MyID

Version 10.8 Update 2

Reporting Web Service 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: IMP1779-07 March 2018

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 installation CD.

- 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 Reporting Web Service allows you to query details from the MyID® database through a web service.

The web service is allowed only to read tables or views that start with the characters `WS_`. This restriction prevents arbitrary access to the database. It is recommended that you tailor the views to your exact requirements.

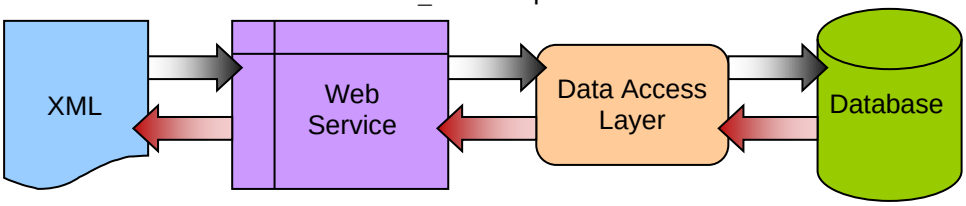
1.1 Prerequisites

To implement the web service, you must have:

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

1.2 How the web service works

The web service uses the `edefice_DAL` component to access the database.



1.3 Change history

Version	Description
TLK1557-01	First version.
TLK1557-02	Added feature codes.
TLK1557-03	Added information on increasing the maximum pool size.
TLK1557-04	Added change history.
TLK1557-05	Added information on restarting the Edefice_BOL component after changing the maximum pool size.
TLK1557-05	Added information on setting the identity for the web service application pool.
IMP1779-01	Removed from Toolkit and incorporated into MyID version 10.3. Instructions for enabling the web service. Removed instructions for adding the web service to an existing system.
IMP1779-02	Updated for MyID 10.4.
IMP1779-03	Updated for MyID 10.5.
IMP1779-04	Updated for MyID 10.6.
IMP1779-05	Updated for MyID 10.7.
IMP1779-06	Updated for MyID 10.8.
IMP1779-07	Updated to include clarification of out-of-the-box authentication settings.

2 Installing and Testing the Web Service

2.1 Enabling access to the web service

As a security feature, when you install the web service, it is configured so that it is impossible to authenticate to the service (setting **ASP.Net Impersonation** 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
```

You must enable access to the web service before you attempt to use it.

If you intend to allow access to this web service you must identify the required authentication mechanism that is suitable for your site, and then configure this in IIS. Windows authentication or 2-way SSL (require client certificates) are valid authentication mechanisms.

Anonymous authentication without enforcing additional 2-way SSL is not suitable for production sites – as this would allow global access to the web service without requiring any authentication.

Once you have configured the authentication mechanism you want to use, disable the **ASP.NET Impersonation** option for the web service in IIS.

2.1.1 Enabling 2-way SSL

Note: There are many options for configuring 2-way SSL in IIS. This is performed by configuring the IIS web server, which is a Microsoft product. Documentation and support for IIS is available from Microsoft, and fully detailing the configuration of IIS is beyond the scope of this guide.

This section shows an example of configuring the `oneToOneCertificateMappings` feature in IIS, which enforces that only clients able to authenticate with specific white-listed client SSL certificates can access the web service.

To enable 2 way SSL to the web service using `oneToOneCertificateMappings`:

1. Ensure that standard SSL is first configured and required for the web service.
2. Ensure that the intended clients to this web service have enrolled client SSL certificates.
3. To use `oneToOneCertificateMappings`, the feature must be installed on the web server running the feature.

   `oneToOneCertificateMappings` is an optional feature on an IIS installation.
4. Edit the `Web.config` file for the `MyIDWebService` directory, to ensure that `clientCertificateMappingAuthentication` is enabled.

   The `security/authentication` section should be as follows:

```
<security>
    <authentication>
    <clientCertificateMappingAuthentication enabled="true" />
    </authentication>
</security>
```

5. In the `MyIDWebservice` virtual directory:
   a) Under **SSL Settings**, select **Require Client Certificates**.
   b) Under the **Configuration Editor**:
      i Enter the following in the **Section** box:

      ```
      system.webServer/security/authentication/
      iisClientCertificateMappingAuthentication
      ```

      ii Set **Enabled** to `True`.
      iii Set **oneToOneCertificateMappingsEnabled** to `True`.
      iv Select `oneToOneMappings`, and add the client SSL certificates that should be available to select, mapping them to the MyID COM user account. The required certificates are entered as Base64 strings without any whitespace or PEM headers.
   c) Under **Authentication**:
      i Enable **Anonymous authentication**, and configure it to use the MyID COM user account.
      ii Ensure all other Authentication mechanisms displayed are disabled.

      Note: **ASP.NET Impersonation** is enabled by default on some versions of MyID – this must be disabled.
6. Open a Windows command prompt and type `iisreset`.
7. Test that the clients using the specified client SSL certificates are able to connect to the web service, and that clients unable to authenticate with a configured client SSL certificate are denied access to the web service.

   In the event that a configuration error is allowing clients without the configured client SSL certificate to connect, disable Anonymous Access while you diagnose the configuration issue, to prevent unauthorized access.

2.1.2 Enabling Windows Authentication

Note: Windows authentication is suitable only for cases where the calling system is a Windows machine that is able to authenticate using a user account that has permission to call the MyID COM+ components.

To enable Windows Authentication to the web service:

1. In Windows Administrative Tools, open Internet Information Services (IIS) Manager.
2. Navigate to the Default Web Site.
3. Set the authentication permissions of the virtual directory:
   a) Select the **MyIDWebService** virtual directory.
   b) Make sure **Features View** is selected.
   c) Double-click **Authentication**.
   d) Click **Windows Authentication** and then click **Enable**.
   e) Ensure all other Authentication mechanisms displayed are disabled.

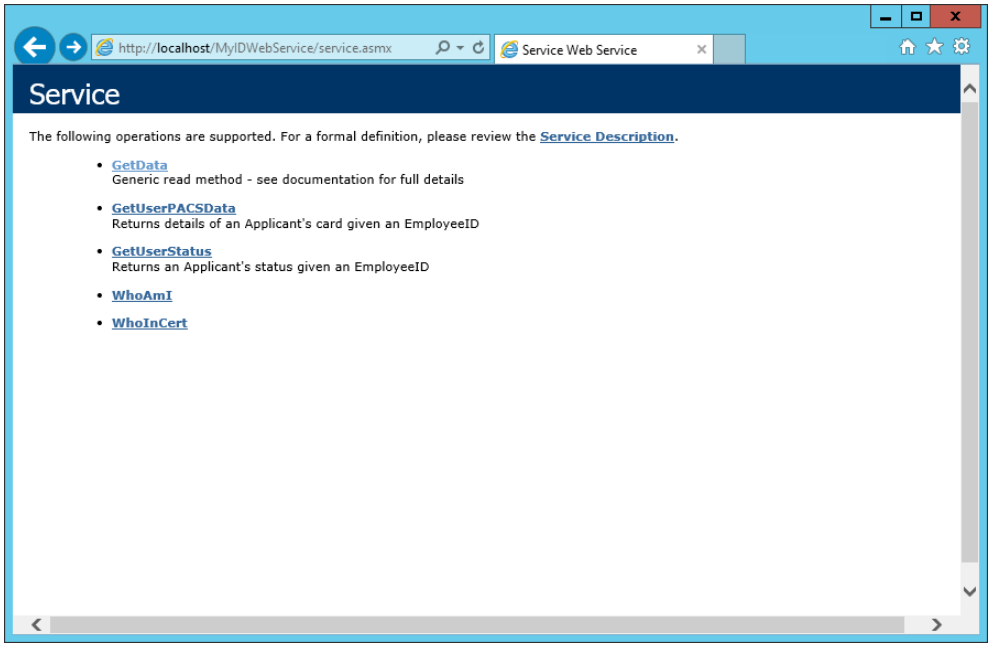
      Note: **ASP.NET Impersonation** is enabled by default on some versions of MyID – this must be disabled.
4. Open a Windows command prompt and type `iisreset`.

2.2 Testing the web service

You can access the web service directly from a web browser. Once you have installed the web service, navigate to:

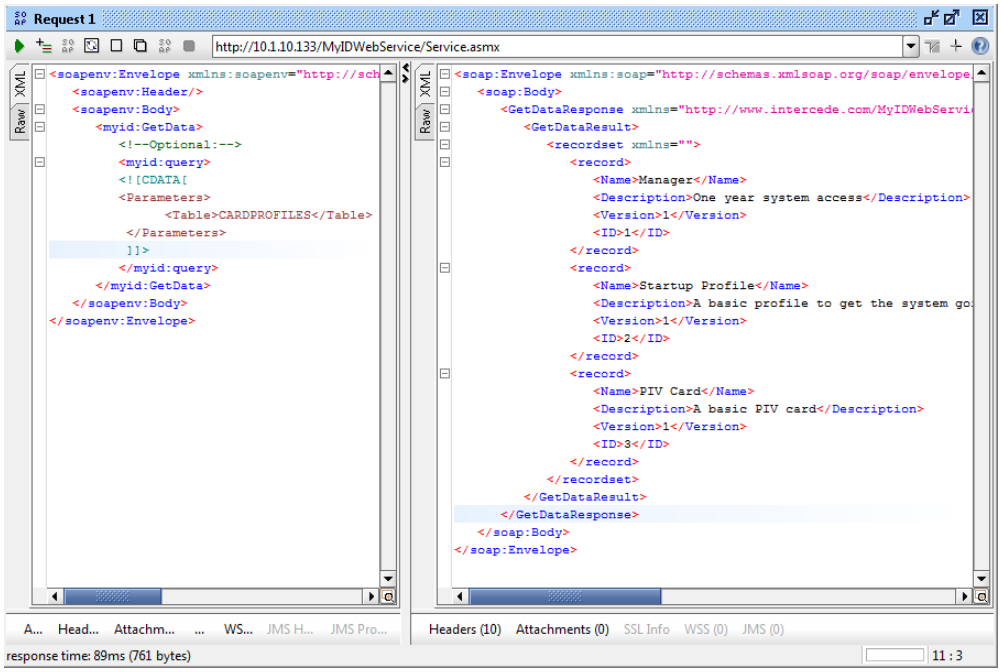
`http://<server_address>/MyIDWebService/service.asmx`

This presents you with a list of the publicly available methods. You can also access the WDSL definition from this page.



Click the `GetData` method to detail the SOAP interface; this also allows you to pass sample data to the method. To pass sample data to the interface, you must run the method on the machine where the service is hosted.

However, you are recommended to test the web service using a SOAP-compliant test client from a remote PC.



3 Restrictions

3.1 Read-only

The web service allows only read operations from the database. There is no mechanism to update data through this interface.

If you want to update data in the MyID database through a web service, you can use the MyIDEnroll web service. See the ***Lifecycle API*** document for details.

3.2 Data source restrictions

The web service is only permitted to read tables or views that start with the characters `WS_`. This restriction prevents arbitrary access to the database. It is recommended that you tailor the views to your exact requirements.

Note: The `WS_` prefix is automatically added to the name of the table or view specified in the `Table` node.

3.3 Service Provider Model restrictions

At present the web service does not support the Service Provider deployment model for MyID; that is, multiple databases sharing a middleware tier. It can connect only to the default database in an installation.

3.4 Auditing

At present the web service does not audit the fact it has been called, other than in the IIS log.

4 GetData – Generic Read Method

The generic read method for the web service is called `GetData`. This method accepts a string XML representation of a SQL statement and runs it against a specified view.

Note: The name of the view passed in is automatically prefixed with `WS_`. See section *3*, *Restrictions*.

4.1 Query XML format

You must pass in well-formed XML. The `Table` node is mandatory; all other nodes are optional.

```
<Parameters>
	<Table> Table name </Table>
	<Distinct />
	<Field><FieldName> Name of a field in the table or view </FieldName></Field>
	...
	<Where>
		<FieldName> Field name </FieldName>
		<FieldValue> Field value </FieldValue>
		<FieldType> Field type </FieldType>
		<Operation> Operation to use </Operation>
		<Conjunction> Whether the nodes are joined with AND or OR.
</Conjunction>
	</Where>
	...
	<OrderBy>
		<OrderFieldName> Field name </OrderFieldName>
		<Desc> Yes|No </Desc>
	</OrderBy>
	...
	<MaxRecords> Maximum number of records </MaxRecords>
</Parameters>
```

4.1.1 Table

The name of the table or view in the MyID database. Do not specify the `WS_` prefix; it is added automatically. For example, if you want to view the data from the `WS_CARDPROFILES` view, use the following:

```
<Table>CARDPROFILES</Table>
```

4.1.2 Distinct

If `<Distinct />` is specified, the `DISTINCT` keyword is added to the SQL query.

4.1.3 Field

You can specify multiple `Field` nodes. If no `Field` nodes are specified, all fields within the specified view are returned. Otherwise only fields specified are returned.

For example the following specifies `SELECT DeviceSerialNumber, DeviceTypeName FROM ...`

```
<Field><FieldName>DeviceSerialNumber</FieldName></Field>
```

```
<Field><FieldName>DeviceTypeName</FieldName></Field>
```

4.1.4 Where

For each <Where> node, a filter criteria is added, filtering the results that are returned.

If no <Where> nodes are specified, no WHERE clause are added to the SQL query.

Each <Where> node needs the following subnodes:

- <FieldName> specifies the name of the field that to be filtered.
- <FieldValue> specifies the value to be filtered against. If the Operation node is equals (=) or not equals (!=), wildcards are allowed, which perform a LIKE or NOT LIKE query. You can have multiple FieldValue nodes for IN operations.

  The following wildcards are supported:

  - * – represents any sequence of characters (equivalent to SQL %).
  - ? – represents a single character (equivalent to SQL _).
- <FieldType> must be one of the following values:
  - string – The field being filtered is a string value.
  - long – The field being filtered is a numeric value.
  - datestring – The field being filtered is a datetime.

  If no FieldType is specified, it defaults to string.
- <Operation> must be one of the following values:
  - = – performs an equals or like query.
  - != – performs a not equals or not like query.
  - lt – less than.
  - gt – greater than.
  - le – less than or equals.
  - ge – greater than or equals.
  - isnull – field is null.
  - notnull – field is not null.
  - IN – perform an IN query for each specified <FieldValue> node. This is the only operation that uses multiple <FieldValue> nodes.

  If no operation is specified it defaults to equals (=).
- <Conjunction> – if you omit this node, all <Where> nodes are joined with AND conjunctions. To specify an OR query, set the <Conjunction> node to or.

4.1.5 OrderBy

Allows ordering of the returned results set. You can omit this node, or include multiple <OrderBy> nodes. Each <OrderBy> has the following subnodes:

- <OrderFieldName> – the field name on which to sort.
- <Desc> – This optional node allows sorting in descending order. If Desc = Yes, the results are returned in descending order. If the node is not present, results are sorted in ascending order.

4.1.6 MaxRecords

`<MaxRecords>` – Specifies the maximum number of records to return.

If you do not specify a value, `MaxRecords` defaults to `1000`.

4.2 Example queries

4.2.1 Example one

```
<Parameters>
       <Table>CARDPROFILES</Table>
</Parameters>
```

The above query returns all records and all fields in the `WS_CARDPROFILES` view.

An example returned XML file is:

```
<?xml version="1.0" encoding="utf-8"?>
<recordset>
  <record>
    <Name>Startup Profile</Name>
    <Description>A basic profile to get the system going.</Description>
    <Version>1</Version>
    <ID>2</ID>
  </record>
  <record>
    <Name>PIV Card</Name>
    <Description>A basic PIV card</Description>
    <Version>1</Version>
    <ID>3</ID>
  </record>
  <record>
    <Name>Manager</Name>
    <Description>One year system access</Description>
    <Version>7</Version>
    <ID>10</ID>
  </record>
  <record>
    <Name>Cardholder</Name>
    <Description>
    </Description>
    <Version>3</Version>
    <ID>12</ID>
  </record>
</recordset>
```

4.2.2 Example two

```
<Parameters>
	<Table>CARDPROFILES</Table>
	<Field><FieldName>Name</FieldName></Field>
	<Field><FieldName>Description</FieldName></Field>
	<Where>
		<FieldName>Version</FieldName>
		<FieldValue>1</FieldValue>
		<FieldType>long</FieldType>
		<Operation>=</Operation>
	</Where>
	<OrderBy>
		<OrderFieldName>Name</OrderFieldName>
		<Desc>Yes</Desc>
	</OrderBy>
	<MaxRecords>10</MaxRecords>
</Parameters>
```

The above query returns the `Name` and `Description` fields from the `WS_CARDPROFILES` view, filtering on a `long` field called `Version` to return only those rows where `Version = 1`. A maximum of `10` records is returned, ordered by the `Name` in descending order.

An example returned XML file is:

```
<?xml version="1.0" encoding="utf-8"?>
<recordset>
  <record>
    <Name>Startup Profile</Name>
    <Description>A basic profile to get the system going.</Description>
    <ID>2</ID>
  </record>
  <record>
    <Name>PIV Card</Name>
    <Description>A basic PIV card</Description>
    <ID>3</ID>
  </record>
</recordset>
```

5 Deprecated Methods

The following methods are available, but are deprecated; do not use them:

- `GetUserStatus`
- `GetUserPACSData`
- `WhoAmI`
- `WhoInCert`

Use the generic, flexible `GetData` method instead of these methods.

6 Troubleshooting

6.1 Extra fields in output XML

Due to the operation of the Microsoft ADODB component, you may see extra fields in the output XML that you have not requested – these are recordset identifiers and can be ignored.

6.2 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.

6.3 Increasing the maximum pool size

When using MyID web services 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.