MyID Enterprise

Version 12.14

Credential Web Service

Lutterworth Hall, St Mary's Road, Lutterworth, Leicestershire, LE17 4PS, UK
www.intercede.com | info@intercede.com | @intercedemyid | +44 (0)1455 558111

Document reference: IMP1945-12.14.0-Enterprise

March 2025

Copyright

Trademarks

Licenses

This software includes packages provided under a variety of licenses. The *About the documentation* page in the HTML version of the MyID CMS documentation, available with the MyID CMS software or on the Intercede customer portal website, contains a full list.

Conventions used in this document

- Lists:
  - Numbered lists are used to show the steps involved in completing a task when the order is important.
  - Bulleted lists are used when the order is unimportant or to show alternatives.
- **Bold** is used for menu items and for labels.

  For example:
  - Record a valid email address in **'From' email address**.
  - Select **Save** from the **File** menu.
- *Italic* is used for emphasis:

  For example:
  - Copy the file *before* starting the installation.
  - Do *not* remove the files before you have backed them up.
- ***Bold and italic*** hyperlinks are used to identify the titles of other documents.

  For example: "See the ***Release Notes*** for further information."

  Unless otherwise explicitly stated, all referenced documentation is available on the product installation media.
- A `fixed width` font is used where the identification of spaces is important, including filenames, example SQL queries and any entries made directly into configuration files or the database.
- **Notes** are used to provide further information, including any prerequisites or configuration additional to the standard specifications.

  For example:

  Note: This issue only occurs if updating from a previous version.
- Warnings are used to indicate where failure to follow a particular instruction may result in either loss of data or the need to manually configure elements of the system.

  For example:

  Warning: You must take a backup of your database before making any changes to it.

Contents

1 Introduction

The Credential Web Service is an API for MyID® that supports the following:

- Allow external sources to request the issuance of Mobile Devices and Microsoft Virtual Smart Cards (VSCs) by MyID.
- Add and remove Additional Identities from MyID users.
- Unlock smart cards and tokens that support challenge/response unlocking.

This API contains no MyID authentication and relies instead on platform authentication such as mutual SSL on IIS. For information relating to configuring SSL on IIS, see the *Configuring SSL/TLS (HTTPS)* section in the ***Securing Websites and Web Services*** document.

2 Overview

This section provides details of the following:

- Prerequisites for using the web service.

  See section *2.1*, *Prerequisites*.
- Terminology used in the web service.

  See section *2.2*, *Terminology*.

2.1 Prerequisites

To implement the web service, you must have:

- A functioning installation of MyID.
- Knowledge of XML and web services.

2.1.1 Mobile identity issuance

If you want to use `RequestCredential` to enable issuing a mobile phone you must make sure that your system is configured for issuing mobile identities. See the ***Mobile Identity Management*** guide for details.

For mobile credentials, recipient user accounts must have an email address, and, for SMS-based notifications, a cell phone number.

2.1.2 Microsoft Virtual Smart Card issuance

If you want to use `RequestCredentialForDevice` to enable issuing a VSC you must make sure that your system is configured for issuing VSCs. See the***Microsoft VSC Integration Guide*** for details.

2.2 Terminology

The following terminology is used for the Credential Web Service.

Name	Description	Examples
Person	A user account within MyID that will require issuance of credentials to represent their identity.	Employees, system administrators, MyID operators.
Device	A physical entity (typically with some form of computer processor) that will require issuance of credentials to represent its identity. It may comprise of one or more Device Elements. These items are held within the `Carriers` database table in MyID.	Computers, mobile phones, tablets, routers, firewalls.
Profile	A definition within MyID of the credentials to be issued, the device element to be used, the lifetime of the credentials, issuance process to be used and access permissions to the profile. These are managed within the **Credential Profiles** workflow in MyID.	See the *Working with credential profiles* section in the ***Administration Guide*** for further details.
Job	An action within MyID that can be executed at a later date. Jobs may require validation before they are allowed to be actioned.	Card Issuance Job, Card Cancellation Job, Replacement Card Job.
Identity	A single Additional Identity. This identity consists of a DN, UPN, email address, and certificate policy.	A user may have an additional identity with credentials to authenticate to a second network. See the *Additional identities* section in the ***Administration Guide*** for further details.
Identities	A collection of Identity items.	A user may have a series of Additional Identities to access many secure environments.

3 Installing and configuring the web service

This section contains information on the following:

- Installing the web service using the MyID installation program.

  See section *3.1*, *Installing the web service*.
- Enabling access to the web service by configuring the security on the web server.

  See section *3.2*, *Enabling access to the web service*.
- Viewing the WSDL for the web service.

  See section *3.3*, *WSDL*.
- Setting up SMS notifications for the web service.

  See section *3.4*, *Setting up SMS notifications*.

3.1 Installing the web service

To enable this API, select the **Credential Web Service** option when installing MyID.

The Credential Web Service is written as a WCF service in C#, with the intention that it is to be hosted on an IIS server with very restrictive access.

If the CWS is installed on a server that does not have the MyID application server software installed on it, you must install DCOM proxies to link it to a MyID application server. For information on setting up DCOM proxies, see the *Split deployment* section in the ***Installation and Configuration Guide***.

3.2 Enabling access to the web service

See the *Configuring SSL/TLS (HTTPS)* section in the ***Securing Websites and Web Services*** document for details of setting up security on MyID web services. You must set up security on the Credential Web Service in the same way as the other MyID server-to-server web services.

Note: By default, all methods on the CWS are locked down.

3.2.1 Method lockdown

By default, all methods on CWS are blocked. Attempting to invoke one produces an HTTP 501 Error code. You can enable individual methods by modifying the `MethodAccessControl.xml` file in the `CredentialWebService` folder. This XML file allows you to explicitly enable or disable individual methods, and also to enable or disable any future methods implemented in later versions of the web service.

Note: Once you have modified and saved the `MethodAccessControl.xml` file, you must restart IIS for the settings to take effect.

Once deployed, this file is never updated by CWS upgrades. If CWS is updated to have new methods on it, those methods may not be enabled until you have manually adjusted this file.

3.2.2 Default settings

The default content of the `MethodAccessControl.xml` file is:

```
<?xml version="1.0" encoding="utf-8" ?>
<Methods default="no">
  <Method name="RequestCredential">no</Method>
  <Method name="RequestCredentialForDevice">no</Method>
  <Method name="AssignAdditionalIdentities">no</Method>
  <Method name="RemoveAdditionalIdentities">no</Method>
  <Method name="RemoveAllAdditionalIdentities">no</Method>
  <Method name="RequestUnlockCodeForDevice">no</Method>
  <Method name="RenewCertificate">no</Method>
  <Method name="RequestCertificate">no</Method>
  <Method name="GetCertificate">no</Method>
  <Method name="RequestCertificatePfx">no</Method>
  <Method name="GetCertificatePfx">no</Method>
  <Method name="IsAlive">no</Method>
</Methods>
```

To enable a method, change the corresponding value from `no` to `yes`. Any value other than `yes` (case sensitive) results in the method being blocked.

The attribute `default` on the root node dictates what should happen to methods not explicitly mentioned in the allowed list.

To allow all methods, and all future methods, you can modify the file to have the following content:

```
<?xml version="1.0" encoding="utf-8" ?>
<Methods default="yes" />
```

3.3 WSDL

You can obtain the WSDL for the web service by browsing to:

`http://myserver.example.com/CredentialWebService/CredentialAPI.svc?singleWsdl`

where `myserver.example.com` is the name of the server on which you have installed the Credential Web Service.

3.4 Setting up SMS notifications

You can configure MyID to sent one time password (OTP) codes as SMS messages.

3.4.1 Configuring SMS messages

To allow MyID to send SMS messages, set the **SMS email notifications** on the **General** tab of the **Operation Settings** workflow to `Yes`.

3.4.2 Customizing the format of SMS messages

By default, SMS messages are sent to an Email to SMS gateway, in the format `<cellnumber>@<gateway>`, where:

- `<cellnumber>` – the cell phone number from the user's record.
- `<gateway>` – the URL from the **SMS gateway URL for notifications** option on the **General** tab of the **Operation Settings** workflow.

For example: `00447700900123@smsgateway.com`

If this is not suitable, you can customize the `sp_CustomPrepareSMS` stored procedure in the MyID database.

3.4.3 Configuring CWS to send OTP codes through SMS

Note: You cannot use the **Send Mobile OTP via SMS** configuration option to send notifications – this option is used for mobile credentials requested through the MyID user interface only.

OTP codes sent from MyID for mobile credentials requested using CWS use the **Mobile Provisioning Code** email template. You can edit this template using the **Email Templates** workflow.

If you want to send the OTP code to the mobile device using SMS, you must edit the **Mobile Provisioning Code** email template and set the **Transport** option to **sms**.

If you do not want to send the OTP code to the end user (because the OTP is returned to the CWS method as the `CollectionCode`, and you have a system in place to relay this to the user) you can deselect the **Enabled** option on the **Mobile Provisioning Code** email template.

4 API reference

This section contains information about the Credential Web Service API, including:

- Naming conventions for classes.
  See section *4.1*, *Naming conventions*.
- Information on data types used when passing parameters to the API.
  See section *4.2*, *Data types*.
- The `RequestCredential` method.
  See section *4.3*, *RequestCredential*.
- The `RequestCredentialForDevice` method.
  See section *4.4*, *RequestCredentialForDevice*.
- The `AssignAdditionalIdentities` method.
  See section *4.5*, *AssignAdditionalIdentities*.
- The `RemoveAdditionalIdentities` method.
  See section *4.6*, *RemoveAdditionalIdentities*.
- The `RemoveAllAdditionalIdentities` method.
  See section *4.7*, *RemoveAllAdditionalIdentities*.
- The `RequestUnlockCodeForDevice` method.
  See section *4.8*, *RequestUnlockCodeForDevice*.
- The `RenewCertificate` method.
  See section *4.9*, *RenewCertificate*.
- The `RequestCertificate` method.
  See section *4.10*, *RequestCertificate*.
- The `GetCertificate` method.
  See section *4.11*, *GetCertificate*.
- The `RequestCertificatePfx` method.
  See section *4.12*, *RequestCertificatePfx*.
- The `GetCertificatePfx` method.
  See section *4.13*, *GetCertificatePfx*.
- The `IsAlive` method.
  See section *4.14*, *IsAlive*.

4.1 Naming conventions

The naming convention for classes are:

- `[Name]` – Enough information to uniquely identify an entity of type `[Name]`.
- `[Name]Details` – Enough information to register a new entity of type `[Name]`.
- `[Name]Response` – Details returned from the server about the consequences of what just happened.
- `[Name]s` – a collection of entities of type `[Name]`.

4.2 Data types

When passing optional parameters into the API it is quite forgiving when the data type is a `String` but for other data types it is not so forgiving.

The following examples pass an empty string as the parameter value:

```
<JobLabel/>
```

```
<JobLabel></JobLabel>
```

If the intention is to pass a null value then the node must be omitted entirely.

The following example will generate an error as the `ExplicitExpiryDate` is a data type `DateTime` which cannot accept an empty string so must contain a valid date or not be included.

```
<RequestCredential>
    <ProfileRequest>
        <ProfileName>My Credential Profile</ProfileName>
        <ExplicitExpiryDate/>
        <JobLabel/>
    </ProfileRequest>
    <UserAccount>
        <LogonName>Joe Bloggs</LogonName>
        <EmailAddress></EmailAddress>
        <MobileNumber/>
    </UserAccount>
</RequestCredential>
```

The following is a valid example with the optional nodes missing entirely.

```
<RequestCredential>
    <ProfileRequest>
        <ProfileName>My Credential Profile</ProfileName>
    </ProfileRequest>
    <UserAccount>
        <LogonName>Joe Bloggs</LogonName>
    </UserAccount>
</RequestCredential>
```

4.3 RequestCredential

```
ProfileRequestResponse RequestCredential(
    ProfileRequest profileRequest,
    UserAccount target);
```

Creates a job to issue a credential to a user. This is restricted to Identity Agent credentials. Requesting a credential that is not configured to be an Identity Agent credential will result in an error. See the ***Mobile Identity Management*** guide for further details.

Note: Requests through the API do not honor validation or role restrictions assigned to the credential profile.

The immediate response will return the fields `CollectionLink` and `CollectionCode`. These are the values that will be sent to the user as part of the issuance process.

The system will send the user an email message containing the link to start the process, and an SMS message containing the OTP required to complete the process; alternatively, your system may be configured so that users do not receive notifications as you have your own method of relaying this information to your users.

4.3.1 Inputs

Class	Field	Data Type	Description	Allow Null?
`ProfileRequest`			Parameters defining the credentials to be requested.	No
	`ProfileName`	String	The name of the credential profile that the Mobile Identity is to receive. Profiles are defined in MyID using the **Credential Profiles** workflow. The latest version of the specified profile will be used.	No
	`ExplicitExpiryDate`	DateTime	If present, the credential will expire on the specified date. It is not possible for this to extend the life of a credential beyond its profile value.	Yes
	`JobLabel`	String	If present, this will be passed through to the Job and can be used to search for the job.	Yes
`UserAccount`			Parameters defining the user who is to be issued the Mobile Identity.	No

Class	Field	Data Type	Description	Allow Null?
	LogonName	String	The identifier for the system account that will own the Device and the credentials. A Device (phone) can have only a single owner.	No
	EmailAddress	String	If specified this value will be used to update the user's current email address; if not specified their current email address will be left unchanged.	Yes
	MobileNumber	String	If specified this value will be used to update the user's current mobile phone number; if not specified their current mobile phone number will be left unchanged.	Yes

4.3.2 Output

Class	Field	Date Type	Description	Allow Null?
`ProfileRequestResponse`			Reports the details of the Job created.	No
	`JobID`	Integer	The MyID identifier for the request.	No
	`JobStatus`	String	"Created" if a certificate is to be issued prior to collection. This will be accompanied by the `DelayedProcess` node. "Awaiting Issue" if it is ready for immediate collection.	No
	`CollectionLink`	String	The link that the user should click to initiate the provisioning of the Mobile Identity. It is also sent in an email.	No
	`CollectionCode`	String	The code that the user should enter to authenticate the provisioning of the Mobile Identity. It is also sent in an SMS.	No
	`DelayedProcess`	Boolean	If this has a value of `true`, the request requires MyID to issue a certificate before it can be collected. Typically this takes a minute.	No

4.4 RequestCredentialForDevice

```
ProfileRequestResponse RequestCredentialForDevice(
    ProfileRequest profileRequest,
    UserAccount target,
    DeviceDetails deviceDetails);
```

Creates a job to issue a credential to a user for a specific device. This is restricted to VSC credentials and Identity Agent credentials. Requesting a credential that is not configured to be a VSC or an Identity Agent will result in an error.

4.4.1 Inputs

Class	Field	Data Type	Description	Allow Null?
ProfileRequest			Parameters defining the credentials to be requested.	No
	ProfileName	String	The name of the credential profile to use for the VSC. Profiles are defined in MyID using the **Credential Profiles** workflow. The latest version of the specified profile will be used.	No
	ExplicitExpiryDate	DateTime	If present, the credential will expire on the specified date. It is not possible for this to extend the life of a credential beyond its profile value.	Yes
	JobLabel	String	If present, this will be passed through to the Job and can be used to search for the job.	Yes
UserAccount			Parameters defining the user who is to be issued the credential.	No
	LogonName	String	The identifier for the system account that will be issued the credential.	No
	EmailAddress	String	Not used.	Yes
	MobileNumber	String	Not used.	Yes

Class	Field	Data Type	Description	Allow Null?
DeviceDetails			Identifies the Device that will host of the VSC. This must include at a minimum a SerialNumber or DNS.	No
	SerialNumber	String	A value to identify the Device uniquely. If not supplied, a GUID will be generated for the Device.	Yes
	Type	String	The category of the device; for example "Workstation", "Mobile", “Appliance”. If not supplied, it will default to “Asset”.	Yes
	Description	String	A text description of the Device.	Yes
	DNS	String	The DNS entry for the device on the network.	Yes
	DN	String	The DN for the device. If left blank this will be constructed from the DNS entry.	Yes
	Active	Boolean	Is the Device currently active? If blank defaults to false. Setting this to false will prevent it from being used in new requests.	Yes
	Model	String	The model of the device.	Yes
	OS	String	The operating system on the device.	Yes
	Fields		A collection of additional fields describing the Device. These fields are defined with the Project Designer tool. The Fields is a List of the type ExtendedField. ExtendedField contains two strings – Name and Value.	Yes

4.4.2 Output

Class	Field	Date Type	Description	Allow Null?
`ProfileRequestResponse`			Reports the details of the Job created.	No
	`JobID`	Integer	The MyID identifier for the request.	No
	`JobStatus`	String	"Awaiting Issue" if it is ready for immediate collection. “Awaiting Validation” if it requires validation before it can be collected.	No
	`CollectionLink`	String	Not used.	No
	`CollectionCode`	String	Not used.	No
	`DelayedProcess`	Boolean	Not used.	No

4.5 AssignAdditionalIdentities

```
int AssignAdditionalIdentities(
    UserAccount target,
    Identities identities);
```

The target is assigned the provided identities in addition to any they already have. A card update job is then created for each device the target has assigned to them that supports Additional Identities. Collecting this job will issue the new identities to that device.

4.5.1 Inputs

Class	Field	Data Type	Description	Allow Null?
UserAccount			Parameters defining the user who is to be assigned the Identities.	No
	LogonName	String	The identifier defining the user who is to be assigned the Identities.	No
	EmailAddress	String	The email address for the user who is to be assigned the Identities.	Yes
	MobileNumber	String	The mobile number for the user who is to be assigned the Identities.	Yes
Identities			A collection of Identity items.	No
> Identity				Yes
	DistinguishedName	String	The distinguished name for the Identity.	No
	UPN	String	The user principal name for the Identity.	No
	EmailAddress	String	The email address for the Identity.	No
	CertificatePolicy	String	The display name of the certificate policy that is to be issued, as displayed in the **Certificate Authorities** workflow; for example, "PIVAuthentication" or "CIVContentSigningCert on VINF2019R2DC15". Ensure the certificate policy is enabled for Identity Mapping in the **Certificate Authorities** workflow.	No

4.5.2 Output

The method returns an integer. This number is the number of device update jobs that were created as a result of this action.

Note: The method creates update jobs whether or not the **Automatically create card update jobs when additional identities are modified** configuration option is set; note, however, that the method reports only the number of update jobs created by the API itself; if the configuration option is set to Yes, the MyID server components create the update jobs, and the method reports that no update jobs have been created.

4.6 RemoveAdditionalIdentities

```
int RemoveAdditionalIdentities(
    UserAccount target,
    Identities identities);
```

The target will have the specified identities removed from their account. If the target has a matching identity, any certificates issued to it will be immediately revoked. A card update job is then created for each device the target has assigned to them that supports Additional Identities. Collecting this job will erase the removed identities from that device.

If no `CertificatePolicy` is specified for an Identity, all Identities that match the `DistinguishedName`, `UPN` and `EmailAddress`s are removed, regardless of their `CertificatePolicy`.

If the target does not have a matching Identity, no action is taken against that target. A device update job is still created.

4.6.1 Inputs

Class	Field	Data Type	Description	Allow Null?
`UserAccount`			Parameters defining the user who is to have Identities removed.	No
	`LogonName`	String	The identifier defining the user who is to have Identities removed.	No
	`EmailAddress`	String	The email address for the user who is to have Identities removed.	Yes
	`MobileNumber`	String	who is to have Identities removed.	Yes
`Identities`			A collection of Identity items.	No
`> Identity`				Yes
	`DistinguishedName`	String	The distinguished name for the Identity to be removed.	No
	`UPN`	String	The user principal name for the Identity to be removed.	No
	`EmailAddress`	String	The email address for the Identity to be removed.	No
	`CertificatePolicy`	String	The type of certificate that is to be issued. A blank value will match against all Identities.	Yes

4.6.2 Output

The method returns an integer. This number is the number of device update jobs that were created as a result of this action.

Note: The method creates update jobs whether or not the **Automatically create card update jobs when additional identities are modified** configuration option is set; note, however, that the method reports only the number of update jobs created by the API itself; if the configuration option is set to Yes, the MyID server components create the update jobs, and the method reports that no update jobs have been created.

4.7 RemoveAllAdditionalIdentities

```
int RemoveAllAdditionalIdentities(
    UserAccount target);
```

The target will have all Identities removed from their account. Any certificates issued to these identities will be immediately revoked. A card update job is then created for each device the target has assigned to them that supports Additional Identities. Collecting this job will erase the removed identities from that device.

If the target does not have a matching Identity, no action is taken against that target. A device update job is still created.

4.7.1 Inputs

Class	Field	Data Type	Description	Allow Null?
`UserAccount`			Parameters defining the user who is to have Identities removed.	No
	`LogonName`	String	The identifier defining the user who is to have Identities removed.	No
	`EmailAddress`	String	The email address for the user who is to have Identities removed.	Yes
	`MobileNumber`	String	who is to have Identities removed.	Yes

4.7.2 Output

The method returns an integer. This number is the number of device update jobs that were created as a result of this action.

Note: The method creates update jobs whether or not the **Automatically create card update jobs when additional identities are modified** configuration option is set; note, however, that the method reports only the number of update jobs created by the API itself; if the configuration option is set to Yes, the MyID server components create the update jobs, and the method reports that no update jobs have been created.

4.8 RequestUnlockCodeForDevice

```
UnlockCodeResponse RequestUnlockCodeForDevice(
    Device target, string challenge);
```

Most devices support a challenge/response mechanism to unblock the user PIN. Supplying the device details and a challenge from a device will calculate and return the unlock code for that device.

If the end user has more than one device, you can use the Reporting web service API to retrieve a list of devices per person, and then allow the user to select the device they want to unlock.

MyID supports this unlock method for all smart cards and tokens that you can issue in MyID, with the exception of the following device families:

- Android System Store
- Android Work Profile Store
- Android Work Profile Wi-Fi
- HID Prox
- iOS System Store
- MyID Platform TPM v1.2
- Windows Hello

Note: Your system may not support these types of device. This list contains device families that are available in the current version of MyID, were available in a previous version of MyID, or are currently available only as proof-of-concept in MyID, and is not to be interpreted as a list of device families that are currently supported in MyID.

4.8.1 Inputs

Class	Field	Data Type	Description	Allow Null?
`Device`			Describes the device to be unlocked.	No
	`SerialNumber`	`String`	A value to identify the device uniquely.	No
	`DeviceTypeName`	`String`	The type of the device.	No
`challenge`		`String`	The challenge code generated by the device. This must not include formatting. This is case sensitive.	No

4.8.2 Output

The output contains details about the device that the unlock code is for, and the unlock code itself. The unlock code is unformatted and case sensitive.

If the specified device is either unknown, unissued, or disabled, an error is returned.

Class	Field	Date Type	Description	Allow Null?
`UnlockResponse`				No
	`SerialNumber`	String	The unique identifier for the device.	No
	`DeviceTypeName`	String	The type of the device.	No
	`UnlockCode`	String	The unlock code for the device. This is unformatted and case sensitive.	No

4.9 RenewCertificate

```
int RenewCertificate(CertificateDetails certificate);
```

The `RenewCertificate` method requests a certificate renewal job for the given certificate. This follows the normal certificate renewal rules with the following additional rules:

- The auto renewal status of the certificate is ignored. Any valid certificate is a candidate for renewal.
- Revoked or partially issued certificates are not allowed to be renewed.
- Expired certificates are allowed to be renewed only if the configuration flag **Renew Expired Certs Via API** (on the **Certificates** tab of the **Operation Settings** workflow) is set to **Yes**.
- There is no minimum lifetime for the certificate before it can be renewed. This allows the renewal process to also be used for certificate re-issuance or migration.
- Due to inconsistent behavior between CAs, it is unreliable to identify a certificate based on its serial number.

Note: If the certificate renewal process results in the creation of a card renewal job because the target device is due to expire soon, an exception will be thrown.

4.9.1 Inputs

Class	Field	Data Type	Description	Allow Null?
`CertificateDetails`				No
	`SerialNumber`	`String`	The serial number of the device to which the certificate is to be issued. Maps to: `Certificates.DeviceSerialNo`	No
	`DeviceTypeName`	`String`	The device type name of the device to which the certificate is to be issued. Maps to: `Certificates.DeviceTypeName`	No
	`Policy`	`String`	The name for the certificate policy with which the certificate was issued. Maps to: `Certificates.CertPolicy`	No

The certificate that will be renewed will be the most recently issued certificate of the specified policy, on the specified device.

4.9.2 Output

The output is the job ID for the newly-created renewal job. Any failures to create this job will result in an exception being thrown. Possible error scenarios are:

- Certificate Not Found
- Certificate Revoked
- Certificate Expired
- Certificate Unsuitable
- Device Renewal Job Created

4.10 RequestCertificate

```
int RequestCertificate(MyIDCertificateRequest certificateRequest);
```

The `RequestCertificate` method requests a non-archived “Software Certificate” for the specified user, for use as a .CER certificate. This follows these rules:

- Credential Profile must be set for “Software Certificates Only”
- Credential Profile must have only one Certificate Policy

Note: Requests through the API do not honor validation or role restrictions assigned to the credential profile. You can request any credential profile.

4.10.1 Inputs

Class	Field	Data Type	Description	Allow Null?
`MyIDCertificateRequest`				No
	`CertificateRequestPKCS10`	`String`	The P10 Certificate Request, in Base64 format. **Note:** The P10 Certificate Request content provided to this endpoint (such as the SAN) is the responsibility of the caller.	No
	`CredentialProfileName`	`String`	The credential profile that this request will be made against. It must be a profile set for "Software Certificates Only". Maps to: `CardProfiles.Name`	No
	`PersonIdentifier`	`String`	The unique identifier for finding a person in MyID. This will map to whatever field is specified in the MyID.config file. See section *4.10.3, Mapping the user identifier*.	No

4.10.2 Output

The output is the `Certificates.ID` for the new Certificate Request. Any failures to create this request will result in an exception being thrown. Possible error scenarios are:

- Missing data
- The user has not been found
- Credential profile has not been found
- Credential profile is invalid for this certificate request

4.10.3 Mapping the user identifier

When requesting a certificate, you provide a unique identifier for a person in MyID. You can map this to a variety of fields in the MyID database; you can use the following mappings:

- `UserAccountID` – the `ID` from the `UserAccounts` table.
- `LogonName` – the `LogonName` from the `UserAccounts` table.
- `EmployeeID` – the `EmployeeID` from the `People` table.
- `UserObjectID` – the `ObjectID` from the `UserAccounts` table.
- `PersonID` – the `PersonID` from the `UserAccounts` table.
- Any field in the `UserAccountsEx` table; make sure it is unique to an individual person.

To map the user identifier, you must edit the `MyID.config` file for the CWS web service. By default, this file is in the following location:

```
C:\Program Files\Intercede\MyID\SSP\CredentialWebService
```

Open the file in a text editor, and amend the following line:

```
<add key="PersonIdentifier" value="LogonName"/>
```

Set the value to the field you want to map.

Once you have saved the file, recycle the app pool used for the Credential Web Service in IIS. This ensures that the web service is using the latest settings.

4.11 GetCertificate

```
string GetCertificate(int certificateId);
```

The `GetCertificate` method collects a CRT "Software Certificate" P7 for the specified request, for use as a .CER/.CRT certificate.

4.11.1 Inputs

Class	Field	Data Type	Description	Allow Null?
				No
	`certificateId`	`int`	The Certificate ID in the MyID `Certificates` table.	No

4.11.2 Output

The output is a string containing the P7 data, in Base64, for the Certificate Request. Any failures to create this request will result in an exception being thrown. Possible error scenarios are:

- Certificate not found
- Certificate not ready
- Certificate request failed

4.12 RequestCertificatePfx

```
int RequestCertificatePfx(MyIDCertificateRequestPfx certificateRequest);
```

The `RequestCertificatePfx` method requests a PFX “Software Certificate” P12 for the specified user, for use as a .PFX certificate. This follows these rules:

- Credential Profile must be set for “Software Certificates Only”
- Credential Profile must have only one Certificate Policy
- The Certificate Policy assigned to the Credential Profile must be marked for Archival

Note: Requests through the API do not honor validation or role restrictions assigned to the credential profile. You can request any credential profile.

4.12.1 Inputs

Class	Field	Data Type	Description	Allow Null?
`MyIDCertificateRequestPfx`				No
	`CredentialProfileName`	`String`	The credential profile that this request will be made against. It must be a profile set for “Software Certificates Only”. Maps to: `CardProfiles.Name`	No
	`PersonIdentifier`	`String`	The unique identifier for finding a person in MyID. This will map to whatever field is specified in the MyID.config file. See section *4.10.3*, *Mapping the user identifier*.	No

Class	Field	Data Type	Description	Allow Null?
	DistinguishedName	String	Optional Distinguished Name to use for the certificate request. This will be used instead of the Distinguished Name of the person in the MyID database. By default this feature is disabled and needs to be enabled in the MyID.config file if you want to use it. See section *4.12.3, Enabling the Distinguished Name override feature*.	Yes if configuration is set to "Optional", No if set to "Mandatory"

4.12.2 Output

The output is the Certificates.ID for the new Certificate Request. Any failures to create this request will result in an exception being thrown. Possible error scenarios are:

- Missing data
- The user has not been found
- Credential profile has not been found
- Credential profile is invalid for this certificate request
- Invalid certificate policy

4.12.3 Enabling the Distinguished Name override feature

You can override the Distinguished Name used for a PFX certificate request by providing the `DistinguishedName` to the endpoint. This value is then used instead of the value stored against the person in the MyID database. This a powerful feature, and therefore requires an extra configuration change to enable it.

To enable the Distinguished Name override, you must edit the `MyID.config` file for the CWS web service. By default, this file is in the following location:

```
C:\Program Files\Intercede\MyID\SSP\CredentialWebService
```

Open the file in a text editor, and amend the following line:

```
<add key=" AllowDistinguishedNameOverrride" value="Disabled"/>
```

Set the `value` to `Optional` or `Mandatory`.

Once you have saved the file, recycle the app pool used for the Credential Web Service in IIS. This ensures that the web service is using the latest settings.

4.13 GetCertificatePfx

```
string GetCertificatePfx(MyIDGetCertificatePfx requestDetails);
```

The `GetCertificatePfx` method collects a Password Protected PFX “Software Certificate” P12 for the specified request, for use as a .PFX certificate.

4.13.1 Inputs

Class	Field	Data Type	Description	Allow Null?
`MyIDGetCertificatePfx`				No
	`RequestId`	`String`	The Certificate ID in the MyID `Certificates` table.	No
	`Password`	`String`	The password to be used for protecting the PFX data.	No

4.13.2 Output

The output is a string containing the password protected P12 data, in Base64, for the Certificate Request. Any failures to create this request will result in an exception being thrown. Possible error scenarios are:

- Missing Data
- Certificate not found
- Invalid certificate policy
- Certificate not ready
- Certificate request failed

4.14 IsAlive

```
IsAliveResponse IsAlive()
```

The IsAlive method is used to determine if the Credential Web Service is running and able to connect to the database through COM.

4.14.1 Inputs

None.

4.14.2 Output

Class	Field	Data Type	Description	Allow Null?
`IsAliveResponse`			Reports the status of the web service	No
	`IsAlive`	`Boolean`	`True` if the web service is running and can successfully connect to the database through COM	No
	`Error`	`String`	If `IsAlive` is `False`, returns the reason for the failure.	Yes

5 Troubleshooting and known issues

This section contains information on the following:

- Troubleshooting problems that may occur when using the Credential Web Service.

  See section *5.1*, *Troubleshooting*.

- Known issues.

  See section *5.2*, *Known issues*.

5.1 Troubleshooting

- Issue with notifications not being sent

  If, after installing the Credential Web Service, you experience a problem with email or SMS notifications not being sent, restart the MyID application server to refresh the cache and pick up the Credential Web Service configuration changes.

- Phone number not appearing on screen

  If you experience an issue with the mobile phone number not appearing on the stage immediately before sending the SMS (which is required to confirm that the SMS is being sent to the correct mobile device), make sure that the **Mobile Provision Via SMS** option (on the **Devices** tab of the **Operation Settings** workflow) is set to `Yes`.

- Certificate policy eligibility

  The eligibility for a certificate policy to be used for an Additional Identity is checked at the point of issuance, not at the point of assignment. Ensure that only certificate policies that are configured in MyID to allow Identity Mapping are used when assigning identities to users.

5.2 Known issues

- **IKB-278 – Incorrect count of number of jobs returned after requesting addition or removal of additional identities**

  If the calling system requests creation or removal of additional identities using the following methods:

  - `AssignAdditionalIdentities`
  - `RemoveAdditionalIdentities`
  - `RemoveAllAdditionalIdentities`

  the response will be either an error response or an integer. The integer may not accurately represent the number of jobs created; however, any integer (including 0) in this response does indicate that at least one job has been created. The number reported will be accurate only if the configuration option **Automatically create card update jobs when additional identities are modified** is set to No.

6 Error messages

The following table lists the error messages that appear, and the requests that may cause them.

Message	Request
An unknown error has occurred.	Any
Only identity agent issuance is currently supported.	`RequestCredential`
Identity agent issuance is currently disabled.	`RequestCredential`
Credential profile has not been found.	`RequestCredential` `RequestCredentialForDevice`
The user has not been found.	`RequestCredential` `RequestCredentialForDevice`
The user has no contact details specified.	`RequestCredential`
The device must specify a DNS or SerialNumber.	`RequestCredentialForDevice`
More than one device was found, please make your criteria more specific.	`RequestCredential`
The device could not be created.	`RequestCredentialForDevice`
Credential profile is incompatible with this device.	`RequestCredentialForDevice`
The device must be active to request a credential.	`RequestCredentialForDevice`
The maximum number of Additional Identities has been exceeded.	`AssignAdditionalIdentities`
Credential is unknown.	`RequestUnlockCodeForDevice`
Credential is not issued.	`RequestUnlockCodeForDevice`
Credential is disabled.	`RequestUnlockCodeForDevice`
Unlock codes are not supported for this credential.	`RequestUnlockCodeForDevice`
Certificate Not Found.	`RenewCertificate`
Certificate is Revoked.	`RenewCertificate`
Certificate has Expired.	`RenewCertificate`
Certificate is Unsuitable.	`RenewCertificate`
Card Renewal Job created.	`RenewCertificate`
Missing data	`RequestCertificate` `RequestCertificatePfx`
Too many records returned	`RequestCertificate` `RequestCertificatePfx`
Credential profile is invalid for this certificate request	`RequestCertificate` `RequestCertificatePfx`
The user has not been found	`RequestCertificate` `RequestCertificatePfx`

Message	Request
Credential profile has not been found	`RequestCertificate` `RequestCertificatePfx`
Invalid certificate policy	`RequestCertificatePfx`
Value is not Hexadecimal	`GetCertificate`
Certificate Not Found	`GetCertificate` `GetCertificatePfx`
Certificate not ready	`GetCertificate` `GetCertificatePfx`
Certificate request failed	`GetCertificate` `GetCertificatePfx`