MyID

Version 11.7

Lifecycle API

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

Document reference: IMP1954-09

October 2020

Copyright

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 and to indicate references to other sections within the current document:

  For example:
  - "Copy the file *before* starting the installation"
  - "See *Issuing a Card* for further information"

- ***Bold and italic*** 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 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

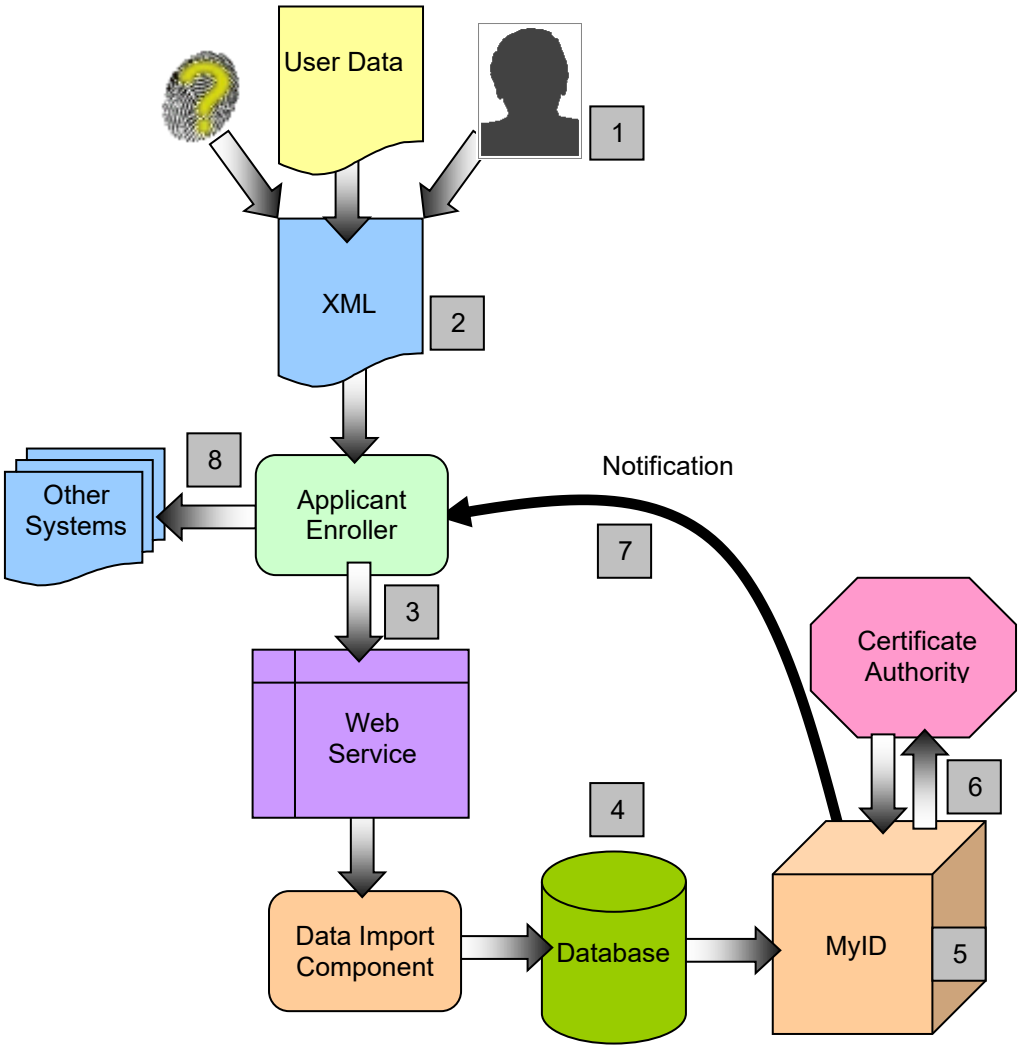
The MyID® Lifecycle API allows you to add and update users in the MyID database by sending XML documents to a web service. You can also request credentials that can then be collected from any MyID card issuing station.

Note: You cannot use the Lifecycle API to request mobile credentials; instead, you must use the Credential Web Service. See the ***Credential Web Service*** document for details.

1.1 Change history

Version	Description
IMP1954-01	Released with MyID 11.0.
IMP1954-02	Released with MyID 11.1.
IMP1954-03	Released with MyID 11.2.
IMP1954-04	Released with MyID 11.3.
IMP1954-05	Released with MyID 11.4.
IMP1954-06	Released with MyID 11.5.
IMP1954-07	Released with MyID 11.6.
IMP1954-08	Released with MyID 11.6.1. Added `VettingDate` field.
IMP1954-09	Released with MyID 11.7. Added `MaxRequestExpiryDate` field.

1.2 The Lifecycle API in the card issuance process



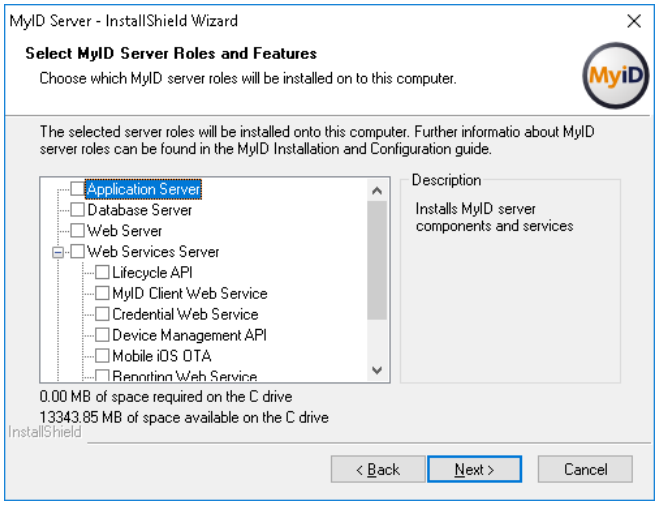
1. External to MyID, capture the user data, including the user's picture.
2. Compile the data into an XML document. Encode the image into this file.
3. Send the XML data to the MyIDEnroll web service.
4. MyID imports the user data, then passes back an XML report containing information on whether the user's record was imported correctly. MyID creates a card request for the added person.
5. If the credential profile being issued requires validation, the card request must be approved in MyID.
6. When the card is issued, MyID submits requests for certificates to the certificate authority.
7. MyID sends a notification through HTTP to the applicant enroller that the card has been issued.

   Contact Intercede professional services for information on setting up notifications.
8. The enroller can then notify other systems that the card has been issued and activated.

2 Lifecycle API Web Service

2.1 Installing the service

To install this API, you must select the **Lifecycle API** option when you install MyID. See the ***Installation and Configuration Guide*** for details.



2.2 Security settings

When the Lifecycle API web service (MyIDEnroll) is installed, it is configured so that it is impossible to authenticate to the service (a combination of **ASP.Net Impersonation** and **Windows Authentication** means you cannot call the web service). If you try to call the web service, you will see an error similar to the following:

```
HTTP Error 500.24 - Internal Server Error
```

The Lifecycle API is a powerful feature that allows a caller to add users, edit users, and request issuance of credentials; therefore you *must* restrict its usage.

If your system requires the Lifecycle API to be available to trusted callers, you must configure IIS to allow this.

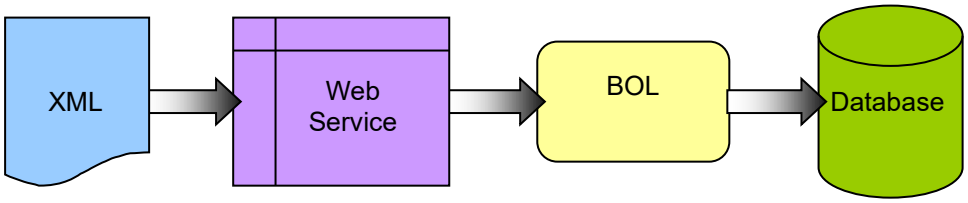
Review the authentication requirements for the MyIDEnroll web service in IIS and configure it. For more information, see the ***Securing Websites and Web Services*** document.

For production systems, you must ensure anonymous access is disabled.

Once you have configured the authentication mechanism you want to use for MyIDEnroll, disable the **ASP.NET Impersonation** option for the web service in IIS; if you are not using **Windows Authentication**, you must also disable this option.

2.3 Overview

The web service communicates with the MyID server components (the BOL) to import the data.



You post the XML document to the web service. This service processes the XML and passes the transformed data to the BOL.

The BOL performs actions on the data, updating the database if necessary, then returns a response to the web service describing the actions it has carried out. The web service then returns the status to the sender.

2.4 Increasing the maximum pool size

When using MyID web services (for example, MyIDEnroll) under heavy load, you may experience some problems; these may be caused by the .Net framework not releasing COM components correctly, affecting the eConfiguration objects used by MyID.

To solve this issue, you can increase the maximum pool size:

1. Open the Windows Component Services MMC snapin.
2. Expand **Component Service > Computers > My Computer > COM+ Applications > Edefice_BOL > Components**.
3. Right-click **EConfiguration.Configuration.1**, then from the pop-up menu select **Properties**.
4. Click the **Activation** tab.
5. Set the **Maximum Pool Size** to `100`.
6. Click **OK**.

Once you have made the change, you must restart the Edefice_BOL component.

1. Open the Windows Component Services MMC snapin.
2. Expand **Component Service > Computers > My Computer > COM+ Applications**.
3. Right-click **Edefice_BOL** and select **Shut down** from the pop-up menu.

   The component will start again automatically when it is next needed.

2.5 Timeouts

If the import process takes too long to process, the Lifecycle API returns an error message stating that the process timed out. For example:

```
<PivImportResponse
xmlns="http://www.intercede.com/MyIDPIVSchema/PIVImportResponse">
      <error>
            <description>System.Runtime.InteropServices.COMException (0x800468CA):
Error: 0x800468ca : AsyncJob - Timed out waiting for import response
Info: ASyncImport.EXE
------------------------
Exception raised in function: CFileImportImpl::ProcessFileImport
In file .\FileImportImpl.cpp at line 475

   at EDEFICE_BOLLib.EdeficeBOLClass.ProcessFileImport(String bstrXMLInstructions)
   at MyIDEnroll.transferData(XmlDocument outputXML)
   at MyIDEnroll.PIVXMLWebImport(String xmlIn)</description>
      </error>
</PivImportResponse>
```

The timeout is set to 60 seconds by default. If you want to change this value, you can edit the registry on the MyID application server.

In the following area of the registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Intercede\Edefice\ImportExport
```

Set the `Timeout` DWORD value to the number of seconds the service will wait for an import to complete.

2.6 Supported options

Items that are marked as "not supported" are part of the schema definition, but must not be used. These items may be legacy functionality that is no longer available, reserved for future use, or require additional changes to your MyID server to enable their use.

2.7 Known issues

- **IKB-67 – Limit to concurrent transactions**

  Performance of this web service may vary based on a range of factors. This includes network load, server performance, and the volume of transactions taking place within the MyID architecture. For example, the API is expected to operate correctly with up to 15 concurrent transactions taking place. However, beyond this level, further issues may occur such as database timeouts. Increasing server performance may help overcome these issues, but the predicted volumes for your business processes should be part of your capacity planning of the solution.

- **IKB-288 – Job ID not returned for some job types**

  When an update or certificate renewal job for a device is requested using the Lifecycle API, an issue has been identified that causes the Job ID not to be returned to the calling system.

  The response received is:

  ```
  <CardRequest>0</CardRequest>
  ```

  However, the job will have been created successfully in MyID.

3 Preparing the data for import

You must create an XML document containing all of the data you want to import using the XML web import service.

The XML document may contain information such as:

- Data for a user
- Core user fields
- User security phrases
- User role and scope to use
- Entrust profile to use
- Extended user fields
- Core group fields
- Extended group fields
- Credential profile to use
- Card expiry date
- Card renewal request
- Images: user photographs and document scans

To make sure the data is transformed into the correct format, you must use an XML schema supported by MyID. See section *9*, *XML schemas* for details of the schemas you can use.

Different schemas are available for adding new users to MyID and updating existing users.

3.1 CMS and PIV differences

While the CMS and PIV schemas have similarities, the PIV schema contains extra information to allow you to import information about PIV applicants and PIV agencies, and is appropriate only for PIV-based systems.

In addition, where the user node in CMS schemas is called `<User>`, in PIV the corresponding node is `<Applicant>`. Similarly, in CMS schemas, the group node is called `<Group>`, while in PIV the corresponding node is `<Agency>`.

Note: Do not use the CMS schemas to request or update PIV cards. You can, however, use the CMS schemas to request or update CIV cards.

3.2 Escaping characters

You must use valid XML for your import files; this includes escaping special characters such as `&`, `<` and `>`. If the XML is invalid, the import service will not process it.

3.2.1 DNs

MyID assumes that the DN is correctly escaped; if it is not, any people in that group will have invalid DNs. To correct this, you must remove the person, correct the group base DN, then add the person again.

The characters that you must escape are:

```
, = + < > \ # ; "
```

To use any of these characters, you must enclose them in double quotes. Additionally, you must prefix any " (double quote) or \ (slash) character with a \ (slash).

For example:

```
John Smith, Jnr
```

should be escaped as

```
cn="John Smith, Jnr", o=...
```

If you have an o value of:

```
Smith "Budget" Cars
```

this should be escaped as:

```
o="Smith \"Budget\" Cars", c=...
```

3.3 Encoding images

You must encode the images with Base64 encoding within the XML document. The XML web import service can then extract the image data and upload the images to the MyID database.

3.4 Advanced settings

Settings are stored in a file called `settings.xml` in the same folder as the `MyIDEnroll.asmx` file.

The following parameters are available:

- `SourceID`
- `IssueDate`
- `GenerateUserDN`
- `ActionOnDuplicate`
- `RolesActionOnDuplicate`
- `DeleteMissingUsers`
- `PushToLDAP`
- `CreateUnknownGroups` – not supported
- `AuditAll`
- `DataType`
- `CheckImportResponseSchema`
- `SynchronousImport` – not supported. Do not use.
- `DefaultUserRole`
- `DisallowCertificateSuspension` – PIV systems only.
- `CardRequestThrottling` – PIV systems only.

You can include these settings in your XML import document; these settings override any default settings from the `settings.xml` file. If the settings are missing from the XML document, the values in the `settings.xml` file are used instead.

See section 5, *Advanced settings* for details of the parameters available in the XML import document.

3.5 Adding and modifying users

When you provide the XML import information to the import web service, if the user does not already exist, the user is created. If the user already exists, but you provide different details for the user, the user is updated in the MyID database.

3.6 Requesting cards

When you provide the XML information to the import web service, if you provide details of a card in the `<Card>` node, a card request is created.

Note: If the user already exists in the database, you must set the `<Renewal>` node to `true`, or card request jobs will *not* be created.

3.7 Enabling, disabling and removing users

You can use the import feature to enable, disable or remove users. To disable or remove a user, add a keyword to the XML import file.

3.7.1 Removing users

In the user node (`User` for CMS, `Applicant` for PIV), add an `<Actions>` node. Within this, add the following to remove the user:

```
<ApplicantAction>Remove</ApplicantAction>
<StatusMappingID>ID</StatusMappingID>
<RevocationComment>Comment</RevocationComment>
```

Where *`ID`* is the ID of the revocation or suspension code from the `StatusMapping` table in the database (see section *3.7.5*, *Status mapping codes*), and *`Comment`* specifies the comment used when revoking or suspending user certificates, and also appears in the system audit reports.

3.7.2 Disabling users

To disable a user, use the following:

```
<ApplicantAction>Disable</ApplicantAction>
<StatusMappingID>ID</StatusMappingID>
<RevocationComment>Comment</RevocationComment>
```

Where *`ID`* is the ID of the revocation or suspension code from the `StatusMapping` table in the database (see section *3.7.5*, *Status mapping codes*), and *`Comment`* specifies the comment used when revoking or suspending user certificates, and also appears in the system audit reports.

The user is removed or disabled, and their cards and certificates are canceled or suspended as appropriate.

- IKB-261 – User certificates not revoked when suspending a user with an `ApplicantAction` of `Disable`

  When disabling a user using the Lifecycle API, certificates are suspended instead of fully revoked, when the `StatusMappingID` specified should result in a full revocation. To work around this issue, trigger device revocation before suspending the user account using `CancelDevices`; see section *3.10.1*, *Canceling cards*.

  Alternatively, adding `DisallowCertificateSuspension` to the `settings.xml` with a value of `1` will allow the status mapping to be used to determine specific revocation actions that take place; see section *5*, *Advanced settings* for details.

3.7.3 Enabling users

If you omit the `<Actions>` node, and the user is already in the database and disabled, the import process enables the user again.

3.7.4 Updating users who were imported from LDAP

If you are using the Lifecycle API to update users who were originally imported from LDAP, and you want the changes to be pushed to the directory, you must include the user's `Group` node to allow the update to determine the LDAP ID.

3.7.5 Status mapping codes

The following table lists the default status mapping codes from the `StatusMapping` table in the MyID database.

Note: These are the default settings, and may be customized for your own installation. When using the Lifecycle API, you cannot specify the mapping codes with negative IDs – these are reserved for system use.

ID	Status	Description
-11	Other Device issued - cancel	System use only, do not use.
-10	Device issued disabled	System use only, do not use.
-9	Automated card update	System use only, do not use.
-8	Automated shared certificate update	System use only, do not use.
-7	Update requested by API	System use only, do not use.
-6	Other Device issued	System use only, do not use.
-5	Revoked/Suspend for timeout at deferred collection	System use only, do not use.
-4	Revoked/Suspend due to timeout in issuance	System use only, do not use.
-3	Revoked/Suspended due to user disabled in LDAP	System use only, do not use.
-2	Revoked due to user removal from LDAP	System use only, do not use.
-1	Revoked due to too many suspensions	System use only, do not use.
0	Unspecified or Automated Processes	System use only, do not use.
1	Lost	The device has been permanently lost.
2	Damaged	The device has been rendered inoperable and should be permanently replaced.
3	Stolen	The device has been stolen and should be replaced.
4	Forgotten	The device has been misplaced and a temporary replacement is required.
5	Permanently Blocked	The device should no longer be used.
6	Compromised	The device has been compromised. A replacement is required.
7	Device holder on leave	The device holder is away temporarily and the device should no longer work until it is re-enabled.

ID	Status	Description
8	Pending Investigation	The device is being investigated and should be suspended temporarily. The device should no longer work until it is re-enabled.
9	Non-payment of services	The device should be disabled permanently.
10	Device holder leaving or changing role	The device holder has left their current role and their device should be blocked permanently.
11	Device holder details change	Information about the device holder has been changed and a new device is needed.
12	Pending Activation	The device is temporary suspended pending activation.
15	Revocation (other)	The device has been permanently blocked for a user-defined reason.
16	Suspension (other)	The device has been temporarily suspended for a user-defined reason.
17	Found Original	This temporary device has been revoked because the device holder found their original device.
18	Original device Compromised	The original device has been compromised so this device must also be considered to have been compromised.
19	Request device Renewal	This device is being replaced or refreshed and shall no longer be active on the system; all certificates will be available for the replacement device.
20	Batch Failed	The device was as part of a batch that failed and must be permanently cancelled.
21	Bureau Failure	There was a failure at the bureau and the device must be permanently cancelled.
22	Processing Failure	There was a processing failure and the device must be permanently cancelled.
25	Poor print quality	The device was printed to an unacceptable standard and must be permanently cancelled.
26	Printing misaligned	The printing on the device is not aligned properly and the device must be permanently cancelled.
27	Poor lamination quality	The device was laminated incorrectly and must be permanently cancelled.
28	Incorrect layout printed	The device was printed with the incorrect layout and must be permanently cancelled.
32	Cancel device and leave Certificates	The device is to be cancel but the certificates are to remain active.
33	Cancel Certificates and leave device	The certificates are to be cancelled but the device is to remain active.
47	Derived Credential Original Revoked	Cancels Derived Credentials on notification of the original credential being revoked.
66	Derived Credential Notification Listener	Cancels Derived Credentials on notification of the original credential being revoked.
70	Compromised – Reissue Shared Certificates	A device has been compromised; any certificates shared with other devices will be reissued.
71	Credential Profile Update (no revocation)	There has been an update to the credential profile, but no certificates will be revoked.
72	Credential Profile Update (full revocation)	There has been an update to the credential profile and all certificates will be revoked.

ID	Status	Description
73	Details Change – re-issue archived certificates	The user's details have been changed and all certificates must be re-issued. Non-archived certificates will not be revoked.
74	Mobile Issued	Mobile issued, cancel Certificate Package.
75	Reissue credentials	Credentials are being re-issued.
76	8 Hour access	8 Hour access.
77	24 Hour access	24 Hour access
78	2 Day access	2 Day access.
79	1 Week access	1 Week access.
80	Cancel temporary card during replacement	Cancel temporary card during replacement certificates are revoked and device cancelled.
81	Unrestricted access	Unrestricted access.
82	Reissue mobile	Reissue a partially issued mobile credential.
83	User details have changed	Used when requesting a card update and the user's details have changed, requiring a reprovision of the card.
84	There is a problem with the device	Used when requesting a card update and there is a problem with the card, requiring a reprovision of the card.
85	New credential profile needs to be applied	Used when requesting a card update and significant credential profile changes, for example data model changes, are needed, requiring a reprovision of the card.
86	New certificates need to be added to the device	Used when requesting a card update when there are only certificate changes required.

Status mapping actions for PIV systems

The following table shows the specific actions for certificates and archived certificates for each status mapping code on a PIV system.

ID	PKI Action	Archive PKI Action (PIV)
-11	Revoke.	Revoke.
-10	Suspend.	Suspend.
-9	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-8	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-7	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-6	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-5	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-4	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-3	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-2	Revoke.	Revoke.
-1	Revoke.	Revoke.
0	Revoke.	Revoke.
1	Revoke.	Revoke.

ID	PKI Action	Archive PKI Action (PIV)
2	Revoke.	Revoke.
3	Revoke.	Revoke.
4	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
5	Revoke.	Revoke.
6	Revoke.	Revoke.
7	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
8	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
9	Revoke.	Revoke.
10	Revoke.	Revoke.
11	Revoke.	Revoke.
12	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
15	Revoke.	Revoke.
16	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
17	Revoke.	Do not revoke. Certificate is available for live recovery on subsequent devices.
18	Revoke.	Revoke.
19	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
20	Revoke.	Revoke.
21	Revoke.	Revoke.
22	Do not revoke.	Revoke.
25	Revoke.	Revoke.
26	Revoke.	Revoke.
27	Revoke.	Revoke.
28	Revoke.	Revoke.
32	Do not revoke. Certificate is available for live recovery on subsequent devices.	Do not revoke. Certificate is available for live recovery on subsequent devices.
33	Revoke.	Revoke.
66	Revoke.	Revoke.
70	Revoke.	Revoke.
71	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
72	Revoke.	Revoke.
73	Do not revoke; allow recovery to new device.	Revoke.
74	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
75	Do not revoke. Certificate is available for live recovery on subsequent devices.	Do not revoke. Certificate is available for live recovery on subsequent devices.

ID	PKI Action	Archive PKI Action (PIV)
76	Revoke.	Revoke.
77	Revoke.	Revoke.
78	Revoke.	Revoke.
79	Revoke.	Revoke.
80	Revoke.	Revoke.
81	Revoke.	Revoke.
82	Revoke.	Revoke.
83	Revoke.	Revoke.
84	Revoke.	Revoke.
85	Revoke.	Revoke.
86	Revoke.	Revoke.

Status mapping actions for non-PIV systems

The following table shows the specific actions for certificates and archived certificates for each status mapping code on a non-PIV system.

ID	PKI Action	Archive PKI Action (non-PIV)
-11	Revoke.	Revoke.
-10	Suspend.	Suspend.
-9	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-8	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-7	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-6	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-5	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-4	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-3	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
-2	Revoke.	Revoke.
-1	Revoke.	Revoke.
0	Revoke.	Revoke.
1	Revoke.	Revoke.
2	Revoke.	Do not revoke. Certificate is available for live recovery on subsequent devices.
3	Revoke.	Revoke.
4	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
5	Revoke.	Do not revoke. Certificate is available for live recovery on subsequent devices.
6	Revoke.	Revoke.
7	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.

ID	PKI Action	Archive PKI Action (non-PIV)
8	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
9	Revoke.	Revoke.
10	Revoke.	Revoke.
11	Revoke.	Revoke.
12	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
15	Revoke.	Revoke.
16	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
17	Revoke.	Do not revoke. Certificate is available for live recovery on subsequent devices.
18	Revoke.	Revoke.
19	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
20	Revoke.	Revoke.
21	Revoke.	Revoke.
22	Do not revoke.	Revoke.
25	Revoke.	Revoke.
26	Revoke.	Revoke.
27	Revoke.	Revoke.
28	Revoke.	Revoke.
32	Do not revoke. Certificate is available for live recovery on subsequent devices.	Do not revoke. Certificate is available for live recovery on subsequent devices.
33	Revoke.	Revoke.
66	Revoke.	Revoke.
70	Revoke.	Revoke.
71	Do not revoke; allow recovery to new device.	Do not revoke. Certificate is available for live recovery on subsequent devices.
72	Revoke.	Revoke.
73	Do not revoke; allow recovery to new device.	Revoke.
74	Suspend.	Do not revoke. Certificate is available for live recovery on subsequent devices.
75	Do not revoke. Certificate is available for live recovery on subsequent devices.	Do not revoke. Certificate is available for live recovery on subsequent devices.
76	Revoke.	Revoke.
77	Revoke.	Revoke.
78	Revoke.	Revoke.
79	Revoke.	Revoke.
80	Revoke.	Revoke.
81	Revoke.	Revoke.
82	Revoke.	Revoke.
83	Revoke.	Revoke.

ID	PKI Action	Archive PKI Action (non-PIV)
84	Revoke.	Revoke.
85	Revoke.	Revoke.
86	Revoke.	Revoke.

3.8 Importing additional fields

Note: Attempting to import an additional field that does not exist in the database causes a failure.

3.8.1 Additional user fields

You can import information into additional extended fields using the data import. In the user node, add an `<AdditionalFields>` node. Within this, add a node for each additional user (`Xu`) field you want to import for the user.

For example:

```
<AdditionalFields>
  <Xu1000>User Information</Xu1000>
  <Xu1001>More User Information</Xu1001>
  <Xu1002>Even More User Information</Xu1002>
</AdditionalFields>
```

3.8.2 Additional group fields

You can also add additional information to the group. For CMS, within the group node, add an `<AdditionalFields>` node. For PIV, within the agency node, add an `<AdditionalAgencyFields>` node.

Within this, add a node for each additional group (`Xg`) field that you want to import into the group's record.

For example:

```
<AdditionalFields>
  <Xg1000>Group Information</Xg1000>
</AdditionalFields>
```

If you add extended user (`Xu`) fields to the `AdditionalFields` node in the group node, or extended group (`Xg`) fields to the `AdditionalFields` node in the user node, these are ignored.

3.8.3 Additional job fields

To add additional information to the job created, add an `<AdditionalFields>` node to the `<Card>` node. Within this, add a node for each additional job (`Xj`) field you want to import.

3.9 Importing users into Entrust

If you add the `<EntrustProfile>` node to the `<Account>` node, you can specify an Entrust profile for the user. If this node is present in the import XML, the user is added to Entrust using the specified Entrust Profile Template.

For example:

```
<Account>
  ...
  <EntrustProfile>MyEntrustTemplate</EntrustProfile>
</Account>
```

3.10 Canceling credentials and jobs

You can cancel credentials (for example, smart cards) and jobs using the `CMSCardRequest` method through the import service.

Note: If you want to cancel credentials or jobs, you do not have to include a `Card` node unless you also want to request a card at the same time.

3.10.1 Canceling cards

To cancel all of a user's credentials, pass an `ApplicantAction` of `CancelDevices`; for example:

```
<Actions>
  <ApplicantAction>CancelDevices</ApplicantAction>
  <StatusMappingID>ID</StatusMappingID>
  <RevocationComment>Comment</RevocationComment>
</Actions>
```

To cancel a specific credential, pass an `ApplicantAction` of `CancelDevice`. To identify the card you must include a `Device` node containing the card's serial number and device type.

You can cancel more than one credential in a single request by including more than one `Device` element.

Because not all systems track the serial number of the card, you can use a lookup mechanism to identify the card. For each field you want to look up, include the following:

```
<DeviceIdentifier>
  <SerialNumber>[identifier]</SerialNumber>
  <SerialNumberField>[field to look up]</SerialNumberField>
</DeviceIdentifier>
```

The field can be the name of any field present in the `vDevices` view; for example, `SerialNumber` for the standard card serial number, or `HIDSerialNumber` for the HID serial number. You can have multiple `DeviceIdentifier` nodes for each device.

Note: If you do not include a `SerialNumberField` node, MyID will use a default value of `SerialNumber`.

For example, to cancel a card with `HIDSerialNumber 0011778`, pass through the following:

```
<Actions>
  <ApplicantAction>CancelDevice</ApplicantAction>
  <StatusMappingID>ID</StatusMappingID>
  <RevocationComment>Comment</RevocationComment>
  <Device>
        <DeviceIdentifier>
        <SerialNumber>0011778</SerialNumber>
        <SerialNumberField>HIDSerialNumber</SerialNumberField>
        </DeviceIdentifier>
  </Device>
</Actions>
```

You can cancel only the cards associated with the applicant. You can cancel suspended cards. You must pass the fields `StatusMappingID` and `RevocationComment` as normal.

3.10.2 Canceling jobs

To cancel all a user's jobs, pass an `ApplicantAction` of `CancelAllJobs`; for example:

```
<Actions>
  <ApplicantAction>CancelAllJobs</ApplicantAction>
</Actions>
```

Note: If the card is currently being processed by the bureau, the job is failed instead of canceled.

To cancel a single job, use an `ApplicantAction` of `CancelJob`, then include a `<Job>` node containing the ID of the job to cancel; for example:

```
<Actions>
  <ApplicantAction>CancelJob</ApplicantAction>
  <Job>12345</Job>
</Actions>
```

You can use the Reporting Web Service API to query the `Jobs` table to find the ID, or use Project Designer to create a custom MI Report that includes the ID in the list of jobs.

3.10.3 Filtering job cancellations

The feature to filter job cancellations using criteria is now "End of Life". The `Filters` node remains in the Lifecycle API schema, but cannot be used.

To cancel a specific job, use:

```
<ApplicantAction>CancelJob</ApplicantAction>
```

See section *3.10.2*, *Canceling jobs* for details.

3.11 Importing operator credentials

As part of a user import, you can import a smart card that has been issued on an external system so that it can be used, unchanged, as an operator card in MyID. If the user account already exists, the operator card is added to the user's record.

You must create a credential profile that will be assigned to imported operator credentials. This credential profile must have the **Contact** capability and must be configured to use the **Imported Authentication** certificate policy in the Unmanaged certificate authority for MyID signing.

To import a device, you need to know the following details about the device:

- The credential profile name you want to use for the imported credential – specify in the `CardProfile` node.
- Serial number – specify in the `SerialNumber` node.
- Device type name – specify in the `DeviceType` node.
- The container name on the card that the certificate is stored in – specify in the `Container` node.
- The Base 64-encoded public part of the certificate to be used for signing – specify in the `Certificate` node.

  The certificate must have a valid date; it cannot have expired, and cannot be valid from a future date. MyID does not carry out a CRL check.

If the **Validate logon certificate** option (on the **Logon** page of the **Security Settings** workflow) is set, the MyID application server must trust the issuing CA and have access to the CRL.

You must set the **Migrated Non-archived Certificate Policy** option (on the **Import & Export** page of the **Operation Settings** workflow) to the following value:

- `Imported Authentication`

This information is passed in through the `Card` node; a sample import may look like this:

```
<Card>
  <CardProfile>Imported Card</CardProfile>
  <ImportCard>true</ImportCard>
  <SerialNumber>1324657980</SerialNumber>
  <DeviceType>Oberthur ID-One PIV</DeviceType>
  <Container>5FC105</Container>
  <Certificate>MIIGlDCCBXygA … =</Certificate>
</Card>
```

An imported card will have an expiry date that is the soonest of that defined in the credential profile and the expiry date of the provided certificate.

The imported card will be blocked from card lifecycle operations such as **Reset Card PIN** and **Erase Card**. You can still cancel the imported card using the **Cancel Credential** workflow; this prevents the card from being used to access MyID, but does not affect the content of the card or revoke its certificates.

Once imported, the device allows authentication to MyID. You can provide the imported user account with the appropriate permissions to access workflows.

Note: If the import of the operator device fails (for example, if the certificate has expired) or if you carry out a card lifecycle operation that is not allowed for imported cards (for example, requesting a replacement card) you may find that the license count has temporarily increased by one; the license count is recalculated every few hours, at which point the license count is set to the correct value.

4 Calling the import service

The MyIDEnroll web service is installed by default to the following location:

`http://<myservername>/MyIDEnroll/MyIDEnroll.asmx`

The MyIDEnroll web service consists of the following methods:

- `CMSXMLWebImport` – used for standard systems.
- `PIVXMLWebImport` – used for PIV/FIPS 201 systems.
- `XMLImport` – used to import data using the `EdeficeData` schema. This option is not supported.

These methods all take XML input parameters, and return an XML string representing the status of the operation.

See section *8*, *MyIDEnroll WSDL* for details of the WSDL for the web service.

4.1 Results

When the import service completes the import, it passes back an XML report containing information on whether the user's record was imported correctly.

See section *7.3*, *CMSImportReponse structure* for details of the XML schema used to return this report.

Section *7.6*, *PivImportResponse structure* contains the PIV version of the response schema.

Any errors (for example, if the XML document contains invalid characters) are returned as the response to the web service call. The errors are displayed as XML to make it easier for your system to parse them.

5 Advanced settings

To add these settings to your XML document, include a node called `Parameters` at the top level of the XML document.

For example:

```
<Parameters>
	<SourceID>Station1</SourceID>
	<IssueDate>2012-01-01</IssueDate>
	<GenerateUserDN>0</GenerateUserDN>
	<ActionOnDuplicate>REPLACE</ActionOnDuplicate>
	<RolesActionOnDuplicate>REPLACE</RolesActionOnDuplicate>
	<DeleteMissingUsers>0</DeleteMissingUsers>
	<PushToLDAP>1</PushToLDAP>
	<CreateUnknownGroups>1</CreateUnknownGroups>
	<AuditAll>1</AuditAll>
	<DataType>CMSRequestCard</DataType>
</Parameters>
```

Note: The parameters are described in the schema as a sequence, which means that the order is important.

The settings you can use are:

Parameter	Description
`SourceID`	The ID of the site.
`IssueDate`	The date the XML document was created.
`GenerateUserDN`	This option is not supported. Set the value to `0`.
`ActionOnDuplicate`	Determines what to do if the user already exists within MyID. The `LogonName` is used as the key to match user records. `REPLACE` – replace the existing user with the new details. `Merge` – merge the records. `MergeEmpty` – add only the field data that was previously empty; do not overwrite any previous data. `Skip` – abort the import. This generates an error.
`RolesActionOnDuplicate`	Determines what to do if the roles already exist within MyID. `REPLACE` or `Merge` – the user's roles and scope will be set to the roles indicated in the supplied XML, if a roles node is supplied. Any other roles held by the user will be removed. `MergeEmpty` – any roles not already held by the user will be added with the scope indicated. If the user already holds the role then the scope will be updated to reflect the supplied data. To remove any existing roles held by the user, specify a `Scope` of `None`. `Skip` – any roles not already held by the user will be added with the scope indicated. If the user already holds the role then it will not be altered. Any existing roles held by the user will not be removed.

Parameter	Description
DeleteMissingUsers	Not supported. Always set to 0.
PushToLDAP	1 – add the user's details to the LDAP directory. 0 – do not add the user's details to the LDAP directory.
CreateUnknownGroups	Not supported. Always set to 1.
AuditAll	1 – add all user information fields to the audit trail. 0 – add only the minimum information (user DN and full name) to the audit trail.
DataType	Determines the specific behavior of the import service, including DN formats and transforms that will be used. CMSRequestCard – import a user's details and request a card. Uses the CMSCardRequest schema. CMSUserUpdate – update an existing user. Uses the CMSUserUpdate schema. PIV or PIVRequestCard – used for PIV card requests. Uses the PivCardRequest schema. PIVApplicantEnrol – used to update existing PIV users. Uses the PivApplicantUpdate schema.
CheckImportResponseSchema	Set to 1 to prevent MyID from checking the returned XML against the response XML schema.
SynchronousImport	Not supported. Do not use.
AllowBioImport	1 – allow the import of fingerprints. 0 – do not allow the import of fingerprints, but allow the import of face scans.
DefaultUserRole	If you do not specify the Role\Name, Scope and LogonMechanism in the Applicant\Account\Roles section of the import file, you can specify them here. Separate the role name, scope and logon mechanism with pipes (|). For example: Applicant|Self|Card specifies the Applicant role, with a scope of Self, and a logon mechanism of Card.
DisallowCertificateSuspension	Set this value to 1, and when you cancel a user account using the Lifecycle API, MyID will not suspend the certificate, but uses the action associated with the StatusMappingID; for example, a status mapping of 100 will permanently revoke the certificate. Set this value to 0 to suspend the certificate when you cancel the user account.

Parameter	Description
CardRequestThrottling	This option prevents the submission of multiple card requests if several requests are received in quick succession. Generating card requests like this can cause the bureau request mechanism to reuse the same FASC-N for more than once card request. Set this value to 1 to enable the throttling behavior. Set this value to 0, or omit the parameter, to disable the throttling behavior.
ReplaceUnassignedCards	Set to 1 to allow MyID to replace unassigned cards. Set to 0 to prevent MyID from replacing unassigned cards.

5.1 Custom schema and transform files

You can customize all Lifecycle API xsd and xsl schema and transform files by editing the web service configuration files. This allows you to override the schema and transform files provided with MyID with your own versions of those files; this means that you will not lose your schema and transform customizations when you update or upgrade your MyID installation.

To allow customizations to the schemas, add the following entries to MyIDEnroll web service web.config file:

```
<configSections>
<section name="MyIDSettings"
type="System.Configuration.DictionarySectionHandler"/>
</configSections>
<MyIDSettings configSource="myid.config" />
```

Once you have done this, you can add entries for schema and transform files to the myid.config file.

Add the key name and value name of the customized schema or transform file.

Note: All key names must be unique. For example:

```
<MyIDSettings>
  <add key="PIV" value="<Replacement schema for PivCardRequest.xsd>"></add>
  <add key="PIVRequestCard" value="<Replacement schema for
PivCardRequest.xsd>"></add>
  <add key="PIVApplicantEnrol" value="<Replacement schema for
PivApplicantUpdate.xsd>"></add>
  <add key="CMSRequestCard" value="<Replacement schema for
CMSCardRequest.xsd>"></add>
  <add key="CMSUpdateCard" value="<Replacement schema for
CMSUserUpdate.xsd>"></add>
  <add key="PIV2MyID" value="<Replacement transform for PIV2MyID.xsl>"></add>
  <add key="CMS2MyID" value="<Replacement transform for CMS2MyID.xsl>"></add>
  <add key="BOL2CMSImportResponse" value="<Replacement transform for
BOL2CMSImportResponse.xsl>"></add>
  <add key="BOL2PIVImportResponse" value="<Replacement transform for
BOL2PIVImportResponse.xsl>"></add>
</MyIDSettings>
```

Note: The names of the keys do not always match the names of the schema files. PIV and PIVRequestCard both override the PivCardRequest.xsd schema – do not specify both in the config file.

For example, if you want to override the PivCardRequest.xsd schema with your own version, which is called MyPIVSchema.xsd, you would add the following:

```
<MyIDSettings>
  <add key="PIVRequestCard" value="MyPIVSchema.xsd"></add>
</MyIDSettings>
```

If you also wanted to replace the standard PIV transform with your own version called `MyPIVTransform.xsl`:

```
<MyIDSettings>
  <add key="PIVRequestCard" value="MyPIVSchema.xsd"></add>
  <add key="PIV2MyID" value="MyPIVTransform.xsl"></add>
</MyIDSettings>
```

6 Troubleshooting

6.1 ASP.NET temporary folder permissions

If your MyID web service user does not have permissions to the .NET framework folder, you may see an error similar to the following:

```
The current identity "mydomain\myiisuser" does not have write access to
'C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files'.
```

To grant access to the web service user, run the `aspnet_regiis` command in the .NET folder mentioned in the error message:

```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727> aspnet_regiis -ga
"mydomain\myuser"
```

Replace `"mydomain\myuser"` with your system's MyID web service user.

6.2 HTTP Error 500.24 – Internal Server Error

This error can occur when you have not configured the authentication mechanisms the MyIDEnroll web service and disabled the **ASP.NET Impersonation** option in IIS.

See section *2.2*, *Security settings* for details.

7 Data Structure

This section describes the structure of the data in the schemas.

The Lifecycle API uses the following XML schemas:

- `MyIDBaseTypes`

  Contains the common schema type for all of the MyID Lifecycle API schemas.

- `CMSSchemaTypes`

  Contains the common schema types for the `CMSCardRequest` and `CMSUserUpdate` schemas.

- `CMSCardRequest`

  Use this schema to create your XML files containing user data to send to the XML web import service. This schema defines the necessary information that you have to include for the import to be successful.

- `CMSUserUpdate`

  Use this schema to create your XML files containing user data when you are updating an existing user in the MyID database.

- `CMSImportResponse`

  This is the schema used when the XML import service passes back information about the results of the user data import.

- `PIVSchemaTypes`

  Contains the common schema types for the `PIVCardRequest` and `PIVApplicantUpdate` schemas.

- `PIVCardRequest`

  Use this schema to create your XML files containing PIV user data to send to the XML web import service. This schema defines the necessary information that you have to include for the import to be successful.

- `PIVApplicantUpdate`

  Use this schema to create your XML files containing user data when you are updating an existing PIV user in the MyID database.

- `PIVImportResponse`

  This is the schema used when the XML import service passes back information about the results of the PIV user data import.

Note: Wherever a node is marked in the schema as a sequence, this order of the contents is important.

7.1 CMSCardRequest structure

This section describes the elements of the CMSCardRequest schema.

7.1.1 CMSCardRequest/Parameters

Contains the settings for the import. See section *5*, *Advanced settings*.

7.1.2 CMSCardRequest/Group

Contains details of the Group to which the user will belong. If the group does not exist, MyID can create the group, depending on your `Parameters` settings.

Note: A group created through the Lifecycle API does not inherit any default roles from its parent group, and you cannot set default roles through the Lifecycle API. This means that for any new users created within this new group, you must specify all of the roles that you want to assign; if you do not specify any roles, the user receives the default Cardholder and Password User roles.

7.1.3 CMSCardRequest/Group/Name

Used to identify the group. Mandatory.

If you are specifying an existing group, this is the only element you need to include to identify the group. If you are specifying a new group, you can complete the rest of the elements to provide the details for the new group.

Note: Group names must be unique. The import will fail if two different groups have the same name as the specified group name.

Note: If you specify a group that has been linked to an LDAP OU in MyID, any users associated with the group will be synchronized with the directory if the `PushToLDAP` parameter is set to `1`.

7.1.4 CMSCardRequest/Group/Description

The description of the group.

7.1.5 CMSCardRequest/Group/OrgUnit

The OU of the group.

7.1.6 CMSCardRequest/Group/Parent

The parent of the group.

Used for new groups only. This is the name of the group beneath which the specified group will be created. For new groups, if you do not specify a parent group, or the parent group does not exist, the new group is placed under the root. If you are specifying an existing group for the user, make sure you omit the `Parent` node.

Note: Parent group names must be unique. The import will fail if two different groups have the same name as the specified parent name.

7.1.7 CMSCardRequest/Group/AdditionalFields

Contains any extended group (`Xg`) fields. See section *3.8*, *Importing additional fields* for details.

7.1.8 CMSCardRequest/Group/User

Contains the details for the user for whom the card is being requested.

You can import a single user in each XML document.

Note: You do not have to include a User node if you are importing the details of a group.

7.1.9 CMSCardRequest/Group/User/Personal

Contains the user's personal details. Mandatory.

7.1.10 CMSCardRequest/Group/User/Personal/FirstName

Internal name: `FirstName`

The user's first name.

7.1.11 CMSCardRequest/Group/User/Personal/LastName

Internal name: `Surname`

The user's last name.

Note: You must include one or both of `FirstName` and `LastName`. If you include both `FirstName` and `LastName`, make sure you specify them in that order – `FirstName` first, `LastName` immediately after. If you intend to supply only a `FirstName` or only a `LastName`, omit the other node – do not simply include an empty string.

7.1.12 CMSCardRequest/Group/User/Personal/Initial

Internal name: `Initial`

The user's middle initials.

7.1.13 CMSCardRequest/Group/User/Personal/Title

Internal name: `Title`

The user's title.

7.1.14 CMSCardRequest/Group/User/Personal/Email

Internal name: `Email`

The user's email address. This address is used for email notifications.

7.1.15 CMSCardRequest/Group/User/Personal/PhoneExt

Internal name: `PhoneExt`

The user's phone extension.

7.1.16 CMSCardRequest/Group/User/Personal/MobileNumber

Internal name: `MobileNumber`

The user's mobile phone number.

7.1.17 CMSCardRequest/Group/User/Personal/PhoneNumber

Internal name: `PhoneNumber`

The user's phone number.

7.1.18 CMSCardRequest/Group/User/Personal/EmployeeID

Internal name: `EmployeeID`

This should be a unique identifier for the applicant.

7.1.19 CMSCardRequest/Group/User/Personal/OptionalLine1

Internal name: `OptionalLine1`

Used to store extra information about the user.

7.1.20 CMSCardRequest/Group/User/Personal/OptionalLine2

Internal name: `OptionalLine2`

Used to store extra information about the user.

7.1.21 CMSCardRequest/Group/User/Personal/OptionalLine3

Internal name: `OptionalLine3`

Used to store extra information about the user.

7.1.22 CMSCardRequest/Group/User/Personal/OptionalLine4

Internal name: `OptionalLine4`

Used to store extra information about the user.

7.1.23 CMSCardRequest/Group/User/Authentication

Contains details of the security phrases to be used for the user.

Note: Currently, the schema does not provide validation on the `Authentication` element.

7.1.24 CMSCardRequest/Group/User/Authentication/SecurityPhrase

Contains a pair of elements – a `Prompt` and an `Answer` – that provide a security phrase for the applicant.

You can have up to five `SecurityPhrase` elements, each containing a `Prompt` and `Answer` pair.

Note: If you provide any security phrases for an existing user, *all* existing security phrases on that user's account are removed and replaced with the newly-provided security phrases. You can use this feature to delete the security phrases from a user's account – provide a single `Prompt` with an empty `Answer`, and the user's security phrases will be removed. (If the minimum number of security phrases is set to 1, the user will still be prompted for a security phrase when attempting to log on, but will be unable to authenticate.)

7.1.25 CMSCardRequest/Group/User/Authentication/SecurityPhrase/Prompt

Internal name: `Question` field in `SecurityQuestions`

Used to import a security question. If the question does not already exist, an entry is added to the `SecurityQuestions` table; this question will subsequently be available for use when setting security phrases in MyID.

The default security questions are:

ID	Question
1	Password
2	Mothers maiden name?
3	N.I number?
4	Favourite food?
5	Car registration number?
6	Name of pet
7	First school attended
8	An old phone number
9	A memorable date
10	A memorable place
11	A memorable event
12	A famous person
13	Place of birth
14	A country

ID	Question
15	A river
16	A movie
17	A song
18	A book
19	A sport
20	A postcode
21	An animal
22	First car type

7.1.26 CMSCardRequest/Group/User/Authentication/SecurityPhrase/Answer

Internal name: `SecurityPhrase` field in `UsersSecurityQuestions`

Used to import an answer to a security question for an applicant. A hash of the answer is stored in the `UsersSecurityQuestions` table.

Note: If you have set up the **Security Phrase Complexity** option within MyID to determine the complexity of security phrases, this is not enforced on any passwords that you import using the Lifecycle API.

You can encrypt the answer using a transport key – this allows you to maintain the security of the security phrase answer when you include it in the XML data.

To encrypt the answer:

1. Import your AES128 transport key into MyID using the **Key Manager** workflow.
2. External to MyID, encrypt the security phrase answer with the following settings:
   - Encode the security phrase answer as little-endian Unicode. Do not use a byte order mark.

     For example, a security phrase of `answer` would be encoded as:

     `0061006E0073007700650072`
   - Pad the data using ISO9797 Method 2 (pad with a single 0x80, then 0x00 until the block-length is reached).
   - Encrypt using CBC or ECB mode encryption.
3. Add the hex-encoded encrypted data to the `Answer` node.
4. Set the `KeyName` attribute of the `Answer` node to the name of the transport key you imported into MyID.

5. Set the `Mode` attribute of the `Answer` node to either `CBC` or `ECB`, depending on how you encrypted the answer.

For example:

- The transport key, `MyTransportKey`, has a value of:

  `206890FC9B4EA1D0137D8C692B3BFCB6`

- The security phrase is:

  `answer`

- In the import, use the following:

  `<Answer KeyName="MyTransportKey" Mode="CBC">6713987B589F76BC4DA0F557D79A6275</Answer>`

If you do not want to encrypt the answer in the XML document, omit the `KeyName` and `Mode` attributes. For example:

`<Answer>plaintextanswer1234</Answer>`

7.1.27 CMSCardRequest/Group/User/Card

Contains details of the card to be requested for the user.

7.1.28 CMSCardRequest/Group/User/Card/CardProfile

If present, this will request a card using the specified credential profile for this applicant. If performing an update, or the credential profile does not exist, no request will be made.

Note: Credential profiles were previously known as card profiles.

7.1.29 CMSCardRequest/Group/User/Card/CardExpiryDate

Used to specify the expiry date for the card when it is issued. The card will expire on the date specified, or on the date the credential profile specifies, whichever is earlier.

The format is `YYYY-MM-DD` – for example, `2016-12-25`.

Note: The card will expire at 00:00:00 UTC on the day of the expiry; that is, at the *start* of the day specified, not the *end*.

7.1.30 CMSCardRequest/Group/User/Card/Renewal

Used to specify whether the card is a renewal of an existing card. If the user already exists, you must set this option to `true` or card request jobs will not be created.

Can be one of the following values:

- `true` – this is a card renewal.
- `false` – this is not a card renewal.

Note: For card renewals, you must specify the card's original serial number and device type using the `CMSCardRequest/Group/User/Card/Update` node.

7.1.31 CMSCardRequest/Group/User/Card/ImportCard

Used to specify whether to import a device (see section *3.11*, *Importing operator credentials*).

Can be one of the following values:

- `true` – import a device.
- `false` – do not import a device.

7.1.32 CMSCardRequest/Group/User/Card/CardRequestedBy

This is the logon name of the person requesting the card. Used for auditing purposes.

7.1.33 CMSCardRequest/Group/User/Card/CancelExisting

Used to specify whether existing jobs of the same type are canceled when the new job is created. Include the empty node if you want to cancel the existing jobs:

```
<CancelExisting/>
```

If you omit this node, existing jobs will not be canceled when the new job is created.

7.1.34 CMSCardRequest/Group/User/Card/JobLabel

Used to specify a label for the job. You can use this label to search for and identify the job within MyID.

7.1.35 CMSCardRequest/Group/User/Card/Update

Used to specify the card being updated or renewed. You must make sure that the card is issued to the user.

For updates, if you do not specify the `OriginalSerialNumber` and `OriginalDeviceType` for the card to be updated, the card currently assigned to the user is used.

For renewals, the `OriginalSerialNumber` and `OriginalDeviceType` for the card to be renewed are mandatory.

7.1.36 CMSCardRequest/Group/User/Card/Update/OriginalSerialNumber

The serial number of the card being updated or renewed. Used to target a specific card.

7.1.37 CMSCardRequest/Group/User/Card/Update/OriginalDeviceType

The device type of the card being updated or renewed.

7.1.38 CMSCardRequest/Group/User/Card/Update/StatusMapping

The status mapping code to be used when the card is updated.

See section *3.7.5*, *Status mapping codes* for details.

7.1.39 CMSCardRequest/Group/User/Card/Replacement

Used to specify the card to be replaced.

Note: The **Card Renewal Period** configuration option (on the **Devices** page of the **Operation Settings** workflow) is ignored when requesting replacement cards through the Lifecycle API – it is possible to request short-lived replacement credentials through the API.

7.1.40 CMSCardRequest/Group/User/Card/Replacement/OriginalSerialNumber

The serial number of the card to be replaced. Mandatory if you specify the `Replacement` node.

7.1.41 CMSCardRequest/Group/User/Card/Replacement/OriginalDeviceType

The device type of the card to be replaced. Mandatory if you specify the `Replacement` node.

7.1.42 CMSCardRequest/Group/User/Card/Replacement/StatusMapping

Used to specify the status mapping ID for certificates on the credentials that are being replaced. This node is used when performing a replacement request; note that if you are performing a card replacement, you use `CMSCardRequest/Group/User/Card/StatusMapping` instead.

If you do not specify a `StatusMapping`, a status mapping ID of `1` – Lost – is used.

See section *3.7.5*, *Status mapping codes* for details.

7.1.43 CMSCardRequest/Group/User/Card/GenerateOTP

Note: This feature reserved for future use. It is not currently possible to use OTPs when collecting jobs. Do not use in the current version of MyID.

Used to generate a One-Time Password (OTP) for use when collecting the job created by the import request. If you specify this element, the user must provide the OTP when collecting the job. You can specify a notification mechanism to provide this OTP to the user by including a `Notification` node.

If you do not specify a `Notification` node, the OTP is sent to the user's email address using the **Job OTP** email template.

7.1.44 CMSCardRequest/Group/User/Card/GenerateOTP/Notification

Used to specify a notification method for OTPs.

7.1.45 CMSCardRequest/Group/User/Card/GenerateOTP/Notification/Name

The `Name` of the notification from the `Notifications` table in the MyID database.

Note: You must have notifications configured correctly within MyID. For more information, contact Intercede professional services.

7.1.46 CMSCardRequest/Group/User/Card/GenerateOTP/Notification/Action

Contains one of the following:

- `Event` – the notification is triggered once using the success data set up in the `Notifications` table. No further notifications are sent, even if the notification fails.
- `Success` – the notification is triggered using the success data set up in the `Notifications` table. If the notification fails, the back-up notifications set up in the fail data in the `Notifications` table are triggered.

- `Fail` – the notification is triggered using the fail data set up in the `Notifications` table. Multiple notifications are sent on a schedule specified by the `RetryDelays` in the `Notifications` table. You can use this option to force reminder notifications; the notification has not failed, but it is treated as a failed notification so that the retry system is triggered.

7.1.47 CMSCardRequest/Group/User/Card/OriginalSerialNumber

Used in conjunction with the `CMSCardRequest/Group/User/Card/OriginalDeviceType` node to specify a credential to be replaced. These settings also specify the credential profile to use when replacing the card. The credential profile used to issue this specified card is used to issue the new card. Alternatively, you can specify the credential profile using the `CMSCardRequest/Group/User/Card/CardProfile` node.

In addition, if you specify the `OriginalSerialNumber` and `OriginalDeviceType` for the card, all credentials on the same physical credential are marked as being replaced.

The `CMSCardRequest/Group/User/Card/StatusMapping` determines what happens to the certificates on the credentials that are being replaced.

7.1.48 CMSCardRequest/Group/User/Card/OriginalDeviceType

Used in conjunction with the `CMSCardRequest/Group/User/Card/OriginalSerialNumber` node to specify a credential to be replaced.

7.1.49 CMSCardRequest/Group/User/Card/SerialNumber

Used in conjunction with the `CMSCardRequest/Group/User/Card/DeviceType` node to specify an exact card to be issued.

7.1.50 CMSCardRequest/Group/User/Card/DeviceType

Used in conjunction with the `CMSCardRequest/Group/User/Card/SerialNumber` node to specify an exact card to be issued.

7.1.51 CMSCardRequest/Group/User/Card/StatusMapping

Used to specify the status mapping ID for certificates on the credentials that are being replaced. This node is used when performing a card request; note that if you are performing a card replacement, you use `CMSCardRequest/Group/User/Card/Replacement/StatusMapping` instead.

If you do not specify a `StatusMapping`, a status mapping ID of `1` – Lost – is used.

See section *3.7.5*, *Status mapping codes* for details.

7.1.52 CMSCardRequest/Group/User/Card/Reprovision

Set the value of this node to `1` to request a reprovision rather than an update.

A reprovision allows you to re-encode a card completely, based on the data in the MyID database, using the latest version of the credential profile used during issuance.

The card will have the same expiry date as the original card. New certificates may have longer expiration times than the original certificates, but these will not exceed the lifetime of the card itself. Certificates that were revoked externally to MyID will be replaced with new active certificates.

When requesting a reprovision, specify a status mapping with one of the following values:

- `83` – User details have changed.
- `84` – There is a problem with the device.
- `85` – New credential profile needs to be applied.

Note: If you specify any other status mapping, it is ignored – only these status mappings carry out reprovisions.

For example:

```
<Card>
 <CardProfile>Yubikey NoOTP</CardProfile>
 <CardRequestedBy>System</CardRequestedBy>
 <OriginalSerialNumber>8115516</OriginalSerialNumber>
 <OriginalDeviceType>YubiKey 4</OriginalDeviceType>
 <StatusMapping>84</StatusMapping>
 <Reprovision>1</Reprovision>
</Card>
```

7.1.53 CMSCardRequest/Group/User/Card/CardLayout

Optionally, contains the name of the card layout to use when printing the requested card.

The card layout name must be valid for the card's credential profile at the point that the card is issued; if the card layout name is not valid, the request will not be processed.

7.1.54 CMSCardRequest/Group/User/Card/Container

Used in conjunction with the `CMSCardRequest/Group/User/Card/ImportCard` node to specify the container that the authentication certificate is located in.

See section *3.11*, *Importing operator credentials*.

7.1.55 CMSCardRequest/Group/User/Card/Certificate

Used in conjunction with the `CMSCardRequest/Group/User/Card/ImportCard` node to specify the authentication certificate on the imported card. This must be the base64 representation of the certificate without headers, footers, or line breaks.

See section *3.11*, *Importing operator credentials*.

7.1.56 CMSCardRequest/Group/User/Card/AdditionalFields

Used to specify additional job (`Xj`) fields. Your system must already have been customized to set up these additional fields.

7.1.57 CMSCardRequest/Group/User/CardUpdate

Note: This `CardUpdate` node is reserved for future use. Do not use. You can use the `CMSCardRequest/Group/User/Card/Update` node instead.

The `CardUpdate` element contains details of card update requirements for existing cards.

7.1.58 CMSCardRequest/Group/User/CardUpdate/SerialNumber

Serial Number of the card to be updated.

7.1.59 CMSCardRequest/Group/User/CardUpdate/DeviceType

Device Type of card to be updated (for example, 'Oberthur v3').

7.1.60 CMSCardRequest/Group/User/CardUpdate/CardRequestedBy

The logon name of the person requesting the card update. Used for auditing purposes.

7.1.61 CMSCardRequest/Group/User/CardUpdate/ParametersXML

XML describing the card updates to be performed. Currently supports only UnlockPIN.

7.1.62 CMSCardRequest/Group/User/CardUpdate/ParametersXML/UnlockPIN

Set to `1` to unlock or set the PIN during the card update.

If you set this option to `1`, the end user must enter and confirm a new PIN using the **Collect Card Update** workflow.

If the end user needs to carry out any operations (for example, adding a certificate) that require the card PIN, you must reset the card's PIN using this method before you do so; this is because the card was originally issued with a random, unknown PIN.

7.1.63 CMSCardRequest/Group/User/CardUpdate/PIN

Reserved for future use.

7.1.64 CMSCardRequest/Group/User/Account

Contains details of the user's account.

7.1.65 CMSCardRequest/Group/User/Account/DN

Internal name: `DistinguishedName`

The DN of the user's account in the LDAP.

7.1.66 CMSCardRequest/Group/User/Account/CN

Internal name: `CommonName`

7.1.67 CMSCardRequest/Group/User/Account/OU

Internal name: `OrganisationalUnit`

7.1.68 CMSCardRequest/Group/User/Account/UPN

Internal name: `UserPrincipalName`

7.1.69 CMSCardRequest/Group/User/Account/SAMAccountName

The SAM Account Name for the user.

7.1.70 CMSCardRequest/Group/User/Account/LogonName

Internal name: `LogonName`

The user's name, used to log into MyID. Must be unique.

A `LogonName` is required for all operations. If no value is supplied, the `CMSCardRequest/Group/User/Personal/EmployeeID` value is used instead. For new user imports, the `EmployeeID` is copied to the user's new `LogonName` record; for actions on existing users, the `EmployeeID` is used to match against the `LogonName`.

7.1.71 CMSCardRequest/Group/User/Account/NewLogonName

Note: This feature reserved for future use.

Allows you to specify a new logon name for the user.

7.1.72 CMSCardRequest/Group/User/Account/UniqueID

Maps onto the `objectGUID` attribute in Active Directory. When MyID synchronizes with the LDAP, it uses this UniqueID to map the users from MyID to the directory.

If your directory does not have an `objectGUID` attribute, the Distinguished Name for the user is used instead.

7.1.73 CMSCardRequest/Group/User/Account/Roles

Contains details of the MyID roles assigned to the user.

Note: You can set any role for a user using the Lifecycle API – the roles you specify override any role restrictions. Also, the user will not receive any default roles that have been set up for their group; if you want to assign roles, you must include them explicitly. If you do *not* specify any roles, the user receives the default Cardholder and Password User roles.

7.1.74 CMSCardRequest/Group/User/Account/Roles/Role

Contains details of the role you want to assign the applicant.

7.1.75 CMSCardRequest/Group/User/Account/Roles/Role/Name

Internal name: `UserProfile`

The name of the role.

7.1.76 CMSCardRequest/Group/User/Account/Roles/Role/Scope

Internal name: `Scope`

The scope of the role that the applicant is to have. The value must be one of the following:

- `None`
- `Self`
- `Department`
- `Division`
- `All`

Note: If you want to remove an existing role from a user's account, set the `Scope` to `None`, and the `RolesActionOnDuplicate` parameter to `MergeEmpty`.

7.1.77 CMSCardRequest/Group/User/Account/Roles/Role/LogonMechanism

Internal name: `LogonMechanism`

Not supported. Do not use.

7.1.78 CMSCardRequest/Group/User/Account/EntrustProfile

Specifies an Entrust profile template to use for the applicant. If specified, the applicant is added to Entrust using the template.

7.1.79 CMSCardRequest/Group/User/Account/MaxRequestExpiryDate

Allows you to set the maximum credential expiry date for a person – this allows you to specify the latest expiry date for any device issued to this person. Note that card requests cannot exceed the **Lifetime** set in the credential profile. Also, the credential profile can override the maximum expiry date for a person using the **Ignore User Expiry Date** option.

This setting affects all future device requests. It does not affect any issued devices or existing requests.

Note: This setting affects device requests made through the MyID Operator Client only. Requests made through MyID Desktop or the Lifecycle API do not take this setting into account.

See the *Requesting a device for a person* section in the ***MyID Operator Client*** guide for more information.

7.1.80 CMSCardRequest/Group/User/Account/UserDataApproved

Internal name: `UserDataApproved`

Contains information on whether card issuance is approved for this applicant. Can be one of the following values:

- `YES` or `1`
- `NO` or `0`

If you set the card profile to have the **Require user data to be approved** option, you must set this option before the card can be issued. This flag confirms that the data being sent to MyID has passed all the enrollment checks required for card issuance.

7.1.81 CMSCardRequest/Group/User/Account/VettingDate

Internal name: `VettingDate`

Contains the date on which the applicant's identity checks were carried out. The format is:

`YYYY-MM-DDThh:mm:ss`

You can use this field to ensure that the applicant's identity checks are renewed periodically. See the *Identity checks* section in the ***Administration Guide*** for details.

7.1.82 CMSCardRequest/Group/User/Photo

Contains the applicant's photo.

7.1.83 CMSCardRequest/Group/User/Photo/Encoding

The type of data encoded in the block. This can be one of the following:

- `jpg` – JPG image data.
- `gif` – GIF image data.
- `png` – PNG image data.
- `bmp` – Windows bitmap image data.
- `378` – INCITS 378 fingerprint data.
- `385` – facial biometrics in ANSI/INCITS 385-2004 format.

For details of importing fingerprints or facial biometrics, contact customer support.

7.1.84 CMSCardRequest/Group/User/Photo/Data

A string of base64-encoded data containing the file to be uploaded.

7.1.85 CMSCardRequest/Group/User/Photo/DateTaken

The date of the data; for example, when the photograph was taken, or when the fingerprints were captured.

This node uses the `xs:dateTime` format:

```
yyyy-mm-ddThh:mm:ss.nnn+zzzzz
```

For example:

```
2010-01-02T14:23:54.123+01:00
```

7.1.86 CMSCardRequest/Group/User/Photo/Source

Not supported.

7.1.87 CMSCardRequest/Group/User/Photo/None

Instead of providing a photograph, you can remove an existing user photograph and replace it with the `noImage.gif` default image.

To remove a user photograph, specify `_NULL_` in the `Photo` node; for example:

```
<Photo>
  <None>_NULL_</None>
<Photo>
```

Note: The image is removed from the user record, but is not removed from the MyID web server.

7.1.88 CMSCardRequest/Group/User/AdminGroups

An optional node that allows you to specify one or more administrative groups for the user. See the Administration Guide for details of setting up and using administrative groups.

The `AdminGroups` node contains one or more `AdminGroup`.

For example:

```
<AdminGroups>
                <AdminGroup>
                                <Name>Group 1</Name>
                </AdminGroup>
</AdminGroups>
```

or:

```
<AdminGroups>
                <AdminGroup>
                                <Name>Group 1</Name>
                </AdminGroup>
                <AdminGroup>
                                <Name>Group 2</Name>
                </AdminGroup>
</AdminGroups>
```

If the user has existing administrative groups, they are replaced by the specified group or groups. If you specify an empty `AdminGroups` node, this clears all of the user's administrative groups. If you do not specify an `AdminGroups` node, the user's administrative groups remain unchanged.

7.1.89 CMSCardRequest/Group/User/AdminGroups/AdminGroup

Specifies an administrative group for the user. You can have multiple `AdminGroup` nodes in the `AdminGroups` parent node. An `AdminGroup` node contains a single `Name` node.

7.1.90 CMSCardRequest/Group/User/AdminGroups/AdminGroup/Name

Specifies the name of the administrative group for the user.

Note: The group must already exist in the MyID database. If the group does not exist, the import will fail.

7.1.91 CMSCardRequest/Group/User/AdditionalFields

You can import information into additional extended user (`Xu`) fields using the data import.

7.1.92 CMSCardRequest/Group/User/Actions

Contains actions to carry out for the applicant; for example, to remove or disable the user, or to cancel the user's credentials.

7.1.93 CMSCardRequest/Group/User/Actions/ApplicantAction

Perform an action on the user. If you do not supply an ApplicantAction element, a disabled user will be re-enabled.

The possible actions are:

- `Disable` – disable the user's account.
- `Remove` – remove the user from the system.
- `CancelDevices` – Cancel all the user's credentials.
- `CancelDevice` – Cancel specific credentials. Specify the cards using the `CMSCardRequest/Group/User/Actions/Device` element.
- `CancelJob` – Cancel a specific job. Specify the job using the `CMSCardRequest/Group/User/Actions/Job` element.
- `CancelAllJobs` – Cancel all jobs for the specified user.
- `UnlockCard` – Unlock a card. Specify the card using the `CMSCardRequest/Group/User/Actions/Device` element.

  Note: This is not currently supported.

- `RenewCertificate` – renew a certificate. Specify the certificate using `CMSCardRequest/Group/User/Actions/CertSerialNumber` and `CMSCardRequest/Group/User/Actions/CertPolicyName`.

7.1.94 CMSCardRequest/Group/User/Actions/CertSerialNumber

Used in conjunction with `CMSCardRequest/Group/User/Actions/CertPolicyName` to specify a certificate to renew.

7.1.95 CMSCardRequest/Group/User/Actions/CertPolicyName

Used in conjunction with `CMSCardRequest/Group/User/Actions/CertSerialNumber` to specify a certificate to renew.

7.1.96 CMSCardRequest/Group/User/Actions/StatusMappingID

The action to apply to the user. This should be an ID that matches an entry in the StatusMappings table. It controls how certificates are revoked.

7.1.97 CMSCardRequest/Group/User/Actions/RevocationComment

A free text description that is stored against revoked certificates in the database.

7.1.98 CMSCardRequest/Group/User/Actions/RevocationDelay

Used to specify a delay in days for the certificate revocation requests.

7.1.99 CMSCardRequest/Group/User/Actions/Device

Specifies the card to cancel when you have selected `CancelDevice` for the `ApplicantAction`, or the card to unlock when you have selected `UnlockCard` for the `ApplicantAction`. (**Note:** `UnlockCard` is not currently supported.)

Contains at least one `DeviceIdentifier` element – you can cancel multiple cards in a single request.

7.1.100 CMSCardRequest/Group/User/Actions/Device/DeviceIdentifier

Specifies the credential to be canceled or unlocked.

7.1.101 CMSCardRequest/Group/User/Actions/Device/DeviceIdentifier/SerialNumber

The serial number of the credential.

7.1.102 CMSCardRequest/Group/User/Actions/Device/DeviceIdentifier/SerialNumberField

The name of the field in which to look for the serial number. The field can be any field present in the `vDevices` view; for example, `SerialNumber` for the standard serial number or `HIDSerialNumber` for the HID serial number.

If you do not include a `SerialNumberField` node, MyID will use a default value of `SerialNumber`.

7.1.103 CMSCardRequest/Group/User/Actions/Device/ProcessStatus

Used to specify the process status to be used for the credential being cancelled.

You can use the following status codes, as listed in the `DeviceProcessStatuses` table in the database.

Status	Description
`Active`	Device is operational
`Assigned`	Device is assigned to a user but has not begun personalization
`AtBureau`	Device is at the bureau
`Collected`	Collected
`Disposed`	Disposed
`Erased`	Erased
`Legacy`	Legacy
`Lost`	Lost
`None`	None (Default the card may be re-issued)
`Not Disposed`	Not Collected
`PendingActivation`	Device is ready to be activated
`PendingDeliveryConfirmation`	Device has been dispatched from the bureau and is in transit
`PendingPersonalisation`	Device is ready to be personalized
`Terminated`	Card has been terminated
`Unassigned`	Device is not assigned to a user

7.1.104 CMSCardRequest/Group/User/Actions/RequestedBy

Used to specify the MyID logon name of the Initiator for any jobs created by the import process. If you do not specify a value here, the Initiator is set to ImportProcessor.

You can use the **Initiator** tab in the **Job Management** workflow to search for jobs created by specific users.

7.1.105 CMSCardRequest/Group/User/Actions/Job

Contains a the ID of a job to cancel. Used in conjunction with an `ApplicantAction` of `CancelJob`.

7.1.106 CMSCardRequest/Group/User/Actions/Filters

The feature to filter job cancellations using criteria is now "End of Life". The `Filters` node remains in the Lifecycle API schema, but cannot be used.

See section *3.10.3*, *Filtering job cancellations* for more information.

7.2 CMSUserUpdate structure

The `CMSUserUpdate` schema uses the same data elements as the `CMSCardRequest` schema. This schema requires less information as a minimum, however.

The following differences exist between the `CMSUserUpdate` and `CMSCardRequest` schema:

- The `CMSUserUpdate` schema allows you to specify a `User` block without including it in a `Group` block.
- The `CMSUserUpdate` schema does not have a `CardUpdate` element.
- The `CMSCardRequest` schema allows only one group and one user, but the `CMSUserUpdate` schema allows any number of groups and users.
- If you specify a group, you must specify one or more users.

7.3 CMSImportReponse structure

The CMSImportResponse schema is used when the XML import service passes back information about the results of the user data import. It contains blocks for the `Group` and the `User`.

7.3.1 CMSImportResponse/Group

Contains information about the group.

7.3.2 CMSImportResponse/Group/Name

Internal name: `Name`

Used to identify the group.

7.3.3 CMSImportResponse/Group/Result

The result of the import. Can be one of the following:

- `Created`
- `Already Exists`
- `Failed`

7.3.4 CMSImportResponse/Group/User

Contains information about the user.

7.3.5 CMSImportResponse/Group/User/FirstName

Internal name: `FirstName`

The first name of the user.

7.3.6 CMSImportResponse/Group/User/LastName

Internal name: `Surname`

The last name of the user.

7.3.7 CMSImportResponse/Group/User/EmployeeID

Internal name: `EmployeeID`

The Employee ID of the user.

7.3.8 CMSImportResponse/Group/User/LogonName

Internal name: `LogonName`

The MyID logon name of the user.

7.3.9 CMSImportResponse/Group/User/CardRequest

Internal name: `Jobs.ID`

The JobID for the card request. If there is no request, returns 0.

7.3.10 CMSImportResponse/Group/User/UnlockCardRequest

The JobID for the unlock card request. If there is no request, returns 0.

Note: This is not currently supported.

7.3.11 CMSImportResponse/Group/User/Result

The result of the import. Can be one of the following:

- `Added`
- `Created`
- `Already Exists`
- `Failed`

- `Removed`
- `License Limit`

7.3.12 CMSImportResponse/Group/User/Reason

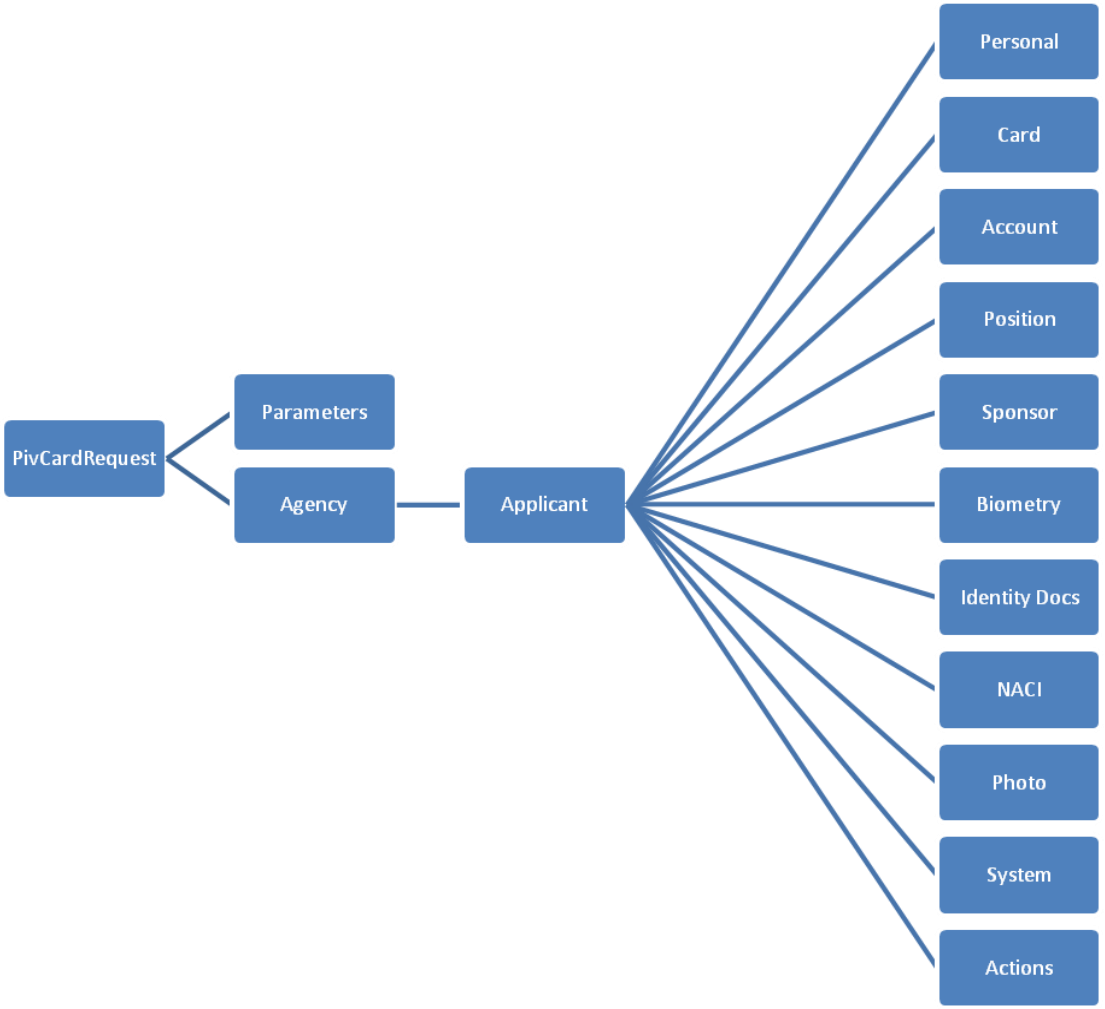
Provides information about the response, including details of the reason why the request failed; for example, the name of a non-existent credential profile.

7.3.13 CMSImportResponse/Group/User/error

Contains XML describing any errors that occurred during the import.

7.4 PivCardRequest

This section provides information on the structure of the XML data, including information on allowed values and the MyID internal name. Use this information in conjunction with the schemas and examples.



7.4.1 PivCardRequest/Parameters

See section *5*, *Advanced settings* for details.

7.4.2 PivCardRequest/Agency

The `Agency` element contains information about the applicant's agency, and contains the block for the applicant.

Note: An agency group created through the Lifecycle API does not inherit any default roles from its parent group, and you cannot set default roles through the Lifecycle API. This means that for any new users created within this new group, you must specify all of the roles that you want to assign; if you do not specify any roles, the user receives the default Cardholder and Password User roles.

7.4.3 PivCardRequest/Agency/DeptCode

Internal name: `Xg16`

Encoded in the PrintedInfo block.

Mandatory only for Add; also mandatory for update if the agency is specified.

7.4.4 PivCardRequest/Agency/AgencyCode

Internal name: `Xg1`

Four alphanumeric characters. Encoded in the FASCN and in the PrintedInfo block.

Mandatory only for Add; also mandatory for update if the agency is specified.

7.4.5 PivCardRequest/Agency/FacilityCode

Internal name: `Xg2`

Encoded in the FASCN and in the PrintedInfo block.

Mandatory only for Add; also mandatory for update if the agency is specified.

7.4.6 PivCardRequest/Agency/AgencyName

Internal name: `Name` and `Xu3`

Used to identify the Agency. Also encoded in the DN for member Applicants.

Mandatory for Add; also mandatory for update if the agency is specified.

Note: Agency names must be unique. The import will fail if the agency name is not unique; this occurs even if a unique parent name is specified.

7.4.7 PivCardRequest/Agency/FacilityName

Not currently used.

7.4.8 PivCardRequest/Agency/DUNS

Internal name: `Xg5`

Needed if the Organizational Category (below) is set to Commercial.

Mandatory only for Add; also mandatory for update if the agency is specified.

7.4.9 PivCardRequest/Agency/OC

Internal name: `Xg3`

Organizational Category. Encoded in the FASCN.

An integer – the values map to the following categories:

- `1` – Federal
- `2` – State
- `3` – Commercial
- `4` – Foreign

Mandatory only for Add; also mandatory for update if the agency is specified.

7.4.10 PivCardRequest/Agency/OI

Internal name: `Xg4`

Organizational Identifier. Encoded in the FASCN.

Mandatory only for Add; also mandatory for update if the agency is specified.

7.4.11 PivCardRequest/Agency/Department

Internal name: `Xg17` and `Xu54`

Encoded in the DN for member Applicants.

7.4.12 PivCardRequest/Agency/BaseDN

Internal name: `OrgUnit` and `Xg15`

If present, this is used as the default DN for member Applicants. The CN field is appended to this value.

7.4.13 PivCardRequest/Agency/Parent

Internal name: `Parent`

The name of the agency beneath which the specified agency should be positioned. For new agencies, if you do not specify a parent agency, or the parent agency does not exist, the new agency is placed under the root. If you are specifying an existing agency for the user, make sure you specify the correct parent agency, or omit the `Parent` node; otherwise, the agency will be moved under the new parent.

Note: Parent agency names must be unique. The import will fail if two different agencies have the same name as the specified parent name.

7.4.14 PivCardRequest/Agency/AgencyAddress

Used for delivery addresses. Contains the following elements:

Field Name	Internal Name
Address1	Xg7
Address2	Xg8
City	Xg9
State	Xg10
Zip	Xg11
PhoneNumber	Xg13
Country	Xg12
Email	Xg14
Contact	Xg6

7.4.15 PivCardRequest/Agency/AdditionalAgencyFields

Contains any extended (Xg) agency fields. See section *3.8.2*, *Additional group fields* for details.

7.4.16 PivCardRequest/Agency/Applicant

The Applicant element contains information about the applicant.

You can import a single applicant in each XML document.

7.4.17 PivCardRequest/Agency/Applicant/Personal

Contains the personal details of the applicant.

Mandatory for Add.

7.4.18 PivCardRequest/Agency/Applicant/Personal/FirstName

Internal name: `FirstName`

Mandatory.

7.4.19 PivCardRequest/Agency/Applicant/Personal/LastName

Internal name: `Surname`

Mandatory.

7.4.20 PivCardRequest/Agency/Applicant/Personal/NickName

Internal name: `Xu49`

7.4.21 PivCardRequest/Agency/Applicant/Personal/MiddleName

Internal name: `Initial`

7.4.22 PivCardRequest/Agency/Applicant/Personal/Suffix

Internal name: `XuPIV4`

7.4.23 PivCardRequest/Agency/Applicant/Personal/Title

Internal name: `Title`

7.4.24 PivCardRequest/Agency/Applicant/Personal/Email

Internal name: `Email`

This address is used for email notifications.

7.4.25 PivCardRequest/Agency/Applicant/Personal/Fax

Internal name: `PhoneExt`

7.4.26 PivCardRequest/Agency/Applicant/Personal/Cell

Internal name: `MobileNumber`

7.4.27 PivCardRequest/Agency/Applicant/Personal/Phone

Internal name: `PhoneNumber`

7.4.28 PivCardRequest/Agency/Applicant/Personal/EmployeeID

Internal name: `EmployeeID`

This should be a unique identifier for the applicant.

Mandatory.

Appears within MyID as the **Security** field on the **Edit PIV Applicant** workflow. For a PIV user (that is, a user who belongs to an agency that does not have a code of 9999) this value is used for the PI (Person Identifier) value in the FASC-N. Due to restrictions on the PI field in the FASC-N, this identifier must be a maximum of 10 numeric digits.

7.4.29 PivCardRequest/Agency/Applicant/Personal/Address

Used for delivery addresses. Contains the following elements:

- Address1

- Address2
- City
- StateZip

7.4.30 PivCardRequest/Agency/Applicant/Personal/BirthDate

Internal name: `Xu37`

Format: `YYYY-MM-DD`

7.4.31 PivCardRequest/Agency/Applicant/Personal/Nationality

Internal name: `XuPIV1`

A code referring to the nationality of the applicant. The possible values are:

Code	Nationality
AFG	Afghanistan
ALA	Aland Islands
ALB	Albania
DZA	Algeria
ASM	American Samoa
AND	Andorra
AGO	Angola
AIA	Anguilla
ATA	Antarctica
ATG	Antigua and Barbuda
ARG	Argentina
ARM	Armenia
ABW	Aruba
AUS	Australia
AUT	Austria

Code	Nationality
AZE	Azerbaijan
BHS	Bahamas
BHR	Bahrain
BGD	Bangladesh
BRB	Barbados
BLR	Belarus
BEL	Belgium
BLZ	Belize
BEN	Benin
BMU	Bermuda
BTN	Bhutan
BOL	Bolivia
BIH	Bosnia and Herzegovina
BWA	Botswana
BVT	Bouvet Island
BRA	Brazil
IOT	British Indian Ocean Territory
BRN	Brunei Darussalam
BGR	Bulgaria
BFA	Burkina Faso
BDI	Burundi
KHM	Cambodia
CMR	Cameroon
CAN	Canada
CPV	Cape Verde
CYM	Cayman Islands
CAF	Central African Republic
TCD	Chad

Code	Nationality
CHL	Chile
CHN	China
CXR	Christmas Island
CCK	Cocos (Keeling) Islands
COL	Colombia
COM	Comoros
COG	Congo
COD	Congo, Democratic Republic of the
COK	Cook Islands
CRI	Costa Rica
CIV	Cote d'Ivoire
HRV	Croatia
CUB	Cuba
CYP	Cyprus
CZE	Czech Republic
DNK	Denmark
DJI	Djibouti
DMA	Dominica
DOM	Dominican Republic
ECU	Ecuador
EGY	Egypt
SLV	El Salvador
GNQ	Equatorial Guinea
ERI	Eritrea
EST	Estonia
ETH	Ethiopia
FLK	Falkland Islands (Malvinas)
FRO	Faroe Islands

Code	Nationality
FJI	Fiji
FIN	Finland
FRA	France
GUF	French Guiana
PYF	French Polynesia
ATF	French Southern Territories
GAB	Gabon
GMB	Gambia
GEO	Georgia
DEU	Germany
GHA	Ghana
GIB	Gibraltar
GRC	Greece
GRL	Greenland
GRD	Grenada
GLP	Guadeloupe[2]
GUM	Guam
GTM	Guatemala
GGY	Guernsey
GIN	Guinea
GNB	Guinea-Bissau
GUY	Guyana
HTI	Haiti
HMD	Heard Island and McDonald Islands
VAT	Holy See (Vatican City State)
HND	Honduras
HKG	Hong Kong
HUN	Hungary

Code	Nationality
ISL	Iceland
IND	India
IDN	Indonesia
IRN	Iran, Islamic Republic of
IRQ	Iraq
IRL	Ireland
IMN	Isle of Man
ISR	Israel
ITA	Italy
JAM	Jamaica
JPN	Japan
JEY	Jersey
JOR	Jordan
KAZ	Kazakhstan
KEN	Kenya
KIR	Kiribati
PRK	Korea, Democratic People's Republic of
KOR	Korea, Republic of
KWT	Kuwait
KGZ	Kyrgyzstan
LAO	Lao People's Democratic Republic
LVA	Latvia
LBN	Lebanon
LSO	Lesotho
LBR	Liberia
LBY	Libyan Arab Jamahiriya
LIE	Liechtenstein
LTU	Lithuania

Code	Nationality
LUX	Luxembourg
MAC	Macao
MKD	Macedonia, the former Yugoslav Republic of
MDG	Madagascar
MWI	Malawi
MYS	Malaysia
MDV	Maldives
MLI	Mali
MLT	Malta
MHL	Marshall Islands
MTQ	Martinique
MRT	Mauritania
MUS	Mauritius
MYT	Mayotte
MEX	Mexico
FSM	Micronesia, Federated States of
MDA	Moldova, Republic of
MCO	Monaco
MNG	Mongolia
MNE	Montenegro
MSR	Montserrat
MAR	Morocco
MOZ	Mozambique
MMR	Myanmar
NAM	Namibia
NRU	Nauru
NPL	Nepal
NLD	Netherlands

Code	Nationality
ANT	Netherlands Antilles
NCL	New Caledonia
NZL	New Zealand
NIC	Nicaragua
NER	Niger
NGA	Nigeria
NIU	Niue
NFK	Norfolk Island
MNP	Northern Mariana Islands
NOR	Norway
OMN	Oman
PAK	Pakistan
PLW	Palau
PSE	Palestinian Territory, Occupied
PAN	Panama
PNG	Papua New Guinea
PRY	Paraguay
PER	Peru
PHL	Philippines
PCN	Pitcairn
POL	Poland
PRT	Portugal
PRI	Puerto Rico
QAT	Qatar
REU	Reunion
ROU	Romania
RUS	Russian Federation
RWA	Rwanda

Code	Nationality
SHN	Saint Helena
KNA	Saint Kitts and Nevis
LCA	Saint Lucia
SPM	Saint Pierre and Miquelon
VCT	Saint Vincent and the Grenadines
WSM	Samoa
SMR	San Marino
STP	Sao Tome and Principe
SAU	Saudi Arabia
SEN	Senegal
SRB	Serbia
SYC	Seychelles
SLE	Sierra Leone
SGP	Singapore
SVK	Slovakia
SVN	Slovenia
SLB	Solomon Islands
SOM	Somalia
ZAF	South Africa
SGS	South Georgia and the South Sandwich Islands
ESP	Spain
LKA	Sri Lanka
SDN	Sudan
SUR	Suriname
SJM	Svalbard and Jan Mayen
SWZ	Swaziland
SWE	Sweden
CHE	Switzerland

Code	Nationality
SYR	Syrian Arab Republic
TWN	Taiwan, Province of China
TJK	Tajikistan
TZA	Tanzania, United Republic of
THA	Thailand
TLS	Timor-Leste
TGO	Togo
TKL	Tokelau
TON	Tonga
TTO	Trinidad and Tobago
TUN	Tunisia
TUR	Turkey
TKM	Turkmenistan
TCA	Turks and Caicos Islands
TUV	Tuvalu
UGA	Uganda
UKR	Ukraine
ARE	United Arab Emirates
GBR	United Kingdom
USA	United States
UMI	United States Minor Outlying Islands
URY	Uruguay
UZB	Uzbekistan
VUT	Vanuatu
VEN	Venezuela
VNM	Viet Nam
VGB	Virgin Islands, British
VIR	Virgin Islands, U.S.

Code	Nationality
WLF	Wallis and Futuna
ESH	Western Sahara
YEM	Yemen
ZMB	Zambia
ZWE	Zimbabwe

7.4.32 PivCardRequest/Agency/Applicant/Personal/Citizenship

Reserved for future use. Do not use.

7.4.33 PivCardRequest/Agency/Applicant/Personal/CountryOfBirth

Internal name: `XuPIV12`.

Used for OPM Adjudication – contact customer support for more information, quoting reference SUP-123.

A two-character NCIC country code specifying the country of birth of the applicant. If the country is the United States of America, use the `PlaceOfBirth` node to specify the state.

7.4.34 PivCardRequest/Agency/Applicant/Personal/PlaceOfBirth

Reserved for future use.

7.4.35 PivCardRequest/Agency/Applicant/Personal/ExportRestrictions

Reserved for future use. Do not use.

7.4.36 PivCardRequest/Agency/Applicant/Authentication

Contains information about the security phrases for a user.

7.4.37 PivCardRequest/Agency/Applicant/Authentication/SecurityPhrase

Contains a pair of elements – a `Prompt` and an `Answer` – that provide a security phrase for the applicant.

You can have up to five `SecurityPhrase` elements, each containing a `Prompt` and `Answer` pair.

Note: If you provide any security phrases for an existing user, *all* existing security phrases on that user's account are removed and replaced with the newly-provided security phrases. You can use this feature to delete the security phrases from a user's account – provide a single `Prompt` with an empty `Answer`, and the user's security phrases will be removed. (If the minimum number of security phrases is set to 1, the user will still be prompted for a security phrase when attempting to log on, but will be unable to authenticate.)

7.4.38 PivCardRequest/Agency/Applicant/Authentication/SecurityPhrase/Prompt

Internal name: `Question` field in `SecurityQuestions`

Used to import a security question. If the question does not already exist, an entry is added to the `SecurityQuestions` table; this question will subsequently be available for use when setting security phrases in MyID.

The default security questions are:

ID	Question
1	Password
2	Mothers maiden name?
3	N.I number?
4	Favourite food?
5	Car registration number?
6	Name of pet
7	First school attended
8	An old phone number
9	A memorable date
10	A memorable place
11	A memorable event
12	A famous person
13	Place of birth
14	A country

ID	Question
15	A river
16	A movie
17	A song
18	A book
19	A sport
20	A postcode
21	An animal
22	First car type

7.4.39 PivCardRequest/Agency/Applicant/Authentication/SecurityPhrase/Answer

Internal name: `SecurityPhrase` field in `UsersSecurityQuestions`

Used to import an answer to a security question for an applicant. A hash of the answer is stored in the `UsersSecurityQuestions` table.

Note: If you have set up the **Security Phrase Complexity** option within MyID to determine the complexity of security phrases, this is not enforced on any passwords that you import using the Lifecycle API.

You can encrypt the answer using a transport key – this allows you to maintain the security of the security phrase answer when you include it in the XML data.

To encrypt the answer:

1. Import your AES128 transport key into MyID using the **Key Manager** workflow.
2. External to MyID, encrypt the security phrase answer with the following settings:
   - Encode the security phrase answer as little-endian Unicode. Do not use a byte order mark.

     For example, a security phrase of `answer` would be encoded as:

     `0061006E0073007700650072`
   - Pad the data using ISO9797 Method 2 (pad with a single 0x80, then 0x00 until the block-length is reached).
   - Encrypt using CBC or ECB mode encryption.
3. Add the hex-encoded encrypted data to the `Answer` node.
4. Set the `KeyName` attribute of the `Answer` node to the name of the transport key you imported into MyID.

5. Set the `Mode` attribute of the `Answer` node to either `CBC` or `ECB`, depending on how you encrypted the answer.

For example:

- The transport key, `MyTransportKey`, has a value of:

  `206890FC9B4EA1D0137D8C692B3BFCB6`

- The security phrase is:

  `answer`

- In the import, use the following:

  `<Answer KeyName="MyTransportKey" Mode="CBC">6713987B589F76BC4DA0F557D79A6275</Answer>`

If you do not want to encrypt the answer in the XML document, omit the `KeyName` and `Mode` attributes. For example:

```
<Answer>plaintextanswer1234</Answer>
```

7.4.40 PivCardRequest/Agency/Applicant/Card

Used to specify details for the applicant's card.

7.4.41 PivCardRequest/Agency/Applicant/Card/CardProfile

If present, this will request a card of the given name for this applicant. If performing an update, or the card profile does not exist, no request will be made.

7.4.42 PivCardRequest/Agency/Applicant/Card/CardExpiryDate

Used to specify the expiry date for the card when it is issued. The card will expire on the date specified, or on the date the card profile specifies, whichever is earlier.

The format is `YYYY-MM-DD` – for example, `2016-12-25`.

Note: The card will expire at 00:00:00 UTC on the day of the expiry; that is, at the *start* of the day specified, not the *end*.

7.4.43 PivCardRequest/Agency/Applicant/Card/Renewal

Used to specify whether the card is a renewal of an existing card. If the user already exists, you must set this option to `true` or card request jobs will not be created.

Can be one of the following values:

- `true` – this is a card renewal.
- `false` – this is not a card renewal.

Note: For card renewals, you must specify the card's original serial number and device type using the `PivCardRequest/Agency/Applicant/Card/Update` node.

7.4.44 PivCardRequest/Group/User/Card/ImportCard

Used to specify whether to import a device (see section *3.11*, *Importing operator credentials*).

Can be one of the following values:

- `true` – import a device.
- `false` – do not import a device.

7.4.45 PivCardRequest/Agency/Applicant/Card/CardRequestedBy

This is the logon name of the person requesting the card. Used for auditing purposes.

7.4.46 PivCardRequest/Agency/Applicant/Card/JobLabel

Allows you to specify a label for the renewal or replacement job. You can then search for this label in the **Job Management** or **Batch Issue Card** workflows.

7.4.47 PivCardRequest/Agency/Applicant/Card/Update

Used to specify the card being updated or renewed. You must make sure that the card is issued to the user.

7.4.48 PivCardRequest/Agency/Applicant/Card/Update/OriginalSerialNumber

The serial number of the card being updated or renewed. Used to target a specific card.

Mandatory if you have included the `Update` node.

7.4.49 PivCardRequest/Agency/Applicant/Card/Update/OriginalDeviceType

The device type of the card being updated or renewed.

Mandatory if you have included the `Update` node.

7.4.50 PivCardRequest/Agency/Applicant/Card/Update/StatusMapping

The status mapping code to be used when the card is updated.

See section *3.7.5*, *Status mapping codes* for details.

7.4.51 PivCardRequest/Agency/Applicant/Card/Replacement

Allows you to provide the details of the credential being replaced.

7.4.52 PivCardRequest/Agency/Applicant/Card/Replacement/OriginalSerialNumber

The serial number of the card being replaced. Mandatory if you specify the `Replacement` node.

7.4.53 PivCardRequest/Agency/Applicant/Card/Replacement/OriginalDeviceType

The device type of the card being replaced. Mandatory if you specify the `Replacement` node.

Note: If you specify a valid serial number and device type for the card being replaced (that is, an existing credential that is currently assigned to the applicant) the original card is marked for cancellation and a new card request is created.

If the serial number is invalid (for example, the card does not exist, or is not assigned to any user) no card is marked for cancellation, but the replacement card is still requested.

If the serial number is valid but is assigned to another user, an error is generated; the card is not cancelled, and no replacement card is requested.

7.4.54 PivCardRequest/Agency/Applicant/Card/Replacement/StatusMapping

The status mapping ID for the replacement.

See section *3.7.5*, *Status mapping codes* for details.

7.4.55 PivCardRequest/Agency/Applicant/Card/Reprovision

Set the value of this node to `1` to request a reprovision rather than an update.

A reprovision allows you to re-encode a card completely, based on the data in the MyID database, using the latest version of the credential profile used during issuance.

The card will have the same expiry date as the original card. New certificates may have longer expiration times than the original certificates, but these will not exceed the lifetime of the card itself. Certificates that were revoked externally to MyID will be replaced with new active certificates.

When requesting a reprovision, specify a status mapping with one of the following values:

- `83` – User details have changed.
- `84` – There is a problem with the device.
- `85` – New credential profile needs to be applied.

Note: If you specify any other status mapping, it is ignored – only these status mappings carry out reprovisions.

For example:

```
<Card>
<CardProfile>Yubikey NoOTP</CardProfile>
<CardRequestedBy>System</CardRequestedBy>
<Update>
 <OriginalSerialNumber>8115516</OriginalSerialNumber>
 <OriginalDeviceType>YubiKey 4</OriginalDeviceType>
 <StatusMapping>84</StatusMapping>
</Update>
<Reprovision>1</Reprovision>
</Card>
```

7.4.56 PivCardRequest/Agency/Applicant/Card/CardLayout

Optionally, contains the name of the card layout to use when printing the requested card.

The card layout name must be valid for the card's credential profile at the point that the card is issued; if the card layout name is not valid, the request will not be processed.

7.4.57 PivCardRequest/Agency/Applicant/Card/SerialNumber

Used in conjunction with the PivCardRequest/Agency/Applicant/Card/DeviceType node to specify an exact card to be issued.

Note: If you have the **Only Issue to Known Serial Numbers** option set in the credential profile, and you specify a serial number for the card that has not previously been added to MyID, the import will fail. This failure will be included in the audit.

7.4.58 PivCardRequest/Agency/Applicant/Card/DeviceType

Used in conjunction with the PivCardRequest/Agency/Applicant/Card/SerialNumber node to specify an exact card to be issued.

7.4.59 PivCardRequest/Group/User/Card/Container

Used in conjunction with the `CMSCardRequest/Group/User/Card/ImportCard` node to specify the container that the authentication certificate is located in.

See section *3.11*, *Importing operator credentials*.

7.4.60 PivCardRequest/Group/User/Card/Certificate

Used in conjunction with the `CMSCardRequest/Group/User/Card/ImportCard` node to specify the authentication certificate on the imported card. This must be the base64 representation of the certificate without headers, footers, or line breaks.

See section *3.11*, *Importing operator credentials*.

7.4.61 PivCardRequest/Agency/Applicant/Account

Contains details of the applicant's account.

7.4.62 PivCardRequest/Agency/Applicant/Account/DN

Internal name: `DistinguishedName`

The DN of the users account in the LDAP

7.4.63 PivCardRequest/Agency/Applicant/Account/AlternateDN

Internal name: `Xu55`

If populated this is used in preference to the DN for cert requests

7.4.64 PivCardRequest/Agency/Applicant/Account/CN

Internal name: `CommonName`

7.4.65 PivCardRequest/Agency/Applicant/Account/OU

Internal name: `OrganisationalUnit`

7.4.66 PivCardRequest/Agency/Applicant/Account/UPN

Internal name: `UserPrincipalName`

7.4.67 PivCardRequest/Agency/Applicant/Account/SAMAccountName

Internal name: `SAMAccountName`

Optional SAMAccountName value for the user. Maximum of 20 characters.

7.4.68 PivCardRequest/Agency/Applicant/Account/Domain

Internal name: `Domain`

Optional Domain value for the user. Maximum of 50 characters.

7.4.69 PivCardRequest/Agency/Applicant/Account/LogonName

Internal name: `LogonName`

User's name, used to log into MyID. Must be unique.

If no value is supplied, the `PivCardRequest/Agency/Applicant/Personal/EmployeeID` value is used instead.

7.4.70 PivCardRequest/Agency/Applicant/Account/Roles

Contains the details of the role you want to assign to the applicant.

Note: You can set any role for a user using the Lifecycle API – the roles you specify override any role restrictions. Also, the user will not receive any default roles that have been set up for their group; if you want to assign roles, you must include them explicitly. If you do *not* specify any roles, the user receives the default Cardholder and Password User roles.

7.4.71 PivCardRequest/Agency/Applicant/Account/Roles/Role

Contains details of the role you want to assign the applicant. Contains the following elements:

Field Name	Internal Name	Description
`Name`	`UserProfile`	The name of the role that the applicant is to have. You must specify a role for the applicant.

`Scope`	`Scope`	The scope of the role that the applicant is to have. The value must be one of the following:
		`None`
		`Self`
		`Department`
		`Division`
		`All`
		Note: If you want to remove an existing role from a user's account, set the `Scope` to `None`, and the `RolesActionOnDuplicate` parameter to `MergeEmpty`.

`LogonMechanism`	`LogonMechanism`	Not supported. Do not use.

7.4.72 PivCardRequest/Agency/Applicant/Account/EntrustProfile

Specifies an Entrust profile template to use for the applicant. If specified, the applicant is added to Entrust using the template. See section *3.9*, *Importing users into Entrust* for details.

7.4.73 PivCardRequest/Agency/Applicant/Account/MaxRequestExpiryDate

Allows you to set the maximum credential expiry date for a person – this allows you to specify the latest expiry date for any device issued to this person. Note that card requests cannot exceed the **Lifetime** set in the credential profile. Also, the credential profile can override the maximum expiry date for a person using the **Ignore User Expiry Date** option.

This setting affects all future device requests. It does not affect any issued devices or existing requests.

Note: This setting affects device requests made through the MyID Operator Client only. Requests made through MyID Desktop or the Lifecycle API do not take this setting into account.

See the *Requesting a device for a person* section in the ***MyID Operator Client*** guide for more information.

7.4.74 PivCardRequest/Agency/Applicant/Position

Contains information about the applicant's position within the agency.

7.4.75 PivCardRequest/Agency/Applicant/Position/EmployeeAssociation

Internal name: `Xu4` and `Xu5`

If "Contractor" this is appended to the Applicant's DN. The numeric version stored in `Xu4`.

Note: This node *must* be populated with a valid lookup value for new user imports.

The value can be one of the following:

- Employee
- Civil Worker
- Executive
- Uniformed
- Contractor
- Afficiate
- Beneficiary

7.4.76 PivCardRequest/Agency/Applicant/Position/EmployeeAffiliation

Internal name: `Xu53`

Used for graphical personalization of the card.

7.4.77 PivCardRequest/Agency/Applicant/Position/Rank

Internal name: `Xu1`

Used for graphical personalization of the card.

The value is restricted to a list of codes (for example, `E-1` or `O-6`). See the schema for a full list.

7.4.78 PivCardRequest/Agency/Applicant/Position/EmergencyRole

Internal name: `Xu8`

Can be used to allow additional card profiles to Emergency Responders.

Can be one of the following values:

- Law Enforcement
- Paramedic
- Doctor
- Firefighter
- Bomb Disposal
- Biohazard
- None

7.4.79 PivCardRequest/Agency/Applicant/Position/Privilege

Internal name: `Xu30`

These details are visible in **View Person** and during card issuance approval.

7.4.80 PivCardRequest/Agency/Applicant/Position/ApplicantPosition

Internal name: `Xu11`

These details are visible in **View Person** and during card issuance approval.

7.4.81 PivCardRequest/Agency/Applicant/Position/ExtraInfo

Internal name: `Xu10`

A free-text field to leave comments about an applicant.

7.4.82 PivCardRequest/Agency/Applicant/Sponsor

Contains information about the applicant's sponsor. These details are visible in **View Person** and during card issuance approval.

7.4.83 PivCardRequest/Agency/Applicant/Sponsor/SponsorName

Internal name: `Xu31`

7.4.84 PivCardRequest/Agency/Applicant/Sponsor/SponsorPosition

Internal name: `Xu32`

7.4.85 PivCardRequest/Agency/Applicant/Sponsor/SponsorAgency

Internal name: `Xu34`

7.4.86 PivCardRequest/Agency/Applicant/Sponsor/SponsorEmail

Internal name: `Xu36`

7.4.87 PivCardRequest/Agency/Applicant/Sponsor/SponsorPhoneNumber

Internal name: `Xu35`

7.4.88 PivCardRequest/Agency/Applicant/Biometry

Contains information about the applicant's biometric details. Some of these details are printed on the reverse of some card layouts. Also included in adjudication requests.

7.4.89 PivCardRequest/Agency/Applicant/Biometry/HeightInches

Internal name: `Xu14`

The applicant's height in inches.

Note: This must be numeric. For example, `73` is accepted; `73"` or `6'1"` or `6 foot 1 inch` are not allowed.

7.4.90 PivCardRequest/Agency/Applicant/Biometry/WeightLbs

Internal name: `Xu15`

The applicant's weight in pounds.

Note: The schema allows a total of eight characters for this value; however, this includes the space and `lbs` that is appended during the import. Accordingly there are four characters available for the numeric part of the weight; for example, `9999`, which becomes `9999 lbs` when it is imported into MyID. Do not include the `lbs` in the import data. `140` is accepted; `140 lbs` or `10 stone` are not allowed.

7.4.91 PivCardRequest/Agency/Applicant/Biometry/Gender

Internal name: `Xu38`

7.4.92 PivCardRequest/Agency/Applicant/Biometry/EyeColor

Internal name: `Xu13`

The value of this element is restricted. See the schema for details.

7.4.93 PivCardRequest/Agency/Applicant/Biometry/HairColor

Internal name: `Xu12`

The value of this element is restricted. The current list is:

Code	Description
`XXX`	Unspecified or unknown
`BAL`	Bald
`BLK`	Black
`BLN`	Blonde or Strawberry

BLU	Blue
BRO	Brown
GRY	Gray or Partially Gray
GRN	Green
ONG	Orange
PNK	Pink
PLE	Purple
RED	Red or Auburn
SDY	Sandy
STR	Streaked
WHI	White

7.4.94 PivCardRequest/Agency/Applicant/Biometry/Race

Internal name: `Xu39`

The value of this element is restricted. See the schema for details.

7.4.95 PivCardRequest/Agency/Applicant/Biometry/BioSample

Contains elements to contain fingerprints, EFT data and facial biometric scans.

The possible elements are:

- `R_Little` – the right little finger. Uses the data type `BioFingerSample`.
- `R_Ring` – the right ring finger. Uses the data type `BioFingerSample`.
- `R_Middle` – the right middle finger. Uses the data type `BioFingerSample`.
- `R_Index` – the right index finger. Uses the data type `BioFingerSample`.
- `R_Thumb` – the right thumb. Uses the data type `BioFingerSample`.
- `L_Little` – the left little finger. Uses the data type `BioFingerSample`.
- `L_Ring` – the left ring finger. Uses the data type `BioFingerSample`.

- `L_Middle` – the left middle finger. Uses the data type `BioFingerSample`.
- `L_Index` – the left index finger. Uses the data type `BioFingerSample`.
- `L_Thumb` – the left thumb. Uses the data type `BioFingerSample`.
- `EFT` – the applicant's EFT sample. Uses the data type `EncodedData` with an encoding type of `bmp`. As this is a standard format, the encoding type is actually ignored.
- `FaceScan` – the applicant's facial biometric sample. Uses the data type `EncodedData` with an encoding type of 385.
- `IrisScan` – the applicant's iris sample. Uses the data type `EncodedData` with an encoding type of 19794.

BioFingerSample

The BioFingerSample block is used to provide the data for a single fingerprint. It contains either:

- An `EncodedData` block (see below), with the `Encoding` type of 378 and a `Data` block of base64 data, and:
- An optional `RolledPrint` element, which can be either `Yes` or `No`, determining whether the fingerprint is a roll or a slap.

or:

- A `ReasonMissing` element, which can be either `Unprintable` or `Amputated`, if you cannot provide the data for the specified finger.

For example:

```
<BioSample>
	<R_Index>
		<Encoding>378</Encoding>
		<Data>Rk1SACAyMAAAgAAAAAAAAAEAAWgAxQDFAQAAAAAQgLQAfVUAgMkApaIAgJUAvV4A
gJMA1wEAgCkBBX4AgKkBIpIAgIkBLYQAgEMBOiYAgGoBPn8AgLABQY8AQIwAQrMA
QI4AYV0AQF0AiwwAQKMAkakAQLsAo08AQIkBF4YAAAA=</Data>
		<RolledPrint>No</RolledPrint>
	</R_Index>
	<L_Index>
		<ReasonMissing>Unprintable</ReasonMissing>
	</L_Index>
...
</BioSample>
```

EncodedData

`EncodedData` blocks are used to include binary data in the imported XML; for example, fingerprint data, facial biometrics, and scanned documents. The `EncodedData` block has the following elements:

- `Encoding` – the type of data encoded in the block. This can be one of the following:
  - `jpg` – JPG image data.
  - `gif` – GIF image data.
  - `png` – PNG image data.
  - `bmp` – Windows bitmap image data.
  - `378` – INCITS 378 fingerprint data.
  - `385` – facial biometrics in ANSI/INCITS 385-2004 format.
  - `19794` – iris data.
- `Data` – a string of base64-encoded data containing the file to be uploaded.
- `DateTaken` – the date of the data; for example, when the photograph was taken, or when the fingerprints were captured.

  For user photos only, this is added to the XML within the `ImageProps` field in the `People` table in the MyID database.

  This node uses the `xs:dateTime` format:

  `yyyy-mm-ddThh:mm:ss.nnn+zzzzz`

  For example:

  `2010-01-02T14:23:54.123+01:00`

  Note: This is important for FIPS 201-2, as it lays down requirements for the maximum age of captured biometrics at the point of card issuance.
- `Source` – For user photos only, this is added to the XML within the `ImageProps` field in the `People` table. You can optionally use this field to store information about where the user photo came from.

Removing and updating fingerprints

You can update the fingerprints for applicants by supplying new fingerprint data. Updated fingerprint data for the specified fingers overwrites the previously-recorded fingerprints. If you are using different fingers (for example, if the applicant has had their right index finger amputated) you can remove the existing fingerprints for specific fingers by including an empty `Data` node.

For example:

```
<BioSample>
	<R_Index>
		<Encoding>378</Encoding>
		<Data/>
```

```
        </R_Index>
</BioSample>
```

Alternatively, you can use the `ReasonMissing` element to remove existing fingerprints:

```
<BioSample>
        <R_Index>
                <ReasonMissing>Amputated</ReasonMissing>
        </R_Index>
</BioSample>
```

All changes to the fingerprints stored for applicants are recorded in the audit trail.

7.4.96 PivCardRequest/Agency/Applicant/Biometry/Signature

Contains the applicant's digitized signature. Uses type `EncodedData` with an `Encoding` type of `jpg`. See *EncodedData* on page 92 for details of the `EncodedData` format.

7.4.97 PivCardRequest/Agency/Applicant/IdentityDocs

Contains scans and information about the applicant's identity documents.

The following elements are included, one for each document:

- `IdentityDoc1`
- `IdentityDoc2`
- `SPF10`

Each document contains the following elements:

Field Name	Internal Name	Description	Allowed Values
`Image`		Contains the scanned document. Uses an `EncodedData` block with an `Encoding` type of `jpg`. See *EncodedData* on page 92 for details of the `EncodedData` format.	

Field Name	Internal Name	Description	Allowed Values
Title	Xu21 Xu26	The type of the identity document	Restricted for IdentityDoc1 and IdentityDoc2 – see the schema for a list of supported documents. IDDocName1 is the list for the first identity document, and IDDocName2 is the list for the second identity document.
IssuedBy	Xu22 Xu27	The authority that issue the document	
Number	Xu23 Xu28	A unique identifier for the document	
Expiry	Xu24 Xu29	The expiry date for the document	YYYY-MM-DD

7.4.98 PivCardRequest/Agency/Applicant/NACI

Contains information about the applicant's NACI checks. The NACI checks are visible in the **View Person** workflow.

For example:

```
<NACI>
	<NACCaseNumber>65522</NACCaseNumber>
	<NACICaseNumber>65522</NACICaseNumber>
	<Check>
		<Type>SII</Type>
		<Status>AP</Status>
	</Check>
	<Check>
		<Type>DCII</Type>
		<Status>AP</Status>
	</Check>
	<Check>
		<Type>FBI CHF</Type>
		<Status>AP</Status>
	</Check>
	<Check>
		<Type>NACI</Type>
		<Status>WR</Status>
	</Check>
	<CardIssuanceApproved>YES</CardIssuanceApproved>
	<Comments></Comments>
</NACI>
```

7.4.99 PivCardRequest/Agency/Applicant/NACI/NACCaseNumber

Internal name: `Xu40`

7.4.100 PivCardRequest/Agency/Applicant/NACI/NACICaseNumber

Internal name: `Xu46`

7.4.101 PivCardRequest/Agency/Applicant/NACI/Check

Internal name: `Xu41`, `Xu42`, `Xu43`, `Xu44`, and `Xu45`

Contains one or more NACI background checks. You can have multiple `Check` elements in the import file. Each `Check` element contains the following elements:

- `Type` – The name of the background check. This can be one of the following:
  - `SII`
  - `DCII`
  - `FBI Name` – this option (which mapped to internal ID `Xu43`) is not supported. If you include this value in the import XML, it is ignored. If your system has been upgraded from an earlier version of MyID, your database may still have the `Xu43` field; however, this field is no longer populated or used.
  - `FBI CHF`
  - `NACI`
- `Status` – the status of the NACI check. This can be one of the following:
  - `NR` – Not Requested
  - `WR` – Waiting Response
  - `AS` – Assessing Response
  - `AP` – Approved
  - `RJ` – Rejected

7.4.102 PivCardRequest/Agency/Applicant/NACI/CardIssuanceApproved

Internal name: `UserDataApproved`

Contains information on whether card issuance is approved for this applicant. Can be one of the following values:

- `YES` or `1`
- `NO` or `0`

This flag confirms that the data being sent to MyID has passed all the enrollment checks required for card issuance. You must set this flag when adding an applicant for the first time, and also on card renewal.

Note: Throughout the ***PIV Integration Guide***, this field is referred to as the *User Data Approved* flag.

7.4.103 PivCardRequest/Agency/Applicant/NACI/Comments

Internal name: `Xu33`

Free text description, up to 255 characters, of any background checks.

7.4.104 PivCardRequest/Agency/Applicant/NACI/VettingDate

Internal name: `VettingDate`

Contains the date on which the applicant's identity checks were carried out. The format is:

`YYYY-MM-DDThh:mm:ss`

You can use this field to ensure that the applicant's identity checks are renewed periodically. See the *Identity checks* section in the ***Administration Guide*** for details.

7.4.105 PivCardRequest/Agency/Applicant/Photo

Contains the applicant's photo. This is an `EncodedData` type block, with an `Encoding` value of `jpg`, `gif`, `bmp` or `png`, and the `Data` block containing a base64-encoded image.

See *EncodedData* on page 92 for details of the `EncodedData` format.

7.4.106 PivCardRequest/Agency/Applicant/System

Contains system information for the application.

Required for PIV issuance.

7.4.107 PivCardRequest/Agency/Applicant/System/GUID

Internal name: `Xu51`

Unique ID used as part of the Card Holder Unique ID (CHUID).

Mandatory.

7.4.108 PivCardRequest/Agency/Applicant/System/CredNo

Internal name: `Xu48`

Credential Number – encoded in the FASC-N.

Mandatory.

7.4.109 PivCardRequest/Agency/Applicant/System/CS

Credential Series – always set to 1, but is ignored in the import anyway and the result is always 1. If you need to increment the credential series, contact customer support for details.

7.4.110 PivCardRequest/Agency/Applicant/System/POA

Legacy field – replaced by Applicant/EmployeeAssociation.

7.4.111 PivCardRequest/Agency/Applicant/System/ICI

Internal name: `Xu47`

The issue number of the next card to be issued. This is encoded in the FASCN. Recommend coding as a `1` always.

Mandatory only for Add.

7.4.112 PivCardRequest/Agency/Applicant/System/UserStatus

Internal name: `Xu50`

The state of the applicant in the process. This is displayed in the View Person workflow. Can be one of the following:

- `Sponsored`
- `EnrollmentBegun`
- `EnrollmentComplete`
- `Vetting`
- `Escalated`
- `Approved`

- `CardRequested`
- `CardActivated`
- `Rejected`

Note: The schema specifies a maximum of 50 characters, but the MyID limit is 40 characters.

7.4.113 PivCardRequest/Agency/Applicant/System/IssuanceNotifyURL

Internal name: `Xu56`

An override for the destination of the Card Issuance Notification XML that is sent when a card for the Applicant is issued.

Note: This notification is sent only when you issue a card through the MyID Desktop application. If you collect a card through the Self-Service App or Self-Service Kiosk, you must set up the Notifications table instead – see the ***PIV Integration Guide*** for details.

7.4.114 PivCardRequest/Agency/Applicant/System/CancelNotifyURL

Specifies an URL to be used for notifications for card cancellations. This allows you to specify a different URL for cancellations and issuance.

7.4.115 PivCardRequest/Agency/Applicant/System/CMSIssuanceNotifyURL

Specifies an URL to be used for notifications for card issuance. This allows you to specify a different URL for cancellations and issuance.

Note: Reserved for future use.

7.4.116 PivCardRequest/Agency/Applicant/AdminGroups

An optional node that allows you to specify one or more administrative groups for the user. See the Administration Guide for details of setting up and using administrative groups.

The `AdminGroups` node contains one or more `AdminGroup`.

For example:

```
<AdminGroups>
                <AdminGroup>
                                <Name>Group 1</Name>
                </AdminGroup>
</AdminGroups>
```

or:

```
<AdminGroups>
                <AdminGroup>
                                <Name>Group 1</Name>
                </AdminGroup>
                <AdminGroup>
                                <Name>Group 2</Name>
                </AdminGroup>
</AdminGroups>
```

If the user has existing administrative groups, they are replaced by the specified group or groups. If you specify an empty `AdminGroups` node, this clears all of the user's administrative groups. If you do not specify an `AdminGroups` node, the user's administrative groups remain unchanged.

7.4.117 PivCardRequest/Agency/Applicant/AdminGroups/AdminGroup

Specifies an administrative group for the user. You can have multiple `AdminGroup` nodes in the `AdminGroups` parent node. An `AdminGroup` node contains a single `Name` node.

7.4.118 PivCardRequest/Agency/Applicant/AdminGroups/AdminGroup/Name

Specifies the name of the administrative group for the user.

Note: The group must already exist in the MyID database. If the group does not exist, the import will fail.

7.4.119 PivCardRequest/Agency/Applicant/AdditionalFields

You can import information into additional extended fields using the data import. See section *3.8.1*, *Additional user fields* for details.

7.4.120 PivCardRequest/Agency/Applicant/Actions

Contains actions to carry out for the applicant; for example, to remove or disable the user, or to cancel the user's credentials.

7.4.121 PivCardRequest/Agency/Applicant/Actions/ApplicantAction

Perform an action on the applicant. If you do not supply an action in the ApplicantAction element, a suspended applicant will be unsuspended.

The possible actions are:

- `Disable` – disable the user's account.
- `Remove` – remove the user from the system.
- `CancelDevices` – cancel all the user's credentials.
- `CancelDevice` – cancel specific credentials. Specify the credentials using the `Device` element.
- `CancelJob` – Cancel a specific job. Specify the job using the `PivCardRequest/Agency/Applicant/Actions/Job` element.
- `CancelAllJobs` – Cancel all jobs for the specified user.
- `RenewCertificate` – renew a certificate. Specify the certificate using `PivCardRequest/Agency/Applicant/Actions/CertSerialNumber` and `PivCardRequest/Agency/Applicant/Actions/CertPolicyName`.

7.4.122 PivCardRequest/Agency/Applicant/Actions/CertSerialNumber

Used in conjunction with an `ApplicantAction` of `RenewCertificate` and `PivCardRequest/Agency/Applicant/Actions/CertPolicyName` to specify a certificate to renew.

7.4.123 PivCardRequest/Agency/Applicant/Actions/CertPolicyName

Used in conjunction with an `ApplicantAction` of `RenewCertificate` and `PivCardRequest/Agency/Applicant/Actions/CertSerialNumber` to specify a certificate to renew.

7.4.124 PivCardRequest/Agency/Applicant/Actions/StatusMappingID

The action to apply to the user. This should be an ID that matches an entry in the `StatusMappings` table. It controls how certificates are revoked.

Note: You cannot specify negative IDs from the `StatusMappings` table – these are reserved for system use.

7.4.125 PivCardRequest/Agency/Applicant/Actions/RevocationComment

A free text description that is stored against revoked certificates in the database.

7.4.126 PivCardRequest/Agency/Applicant/Actions/Device

Specifies the credential (for example, a smart card) to cancel when you have selected `CancelDevice` for the `ApplicantAction`. Contains at least one `DeviceIdentifier` element. You can specify more than one credential to cancel by including multiple `DeviceIdentifier` elements.

7.4.127 PivCardRequest/Agency/Applicant/Actions/Device/DeviceIdentifier

Specifies the credential to be canceled or unlocked.

7.4.128 PivCardRequest/Agency/Applicant/Actions/Device/DeviceIdentifier/SerialNumber

The serial number of the credential.

7.4.129 PivCardRequest/Agency/Applicant/Actions/Device/DeviceIdentifier/SerialNumberField

The name of the field in which to look for the serial number. The field can be any field present in the `vDevices` view; for example, `SerialNumber` for the standard serial number or `HIDSerialNumber` for the HID serial number.

If you do not include a `SerialNumberField` node, MyID will use a default value of `SerialNumber`.

7.4.130 PivCardRequest/Agency/Applicant/Actions/Device/ProcessStatus

Used to specify the process status to be used for the credential being cancelled.

You can use the following status codes, as listed in the `DeviceProcessStatuses` table in the database.

Status	Description
`Active`	Device is operational
`Assigned`	Device is assigned to a user but has not begun personalization
`AtBureau`	Device is at the bureau
`Collected`	Collected
`Disposed`	Disposed
`Erased`	Erased
`Legacy`	Legacy
`Lost`	Lost
`None`	None (Default the card may be re-issued)
`Not Disposed`	Not Collected
`PendingActivation`	Device is ready to be activated
`PendingDeliveryConfirmation`	Device has been dispatched from the bureau and is in transit
`PendingPersonalisation`	Device is ready to be personalized
`Terminated`	Card has been terminated
`Unassigned`	Device is not assigned to a user

7.4.131 PivCardRequest/Agency/Applicant/Actions/Job

Contains a the ID of a job to cancel. Used in conjunction with an `ApplicantAction` of `CancelJob`.

7.5 PivApplicantUpdate structure

The PivApplicantUpdate schema uses the same data elements as the PivCardRequest schema. This schema requires less information as a minimum, however. See the schema for details of the minimum data required.

For example:

- `PivCardRequest/Agency/Applicant/Sponsor` is mandatory for PivCardRequest.
- `PivCardRequest/Agency/Applicant/Sponsor/SponsorName` is mandatory for PivCardRequest.
- `PivCardRequest/Agency/Applicant/Sponsor/SponsorAgency` is mandatory for PivCardRequest.

- `PivCardRequest/Agency/Applicant/System/GUID` is mandatory for PivCardRequest.
- `PivCardRequest/Agency/Applicant/System/CredNo` is mandatory for PivCardRequest.
- `PivCardRequest/Agency/Applicant/System/ICI` is mandatory for PivCardRequest.

7.6 PivImportResponse structure

The PivImportResponse schema is used when the XML import service passes back information about the results of the applicant data import.. It contains blocks for the `Agency` and the `Applicant`.

7.6.1 PivImportResponse/Agency

Contains information about the agency.

7.6.2 PivImportResponse/Agency/AgencyName

Internal name: `Name` and `Xu3`

Used to identify the Agency. Also encoded in the DN for member Applicants.

7.6.3 PivImportResponse/Agency/AgencyCode

Internal name: `Xg1`

Four alphanumeric characters. Encoded in the FASCN and in the PrintedInfo block.

7.6.4 PivImportResponse/Agency/Result

The result of the import for the agency. Can be one of the following:

- `Created`
- `Already Exists`
- `Failed`

7.6.5 PivImportResponse/Agency/Applicant

Contains information about the applicant.

7.6.6 PivImportResponse/Agency/Applicant/FirstName

Internal name: `FirstName`

The applicant's first name.

7.6.7 PivImportResponse/Agency/Applicant/LastName

Internal name: `Surname`

The applicant's last name.

7.6.8 PivImportResponse/Agency/Applicant/EmployeeID

Internal name: `EmployeeID`

This is a unique, numerical identifier for the applicant.

7.6.9 PivImportResponse/Agency/Applicant/CardRequest

Internal name: `Jobs.ID`

The JobID for the card request. If there is no request, returns 0.

7.6.10 PivImportResponse/Agency/Applicant/CardReqestFailReason

The reason the card request failed.

7.6.11 PivImportResponse/Agency/Applicant/Result

The result of the import for the applicant. Can be one of the following:

- `Added`
- `Already Exists`

- `Failed`
- `Removed`
- `License Limit`

7.6.12 PivImportResponse/Agency/Applicant/Reason

Provides information about the response, including details of the reason why the request failed; for example, the name of a non-existent credential profile.

8 MyIDEnroll WSDL

The WSDL definition for the MyIDEnroll web service is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://www.intercede.com/MyIDEnroll" xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://www.intercede.com/MyIDEnroll" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.intercede.com/MyIDEnroll">
      <s:element name="PIVXMLWebImport">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="xmlIn" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="PIVXMLWebImportResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="PIVXMLWebImportResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="CMSXMLWebImport">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="xmlIn" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="CMSXMLWebImportResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="CMSXMLWebImportResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="XMLImport">
        <s:complexType>
          <s:sequence>
```

```
          <s:element minOccurs="0" maxOccurs="1" name="ParametersXML" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="xmlData" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="XMLImportResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="XMLImportResult" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:schema>
</wsdl:types>
<wsdl:message name="PIVXMLWebImportSoapIn">
  <wsdl:part name="parameters" element="tns:PIVXMLWebImport" />
</wsdl:message>
<wsdl:message name="PIVXMLWebImportSoapOut">
  <wsdl:part name="parameters" element="tns:PIVXMLWebImportResponse" />
</wsdl:message>
<wsdl:message name="CMSXMLWebImportSoapIn">
  <wsdl:part name="parameters" element="tns:CMSXMLWebImport" />
</wsdl:message>
<wsdl:message name="CMSXMLWebImportSoapOut">
  <wsdl:part name="parameters" element="tns:CMSXMLWebImportResponse" />
</wsdl:message>
<wsdl:message name="XMLImportSoapIn">
  <wsdl:part name="parameters" element="tns:XMLImport" />
</wsdl:message>
<wsdl:message name="XMLImportSoapOut">
  <wsdl:part name="parameters" element="tns:XMLImportResponse" />
</wsdl:message>
<wsdl:portType name="MyIDEnrollSoap">
  <wsdl:operation name="PIVXMLWebImport">
    <wsdl:input message="tns:PIVXMLWebImportSoapIn" />
    <wsdl:output message="tns:PIVXMLWebImportSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="CMSXMLWebImport">
    <wsdl:input message="tns:CMSXMLWebImportSoapIn" />
    <wsdl:output message="tns:CMSXMLWebImportSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="XMLImport">
    <wsdl:input message="tns:XMLImportSoapIn" />
    <wsdl:output message="tns:XMLImportSoapOut" />
```

```
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="MyIDEnrollSoap" type="tns:MyIDEnrollSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="PIVXMLWebImport">
    <soap:operation soapAction="http://www.intercede.com/MyIDEnroll/PIVXMLWebImport" style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="CMSXMLWebImport">
    <soap:operation soapAction="http://www.intercede.com/MyIDEnroll/CMSXMLWebImport" style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="XMLImport">
    <soap:operation soapAction="http://www.intercede.com/MyIDEnroll/XMLImport" style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="MyIDEnrollSoap12" type="tns:MyIDEnrollSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="PIVXMLWebImport">
    <soap12:operation soapAction="http://www.intercede.com/MyIDEnroll/PIVXMLWebImport" style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="CMSXMLWebImport">
```

```
      <soap12:operation soapAction="http://www.intercede.com/MyIDEnroll/CMSXMLWebImport" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="XMLImport">
      <soap12:operation soapAction="http://www.intercede.com/MyIDEnroll/XMLImport" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="MyIDEnroll">
    <wsdl:port name="MyIDEnrollSoap" binding="tns:MyIDEnrollSoap">
      <soap:address location="http://localhost/MyiDEnroll/MyIDEnroll.asmx" />
    </wsdl:port>
    <wsdl:port name="MyIDEnrollSoap12" binding="tns:MyIDEnrollSoap12">
      <soap12:address location="http://localhost/MyiDEnroll/MyIDEnroll.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

9 XML schemas

This section contains listings of the Lifecycle schemas available.

9.1 Schema file locations

The schema files are installed to the `Schemas` folder the `MyIDEnroll` folder on the MyID web server; by default, this is:

```
C:\Program Files (x86)\Intercede\MyID\Web\MyIDEnroll\Schemas
```

9.2 MyIDBaseTypes

This schema contains the elements common to all the MyID Lifecycle API schemas.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.intercede.com/CommonTypes"
    xmlns:tng="http://www.intercede.com/CommonTypes"
    elementFormDefault="qualified" attributeFormDefault="qualified"
>
  <xs:simpleType name="String">
    <xs:restriction base="xs:string">
      <xs:pattern value="((?!&lt;.*&gt;)(?!_no_match_)(?!]]&gt;).)*" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="NotEmptyString">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NotEmptyString4">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
      <xs:maxLength value="4"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NotEmptyString6">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
      <xs:maxLength value="6"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NotEmptyString8">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NotEmptyString40">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
```

```
      <xs:maxLength value="40"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NotEmptyString50">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
      <xs:maxLength value="50"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NotEmptyString64">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
      <xs:maxLength value="64"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NotEmptyString80">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
      <xs:maxLength value="80"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NotEmptyString100">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
      <xs:maxLength value="100"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NotEmptyString255">
    <xs:restriction base="tng:String">
      <xs:minLength value="1" />
      <xs:maxLength value="255"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="String30">
    <xs:restriction base="tng:String">
      <xs:maxLength value="30"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="String80">
    <xs:restriction base="tng:String">
      <xs:maxLength value="80"/>
    </xs:restriction>
  </xs:simpleType>
```

```
<xs:simpleType name="String100">
  <xs:restriction base="tng:String">
    <xs:maxLength value="100"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String120">
  <xs:restriction base="tng:String">
    <xs:maxLength value="120"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String200">
  <xs:restriction base="tng:String">
    <xs:maxLength value="200"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String255">
  <xs:restriction base="tng:String">
    <xs:maxLength value="255"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String1024">
  <xs:restriction base="tng:String">
    <xs:maxLength value="1024"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String2">
  <xs:restriction base="tng:String">
    <xs:maxLength value="2"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String20">
  <xs:restriction base="tng:String">
    <xs:maxLength value="20"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String8">
  <xs:restriction base="tng:String">
    <xs:maxLength value="8"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String40">
  <xs:restriction base="tng:String">
    <xs:maxLength value="40"/>
  </xs:restriction>
```

```
</xs:simpleType>
<xs:simpleType name="String50">
  <xs:restriction base="tng:String">
    <xs:maxLength value="50"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String64">
  <xs:restriction base="tng:String">
    <xs:maxLength value="64"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String10">
  <xs:restriction base="tng:String">
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String1">
  <xs:restriction base="tng:String">
    <xs:maxLength value="1"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String3">
  <xs:restriction base="tng:String">
    <xs:maxLength value="3"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String4">
  <xs:restriction base="tng:String">
    <xs:maxLength value="4"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="String6">
  <xs:restriction base="tng:String">
    <xs:maxLength value="6"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="StringBase64">
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z0-9+ /=\n\r\w\t]*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Int50">
```

```
    <xs:restriction base="xs:string">
      <xs:maxLength value="50"/>
      <xs:pattern value="[0-9]*"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Int1">
    <xs:restriction base="xs:string">
      <xs:maxLength value="1"/>
      <xs:pattern value="[0-9]"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Int4">
    <xs:restriction base="xs:string">
      <xs:maxLength value="4"/>
      <xs:pattern value="[0-9]*"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Int9">
    <xs:restriction base="xs:string">
      <xs:maxLength value="9"/>
      <xs:pattern value="[0-9]*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="TrueFalse">
    <xs:restriction base="xs:string">
      <xs:enumeration value="true"/>
      <xs:enumeration value="false"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="YesNo">
    <xs:restriction base="xs:string">
      <xs:enumeration value="YES"/>
      <xs:enumeration value="NO"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="YesNoOneZero">
    <xs:restriction base="xs:string">
      <xs:enumeration value="YES"/>
      <xs:enumeration value="NO"/>
      <xs:enumeration value="1"/>
      <xs:enumeration value="0"/>
```

```
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="EncryptedData">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="KeyName" type="xs:string" use="optional" />
        <xs:attribute name="Mode" type="xs:string" use="optional" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="NotEmptyEncryptedData">
    <xs:simpleContent>
      <xs:restriction base="tng:EncryptedData">
        <xs:minLength value="1" />
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="OperationType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="=" />
      <xs:enumeration value="!=" />
      <xs:enumeration value="gt" />
      <xs:enumeration value="lt" />
      <xs:enumeration value="ge" />
      <xs:enumeration value="le" />
      <xs:enumeration value="isnull" />
      <xs:enumeration value="notnull" />
      <xs:enumeration value="leftbracket" />
      <xs:enumeration value="rightbracket" />
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

9.3 CMSSchemaTypes

This schema contains the elements common to the `CMSCardRequest` and `CMSUserUpdate` schemas.

```
<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
  xmlns:myidbase="http://www.intercede.com/CommonTypes"
  targetNamespace="http://www.intercede.com/MyIDSchema/CMSCardRequest"
  xmlns="http://www.intercede.com/MyIDSchema/CMSCardRequest"
  elementFormDefault="qualified" attributeFormDefault="qualified"
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
  >

<xs:import namespace="http://www.intercede.com/CommonTypes" schemaLocation="MyIDBaseTypes.xsd"/>

<xs:complexType name="ParametersBlockType">
    <xs:all>
          <xs:element name="SourceID" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="IssueDate" type="xs:date" minOccurs="0" maxOccurs="1"/>
          <xs:element name="GenerateUserDN" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
          <xs:element name="ActionOnDuplicate" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="RolesActionOnDuplicate" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="DeleteMissingUsers" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
          <xs:element name="PushToLDAP" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
          <xs:element name="CreateUnknownGroups" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
          <xs:element name="AuditAll" type="myidbase:Int1" minOccurs="1" maxOccurs="1"/>
          <xs:element name="DataType" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="CheckImportResponseSchema" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
          <xs:element name="SynchronousImport" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
          <xs:element name="AllowBioImport" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
          <xs:element name="DefaultUserRole" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="DisallowCertificateSuspension" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
          <xs:element name="CardRequestThrottling" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
          <xs:element name="ReplaceUnassignedCards" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="PersonalBlockType">
  <xs:sequence>
    <xs:choice>
      <xs:sequence>
        <xs:element name="FirstName" type="myidbase:NotEmptyString64" minOccurs="1" maxOccurs="1"/>
        <xs:element name="LastName" type="myidbase:NotEmptyString64" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:element name="LastName" type="myidbase:NotEmptyString64" minOccurs="1" maxOccurs="1"/>
    </xs:choice>
    <xs:element name="Initial" type="myidbase:String40" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Title" type="myidbase:String20" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Email" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="PhoneExt" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
```

```
    <xs:element name="MobileNumber" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
    <xs:element name="PhoneNumber" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
    <xs:element name="EmployeeID" type="myidbase:String50" minOccurs="1" maxOccurs="1"/>
    <xs:element name="OptionalLine1" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="OptionalLine2" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="OptionalLine3" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="OptionalLine4" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="AuthenticationBlockType">
  <xs:sequence>
    <xs:element name="SecurityPhrase" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Prompt" type="myidbase:NotEmptyString255" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Answer" type="myidbase:EncryptedData" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="CardBlockType">
  <xs:sequence>
    <xs:element name="CardProfile" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="CardExpiryDate" type="xs:date" minOccurs="0"/>
    <xs:element name="Renewal" type="RenewalString" minOccurs="0"/>
    <xs:element name="ImportCard" type="myidbase:TrueFalse" minOccurs="0"/>
    <xs:element name="CardRequestedBy" type="myidbase:String255" minOccurs="0"/>
    <xs:element name="CancelExisting" minOccurs="0" maxOccurs="1"/>
    <xs:element name="JobLabel" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="Update" minOccurs="0" maxOccurs="1" type="UpdateCardBlockType"/>
    <xs:element name="Replacement" minOccurs="0" type="ReplacementCardBlockType"/>
    <xs:element name="GenerateOTP" minOccurs="0" maxOccurs="1" type="GenerateOTPBlockType"/>
    <xs:element name="OriginalSerialNumber" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
    <xs:element name="OriginalDeviceType" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
    <xs:element name="SerialNumber" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
    <xs:element name="DeviceType" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
    <xs:element name="StatusMapping" type="xs:unsignedByte" minOccurs="0" maxOccurs="1"/>
    <xs:element name="RevocationComment" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Reprovision" type="xs:unsignedByte" minOccurs="0" maxOccurs="1"/>
    <xs:element name="CardLayout" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
    <xs:element name="Container" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
```

```
    <xs:element name="Certificate" type="myidbase:StringBase64" minOccurs="0" maxOccurs="1" />
    <xs:element name="AdditionalFields" minOccurs="0" maxOccurs="1">
      <xs:complexType mixed="true">
        <xs:sequence>
          <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="UpdateCardBlockType">
  <xs:sequence>
    <xs:element name="OriginalSerialNumber" type="myidbase:String50" minOccurs="1" maxOccurs="1" />
    <xs:element name="OriginalDeviceType" type="myidbase:String50" minOccurs="1" maxOccurs="1" />
      <xs:element name="StatusMapping" type="xs:int" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ReplacementCardBlockType">
  <xs:all>
    <xs:element name="OriginalSerialNumber" type="myidbase:String50" minOccurs="1" maxOccurs="1"/>
    <xs:element name="OriginalDeviceType" type="myidbase:String50" minOccurs="1" maxOccurs="1"/>
    <xs:element name="StatusMapping" type="xs:int" minOccurs="0" maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="CardUpdateBlockType">
  <xs:all>
    <xs:element name="SerialNumber" type="myidbase:String255" minOccurs="1" maxOccurs="1"/>
    <xs:element name="DeviceType" type="myidbase:String255" minOccurs="1" maxOccurs="1"/>
    <xs:element name="CardRequestedBy" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="ParametersXML" type="JobParameters" minOccurs="1" maxOccurs="1"/>
    <xs:element name="PIN" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="AccountBlockType">
  <xs:sequence>
    <xs:element name="DN" type="myidbase:String1024" minOccurs="0" maxOccurs="1"/>
    <xs:element name="CN" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="OU" type="myidbase:String1024" minOccurs="0" maxOccurs="1"/>
    <xs:element name="UPN" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="SAMAccountName" type="myidbase:String20" minOccurs="0" maxOccurs="1"/>
```

```
    <xs:element name="Domain" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
    <xs:element name="LogonName" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="NewLogonName" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="UniqueID" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Roles" minOccurs="0" maxOccurs="1" type="RolesBlockType"/>
    <xs:element name="EntrustProfile" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
    <xs:element name="UserDataApproved" type="myidbase:YesNoOneZero" minOccurs="0" maxOccurs="1"/>
    <xs:element name="MaxRequestExpiryDate" type="xs:date" minOccurs="0" maxOccurs="1"/>
    <xs:element name="VettingDate" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="RolesBlockType">
  <xs:sequence>
    <xs:element name="Role" minOccurs="0" maxOccurs="unbounded" type="RoleBlockType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="RoleBlockType">
  <xs:sequence>
    <xs:element name="Name" type="myidbase:String255" minOccurs="1"/>
    <xs:element name="Scope" type="RoleScope" minOccurs="0" maxOccurs="1"/>
    <xs:element name="LogonMechanism" type="RoleLogonMechanism" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="PhotoBlockType">
  <xs:sequence>
       <xs:choice>
      <xs:sequence>
           <xs:element name="Encoding" type="FileEncoding" maxOccurs="1" minOccurs="1"/>
        <xs:element name="Data" type="myidbase:StringBase64" maxOccurs="1" minOccurs="1"/>
        <xs:element name="DateTaken" type="xs:dateTime" maxOccurs="1" minOccurs="0"/>
        <xs:element name="Source" type="myidbase:String50" maxOccurs="1" minOccurs="0"/>
         </xs:sequence>
         <xs:element name="None" type="myidbase:String50" maxOccurs="1" minOccurs="1"/>
       </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="FileEncoding">
  <xs:restriction base="xs:string">
    <xs:enumeration value="jpg"/>
    <xs:enumeration value="gif"/>
```

```
      <xs:enumeration value="png"/>
      <xs:enumeration value="bmp"/>
      <xs:enumeration value="378"/>
      <xs:enumeration value="385"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="RoleScope">
    <xs:restriction base="xs:string">
      <xs:enumeration value="None"/>
      <xs:enumeration value="Self"/>
      <xs:enumeration value="Department"/>
      <xs:enumeration value="Division"/>
      <xs:enumeration value="All"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="RoleLogonMechanism">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Password"/>
      <xs:enumeration value="Card"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="UserAction">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Disable"/>
      <xs:enumeration value="Remove"/>
      <xs:enumeration value="CancelDevices"/>
      <xs:enumeration value="CancelDevice"/>
      <xs:enumeration value="CancelJob"/>
      <xs:enumeration value="CancelAllJobs"/>
      <xs:enumeration value="UnlockCard"/>
      <xs:enumeration value="RenewCertificate"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="RenewalString">
    <xs:restriction base="xs:string">
      <xs:enumeration value="true"/>
      <xs:enumeration value="false"/>
    </xs:restriction>
  </xs:simpleType>
```

```
<xs:complexType name="JobParameters">
  <xs:sequence>
    <xs:element name="UnlockPIN" type="xs:unsignedInt" maxOccurs="1" minOccurs="0" />
  </xs:sequence>
</xs:complexType>



<xs:complexType name="ActionsBlockType">
  <xs:sequence>
    <xs:element name="ApplicantAction" type="UserAction" minOccurs="0" maxOccurs="1"/>
    <!-- MPS-69 to support certificate renewal -->
    <xs:element name="CertSerialNumber" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
    <xs:element name="CertPolicyName" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
    <!-- -->
    <xs:element name="StatusMappingID" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
    <xs:element name="RevocationComment" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="RevocationDelay" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Device" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="DeviceIdentifier" minOccurs="0" maxOccurs="unbounded" type="DeviceIdentifierBlockType"/>
                  <xs:element name="ProcessStatus" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="RequestedBy" type="myidbase:String255" minOccurs="0" maxOccurs="1" />
    <xs:element name="Job" type="xs:unsignedInt" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="Filters" type="FilterCriteria" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="DeviceIdentifierBlockType">
  <xs:sequence>
    <xs:element name="SerialNumber" type="myidbase:String50" minOccurs="1"/>
    <xs:element name="SerialNumberField" type="myidbase:String50" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="FilterCriteria">
  <xs:sequence>
    <xs:element name="Filter" minOccurs="1" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
```

```
            <xs:element name="FieldName" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="FieldValue" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="Operation" minOccurs="1" maxOccurs="1" type="myidbase:OperationType"/>
            <xs:element name="Or" type="xs:boolean" minOccurs="0" maxOccurs="1" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="GenerateOTPBlockType">
    <xs:sequence>
      <xs:element name="Notification" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Name" type="myidbase:String255" minOccurs="1" maxOccurs="1" />
            <xs:element name="Action" type="myidbase:String10" minOccurs="0" maxOccurs="1" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="AdminGroupsBlockType">
    <xs:sequence>
      <xs:element name="AdminGroup" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Name" minOccurs="1" type="myidbase:String100" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  </xs:schema>
```

9.4 CMSCardRequest schema

This schema is used for user data sent to the import service to request cards.

```
<?xml version="1.0"?>
<xs:schema
  id="CMSCardRequest"
```

```
 targetNamespace="http://www.intercede.com/MyIDSchema/CMSCardRequest"
 xmlns:myidbase="http://www.intercede.com/CommonTypes"
 xmlns="http://www.intercede.com/MyIDSchema/CMSCardRequest"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
 attributeFormDefault="unqualified" elementFormDefault="qualified"
>
 <xs:include schemaLocation="CMSSchemaTypes.xsd" />
 <xs:import namespace="http://www.intercede.com/CommonTypes" schemaLocation="MyIDBaseTypes.xsd"/>

 <xs:element name="CMSCardRequest" msdata:IsDataSet="true" msdata:Locale="en-GB" msdata:EnforceConstraints="false">
            <xs:complexType>
                  <xs:sequence>
       <xs:element name="Parameters" minOccurs="0" maxOccurs="1" type="ParametersBlockType"/>
       <xs:element name="Group" minOccurs="0" maxOccurs="1" type="GroupBlockType"/>
                  </xs:sequence>
            </xs:complexType>
     </xs:element>

 <xs:complexType name="GroupBlockType">
   <xs:sequence>
     <xs:element name="Name" type="myidbase:String100" minOccurs="1"/>
     <xs:element name="Description" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
     <xs:element name="OrgUnit" type="myidbase:String1024" minOccurs="0"/>
     <xs:element name="Parent" type="myidbase:String80" minOccurs="0"/>
     <xs:element name="AdditionalFields" minOccurs="0" maxOccurs="1">
       <xs:complexType mixed="true">
         <xs:sequence>
           <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
         </xs:sequence>
       </xs:complexType>
     </xs:element>
     <xs:element name="User" minOccurs="0" maxOccurs="1" type="UserBlockType"/>
   </xs:sequence>
 </xs:complexType>

 <xs:complexType name="UserBlockType">
   <xs:sequence>
     <xs:element name="Personal" type="PersonalBlockType"/>
     <xs:element name="Authentication" type="AuthenticationBlockType" minOccurs="0" maxOccurs="1" />
     <xs:element name="Card" minOccurs="0" type="CardBlockType"/>
     <xs:element name="CardUpdate" minOccurs="0" type="CardUpdateBlockType"/>
     <xs:element name="Account" minOccurs="0" type="AccountBlockType" />
     <xs:element name="Photo" minOccurs="0" type="PhotoBlockType"/>
```

```
      <xs:element name="AdminGroups" minOccurs="0" maxOccurs="1" type="AdminGroupsBlockType" />
      <xs:element name="AdditionalFields" minOccurs="0" maxOccurs="1">
        <xs:complexType mixed="true">
          <xs:sequence>
            <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Actions"  minOccurs="0" maxOccurs="1" type="ActionsBlockType"/>
    </xs:sequence>
  </xs:complexType>



</xs:schema>
```

9.5 CMSUserUpdate schema

This schema is used to update existing users.

```
<?xml version="1.0"?>
<xs:schema
  id="CMSCardRequest"
  targetNamespace="http://www.intercede.com/MyIDSchema/CMSCardRequest"
  xmlns:myidbase="http://www.intercede.com/CommonTypes"
  xmlns="http://www.intercede.com/MyIDSchema/CMSCardRequest"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
  attributeFormDefault="unqualified" elementFormDefault="qualified"
 >

  <xs:include schemaLocation="CMSSchemaTypes.xsd" />
  <xs:import namespace="http://www.intercede.com/CommonTypes" schemaLocation="MyIDBaseTypes.xsd"/>

  <xs:element name="CMSCardRequest" msdata:IsDataSet="true" msdata:Locale="en-GB" msdata:EnforceConstraints="false">
            <xs:complexType>
                  <xs:sequence>
       <xs:element name="Parameters" minOccurs="0" maxOccurs="1" type="ParametersBlockType"/>
       <xs:element name="Group" maxOccurs="unbounded" minOccurs="0" type="GroupBlockType"/>
                        <xs:element name="User" minOccurs="0" maxOccurs="unbounded" type="UserBlockType"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
```

```
  <xs:complexType name="GroupBlockType">
    <xs:sequence>
      <xs:element name="Name" type="myidbase:String100" minOccurs="1"/>
         <xs:element name="Description" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
      <xs:element name="OrgUnit" type="myidbase:String1024" minOccurs="0"/>
      <xs:element name="Parent" type="myidbase:String80" minOccurs="0"/>
      <xs:element name="AdditionalFields" minOccurs="0" maxOccurs="1">
        <xs:complexType mixed="true">
          <xs:sequence>
            <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="User" maxOccurs="unbounded" type="UserBlockType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="UserBlockType">
    <xs:sequence>
      <xs:element name="Personal" type="PersonalBlockType"/>
      <xs:element name="Authentication" minOccurs="0" maxOccurs="1" type="AuthenticationBlockType"/>
      <xs:element name="Card" minOccurs="0" type="CardBlockType"/>
      <xs:element name="Account" minOccurs="0" type="AccountBlockType" />
      <xs:element name="Photo" minOccurs="0" type="PhotoBlockType"/>
      <xs:element name="AdminGroups" minOccurs="0" maxOccurs="1" type="AdminGroupsBlockType" />
      <xs:element name="AdditionalFields" minOccurs="0" maxOccurs="1">
        <xs:complexType mixed="true">
          <xs:sequence>
            <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Actions"  minOccurs="0" maxOccurs="1" type="ActionsBlockType"/>
    </xs:sequence>
  </xs:complexType>


</xs:schema>
```

9.6 CMSImportResponse schema

This schema is used by the import service to pass back information about the user import.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="CMSImportResponse" targetNamespace="http://www.intercede.com/MyIDSchema/CMSImportResponse"
xmlns:mstns="http://www.intercede.com/MyIDSchema/CMSImportResponse" attributeFormDefault="unqualified"
elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="CMSImportResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Group"  minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Name"   type="mstns:String80"            minOccurs="1" maxOccurs="1"/>
              <xs:element name="Result" type="mstns:ActionStatus"  minOccurs="1" maxOccurs="1" />
              <xs:element name="User"   type="mstns:User"                minOccurs="0" maxOccurs="unbounded"  />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="User" type="mstns:User" minOccurs="0" maxOccurs="unbounded"  />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

      <xs:complexType name="User">
            <xs:sequence>
                  <xs:element name="FirstName"     type="mstns:String64" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="LastName"      type="mstns:String64" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="EmployeeID"    type="mstns:String50" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="LogonName"     type="mstns:String50" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="CardRequest"   type="xs:int" minOccurs="1" maxOccurs="1" />
                  <xs:element name="UnlockCardRequest"    type="xs:int" minOccurs="1" maxOccurs="1" />
                  <xs:element name="Result"        type="mstns:ActionStatus"  minOccurs="1" maxOccurs="1"/>
                  <xs:element name="Reason"        type="xs:string"  minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
      </xs:complexType>


  <xs:simpleType name="String50">
    <xs:restriction base="xs:string">
      <xs:maxLength value="50" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="String80">
    <xs:restriction base="xs:string">
      <xs:maxLength value="80" />
    </xs:restriction>
```

```
  </xs:simpleType>
  <xs:simpleType name="String64">
    <xs:restriction base="xs:string">
      <xs:maxLength value="64" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ActionStatus">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Added"/>
      <xs:enumeration value="Created"/>
      <xs:enumeration value="Already Exists"/>
      <xs:enumeration value="Failed"/>
         <xs:enumeration value="Removed"/>
         <xs:enumeration value="License Limit"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

9.7 PIVSchemaTypes

```
<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:myidbase="http://www.intercede.com/CommonTypes"
    targetNamespace="http://www.intercede.com/MyIDPIVSchema/PivCardRequest"
    xmlns="http://www.intercede.com/MyIDPIVSchema/PivCardRequest"
    elementFormDefault="qualified" attributeFormDefault="qualified"
>
  <xs:import namespace="http://www.intercede.com/CommonTypes" schemaLocation="MyIDBaseTypes.xsd"/>

  <xs:complexType name="ParametersBlockType">
            <xs:all>
                  <xs:element name="SourceID" type="xs:string" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="IssueDate" type="xs:date" minOccurs="0" maxOccurs="1"/>
                  <xs:element name="GenerateUserDN" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
                  <xs:element name="ActionOnDuplicate" type="xs:string" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="RolesActionOnDuplicate" type="xs:string" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="DeleteMissingUsers" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="PushToLDAP" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="CreateUnknownGroups" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="AuditAll" type="myidbase:Int1" minOccurs="1" maxOccurs="1"/>
                  <xs:element name="DataType" type="xs:string" minOccurs="0" maxOccurs="1"/>
                  <xs:element name="CheckImportResponseSchema" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
                  <xs:element name="SynchronousImport" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
```

```
                    <xs:element name="AllowBioImport" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="DefaultUserRole" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="DisallowCertificateSuspension" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>

                    <xs:element name="CardRequestThrottling" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="ReplaceUnassignedCards" type="myidbase:Int1" minOccurs="0" maxOccurs="1"/>
                </xs:all>
            </xs:complexType>

  <xs:complexType name="AgencyAddressBlockType">
    <xs:sequence>
      <xs:element name="Address1" type="myidbase:String40" minOccurs="0"/>
      <xs:element name="Address2" type="myidbase:String40" minOccurs="0"/>
      <xs:element name="City" type="myidbase:String40" minOccurs="1"/>
      <xs:element name="State" type="myidbase:String40" minOccurs="0"/>
      <xs:element name="Zip" type="myidbase:String10" minOccurs="0"/>
      <xs:element name="PhoneNumber" type="myidbase:String30" minOccurs="0"/>
      <xs:element name="Country" type="myidbase:String40" minOccurs="0"/>
      <xs:element name="Email" type="myidbase:String120" minOccurs="0"/>
      <xs:element name="Contact" type="myidbase:String80" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Complex PIV elements -->
  <xs:complexType name="PersonalBlockType">
    <xs:sequence>
      <xs:element name="FirstName" type="myidbase:NotEmptyString64" minOccurs="1"/>
      <xs:element name="LastName" type="myidbase:NotEmptyString64" minOccurs="1"/>
      <xs:element name="NickName" type="myidbase:String80" minOccurs="0"/>
      <xs:element name="MiddleName" type="myidbase:String50" minOccurs="0"/>
      <xs:element name="Suffix" type="myidbase:String20" minOccurs="0"/>
      <xs:element name="Title" type="myidbase:String20" minOccurs="0"/>
      <xs:element name="Email" type="myidbase:String255" minOccurs="0"/>
      <xs:element name="Fax" type="myidbase:String50" minOccurs="0"/>
      <xs:element name="Cell" type="myidbase:String50" minOccurs="0"/>
      <xs:element name="Phone" type="myidbase:String50" minOccurs="0"/>
      <xs:element name="EmployeeID" type="myidbase:Int50" minOccurs="1"/>
      <xs:element name="Address" minOccurs="0" type="PersonalAddressBlockType" />
      <xs:element name="BirthDate" type="xs:date" minOccurs="0"/>
      <xs:element name="Nationality" type="myidbase:String4" minOccurs="0"/>
      <xs:element name="Citizenship" type="CitizenshipString" minOccurs="0"/>
        <xs:element name="CountryOfBirth" type="myidbase:String2" minOccurs="0"/>
      <xs:element name="PlaceOfBirth" type="myidbase:String2" minOccurs="0"/>
      <xs:element name="ExportRestrictions" type="ExportRestrictionsString" minOccurs="0"/>
```

```
  </xs:sequence>
</xs:complexType>

<xs:complexType name="CardBlockType">
  <xs:sequence>
    <xs:element name="CardProfile" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="CardExpiryDate" type="xs:date" minOccurs="0"/>
    <xs:element name="Renewal" type="RenewalString" minOccurs="0"/>
    <xs:element name="ImportCard" type="myidbase:TrueFalse" minOccurs="0"/>
    <xs:element name="CardRequestedBy" type="myidbase:String255" minOccurs="0"/>
    <xs:element name="JobLabel" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="Update" type="UpdateCardBlockType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Replacement" type="ReplacementCardBlockType" minOccurs="0"/>
    <xs:element name="Reprovision" type="xs:unsignedByte" minOccurs="0" maxOccurs="1"/>
    <xs:element name="CardLayout" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
    <xs:element name="SerialNumber" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
    <xs:element name="DeviceType" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
    <xs:element name="Container" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
    <xs:element name="Certificate" type="myidbase:StringBase64" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="UpdateCardBlockType">
  <xs:sequence>
    <xs:element name="OriginalSerialNumber" type="myidbase:String50" minOccurs="1" maxOccurs="1" />
    <xs:element name="OriginalDeviceType" type="myidbase:String50" minOccurs="1" maxOccurs="1" />
       <xs:element name="StatusMapping" type="xs:int" minOccurs="0" maxOccurs="1"/>
    <xs:element name="RevocationComment" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ReplacementCardBlockType">
  <xs:all>
    <xs:element name="OriginalSerialNumber" type="myidbase:String50" minOccurs="1" maxOccurs="1"/>
    <xs:element name="OriginalDeviceType" type="myidbase:String50" minOccurs="1" maxOccurs="1"/>
    <xs:element name="StatusMapping" type="xs:int" minOccurs="0" maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="PersonalAddressBlockType">
  <xs:sequence>
    <xs:element name="Address1" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="Address2" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="City" type="myidbase:String50" minOccurs="0"/>
```

```
    <xs:element name="StateZip" type="myidbase:String50" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="AuthenticationBlockType">
  <xs:sequence>
    <xs:element name="SecurityPhrase" minOccurs="0" maxOccurs="unbounded" >
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Prompt" type="myidbase:NotEmptyString255" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Answer" type="myidbase:EncryptedData" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="AccountBlockType">
  <xs:sequence>
    <xs:element name="DN" type="myidbase:String1024" minOccurs="0"/>
    <xs:element name="AlternateDN" type="myidbase:String1024" minOccurs="0"/>
    <xs:element name="CN" type="myidbase:String255" minOccurs="0"/>
    <xs:element name="OU" type="myidbase:String1024" minOccurs="0"/>
    <xs:element name="UPN" type="myidbase:String255" minOccurs="0"/>
    <xs:element name="SAMAccountName" type="myidbase:String20" minOccurs="0"/>
    <xs:element name="Domain" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
    <xs:element name="LogonName" type="myidbase:String255" minOccurs="0"/>
    <xs:element name="Roles" minOccurs="0" maxOccurs="1" type="RolesBlockType"/>
    <xs:element name="EntrustProfile" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="MaxRequestExpiryDate" type="xs:date" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="RolesBlockType">
  <xs:sequence>
    <xs:element name="Role" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Name" type="myidbase:String255" minOccurs="1"/>
          <xs:element name="Scope" type="RoleScope" minOccurs="0" maxOccurs="1"/>
          <xs:element name="LogonMechanism" type="RoleLogonMechanism" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
```

```
  </xs:sequence>
</xs:complexType>

<xs:complexType name="PositionBlockType">
  <xs:sequence>
    <xs:element name="EmployeeAssociation" type="Association" minOccurs="0"/>
    <xs:element name="EmployeeAffiliation" type="myidbase:String64" minOccurs="0"/>
    <xs:element name="Rank" type="Rank" minOccurs="0"/>
    <xs:element name="EmergencyRole" type="EmergencyRole" minOccurs="0"/>
    <xs:element name="Privilege" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="ApplicantPosition" type="myidbase:String30" minOccurs="0"/>
    <xs:element name="ExtraInfo" type="myidbase:String80" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="BiometryBlockType">
  <xs:sequence>
    <xs:element name="HeightInches" type="myidbase:String8" minOccurs="0"/>
    <xs:element name="WeightLbs" type="myidbase:String8" minOccurs="0"/>
    <xs:element name="Gender" type="Gender" minOccurs="0"/>
    <xs:element name="EyeColor" type="EyeColor" minOccurs="0"/>
    <xs:element name="HairColor" type="HairColor" minOccurs="0"/>
    <xs:element name="Race" type="Race" minOccurs="0"/>
    <xs:element name="BioSample" minOccurs="0" type="BioSampleBlockType"/>
    <xs:element name="Signature" minOccurs="0">
      <xs:complexType>
        <xs:group ref="EncodedData"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="BioSampleBlockType">
  <xs:all>
    <xs:element name="R_Little" minOccurs="0">
      <xs:complexType>
        <xs:group ref="BioFingerSample"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="R_Ring" minOccurs="0">
      <xs:complexType>
        <xs:group ref="BioFingerSample"/>
      </xs:complexType>
    </xs:element>
```

```
<xs:element name="R_Middle" minOccurs="0">
  <xs:complexType>
    <xs:group ref="BioFingerSample"/>
  </xs:complexType>
</xs:element>
<xs:element name="R_Index" minOccurs="0">
  <xs:complexType>
    <xs:group ref="BioFingerSample"/>
  </xs:complexType>
</xs:element>
<xs:element name="R_Thumb" minOccurs="0">
  <xs:complexType>
    <xs:group ref="BioFingerSample"/>
  </xs:complexType>
</xs:element>
<xs:element name="L_Little" minOccurs="0">
  <xs:complexType>
    <xs:group ref="BioFingerSample"/>
  </xs:complexType>
</xs:element>
<xs:element name="L_Ring" minOccurs="0">
  <xs:complexType>
    <xs:group ref="BioFingerSample"/>
  </xs:complexType>
</xs:element>
<xs:element name="L_Middle" minOccurs="0">
  <xs:complexType>
    <xs:group ref="BioFingerSample"/>
  </xs:complexType>
</xs:element>
<xs:element name="L_Index" minOccurs="0">
  <xs:complexType>
    <xs:group ref="BioFingerSample"/>
  </xs:complexType>
</xs:element>
<xs:element name="L_Thumb" minOccurs="0">
  <xs:complexType>
    <xs:group ref="BioFingerSample"/>
  </xs:complexType>
</xs:element>
<xs:element name="EFT" minOccurs="0">
  <xs:complexType>
    <xs:group ref="EncodedData"/>
  </xs:complexType>
```

```
    </xs:element>
    <xs:element name="FaceScan" minOccurs="0">
      <xs:complexType>
        <xs:group ref="EncodedData"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="IrisScan" minOccurs="0">
      <xs:complexType>
        <xs:group ref="EncodedData"/>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>

<xs:group name="BioFingerSample">
  <xs:sequence>
    <xs:choice>
      <xs:sequence>
        <xs:group ref="EncodedData"/>
        <xs:element name="RolledPrint" type="myidbase:YesNo" maxOccurs="1" minOccurs="0"/>
      </xs:sequence>
      <xs:element name="ReasonMissing" type="MissingFingerReason" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
</xs:group>

<xs:group name="EncodedData">
  <xs:sequence>
    <xs:element name="Encoding" type="FileEncoding" maxOccurs="1" minOccurs="1"/>
    <xs:element name="Data" type="myidbase:StringBase64" maxOccurs="1" minOccurs="1"/>
    <xs:element name="DateTaken" type="xs:dateTime" maxOccurs="1" minOccurs="0"/>
    <xs:element name="Source" type="myidbase:String50" maxOccurs="1" minOccurs="0"/>
  </xs:sequence>
</xs:group>

<xs:complexType name="IdentityDocumentsBlockType">
  <xs:sequence>
    <xs:element name="IdentityDoc1" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Image">
            <xs:complexType>
              <xs:group ref="EncodedData"/>
            </xs:complexType>
```

```
          </xs:element>
          <xs:element name="Title" type="IDDocName1"/>
          <xs:element name="IssuedBy" type="myidbase:String80"/>
          <xs:element name="Number" type="myidbase:String20"/>
          <xs:element name="Expiry" type="xs:date"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="IdentityDoc2" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Image">
            <xs:complexType>
              <xs:group ref="EncodedData"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="Title" type="IDDocName2"/>
          <xs:element name="IssuedBy" type="myidbase:String80"/>
          <xs:element name="Number" type="myidbase:String20"/>
          <xs:element name="Expiry" type="xs:date"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="SPF10" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Image">
            <xs:complexType>
              <xs:group ref="EncodedData"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="Title"/>
          <xs:element name="IssuedBy"/>
          <xs:element name="Number"/>
          <xs:element name="Expiry"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="IDDocName1">
  <xs:restriction base="xs:string">
    <!-- the following items are allowable under FIPS201-2 -->
```

```
      <xs:enumeration value="U.S. Passport"/>
      <xs:enumeration value="Foreign Passport"/>
      <xs:enumeration value="Alien Regn Receipt Card"/>
      <xs:enumeration value="Employment Authorization"/>
      <xs:enumeration value="Drivers License"/>
      <xs:enumeration value="U.S. Military Card"/>
      <xs:enumeration value="U.S. Military dependent ID Card"/>
      <xs:enumeration value="PIV Card"/>
      <!-- the following items are legacy items which need to be retained to support existing functionality -->
      <xs:enumeration value="Cert. of U.S. Citizenship"/>
      <xs:enumeration value="Cert. of Naturalization"/>
      <xs:enumeration value="Temporary Card"/>
      <xs:enumeration value="Reentry Permit"/>
      <xs:enumeration value="Refugee Travel Document"/>
      <xs:enumeration value="Photo ID Card"/>
      <xs:enumeration value="School ID Card"/>
      <xs:enumeration value="Voters Regn Card"/>
      <xs:enumeration value="Merchant Mariner Card"/>
      <xs:enumeration value="Native American Tribal"/>
      <xs:enumeration value="Canadian Drivers License"/>
      <xs:enumeration value="School Report"/>
      <xs:enumeration value="Hospital Record"/>
      <xs:enumeration value="Nursery Record"/>
      <xs:enumeration value="U.S. Social Security"/>
      <xs:enumeration value="Cert. of Birth Abroad"/>
      <xs:enumeration value="Birth Certificate"/>
      <xs:enumeration value="U.S. Citizen ID"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="IDDocName2">
    <xs:restriction base="xs:string">
      <!-- the following items are allowable under FIPS201-2 -->
      <xs:enumeration value="U.S. Passport"/>
      <xs:enumeration value="Cert. of U.S. Citizenship"/>
      <xs:enumeration value="Cert. of Naturalization"/>
      <xs:enumeration value="Foreign Passport"/>
      <xs:enumeration value="Alien Regn Receipt Card"/>
      <xs:enumeration value="Temporary Card"/>
      <xs:enumeration value="Reentry Permit"/>
      <xs:enumeration value="Refugee Travel Document"/>
      <xs:enumeration value="Employment Authorization"/>
      <xs:enumeration value="Drivers License"/>
      <xs:enumeration value="Photo ID Card"/>
```

```
      <xs:enumeration value="Voters Regn Card"/>
      <xs:enumeration value="U.S. Military Card"/>
      <xs:enumeration value="Merchant Mariner Card"/>
      <xs:enumeration value="Native American Tribal"/>
      <xs:enumeration value="Canadian Drivers License"/>
      <xs:enumeration value="U.S. Social Security"/>
      <xs:enumeration value="Cert. of Birth Abroad"/>
      <xs:enumeration value="Birth Certificate"/>
      <xs:enumeration value="U.S. Citizen ID"/>
      <xs:enumeration value="U.S. Resident Citizen Card (I-179)"/>
      <xs:enumeration value="Employment Auth Card (I-688A)"/>
      <xs:enumeration value="Employment Auth Doc (DHS)"/>
      <xs:enumeration value="Employment Auth Doc (I-688B)"/>
      <!-- the following items are legacy items which need to be retained to support existing functionality -->
      <xs:enumeration value="School ID Card"/>
      <xs:enumeration value="School Report"/>
      <xs:enumeration value="Hospital Record"/>
      <xs:enumeration value="Nursery Record"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="NACIStatusBlockType">
    <xs:sequence>
      <xs:element name="NACCaseNumber" type="xs:unsignedShort" minOccurs="0"/>
      <xs:element name="NACICaseNumber" type="xs:unsignedShort" minOccurs="0"/>
      <xs:element maxOccurs="unbounded" name="Check">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Type" type="myidbase:String20"/>
            <xs:element name="Status" type="NACState"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="CardIssuanceApproved" type="myidbase:YesNoOneZero" minOccurs="0"/>
      <xs:element name="Comments" type="myidbase:String255" minOccurs="0"/>
      <xs:element name="VettingDate" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ActionsBlockType">
    <xs:sequence>
      <xs:element name="ApplicantAction" type="UserAction" minOccurs="0" maxOccurs="1"/>
      <xs:element name="CertSerialNumber" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
      <xs:element name="CertPolicyName" type="myidbase:String50" minOccurs="0" maxOccurs="1"/>
```

```
    <xs:element name="StatusMappingID" type="xs:unsignedInt" minOccurs="0" maxOccurs="1"/>
    <xs:element name="RevocationComment" type="myidbase:String255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Device" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="DeviceIdentifier" minOccurs="0" maxOccurs="unbounded" type="DeviceIdentifierBlockType"/>
                  <xs:element name="ProcessStatus" type="myidbase:String50" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Job" type="xs:unsignedInt" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="DeviceIdentifierBlockType">
  <xs:sequence>
    <xs:element name="SerialNumber" type="myidbase:String50" minOccurs="1"/>
    <xs:element name="SerialNumberField" type="myidbase:String50" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- simple/base PIV types -->
<xs:simpleType name="RenewalString">
  <xs:restriction base="xs:string">
    <xs:enumeration value="true"/>
    <xs:enumeration value="false"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="CitizenshipString">
  <xs:restriction base="xs:string">
    <xs:enumeration value="US"/>
    <xs:enumeration value="Non US"/>
    <xs:enumeration value="PR"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ExportRestrictionsString">
  <xs:restriction base="xs:string">
    <xs:enumeration value="EXP"/>
    <xs:enumeration value=""/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="RoleScope">
  <xs:restriction base="xs:string">
    <xs:enumeration value="None"/>
    <xs:enumeration value="Self"/>
    <xs:enumeration value="Department"/>
    <xs:enumeration value="Division"/>
    <xs:enumeration value="All"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="RoleLogonMechanism">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Password"/>
    <xs:enumeration value="Card"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Association">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Employee"/>
    <xs:enumeration value="Civil Worker"/>
    <xs:enumeration value="Executive"/>
    <xs:enumeration value="Uniformed"/>
    <xs:enumeration value="Contractor"/>
    <xs:enumeration value="Afficiate"/>
    <xs:enumeration value="Beneficiary"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Rank">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Civilian"/>
    <xs:enumeration value="E-1"/>
    <xs:enumeration value="E-2"/>
    <xs:enumeration value="E-3"/>
    <xs:enumeration value="E-4"/>
    <xs:enumeration value="E-5"/>
    <xs:enumeration value="E-6"/>
    <xs:enumeration value="E-7"/>
    <xs:enumeration value="E-8"/>
    <xs:enumeration value="E-9"/>
    <xs:enumeration value="ES-1"/>
    <xs:enumeration value="ES-2"/>
    <xs:enumeration value="ES-3"/>
```

```
      <xs:enumeration value="ES-4"/>
      <xs:enumeration value="ES-5"/>
      <xs:enumeration value="ES-6"/>
      <xs:enumeration value="GS-1"/>
      <xs:enumeration value="GS-10"/>
      <xs:enumeration value="GS-11"/>
      <xs:enumeration value="GS-12"/>
      <xs:enumeration value="GS-13"/>
      <xs:enumeration value="GS-14"/>
      <xs:enumeration value="GS-15"/>
      <xs:enumeration value="GS-1"/>
      <xs:enumeration value="GS-2"/>
      <xs:enumeration value="GS-3"/>
      <xs:enumeration value="GS-4"/>
      <xs:enumeration value="GS-5"/>
      <xs:enumeration value="GS-6"/>
      <xs:enumeration value="GS-7"/>
      <xs:enumeration value="GS-8"/>
      <xs:enumeration value="GS-9"/>
      <xs:enumeration value="O-10"/>
      <xs:enumeration value="O-11"/>
      <xs:enumeration value="O-1"/>
      <xs:enumeration value="O-2"/>
      <xs:enumeration value="O-3"/>
      <xs:enumeration value="O-4"/>
      <xs:enumeration value="O-5"/>
      <xs:enumeration value="O-6"/>
      <xs:enumeration value="O-7"/>
      <xs:enumeration value="O-8"/>
      <xs:enumeration value="O-9"/>
      <xs:enumeration value="W-1"/>
      <xs:enumeration value="W-2"/>
      <xs:enumeration value="W-3"/>
      <xs:enumeration value="W-4"/>
      <xs:enumeration value="W-5"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="EmergencyRole">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Law Enforcement"/>
      <xs:enumeration value="Paramedic"/>
      <xs:enumeration value="Doctor"/>
      <xs:enumeration value="Firefighter"/>
```

```
      <xs:enumeration value="Bomb Disposal"/>
      <xs:enumeration value="Biohazard"/>
      <xs:enumeration value="None"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Gender">
    <xs:restriction base="xs:string">
      <xs:enumeration value="F"/>
      <xs:enumeration value="M"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="EyeColor">
    <xs:restriction base="xs:string">
      <xs:enumeration value="BLK"/>
      <xs:enumeration value="BLU"/>
      <xs:enumeration value="BRO"/>
      <xs:enumeration value="GRY"/>
      <xs:enumeration value="GRN"/>
      <xs:enumeration value="HAZ"/>
      <xs:enumeration value="MAR"/>
      <xs:enumeration value="PNK"/>
      <xs:enumeration value="XXX"/>
      <xs:enumeration value="MUL"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="HairColor">
    <xs:restriction base="xs:string">
      <xs:enumeration value="XXX"/>
      <xs:enumeration value="BAL"/>
      <xs:enumeration value="BLK"/>
      <xs:enumeration value="BLN"/>
      <xs:enumeration value="BLU"/>
      <xs:enumeration value="BRO"/>
      <xs:enumeration value="GRY"/>
      <xs:enumeration value="GRN"/>
      <xs:enumeration value="ONG"/>
      <xs:enumeration value="PNK"/>
      <xs:enumeration value="PLE"/>
      <xs:enumeration value="RED"/>
      <xs:enumeration value="SDY"/>
      <xs:enumeration value="STR"/>
```

```
      <xs:enumeration value="WHI"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Race">
    <xs:restriction base="xs:string">
      <xs:enumeration value="A"/>
      <xs:enumeration value="B"/>
      <xs:enumeration value="I"/>
      <xs:enumeration value="W"/>
      <xs:enumeration value="O"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="MissingFingerReason">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Unprintable"/>
      <xs:enumeration value="Amputated"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="NACState">
    <xs:restriction base="xs:string">
      <xs:enumeration value="NR"/>
      <xs:enumeration value="WR"/>
      <xs:enumeration value="AS"/>
      <xs:enumeration value="AP"/>
      <xs:enumeration value="RJ"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="UserAction">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Disable"/>
      <xs:enumeration value="Remove"/>
      <xs:enumeration value="CancelDevices"/>
      <xs:enumeration value="CancelDevice"/>
      <xs:enumeration value="CancelJob"/>
      <xs:enumeration value="CancelAllJobs"/>
      <xs:enumeration value="RenewCertificate"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="FileEncoding">
```

```
    <xs:restriction base="xs:string">
      <xs:enumeration value="jpg"/>
      <xs:enumeration value="gif"/>
      <xs:enumeration value="png"/>
      <xs:enumeration value="bmp"/>
      <xs:enumeration value="378"/>
      <xs:enumeration value="385"/>
      <xs:enumeration value="19794"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="ApplicantStatus">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Sponsored"/>
      <xs:enumeration value="EnrollmentBegun"/>
      <xs:enumeration value="EnrollmentComplete"/>
      <xs:enumeration value="Vetting"/>
      <xs:enumeration value="Escalated"/>
      <xs:enumeration value="Approved"/>
      <xs:enumeration value="CardRequested"/>
      <xs:enumeration value="CardActivated"/>
      <xs:enumeration value="Rejected"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="AgencyCategory">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Federal"/>
      <xs:enumeration value="State"/>
      <xs:enumeration value="Commercial"/>
      <xs:enumeration value="Foreign"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="AgencyType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Agency"/>
      <xs:enumeration value="State"/>
      <xs:enumeration value="Company"/>
      <xs:enumeration value="Country"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="AdminGroupsBlockType">
```

```
    <xs:sequence>
      <xs:element name="AdminGroup" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Name" minOccurs="1" type="myidbase:String100" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

9.8 PivCardRequest

```
<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:myidbase="http://www.intercede.com/CommonTypes"
    targetNamespace="http://www.intercede.com/MyIDPIVSchema/PivCardRequest"
    xmlns="http://www.intercede.com/MyIDPIVSchema/PivCardRequest"
    id="PivCardRequest"
    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
    elementFormDefault="qualified" attributeFormDefault="qualified"
 >
  <xs:include schemaLocation="PIVSchemaTypes.xsd" />
  <xs:import namespace="http://www.intercede.com/CommonTypes" schemaLocation="MyIDBaseTypes.xsd"/>

  <xs:element name="PivCardRequest" msdata:IsDataSet="true" msdata:Locale="en-GB" msdata:EnforceConstraints="false">
              <xs:complexType>
                    <xs:sequence>
        <xs:element name="Parameters" minOccurs="0" maxOccurs="1" type="ParametersBlockType"/>
                          <xs:element name="Agency" minOccurs="0" maxOccurs="1" type="AgencyBlockType"/>
                    </xs:sequence>
              </xs:complexType>
       </xs:element>

  <xs:complexType name="AgencyBlockType">
    <xs:sequence>
      <xs:element name="DeptCode" type="myidbase:String6" minOccurs="1"/>
      <xs:element name="AgencyCode" type="myidbase:String4" minOccurs="1"/>
      <xs:element name="FacilityCode" type="myidbase:String4" minOccurs="1"/>
      <xs:element name="AgencyName" type="myidbase:String100" minOccurs="1"/>
      <xs:element name="FacilityName" type="myidbase:String80" minOccurs="0"/>
```

```
    <xs:element name="DUNS" type="myidbase:Int9" minOccurs="1"/>
    <xs:element name="OC" type="myidbase:Int1" minOccurs="1"/>
    <xs:element name="OI" type="myidbase:Int4" minOccurs="1"/>
    <xs:element name="Department" type="myidbase:String64" minOccurs="0"/>
    <xs:element name="BaseDN" type="myidbase:String1024" minOccurs="0"/>
    <xs:element name="Parent" type="myidbase:String80" minOccurs="0"/>
    <xs:element name="AgencyAddress" minOccurs="0" maxOccurs="1" type="AgencyAddressBlockType"/>
    <xs:element name="AdditionalAgencyFields" minOccurs="0" maxOccurs="1">
      <xs:complexType mixed="true">
        <xs:sequence>
          <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Applicant" minOccurs="0" maxOccurs="1" type="ApplicantBlockType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ApplicantBlockType">
  <xs:sequence>
    <xs:element name="Personal" type="PersonalBlockType"/>
    <xs:element name="Authentication" type="AuthenticationBlockType" minOccurs="0" maxOccurs="1" />
    <xs:element name="Card" minOccurs="0" maxOccurs="1" type="CardBlockType"/>
    <xs:element name="Account" minOccurs="0" type="AccountBlockType"/>
    <xs:element name="Position" minOccurs="0" type="PositionBlockType"/>
    <xs:element name="Sponsor" minOccurs="1" maxOccurs="1" type="SponsorBlockType"/>
    <xs:element name="Biometry" minOccurs="0" type="BiometryBlockType"/>
    <xs:element name="IdentityDocs" minOccurs="0" type="IdentityDocumentsBlockType"/>
    <xs:element name="NACI" minOccurs="0" type="NACIStatusBlockType"/>
    <xs:element name="Photo" minOccurs="0">
      <xs:complexType>
        <xs:group ref="EncodedData"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="System" minOccurs="0" maxOccurs="1" type="SystemBlockType"/>
    <xs:element name="AdminGroups" minOccurs="0" maxOccurs="1" type="AdminGroupsBlockType" />
    <xs:element name="AdditionalFields" minOccurs="0" maxOccurs="1">
      <xs:complexType mixed="true">
        <xs:sequence>
          <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Actions" minOccurs="0" maxOccurs="1" type="ActionsBlockType"/>
```

```
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="SponsorBlockType">
    <xs:sequence>
      <xs:element name="SponsorName" type="myidbase:String80" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SponsorPosition" type="myidbase:String30" minOccurs="0" maxOccurs="1"/>
      <xs:element name="SponsorAgency" type="myidbase:String80" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SponsorEmail" type="myidbase:String120" minOccurs="0" maxOccurs="1"/>
      <xs:element name="SponsorPhoneNumber" type="myidbase:String30" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="SystemBlockType">
    <xs:sequence>
      <xs:element name="GUID" type="myidbase:String50" minOccurs="1"/>
      <xs:element name="CredNo" type="myidbase:String8" minOccurs="1"/>
      <xs:element name="CS" type="xs:unsignedInt" minOccurs="0"/>
      <xs:element name="POA" type="xs:unsignedInt" minOccurs="0"/>
      <xs:element name="ICI" type="xs:unsignedInt" minOccurs="1"/>
      <xs:element name="UserStatus" type="myidbase:String50" minOccurs="0"/>
      <xs:element name="IssuanceNotifyURL" type="myidbase:String255" minOccurs="0"/>
      <xs:element name="CancelNotifyURL" type="myidbase:String255" minOccurs="0"/>
      <xs:element name="CMSIssuanceNotifyURL" type="myidbase:String255" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

9.9 PivApplicantUpdate

```
<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:myidbase="http://www.intercede.com/CommonTypes"
    targetNamespace="http://www.intercede.com/MyIDPIVSchema/PivCardRequest"
    xmlns="http://www.intercede.com/MyIDPIVSchema/PivCardRequest"
    id="PivApplicantUpdate"
    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
    elementFormDefault="qualified" attributeFormDefault="qualified"
>
  <xs:include schemaLocation="PIVSchemaTypes.xsd" />
  <xs:import namespace="http://www.intercede.com/CommonTypes" schemaLocation="MyIDBaseTypes.xsd"/>
```

```
<xs:element name="PivCardRequest" msdata:IsDataSet="true" msdata:Locale="en-GB" msdata:EnforceConstraints="false">
            <xs:complexType>
                  <xs:sequence>
      <xs:element name="Parameters" minOccurs="0" maxOccurs="1" type="ParametersBlockType"/>
      <xs:element name="Agency" minOccurs="0" maxOccurs="unbounded" type="AgencyBlockType"/>
                        <xs:element name="Applicant" minOccurs="0" maxOccurs="unbounded" type="ApplicantBlockType"/>
                  </xs:sequence>
            </xs:complexType>
     </xs:element>

<xs:complexType name="AgencyBlockType">
  <xs:sequence>
    <xs:element name="DeptCode" type="myidbase:String6" minOccurs="1"/>
    <xs:element name="AgencyCode" type="myidbase:String4" minOccurs="1"/>
    <xs:element name="FacilityCode" type="myidbase:String4" minOccurs="1"/>
    <xs:element name="AgencyName" type="myidbase:String100" minOccurs="1"/>
    <xs:element name="FacilityName" type="myidbase:String80" minOccurs="0"/>
    <xs:element name="DUNS" type="myidbase:Int9" minOccurs="1"/>
    <xs:element name="OC" type="myidbase:Int1" minOccurs="1"/>
    <xs:element name="OI" type="myidbase:Int4" minOccurs="1"/>
    <xs:element name="Department" type="myidbase:String64" minOccurs="0"/>
    <xs:element name="BaseDN" type="myidbase:String1024" minOccurs="0"/>
    <xs:element name="Parent" type="myidbase:String80" minOccurs="0"/>
    <xs:element name="AgencyAddress" minOccurs="0" maxOccurs="1" type="AgencyAddressBlockType"/>
    <xs:element name="AdditionalAgencyFields" minOccurs="0" maxOccurs="1">
      <xs:complexType mixed="true">
        <xs:sequence>
          <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Applicant" minOccurs="0" maxOccurs="unbounded" type="ApplicantBlockType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ApplicantBlockType">
  <xs:sequence>
    <xs:element name="Personal" minOccurs="0" type="PersonalBlockType"/>
    <xs:element name="Authentication" minOccurs="0" maxOccurs="1" type="AuthenticationBlockType"/>
    <xs:element name="Card" minOccurs="0" maxOccurs="1" type="CardBlockType"/>
    <xs:element name="Account" minOccurs="0" type="AccountBlockType"/>
    <xs:element name="Position" minOccurs="0" type="PositionBlockType"/>
    <xs:element name="Sponsor" minOccurs="0" maxOccurs="1" type="SponsorBlockType"/>
```

```
    <xs:element name="Biometry" minOccurs="0" type="BiometryBlockType"/>
    <xs:element name="IdentityDocs" minOccurs="0" type="IdentityDocumentsBlockType"/>
    <xs:element name="NACI" minOccurs="0" type="NACIStatusBlockType"/>
    <xs:element name="Photo" minOccurs="0">
      <xs:complexType>
        <xs:group ref="EncodedData"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="System" minOccurs="0" type="SystemBlockType"/>
    <xs:element name="AdminGroups" minOccurs="0" maxOccurs="1" type="AdminGroupsBlockType" />
    <xs:element name="AdditionalFields" minOccurs="0" maxOccurs="1">
      <xs:complexType mixed="true">
        <xs:sequence>
          <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Actions" minOccurs="0" maxOccurs="1" type="ActionsBlockType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SponsorBlockType">
  <xs:sequence>
    <xs:element name="SponsorName" type="myidbase:String80" minOccurs="0" maxOccurs="1"/>
    <xs:element name="SponsorPosition" type="myidbase:String30" minOccurs="0" maxOccurs="1"/>
    <xs:element name="SponsorAgency" type="myidbase:String80" minOccurs="0" maxOccurs="1"/>
    <xs:element name="SponsorEmail" type="myidbase:String120" minOccurs="0" maxOccurs="1"/>
    <xs:element name="SponsorPhoneNumber" type="myidbase:String30" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SystemBlockType">
  <xs:sequence>
    <xs:element name="GUID" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="CredNo" type="myidbase:String8" minOccurs="0"/>
    <xs:element name="CS" type="xs:unsignedInt" minOccurs="0"/>
    <xs:element name="POA" type="xs:unsignedInt" minOccurs="0"/>
    <xs:element name="ICI" type="xs:unsignedInt" minOccurs="0"/>
    <xs:element name="UserStatus" type="myidbase:String50" minOccurs="0"/>
    <xs:element name="IssuanceNotifyURL" type="myidbase:String255" minOccurs="0"/>
    <xs:element name="CancelNotifyURL" type="myidbase:String255" minOccurs="0"/>
    <xs:element name="CMSIssuanceNotifyURL" type="myidbase:String255" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

```
</xs:schema>
```

9.10 PivImportResponse

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="PIVImportResponse" targetNamespace="http://www.intercede.com/MyIDPIVSchema/PIVImportResponse"
xmlns:mstns="http://www.intercede.com/MyIDPIVSchema/PIVImportResponse" attributeFormDefault="unqualified"
elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="PivImportResponse">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="Agency" minOccurs="0" maxOccurs="unbounded">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="AgencyName" type="mstns:String80" minOccurs="1" maxOccurs="1"/>
							<xs:element name="AgencyCode" type="mstns:String4" minOccurs="1" maxOccurs="1" />
							<xs:element name="Result" type="mstns:ActionStatus" minOccurs="1" maxOccurs="1" />
							<xs:element name="Applicant" minOccurs="0" maxOccurs="unbounded"
type="mstns:ApplicantType"/>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
				<xs:element name="Applicant" minOccurs="0" maxOccurs="unbounded" type="mstns:ApplicantType"/>
				<xs:element name="error" minOccurs="0" maxOccurs="1">
					<xs:complexType mixed="true">
						<xs:sequence>
							<xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:complexType name="ApplicantType">
		<xs:sequence>
			<xs:element name="FirstName" type="mstns:String64" minOccurs="1" maxOccurs="1"/>
			<xs:element name="LastName" type="mstns:String64" minOccurs="1" maxOccurs="1"/>
			<xs:element name="EmployeeID" type="mstns:Int50" minOccurs="1" maxOccurs="1"/>
			<xs:element name="CardRequest" type="xs:int" minOccurs="1" maxOccurs="1" />
	<xs:element name="CardReqestFailReason" type="mstns:String64"  minOccurs="0" maxOccurs="1"/>
	<xs:element name="Result" type="mstns:ActionStatus"  minOccurs="1" maxOccurs="1"/>
	  <xs:element name="Reason"            type="xs:string"  minOccurs="0" maxOccurs="1"/>
		</xs:sequence>
```

```
	</xs:complexType>
	<xs:simpleType name="String50">
		<xs:restriction base="xs:string">
			<xs:maxLength value="50" />
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="String80">
		<xs:restriction base="xs:string">
			<xs:maxLength value="80" />
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="String64">
		<xs:restriction base="xs:string">
			<xs:maxLength value="64" />
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="String4">
		<xs:restriction base="xs:string">
			<xs:maxLength value="4" />
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="Int50">
		<xs:restriction base="xs:string">
			<xs:maxLength value="50" />
			<xs:pattern value="[0-9]*" />
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="ActionStatus">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Added"/>
			<xs:enumeration value="Created"/>
			<xs:enumeration value="Already Exists"/>
			<xs:enumeration value="Failed"/>
			<xs:enumeration value="Removed"/>
			<xs:enumeration value="License Limit"/>
		</xs:restriction>
	</xs:simpleType>
</xs:schema>
```