1 です。この値を 5 に設定すると、このエラーが発生します。 **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 A の 「入力データの検証」を参照してください。
	2056	操作に使用されたいずれかのパラメータの値に無効な文字が含まれています。	**考えられる原因**： `ParameterKey` で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` で指定されるパラメータに有効な文字を入力します。

表 D-5. レスポンス コードと理由コード

レスポンス コード	理由コード	説明	考えられる障害の原因
1050	2057	操作に使用されたいずれかのパラメータがフォーマットの要件を満たしていません。	**考えられる原因**： ParameterKey で指定されたパラメータのフォーマットが無効です。 **解決方法**： ParameterKey で指定されるパラメータを有効なフォーマットにします。
	2058	パスワードに含まれる英字の数が最小許容値を下回っています。	**考えられる原因**： 指定したパスワードに含まれる英字の数が、パスワード強度ポリシーで許可された数を下回っています。 **解決方法**： 関連するパスワード ポリシーを参照し、パスワードの強度が正しく設定されていることを確認します。
	2059	パスワードに含まれる数字の数が最小許容値を下回っています。	**考えられる原因**： 指定したパスワードに含まれる数字の数が、パスワード強度ポリシーで許可された数を下回っています。 **解決方法**： 関連するパスワード ポリシーを参照し、パスワードの強度が正しく設定されていることを確認します。
	2060	パスワードに含まれる ASCII 特殊文字の数が最小許容値を下回っています。	**考えられる原因**： 指定したパスワードに含まれる ASCII 特殊文字の数が、パスワード強度ポリシーで許可された数を下回っています。 **解決方法**： 関連するパスワード ポリシーを参照し、パスワードの強度が正しく設定されていることを確認します。

表 D-5. レスポンス コードと理由コード

レスポンス コード	理由コード	説明	考えられる障害の原因
1050	6000	重複した質問はサポートされていません。	**考えられる原因**： 2 つ以上の質問が同じです。 **解決方法**： 異なる質問を指定します。
	6001	重複した回答はサポートされていません。	**考えられる原因**： 2 つ以上の回答が同じです。 **解決方法**： 異なる回答を指定します。
	6002	質問をいずれかの回答と同じにすることはできません。	**考えられる原因**： 質問がいずれかの回答と同じである可能性があります。 **解決方法**： 異なる質問と回答を指定します。
1051	0	無効なリクエスト。	**考えられる原因**： 受信したパケットが無効です。 解決方法： 1. 正しい SDK がサーバを参照していていることを確認します。 2. クライアント側に設定されたポートが適切なサーバ プロトコルを参照していることを確認します。
1100	0	組織が見つかりません。	**考えられる原因**： 指定した組織が存在しません。 解決方法： 1. 指定した名前を持つ組織が作成されているかどうかを確認します。 2. 組織を作成した後は、サーバのキャッシュをリフレッシュする必要がある場合があります。サーバのキャッシュをリフレッシュします。 3. 渡された組織の名前が正しいかどうかを確認します。

表 D-5. レスポンス コードと理由コード

レスポンス コード	理由コード	説明	考えられる障害の原因
1101	0	組織のクレデンシャルの設定が見つかりません。	**考えられる原因**： 指定したクレデンシャルの設定が存在しません。 **解決方法**： 1. この組織用の設定が作成されているかどうかを確認します。 2. 設定がこの組織に割り当てられているかどうかを確認します。 3. 設定を作成し、割り当てた場合は、キャッシュをリフレッシュする必要があります。サーバのキャッシュをリフレッシュします。
1102	0	ユーザが見つかりません。	**考えられる原因**： ユーザが存在しません。 **解決方法**： ユーザを作成するか、またはユーザの情報を正しく設定します。
1103	0	組織がアクティブではありません。	**考えられる原因**： 組織がアクティブではありません。 **解決方法**： Administration Console を使用して、組織をアクティブにします。
1150	0	ユーザのステータスがアクティブではありません。	**考えられる原因**： ユーザのステータスがアクティブではありません。 **解決方法**： ユーザをアクティブにします。
1151	0	ユーザはすでに存在しています。	**考えられる原因**： ユーザがシステム内にすでに存在します。 **解決方法**： 別のユーザ名でユーザを作成するか、またはユーザの詳細を正しく指定します。

表 D-5. レスポンス コードと理由コード

レスポンス コード	理由コード	説明	考えられる障害の原因
1152	0	クレデンシャルが無効です。	**考えられる原因**： ユーザのクレデンシャルがすでに存在します。 **解決方法**： ユーザのクレデンシャルがすでに存在する場合は、クレデンシャルを作成しないでください。
5500	0	プロセッサが無効です。 プロセッサとは、認証メカニズムのことです。	**考えられる原因**： リクエストされたメカニズムは、システムによってサポートされていません。 **解決方法**： システムによってサポートされているメカニズムを使用します。
5604	0	組織が無効です。	**考えられる原因**： 指定した組織名が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。
5605	0	SSL 信頼ストアの組織名が無効です。	**考えられる原因**： 指定した組織名が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。
5700	0	認証試行回数が限度を超えました。	**考えられる原因**： クレデンシャルの認証試行回数が許容限度を超えました。 **解決方法**： 管理者は、クレデンシャルのステータスをロック済みからアクティブに変更する必要があります。
5701	0	認証トークンの有効期限が切れています。	**考えられる原因**： ユーザによってサブミットされた認証トークンの有効期限が切れています。 **解決方法**： 再度認証します。

表 D-5. レスポンス コードと理由コード

レスポンス コード	理由コード	説明	考えられる障害の原因
5702	0	チャレンジの有効期限が切れています。	**考えられる原因**： チャレンジの有効期限が切れています。 **解決方法**： チャレンジを再度リクエストします。
5704	0	クレデンシャルの有効期限が切れています。	**考えられる原因**： ユーザによって指定されたクレデンシャルの有効期限が切れています。 **解決方法**： 新しいクレデンシャルを取得します。
5705	0	クレデンシャルがアクティブではありません。	**考えられる原因**： ユーザによって指定されたクレデンシャルがアクティブではありません。 **解決方法**： 管理者がクレデンシャルをアクティブにする必要があります。
5706	0	クレデンシャルが再発行されています。	**考えられる原因**： クレデンシャルが再発行されています。
5707	0	指定されたクレデンシャルの詳細が正しくありません。	**考えられる原因**： ユーザによって指定されたクレデンシャルの詳細が正しくありません。 **解決方法**： クレデンシャルの詳細を正しく指定します。
5800	0	ユーザのクレデンシャルが見つかりません。	**考えられる原因**： ユーザのクレデンシャルが存在しません。 **解決方法**： クレデンシャルを作成します。 **考えられる原因**： ユーザによって指定された詳細が正しくない可能性があります。 **解決方法**： 詳細を正しく指定します。
5801	0	ユーザのクレデンシャルがすでに存在します。	**考えられる原因**： ユーザのクレデンシャルがすでに存在します。

付録 E
WebFort サンプル アプリケーション

WebFort には、Java API の使用方法の実例を示すサンプル アプリケーションが付属しています。

重要 : サンプル アプリケーションは運用環境用ではないため、コードの参照目的にのみ使用してください。

サンプル アプリケーションを使用する前に、WebFort サーバと通信できるように設定する必要があります。この付録では、以下のトピックについて説明します。

- サンプル アプリケーションの設定
- ArcotID Client の選択
- サンプル アプリケーションのログ ファイルの設定

注 : この付録では、サンプル アプリケーションの設定方法と ArcotID Client タイプの選択方法についてのみ説明します。サンプル アプリケーションで簡単に実行できるその他の操作については取り上げません。

サンプル アプリケーションの設定

［**Setup**］ページを使用して、WebFort サーバのホスト名または IP アドレス、認証サービスおよび発行サービスを使用できるポート、およびサンプル アプリケーションのログ ファイル名と場所を設定できます。これを行うには、以下の手順に従います。

1. Web ブラウザ ウィンドウでサンプル アプリケーションを起動します。サンプル アプリケーションのデフォルトの URL は次のとおりです。

   http://<host>:<port>/webfort-6.2-sample-application

   ［WebFort 6.2 Sample Application］ページが表示されます。

2. サイドバーから、［**Setup**］ボタンを展開し、［**Server Connectivity**］リンクをクリックすると、［WebFort Server Connectivity］ページが表示されます。

3. 表 E-1 に示す設定パラメータの値を指定します。

表 E-1. 設定パラメータ

フィールド	デフォルト値	説明
[IP Address]	localhost	WebFort サーバが利用可能なホスト名または IP アドレス。
[Port]	**Authentication Service:** 9742 **Issuance Service:** 9744	認証サービスまたは発行サービスが利用可能なポート。
[Maximum Active Connections]	64	サンプル アプリケーションと WebFort サーバ間で維持される最大接続数。

4. ［**Set Up**］をクリックして接続を設定します。

追加の WebFort サーバ インスタンスと通信するようにサンプル アプリケーションを設定する方法

1. ［**Additional Server Configurations**］の前にある［+］記号をクリックします。
2. ［**IP Address**］および［**Port**］接続パラメータを指定します。
3. ［**Set Up**］をクリックして接続を設定します。

ArcotID Client の選択

ArcotID 関連の操作を実行する前に、使用する ArcotID Client の適切なタイプと、ダウンロードされた ArcotID を保存するストレージ メディアを選択する必要があります。

注： ArcotID Client のタイプおよびこのページで選択するダウンロード タイプは、現在のブラウザ セッションの間、永続します。

ArcotID Client を選択する方法

1. サイドバーで［**Setup**］ボタンを展開し、［**ArcotID Client**］リンクをクリックすると、［ArcotID Client Settings］ページが表示されます。

2. ［**Choose ArcotID Client**］セクションで、使用する ArcotID Client のタイプを選択します。

注： Flash クライアントを選択した場合は、サンプル アプリケーションが HTTPS に対応している必要があります。

3. ［**Choose ArcotID Download Type**］セクションで、ArcotID を保存するメディアのタイプを選択します。

   サポートされるダウンロード タイプの詳細については、5-70 ページの「ArcotID のダウンロード」を参照してください。

4. ［**Select**］をクリックして設定を保存します。

サンプル アプリケーションのログ ファイルの設定

サンプル アプリケーションがログの書き込みに使用するログ ファイルを設定する方法

1. サイドバーから、［**Setup**］ボタンを展開し、［**Logger**］リンクをクリックすると、［Logger Configuration］ページが表示されます。
2. ［**Log File Path**］フォルダに、サンプル アプリケーションのログ ファイルへの絶対パスを入力します。デフォルトでは、サンプル アプリケーションのログ ファイルは *<APP_SERVER_HOME>* フォルダ内に生成されます。
3. ［**Log Level**］を選択します。ログ レベルの詳細については、表 B-3 を参照してください。
4. ［**Set Up**］をクリックしてログ ファイルを設定します。

付録 F
用語集

Administration Console	WebFort サーバとそのコンポーネントの間の通信モードを設定し、管理タスクを実行するための Web ベースのコンソール。
Adobe 署名サービス プロトコル	「ASSP」を参照してください。
ArcotID	ソフトウェアの形でのハードウェア レベルの認証を可能にするセキュアなソフトウェア クレデンシャル。
ASSP	Acrobat と Reader のユーザがデジタル署名のローミング クレデンシャルにアクセスできるようにします。ASSP はハッシュを ASSP 対応サーバに渡して署名をし、その後でハッシュをエンド ユーザのドキュメントに埋め込みます。
Digest-MD5	広く使用される暗号化ハッシュ関数。ハッシュ値は 128 ビットです。
FYP（Forgot Your Password、パスワードを忘れた場合）	ユーザが ArcotID のパスワードを忘れた場合、ユーザと WebFort の間で Q&A セッションが実施されます。ユーザは簡単な質問に答えた後で、新しい ArcotID パスワードの入力を求められ、新しい ArcotID が発行されます。
OTP	「ワンタイム パスワード」を参照してください。
OTT	「ワンタイム トークン」を参照してください。
PKCS	RSA によって考案され発行された公開キー暗号化標準のグループ。詳細については、「公開キー暗号化法」を参照してください。
PKCS#12	パスワードベースの対称キーで保護された添付の公開キー証明書を使用して秘密キーを保存するために共通で使用されるファイル フォーマットの定義。
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。
Q&A	チャレンジ レスポンス認証メカニズム。Q&A を使用することにより、ユーザ エージェントとサーバの間で情報を交換するダイアログが表示されます。サーバは任意の数の質問をし、ユーザは正しい回答を提供します。
RADIUS **Remote Authentication Dial-In User Service**	一元化された認証、許可、および監査（AAA）用のプロトコル。

SAML	ID プロバイダ（アサーションを提供します）とサービス プロバイダ（アサーションを使用します）の間の認証データの交換のための XML 基準。
Security Assertion Markup Language	「SAML」を参照してください。
SHA（Secure Hash Algorithm）	暗号化ハッシュ関数のセット。
SSL **（Secure Sockets Layer）**	データの暗号化により、公衆ネットワーク間での通信のセキュリティ保護および認証を目的とするプロトコル。
SSO（Single Sign-On）	複数のシステム間で共有される単一の ID。SSO によって、ユーザがコンピュータまたはネットワークに 1 回ログオンすると、単一のクレデンシャルを使用して複数のアプリケーションおよびシステムにアクセスできます。
WebFort	エンド ユーザを認証するための強力な認証システム。
WebFort サーバ	WebFort SDK を介してアプリケーションと通信し、発行リクエストと認証リクエストを受理するサーバ コンポーネント。
インスタンス	指定されたポートで WebFort サーバが利用可能なシステム。
エラー メッセージ	エラーが発生した場合に、その状況に関してユーザ エージェントに報告するためにアプリケーションによって返されるメッセージ。
クレデンシャル	ユーザ ID を証明するもの。デジタル クレデンシャルがスマート 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。
クレデンシャル プロファイル	複数の組織および複数のユーザの間で共有可能な、すぐに使える共通のクレデンシャル設定。
サンプル アプリケーション	WebFort Java API の使用方法と、ユーザのアプリケーションを WebFort に統合する方法を示す例。また、WebFort が正常にインストールされたかどうか、発行操作と認証操作を実行できるかどうかを確認するために使用できます。
デジタル証明書	証明書は、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 PKI 暗号化法に基づいています。
デフォルトの組織	ユーザが Administration Console を展開するときにデフォルトで作成される組織。
ユーザ名 / 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。
ワンタイム トークン	認証成功後に WebFort サーバによって返されるトークン。

ワンタイム パスワード	単一セッションで有効なパスワード クレデンシャル。WebFort には、複数回使用できる OTP が実装されています。
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。
暗号化ハッシュ関数	認証などのセキュリティ関連アプリケーションで使用される、セキュリティ プロパティが追加されたハッシュ関数。
公開キー	公開キー暗号化法で使用される 1 対のキーの一方。公開キーは自由に配布され、証明書の一部として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。
公開キー暗号化法	秘密の共有について事前に合意しなくてもユーザが安全に通信することができる最新の暗号化形式。この方法では、対称暗号化法と異なり、全員が知っている公開キーと、公開キーと秘密キーのペアの所有者のみが知っている秘密キーという 2 つのキーを使用します。公開キー暗号化法は、非対称暗号化法とも呼ばれます。
再試行許容回数 N 回	ユーザが認証を失敗できる最大回数。この数を超えると、ユーザはロック アウトされます。
質問と回答	「Q&A」を参照してください。
組織	企業全体（すなわち会社）、特定の部門、または企業内の他のエンティティにマッピングできる WebFort の単位。
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 SDK	WebFort サーバに認証リクエストを転送するためにアプリケーションによって呼び出し可能な API。
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。
認証ポリシー	認証プロセスを制御するルールのセット。
発行 SDK	WebFort へのユーザ登録およびユーザのクレデンシャル作成のために WebFort サーバに発行リクエストを転送するためにアプリケーションによって呼び出し可能な API。
秘密キー	公開キー暗号化法で使用される 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。

索引