



MyID Enterprise

Version 12.14

REST Web Service Notifications

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

Copyright

© 2001-2025 Intercede Limited. All rights reserved.

Information in this document is subject to change without notice. The software described in this document is furnished exclusively under a restricted license or non-disclosure agreement. Copies of software supplied by Intercede Limited may not be used resold or disclosed to third parties or used for any commercial purpose without written authorization from Intercede Limited and will perpetually remain the property of Intercede Limited. They may not be transferred to any computer without both a service contract for the use of the software on that computer being in existence and written authorization from Intercede Limited.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Intercede Limited.

Whilst Intercede Limited has made every effort in the preparation of this manual to ensure the accuracy of the information, the information contained in this manual is delivered without warranty, either express or implied. Intercede Limited will not be held liable for any damages caused, or alleged to be caused, either directly or indirectly by this manual.

Trademarks

The Intercede[®] and MyID[®] word marks and the MyID[®] logo are registered trademarks of Intercede in the UK, US and other countries.

Microsoft and Windows are registered trademarks of Microsoft Corporation. Other brands and their products are trademarks or registered trademarks of their respective holders and should be noted as such. All other trademarks acknowledged.

Licenses

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

Conventions used in this document

- Lists:
 - Numbered lists are used to show the steps involved in completing a task when the order is important.
 - Bulleted lists are used when the order is unimportant or to show alternatives.
- **Bold** is used for menu items and for labels.
For example:
 - Record a valid email address in '**From**' email address.
 - Select **Save** from the **File** menu.
- *Italic* is used for emphasis:
For example:
 - Copy the file *before* starting the installation.
 - Do *not* remove the files before you have backed them up.
- ***Bold and italic*** hyperlinks are used to identify the titles of other documents.
For example: "See the ***Release Notes*** for further information."
Unless otherwise explicitly stated, all referenced documentation is available on the product installation media.
- A `fixed width` font is used where the identification of spaces is important, including filenames, example SQL queries and any entries made directly into configuration files or the database.
- **Notes** are used to provide further information, including any prerequisites or configuration additional to the standard specifications.
For example:
Note: This issue only occurs if updating from a previous version.
- **Warnings** are used to indicate where failure to follow a particular instruction may result in either loss of data or the need to manually configure elements of the system.
For example:
Warning: You must take a backup of your database before making any changes to it.

Contents

REST Web Service Notifications	1
Copyright	2
Conventions used in this document	3
Contents	4
1 Introduction	5
2 Configuring REST web service notifications	6
2.1 Standard REST notifications	7
2.1.1 DisableCard notification	10
2.1.2 EnableCard notification	11
2.1.3 REST Certificate Added	12
2.1.4 REST Device Cancelled notification	17
2.1.5 REST Device Issued notification	18
2.1.6 REST Device Reassigned notification	19
2.1.7 REST Person Added notification	20
2.1.8 REST Person Deleted notification	21
2.1.9 REST Person Disabled notification	22
2.1.10 REST Person Edited notification	23
2.1.11 REST Person Enabled notification	24
2.1.12 REST Request Added notification	25
2.1.13 REST Request Updated notification	26
2.2 Creating a mapping file	28
2.2.1 Notification	28
2.2.2 DataSources	28
2.2.3 Endpoint	31
2.2.4 Body	32
2.3 Configuring an external system for REST notifications	35
2.3.1 Enabling and disabling a notification	38
2.3.2 REST notification name mapping	38
3 Troubleshooting REST notifications	39

1 Introduction

The REST web service notifications system allows you to send notifications from MyID to REST web services for lifecycle events. For example, you can update a physical access control system (PACS) to enable or disable access for a particular card when it is issued or managed in MyID, or inform connected systems such as a workflow automation platform, endpoint management solution, or stock control system when credentials have been issued to a person.

MyID provides standard notifications for:

- Issuing, canceling, enabling, and disabling devices.
- Adding, editing, enabling, disabling, and deleting people.
- Adding and updating Issue Card requests.

To implement each of these notifications, you must set up an external system within MyID that provides details of your REST web service, including authentication details, and a mapping file that describes the format of the payload that you want to send to the web service endpoint. See section [2.3, *Configuring an external system for REST notifications*](#).

MyID provides sample mapping files for each of the standard notifications. You can write your REST web service to consume the payload provided by these notifications, or use the provided mapping files as a basis for your own requirements. See section [2.1, *Standard REST notifications*](#) for details of the data provided by the standard notifications.

For information on creating your own mapping files, see section [2.2, *Creating a mapping file*](#).

You can manage REST web service notifications using the **Notifications Management** workflow; see the *Using the Notifications Management workflow* section in the [Administration Guide](#) for details. From the **Notification Type** drop-down list, select **Rest Service** to search for notifications sent to REST web services.

2 Configuring REST web service notifications

To configure REST web service notifications, you must choose a mapping file.

You can use either:

- A standard REST web service notifications mapping file.

These mapping files provide a sample endpoint and a definition for the payload to be passed; you can create this endpoint on your own system to consume the notification.

See section [2.1, Standard REST notifications](#).

or

- A new mapping file.

If you have an existing system that you want to receive the notification, you can create a mapping file that specifies the endpoint on your system to use, and a definition for the data payload that the endpoint is expecting.

See section [2.2, Creating a mapping file](#).

Once you have chosen or created your mapping file, you can configure MyID with an external system for REST service notifications; see section [2.3, Configuring an external system for REST notifications](#).

2.1 Standard REST notifications

By default, MyID provides the following standard REST notifications with corresponding mapping files:

Notification	Mapping file	Description
DisableCard	DisableCard.xml	Sends a notification to a REST web service when a device is disabled in MyID. See section 2.1.1, DisableCard notification .
EnableCard	EnableCard.xml	Sends a notification to a REST web service when a device is enabled in MyID. See section 2.1.2, EnableCard notification .
REST Certificate Added	RESTCertificateAddedMyID.xml <i>or:</i> RESTCertificateAdded.xml	Sends a notification to a REST web service (or, specifically, another MyID CMS server) when a certificate is added. See section 2.1.3, REST Certificate Added .
REST Device Cancelled	RESTDeviceCancelled.xml	Sends a notification to a REST web service when a device is canceled and therefore removed from association with any people. See section 2.1.4, REST Device Cancelled notification .
REST Device Issued	RESTDeviceIssued.xml	Sends a notification to a REST web service when a device is issued and has a status of Active. See section 2.1.5, REST Device Issued notification .

Notification	Mapping file	Description
REST Device Reassigned	RESTDeviceReassigned.xml	Sends a notification to a REST web service when a device is reassigned. See section 2.1.6, REST Device Reassigned notification .
REST Person Added	RESTPersonAdded.xml	Sends a notification to a REST web service when a person is added to MyID. See section 2.1.7, REST Person Added notification .
REST Person Deleted	RESTPersonDeleted.xml	Sends a notification to a REST web service when a person is deleted from MyID. See section 2.1.8, REST Person Deleted notification .
REST Person Disabled	RESTPersonDisabled.xml	Sends a notification to a REST web service when a person is disabled in MyID. See section 2.1.9, REST Person Disabled notification .
REST Person Edited	RESTPersonEdited.xml	Sends a notification to a REST web service when a person is edited in MyID. See section 2.1.10, REST Person Edited notification .

Notification	Mapping file	Description
REST Person Enabled	RESTPersonEnabled.xml	Sends a notification to a REST web service when a person is enabled in MyID. See section 2.1.11 , <i>REST Person Enabled notification</i> .
REST Request Added	RESTRequestAdded.xml	Sends a notification to a REST web service when an Issue Card request is added in MyID. See section 2.1.12 , <i>REST Request Added notification</i> .
REST Request Updated	RESTRequestUpdated.xml	Sends a notification to a REST web service when an Issue Card request is updated in MyID. See section 2.1.13 , <i>REST Request Updated notification</i> .

You can create an external system for each of these REST notifications. See section [2.3](#), *Configuring an external system for REST notifications*.

2.1.1 DisableCard notification

The `DisableCard` event sends a notification to a REST web service when a device is disabled in MyID.

If you configure an external system with the name `DisableCard` and specify the corresponding mapping file `DisableCard.xml`, you can configure MyID to respond to the event with the following.

2.1.1.1 Endpoint

```
POST /devices/{device GUID}/deviceDisabled
```

2.1.1.2 Data

```
{
  "device": {
    "id": "<device guid>",
    "sn": "<device serial number>",
    "dt": "<MyID device type>",
    "validity": {
      "from": "<ISO format issuance date>",
      "to": "<ISO format expiration date>",
      "enabled": <true|false>
    },
    "hid": {
      "serialNumber": "<HID loop ID>",
      "facilityCode": "<HID facility code>"
    }
  },
  "person": {
    "logonName": "<logon name>"
  }
}
```

2.1.1.3 Expected response

A 200 OK response in the event of success.

2.1.2 EnableCard notification

The `EnableCard` event sends a notification to a REST web service when a device is enabled in MyID.

If you configure an external system with the name `EnableCard` and specify the corresponding mapping file `EnableCard.xml`, you can configure MyID to respond to the event with the following.

2.1.2.1 Endpoint

POST `/devices/{device GUID}/deviceEnabled`

2.1.2.2 Data

```
{
  "device": {
    "id": "<device guid>",
    "sn": "<device serial number>",
    "dt": "<MyID device type>",
    "validity": {
      "from": "<ISO format issuance date>",
      "to": "<ISO format expiration date>",
      "enabled": <true|false>
    },
    "hid": {
      "serialNumber": "<HID loop ID>",
      "facilityCode": "<HID facility code>"
    }
  },
  "person": {
    "logonName": "<logon name>"
  }
}
```

2.1.2.3 Expected response

A 200 OK response in the event of success.

2.1.3 REST Certificate Added

The `REST Certificate Added` event sends a notification to a REST web service when a certificate is added to the system. You must configure the **Certificate Authorities** workflow for each certificate policy for which you want to trigger the notification; see section [2.1.3.4, *Setting up your certificate policies*](#).

The `REST Certificate Added` notification can use the following mapping files:

- `RESTCertificateAddedMyID.xml` – specifically designed to send a notification to another MyID CMS system. Provides information on the certificate so that the target MyID CMS system can import it.
See section [2.1.3.5, *Setting up an external MyID CMS system to receive notifications*](#).
- `RESTCertificateAdded.xml` – a generic notification that provides as much information about the certificate as possible so that you can extract the information you need on your own system.

You can configure multiple notifications of this type, which allows you to send notifications from different certificate policies to different external systems.

2.1.3.1 Endpoint

The endpoint is determined by the mapping file you use.

- For the `RESTCertificateAddedMyID.xml` mapping file:

```
POST /api/Certificates/import
```

This is a MyID Core API endpoint. For more information about this endpoint, see the *Importing certificates* section in the [MyID Core API](#) guide. Note, however, that while the endpoint can handle importing either X.509 files (containing certificates) and PKCS#12 files (containing certificates and private keys), in this case the notifications contain only X.509 certificate data, and do not contain private keys.

- For the `RESTCertificateAdded.xml` mapping file:

```
POST /certificates/{Certificate.ObjectID}/added
```

This is a generic endpoint that you can create on your own system to consume the notification; alternatively, you can copy this mapping file and edit it to create a mapping file that uses an endpoint of your own choosing. See section [2.2, *Creating a mapping file*](#).

2.1.3.2 Data

The data is determined by the mapping file you use.

When you use the `RESTCertificateAddedMyID.xml` mapping file, the data is tailored to providing information to another MyID CMS system to allow it to handle the certificate. If you set up your certificate policy in the **Certificate Authorities** workflow to specify the object ID of a certificate policy on the target system, the target MyID CMS system imports the certificate using this certificate policy (usually the `Unmanaged Imported` policy).

When you use the `RESTCertificateAdded.xml` mapping file, MyID sends as much information as possible about the certificate so that you can process the information that is important to your own target system.

The data sent when using the `RESTCertificateAdded.xml` mapping file is:

```
{
  "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "target": {
    "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "name": "John Smith"
  },
  "automaticRenewal": "0 | 1",
  "cAPath": "Certificates.CAPath",
  "certPolicy": "Certificates.CertPolicy",
  "certSerialNo": "Certificates.CertSerialNo",
  "status": "1",
  "certStatusDescription": "Certificates.CertStatusDescription",
  "collected": "1",
  "containerName": "Certificates.ContainerName",
  "number": "Certificates.ID",
  "imported": "Certificates.Imported",
  "issuance": {
    "completeTime": "yyyy-MM-dd HH:mm:ss",
    "startTime": "yyyy-MM-dd HH:mm:ss"
  },
  "issuerDN": "Certificates.IssuerDN",
  "keyArchivalDisplayName": "Certificates.KeyArchivalDisplayName",
  "keyArchived": "Certificates.KeyArchived",
  "expiryDate": "Certificates.NotAfter",
  "dateIssued": "Certificates.NotBefore",
  "renewalDate": "yyyy-mm-dd",
  "requestID": "Certificates.RequestID",
  "response": "Certificates.Response",
  "statusMapping": {
    "id": "1"
  },
  "userDN": "Certificates.UserDN",
  "userSid": "Certificates.UserSID",
}
```

This is based on the data that you can retrieve for a certificate using the following endpoint in the MyID Core API:

```
GET /api/Certificates/{id}
```

2.1.3.3 Expected response

A 200 OK response in the event of success.

2.1.3.4 Setting up your certificate policies

If you have set up an external system using the **REST Certificate Added** notification, the following additional options appear when editing a certificate policy in the **Certificate Authorities** workflow:

Enabled (Allow Issuance)

Display Name:

Description:

Allow Identity Mapping:

Reverse DN:

Archive Keys: ▼

Certificate Lifetime:

Automatic Renewal:

Certificate Storage: Hardware Software Both

Recovery Storage: Hardware Software Both None

Key Algorithm: ▼

Key Purpose: ▼

External Notification Data:

External System Notifications:

[Edit Attributes](#)

[Supersede](#)

- **External Notification Data** – type the object ID of the certificate policy on the target MyID CMS system under which you want to import the certificate.

This is used when sending a notification to another MyID CMS system. You can find the object ID of the certificate policy in the database of the target MyID CMS system; typically, you want to use the `Unmanaged Imported` certificate policy. Run the following SQL against the MyID database:

```
select ObjectID from CertPolicies where Template='Unmanaged Imported'
```

- **External System Notifications** – select the notifications you want to trigger when a certificate using this policy is added to the system. The box lists all of the external systems you have set up with a **Notification** type of **Rest Certificate Added**. If you want to trigger the notification to multiple systems, hold CTRL or SHIFT and click to select multiple items in the list.

2.1.3.5 Setting up an external MyID CMS system to receive notifications

You can connect your primary MyID CMS system to an external MyID CMS system through the `REST Certificate Added` notifications so that every time you add a certificate to the primary MyID CMS system, it sends a notification to the external MyID CMS system containing the PKCS #7 certificate data so that it can import the certificate.

This connection is made through the MyID Core API, using the following endpoint:

```
POST /api/Certificates/import
```

To configure your external MyID CMS system to receive notifications, you must set it up for server-to-server OAuth2 authentication; see the *Server-to-server authentication* section in the [MyID Core API](#) guide. Make sure you grant the user account permissions to the `/api/Certificates/import` endpoint; see the *Accessing the API features* section.

Once you have set up the external MyID CMS system for server-to-server authentication, you can set up the external system in your primary MyID CMS system so that your primary MyID CMS system can authenticate to the external system:

1. From the **Configuration** category, select **External Systems**.

You can also launch this workflow from the **Connections and Notifications** section of the **More** category in the MyID Operator Client. See the *Using Connections and Notifications workflows* section in the [MyID Operator Client](#) guide for details.

2. Click **New**.
3. From the **Listener Type** drop-down list, select **RESTService**.
4. Complete the following details:
 - **Name** – Type a name for your REST notification external system.
 - **Description** – Type a description for the external system.
 - **Enabled** – Select this option to enable the notification.
 - **Mapping File** – Select `RESTCertificateAddedMyID.xml` from the drop-down list.
 - **Notification** – Select `REST Certificate Added` from the drop-down list.
 - **API Location** – Type the base URL of the MyID Core API on the external MyID CMS system.

For example:

```
https://external.example.com/rest.core
```

5. Provide the server-to-server authentication details you configured on the external MyID CMS system.

- **OAuth Token Endpoint** – Type the URL of the token endpoint on the MyID web.oauth2 web service on the external MyID CMS system.

For example:

```
https://external.example.com/web.oauth2/connect/token
```

- **Client ID** – Type the client ID that you created when you set up server-to-server authentication on the external MyID CMS system.

For example:

```
myid.mysystem
```

- **Requested Scopes** – Provide the scope you configured when you set up server-to-server authentication.

Typically, this is:

```
myid.rest.basic
```

- **Client Secret** – Type the client secret you created when you set up server-to-server authentication.
- **Confirm Client Secret** – Type the client secret again to confirm it.

Note: You do not have to provide a **Bearer** token; MyID CMS requests the bearer token automatically using the above authentication details.

6. Click **Save**.

2.1.4 REST Device Cancelled notification

The `REST Device Cancelled` event sends a notification to a REST web service when a device is canceled and therefore removed from association with any people. This may be caused by a variety of operations; for example, canceling a device, erasing a device, or reaching a limit of devices per person, for example.

If you configure an external system with the name `REST Device Cancelled` and specify the corresponding mapping file `RESTDeviceCancelled.xml`, you can configure MyID to respond to the event with the following.

2.1.4.1 Endpoint

POST `/devices/{device GUID}/deviceCancelled`

2.1.4.2 Data

```
{
  "person": {
    "id": "<person guid>",
    "group": {
      "id": "<group guid>",
      "name": "<group name>"
    },
    "name": {
      "first": "<first name>",
      "last": "<surname>"
    },
    "contact": {
      "emailAddress": "<email address>"
    },
    "employeeId": "<employee ID / security number>",
    "logonName": "<logon name>"
  },
  "device": {
    "id": "<device guid>",
    "sn": "<device serial number>",
    "dt": "<MyID device type>",
    "hid": {
      "serialNumber": "<HID loop ID>",
      "facilityCode": "<HID facility code>"
    }
  }
}
```

2.1.4.3 Expected response

A 200 OK response in the event of success.

2.1.5 REST Device Issued notification

The `REST Device Issued` event sends a notification to a REST web service when a device is issued and has a status of `Active`.

If you configure an external system with the name `REST Device Issued` and specify the corresponding mapping file `RESTDeviceIssued.xml`, you can configure MyID to respond to the event with the following.

2.1.5.1 Endpoint

POST `/devices/deviceIssued`

2.1.5.2 Data

```
{
  "person": {
    "id": "<person guid>",
    "group": {
      "id": "<group guid>",
      "name": "<group name>"
    },
    "name": {
      "first": "<first name>",
      "last": "<surname>"
    },
    "contact": {
      "emailAddress": "<email address>"
    },
    "employeeId": "<employee ID / security number>",
    "photo": "<base 64 encoded user image>",
    "logonName": "<logon name>"
  },
  "device": {
    "id": "<device guid>",
    "sn": "<device serial number>",
    "dt": "<MyID device type>",
    "validity": {
      "from": "<ISO format issuance date>",
      "to": "<ISO format expiration date>",
      "enabled": <true|false>
    },
    "hid": {
      "serialNumber": "<HID loop ID>",
      "facilityCode": "<HID facility code>"
    }
  }
}
```

2.1.5.3 Expected response

A `200 OK` response in the event of success.

2.1.6 REST Device Reassigned notification

The `REST Device Reassigned` event sends a notification to a REST web service when a device is reassigned.

Note: You can reassign devices only using the MyID Core API; see the *Reassigning devices* section in the *MyID Core API* guide for details.

If you configure an external system with the name `REST Device Reassigned` and specify the corresponding mapping file `RESTDeviceReassigned.xml`, you can configure MyID to respond to the event with the following.

2.1.6.1 Endpoint

POST `/devices/deviceReassigned`

2.1.6.2 Data

```
{
  "devices": {
    "id": "<device guid>",
    "previousOwnerID": "<previous owner ID>",
    "newOwnerID": "<new owner ID>",
    "operation": "reassign"
  }
}
```

2.1.6.3 Expected response

A `200 OK` response in the event of success.

2.1.7 REST Person Added notification

The `REST Person Added` event sends a notification to a REST web service when a person's account is added to MyID.

If you configure an external system with the name `REST Person Added` and specify the corresponding mapping file `RESTPersonAdded.xml`, you can configure MyID to respond to the event with the following.

2.1.7.1 Endpoint

```
PATCH /people/{People.ObjectID}/personAdded
```

2.1.7.2 Data

```
{
  "person": {
    "id": "<object ID of the person>",
    "account": {
      "dn": "<person Distinguished Name>",
      "domain": "<domain>",
      "samAccountName": "<SAM account name>",
      "upn": "<UPN>"
    },
    "contact": {
      "emailAddress": "<email>"
    },
    "employeeId": "<employee id>",
    "enabled": "<0|1>",
    "name": {
      "first": "<first name>",
      "fullName": "<full name>",
      "last": "<last name>"
    },
    "group": {
      "id": "<object ID of the group>",
      "name": "<group name>"
    },
    "logonName": "<logon name>"
  }
}
```

2.1.7.3 Expected response

A 200 OK response in the event of success.

2.1.8 REST Person Deleted notification

The `REST Person Deleted` event sends a notification to a REST web service when a person's account is deleted from MyID.

If you configure an external system with the name `REST Person Deleted` and specify the corresponding mapping file `RESTPersonDeleted.xml`, you can configure MyID to respond to the event with the following.

2.1.8.1 Endpoint

```
DELETE /people/{People.ObjectID}/personDeleted
```

2.1.8.2 Data

```
{
  "person": {
    "id": "<object ID of the person>",
    "account": {
      "dn": "<person Distinguished Name>",
      "domain": "<domain>",
      "samAccountName": "<SAM account name>",
      "upn": "<UPN>"
    },
    "contact": {
      "emailAddress": "<email>"
    },
    "enabled": "0",
    "name": {
      "first": "<first name>",
      "fullName": "<full name>",
      "last": "<last name>"
    },
    "group": {
      "id": "<object ID of the group>",
      "name": "<group name>"
    },
    "logonName": "<logon name>"
  }
}
```

2.1.8.3 Expected response

A 200 OK response in the event of success.

2.1.9 REST Person Disabled notification

The `REST Person Disabled` event sends a notification to a REST web service when a person's account is disabled in MyID.

If you configure an external system with the name `REST Person Disabled` and specify the corresponding mapping file `RESTPersonDisabled.xml`, you can configure MyID to respond to the event with the following.

2.1.9.1 Endpoint

POST `/people/{People.ObjectID}/personDisabled`

2.1.9.2 Data

```
{
  "person": {
    "id": "<object ID of the person>",
    "account": {
      "dn": "<person Distinguished Name>",
      "domain": "<domain>",
      "samAccountName": "<SAM account name>",
      "upn": "<UPN>"
    },
    "contact": {
      "emailAddress": "<email>"
    },
    "enabled": "0",
    "name": {
      "first": "<first name>",
      "fullName": "<full name>",
      "last": "<last name>"
    },
    "group": {
      "id": "<object ID of the group>",
      "name": "<group name>"
    },
    "logonName": "<logon name>"
  }
}
```

2.1.9.3 Expected response

A 200 OK response in the event of success.

2.1.10 REST Person Edited notification

The `REST Person Edited` event sends a notification to a REST web service when a person's account is edited in MyID.

If you configure an external system with the name `REST Person Edited` and specify the corresponding mapping file `RESTPersonEdited.xml`, you can configure MyID to respond to the event with the following.

2.1.10.1 Endpoint

```
PATCH /people/{People.ObjectID}/personEdited
```

2.1.10.2 Data

```
{
  "person": {
    "id": "<object ID of the person>",
    "account": {
      "dn": "<person Distinguished Name>",
      "domain": "<domain>",
      "samAccountName": "<SAM account name>",
      "upn": "<UPN>"
    },
    "contact": {
      "emailAddress": "<email>"
    },
    "employeeId": "<employee id>",
    "enabled": "<0|1>",
    "name": {
      "first": "<first name>",
      "fullName": "<full name>",
      "last": "<last name>"
    },
    "group": {
      "id": "<object ID of the group>",
      "name": "<group name>"
    },
    "logonName": "<logon name>"
  }
}
```

2.1.10.3 Expected response

A 200 OK response in the event of success.

2.1.11 REST Person Enabled notification

The `REST Person Enabled` event sends a notification to a REST web service when a person's account is enabled in MyID.

If you configure an external system with the name `REST Person Enabled` and specify the corresponding mapping file `RESTPersonEnabled.xml`, you can configure MyID to respond to the event with the following.

2.1.11.1 Endpoint

POST `/people/{People.ObjectID}/personEnabled`

2.1.11.2 Data

```
{
  "person": {
    "id": "<object ID of the person>",
    "account": {
      "dn": "<person Distinguished Name>",
      "domain": "<domain>",
      "samAccountName": "<SAM account name>",
      "upn": "<UPN>"
    },
    "contact": {
      "emailAddress": "<email>"
    },
    "employeeId": "<employee id>",
    "enabled": "1",
    "name": {
      "first": "<first name>",
      "fullName": "<full name>",
      "last": "<last name>"
    },
    "group": {
      "id": "<object ID of the group>",
      "name": "<group name>"
    },
    "logonName": "<logon name>"
  }
}
```

2.1.11.3 Expected response

A 200 OK response in the event of success.

2.1.12 REST Request Added notification

The `REST Request Added` event sends a notification to a REST web service when an Issue Card request is created in MyID.

This notification is triggered when the Issue Card request reaches a status of `Awaiting Issue`.

If you configure an external system with the name `REST Request Added` and specify the corresponding mapping file `RESTRequestAdded.xml`, you can configure MyID to respond to the event with the following.

2.1.12.1 Endpoint

POST `/requests/{Requests.ObjectID}/requestAdded`

2.1.12.2 Data

```
{
  "request": {
    "target": {
      "id": "<object ID of the target of the request>",
      "name": "<target name>",
      "logonName": "<target logon name>"
    },
    "task": {
      "desc": "IssueCard"
    },
    "id": "<object ID of the request>",
    "initiationDate": "<date the request was created>",
    "status": "<request status>",
    "credProfile": {
      "id": "<object ID of the credential profile>",
      "name": "<credential profile name>"
    }
  }
}
```

2.1.12.3 Expected response

A `200 OK` response in the event of success.

2.1.13 REST Request Updated notification

The `REST Request Updated` event sends a notification to a REST web service when an Issue Card request is updated in MyID.

This notification is triggered when the Issue Card request reaches one of the following statuses:

- Cancelled
- AutoDisabled
- Await Bureau
- Cancelled
- Completed
- Completed With Errors
- Created
- Failed
- In Progress
- In Transit
- Suspended

If you configure an external system with the name `REST Request Updated` and specify the corresponding mapping file `RESTRequestUpdated.xml`, you can configure MyID to respond to the event with the following.

2.1.13.1 Endpoint

```
PATCH /request/{Requests.ObjectID}/requestUpdated
```

2.1.13.2 Data

```
{
  "request": {
    "target": {
      "id": "<object ID of the target of the request>",
      "name": "<target name>",
      "logonName": "<target logon name>"
    },
    "task": {
      "desc": "IssueCard"
    },
    "id": "<object ID of the request>",
    "initiationDate": "<date the request was created>",
    "status": "<request status>",
    "credProfile": {
      "id": "<object ID of the credential profile>",
      "name": "<credential profile name>"
    }
  }
}
```

2.1.13.3 Expected response

A 200 OK response in the event of success.

2.2 Creating a mapping file

MyID provides example mapping files for the standard notifications; see section 2.1, *Standard REST notifications* for details of these files and their output.

You can use these files as a basis for your own files, or create your own files from scratch. Do not edit the content of the provided files, as they may be overwritten if you update or upgrade your system.

Mapping files are stored in the following folder by default:

```
C:\Program Files\Intercede\MyID\Components\ExternalSystemMappings\
```

Any XML file you place in this folder is available for selection from the **Mapping File** drop-down list in the **External Systems** workflow.

Note: If you make changes to your mapping file after setting up your external system, you must edit your external system and save it again so that MyID picks up your changes.

2.2.1 Notification

The `<Notification>` node is the root node of the XML file. Each mapping file has one and only one `<Notification>` node, which contains the `<DataSources>`, `<Endpoint>`, and `<Body>` nodes.

2.2.2 DataSources

The `<DataSources>` node contains one or more `<DataSource>` nodes, each of which specifies the name of a table or view in the MyID database. You must specify which field in the table or view you are matching against the device, person, or job ID for the notification. You can also specify an optional WHERE clause to restrict the data further; MyID does not support multiple records in a notification, so if the data source and where clause result in multiple records, the first record is used.

Each `<DataSource>` node has the following format:

```
<DataSource ID="id" View="view or table name" Lookup="identifier" FieldName="field" />
```

where:

- **ID** – a unique identifier for the data source. You use this to identify the data you want to use in the endpoint and body.
- **View** – the name of the view or table in the MyID database that contains the data you want to use.
- **Lookup** – the identifier used to look up the appropriate record in the table or view. You can use the following values:
 - **DeviceID** – the ID of the device that is the subject of the notification.
 - **PersonID** – the ID of the person who is the subject of the notification.
 - **JobID** – the ID of the job of the operation that is being carried out.
- **FieldName** – optionally, if it is not the same as the `Lookup` value, the name of the field in the view or table to compare against the identifier.

For example:

```
<DataSources>
  <DataSource ID="People" View="vPeopleUserAccounts" Lookup="PersonID" />
  <DataSource ID="Devices" View="vDevices" Lookup="DeviceID" />
</DataSources>
```

This creates two data sources:

- A data source called `People` that contains the data from the `vPeopleUserAccounts` view in the MyID database. The ID of the person who is the subject of the notification is used to match against the `PersonID` in the view.
- A data source called `Devices` that contains the data from the `vDevices` view in the MyID database. The ID of the device that is the subject of the notification is used to match against the `DeviceID` field in the view.

You can now access any field in the `vPeopleUserAccounts` view for the person, or the `vDevices` view for that device; for example, to include the device GUID in the endpoint:

```
<Endpoint Verb="POST" URL="/devices/{Devices.ObjectID}/deviceCancelled" />
```

or, to include the person's email address in the body payload:

```
<Source Field="Email" Retrieval="People" />
```

2.2.2.1 Where clauses

Optionally, you can add one or more `<Where>` nodes inside the `<DataSource>` node. This node has the following format:

```
<Where FieldName="fieldname" FieldValue="fieldvalue" Operation="operation" Conjunction="
[and|or]" />
```

where:

- `FieldName` – the name of a field in the view or table.
- `FieldValue` – the value to compare against the content of this field.
- `Operation` – the comparison operation. You can use the following:
 - `=` equal to – the default when a single field value is specified
 - `!=` not equal to
 - `lt` less than
 - `le` less than or equal to
 - `gt` greater than
 - `ge` greater than or equal to
 - `like` like
 - `isnull` value is null
 - `notnull` value is not null
 - `IN` value is one of the listed values – the default when multiple values are specified
- `Conjunction` – optionally, the conjunction where you have multiple `<where>` nodes. You can use the following:
 - `and`
 - `or`

For example:

```
<DataSource ID="PivAuthCert" View="vCertInstances" Lookup="DeviceID">
  <Where>
    <FieldName>ContainerName</FieldName>
    <FieldValue>5FC105</FieldValue>
    <Operation>=</Operation>
  </Where>
</DataSource>
```

This returns the details from the `vCertInstances` view for the device that is the subject of the notification; as there are multiple certificates for a single device, the `<where>` node restricts the returned data to the certificate where the container name matches the name for the PIV Authentication certificate.

2.2.3 Endpoint

The `<Endpoint>` node specifies the REST web service endpoint to which MyID sends the notification. The format is:

```
<Endpoint Verb="verb" URL="url"/>
```

where:

- **Verb** – the HTTP verb used to call the REST web service. You can use the following:
 - POST (this is the default)
 - PUT
 - DELETE
 - PATCH
- **URL** – the endpoint on the server.

The URL to which MyID sends the notification is constructed from the **API Location** field in the external system (see section 2.3, *Configuring an external system for REST notifications*) and the URL you provide in the mapping file.

You can include substitutions from your data sources in this endpoint; for example, you may want to include the device GUID. Use the following format:

```
{DataSource.FieldName}
```

For example:

```
{Devices.ObjectID}
```

For example:

```
<Endpoint Verb="POST" URL="/devices/{Devices.ObjectID}/deviceCancelled" />
```

If the **API Location** field in the external system is:

```
https://myserver/Notify
```

and the device GUID (as obtained from the `ObjectID` field in the `Devices` data source, which refers to the `vDevices` view in the database) is:

```
79036d80-8322-404f-b146-f37af6b016b4
```

then MyID sends a POST request to:

```
https://myserver/Notify/devices/79036d80-8322-404f-b146-f37af6b016b4/deviceCancelled
```

2.2.4 Body

The `<Body>` node specifies the data that is sent to the REST web service.

Within the `<Body>` node, you can include one or more `<Property>` nodes. Each `<Property>` node has a `JPath` attribute that specifies the location in the JSON output where the value is to be included, and a `<Source>` node that specifies the data to include in the JSON output for the specified `JPath`.

The format is:

```
<Property JPath="path">
  <Source
    Retrieval="datasource"
    Field="fieldname"
    EncodingFormat="format"
    DataType="type"
    Default="staticvalue" />
</Property>
```

where:

- `JPath` – the path in the JSON output to create.
For example, `device.id` creates a top-level element called `device` and within that an element called `id`.
- `Retrieval` – the name of the data source from which you want to obtain the data.
- `Field` – the name of the field in the data source you want to include.
- `EncodingFormat` – if you are retrieving Boolean or date information from the database, specify one of the following:
 - `Boolean` – the value is output as `true` or `false`.
 - `Date` – the value is output in ISO8601 date time format: `yyyy-MM-ddTHH:mm:ss.000Z`
- `DataType` – if you are retrieving image, base64, or GUID data from the database, specify one of the following:
 - `Image` – the image is stored as binary object data in the database.
Note: You cannot access image data when the data is stored as a file; you can access only image data that is stored as binary object data in the database.
 - `HexedB64Certificate` – the retrieved data is in hexed base64 format. The data is converted to base64 format and any carriage return or line feed characters are removed before they are written to the JSON body.
 - `Guid` – when retrieving a GUID from the database (for example, the `ObjectID` for a device) it is enclosed in braces `{ }` – you cannot include these in a JSON file. Specify a `DataType` of `Guid` to strip the braces from the GUID before writing it to the JSON body.

- **Default** – a static value to use if the retrieval from the database does not produce a value. Alternatively, you can use the `Default` attribute instead of the `Retrieval` and `Field` attributes to provide a static value.

If you do not provide a `Default` value, and the retrieval from the database does not produce a value, the path specified in the `JPath` is not included in the JSON output.

For example:

```
<Property JPath="device.validity.enabled">
  <Source EncodingFormat="Boolean" Field="Enabled" Retrieval="Devices" />
</Property>
```

This obtains the value for `Enabled` field in the `Devices` data source, which refers to the `vDevices` view in the database; this value is 0 or 1 in the database, but the `EncodingFormat` tells MyID to output it as `true` or `false` to the appropriate place in the JSON output; for example:

```
{
  "device": {
    "validity": {
      "enabled": true
    },
  }
}
```

2.2.4.1 Additional processing

Optionally, you can add a `<Processor>` node inside the `<Source>` node to carry out additional processing on the value obtained from the database. This node has the following format:

```
<Processor Type="type" P1="Param1" P2="Param2" />
```

where:

- `Type` – currently, this must be:
 - `substring` – obtains a part of the string value.
- `P1` – the first parameter. For `substring`, this is the start location (based on zero-based indexing).
- `P2` – the second parameter. For `substring`, this is the number of characters. If you do not specify a number of characters, the value returned is from the start location to the end of the string.

For example:

```
<Property JPath="device.fascn">
  <Source Field="SN3" Retrieval="Devices">
    <Processor Type="substring" P1="14" P2="6"/>
  </Source>
</Property>
```

This obtains the value for the `SN3` field in the `Devices` data source, which refers to the `vDevices` view in the database, then obtains a 6-character section of this value starting at position 14 (based on zero-based indexing).

If the `SN3` value is:

0011 - 0000 - 250018 - 1 - 1 - 0000000001 1 1234 4 - 28

the output is:

```
{
  "device": {
    "fascn": "250018"
  },
}
```

2.3 Configuring an external system for REST notifications

To enable a REST notification, you must create a connection to your REST web service using the **External Systems** workflow.

To create an external system:

1. From the **Configuration** category, select **External Systems**.

You can also launch this workflow from the **Connections and Notifications** section of the **More** category in the MyID Operator Client. See the *Using Connections and Notifications workflows* section in the *MyID Operator Client* guide for details.

2. Click **New**.
3. From the **Listener Type** drop-down list, select **RESTService**.

The details for a REST notification external system appear.

The screenshot shows the 'External System' configuration form. It includes the following fields and controls:

- Name:** Text input field.
- Description:** Text input field.
- Listener Type:** Dropdown menu with 'RESTService' selected.
- Enabled:** Checkmark icon indicating the system is enabled.
- Mapping File:** Dropdown menu with 'Please select...' selected.
- Contents of Mapping File:** Text area for the mapping file content.
- Notification:** Dropdown menu.
- API Location:** Text input field.
- OAuth Token Endpoint:** Text input field.
- Client ID:** Text input field.
- Requested Scopes:** Text input field.
- Client Secret:** Text input field.
- Confirm Client Secret:** Text input field.
- Bearer token:** Text input field.
- Confirm Bearer token:** Text input field.

At the bottom of the form, there are three buttons: '< Back', 'Save', and 'Cancel'.

4. Complete the following details:
 - **Name** – Type a name for your REST notification external system.
 - **Description** – Type a description for the external system.
 - **Enabled** – Select this option to enable the notification, or deselect it to disable the notification. When the notification is disabled, MyID does not attempt to send this notification to the external system.

- **Mapping File** – Select the mapping file you want to use from the drop-down list.

Mapping files are stored on the MyID application server in the following folder by default:

```
C:\Program Files\Intercede\MyID\Components\ExternalSystemMappings\
```

For details of the provided standard mapping files, see section [2.1, *Standard REST notifications*](#).

For details of creating your own mapping file, see section [2.2, *Creating a mapping file*](#).

Once you have selected the mapping file, its contents appear on screen.

- **Notification** – Select a notification type.

See section [2.1, *Standard REST notifications*](#) for details of these notifications.

- **API Location** – Type the base URL of your REST web service API.

The endpoint for the notification is generated by taking this API location and appending the endpoint URL specified by the mapping file; for example, if your API Location is:

```
https://myserver.example.com/Notify
```

and the mapping file specifies:

```
<Endpoint Verb="POST" URL="/devices/deviceIssued" />
```

MyID sends a `POST` notification to:

```
https://myserver.example.com/Notify/devices/deviceIssued
```

5. Provide the authentication details for your web service.

You can either provide a pre-requested bearer token, or use an OAuth2 provider to request a bearer token.

To use a pre-requested bearer token, provide the following information:

- **Bearer token** – Type the bearer token you want to use.
- **Confirm Bearer token** – Type the bearer token again to confirm it.

Note: MyID does not validate the bearer token before using it to send the notification. You must ensure that the token is valid.

Alternatively, to use an OAuth2 provider to request a bearer token, provide the following information:

- **OAuth Token Endpoint** – Type the URL of the web service token provider.
- **Client ID** – Type the client ID that has been registered with the OAuth2 token provider.
- **Requested Scopes** – Optionally, provide the scopes you want for the requested token. If you do not provide any scopes, the default scope configured for the token provider is used.
- **Client Secret** – Type the shared secret you have configured for the token provider.
- **Confirm Client Secret** – Type the shared secret again to confirm it.

Note: The token is issued with the default validity period for the bearer token scope; you cannot specify a validity period. MyID caches the provided token and requests a new one once the token's validity period has expired.

6. Click **Save**.

2.3.1 Enabling and disabling a notification

To enable or disable a notification:

1. From the **Configuration** category, select **External Systems**.
2. From the Name drop-down list, select the external system you created for the notification you want to enable or disable.
3. Click **Edit**.
4. Set the **Enabled** option.

When the notification is disabled, MyID does not attempt to send this notification to the external system.

5. Click **Save**.

2.3.2 REST notification name mapping

In previous versions of MyID, before the introduction of the **Notification** drop-down list to determine the type of notification, there was a one-to-one mapping between the **Name** of an external system and the event that triggered the notification.

You can now create multiple external systems that are triggered by the same notification event by creating multiple external systems that have the same setting in the **Notification** drop-down list.

However, if you do not specify an option in the **Notification** drop-down list, for backward compatibility purposes, MyID reverts to the previous behavior, and if the name of the external system matches a notification generated by MyID for a device lifecycle event, the notification is triggered.

This is a legacy feature and the list of notifications that use it is not being extended; you are recommended to set an option in the **Notification** drop-down list.

The list of supported notifications that use name mapping is:

- DisableCard
- EnableCard
- REST Device Cancelled
- REST Device Issued
- REST Device Reassigned
- REST Person Added
- REST Person Deleted
- REST Person Disabled
- REST Person Edited
- REST Person Enabled
- REST Request Added
- REST Request Updated

These values are case sensitive.

3 Troubleshooting REST notifications

Notifications are sent outside the main workflow process; this means that a failure to send a notification does not result in a workflow failure, is not audited, and does not appear in the system events.

If a notification cannot be sent, the notification remains in progress and MyID will retry on the following default schedule:

- After 10 minutes.
- After 30 minutes.
- After 60 minutes.
- After 4 hours.
- After 12 hours.
- After 24 hours.

If the notification has not been sent after this time, MyID stops attempting to send it.

You can view the status of the notification in the **Notifications Management** workflow; see the *Using the Notifications Management workflow* section in the [Administration Guide](#) for details.

If you make changes to your mapping file after setting up your external system, you must edit your external system and save it again so that MyID picks up your changes.; see section [2.2, Creating a mapping file](#).

To view the reason for a failure to send a notification, you can check the `ServerResponse` field in the `NotificationsInProgress` table in the MyID database.

You can also configure logging for the Notifications.Net component; see the *Log4Net* section of the [Configuring Logging](#) guide for details.