



## MyID

# Customizing the MyID Operator Client

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

## Copyright

© 2001-2024 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.

### **Licenses and Trademarks**

The Intercede<sup>®</sup> and MyID<sup>®</sup> word marks and the MyID<sup>®</sup> 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.

## 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

- Customizing the MyID Operator Client** ..... **1**
- Copyright** ..... **2**
- Conventions used in this document** ..... **3**
- Contents** ..... **4**
- 1 Introduction** ..... **5**
  - 1.1 Change history ..... 5
- 2 Adding links to the database** ..... **6**
  - 2.1 Use cases ..... 7
  - 2.2 Adding a link ..... 8
  - 2.3 Configuring access to a link ..... 9
  - 2.4 Removing a link ..... 10
  - 2.5 Providing substitution codes ..... 10
  - 2.6 Available icons ..... 11
  - 2.7 Obtaining the authentication token ..... 13
  - 2.8 Translating link text ..... 13
- 3 Adding links to the configuration file** ..... **14**
  - 3.1 Enabling external links ..... 14
  - 3.2 Adding a link ..... 15
- 4 Customizing the theme** ..... **17**
  - 4.1 Theme limitations ..... 19
    - 4.1.1 Legacy workflows in MyID Desktop ..... 19
    - 4.1.2 Operator Client authentication ..... 20
  - 4.2 Setting the theme ..... 21
    - 4.2.1 Changing the MyID logo ..... 22
    - 4.2.2 Adding a banner image ..... 24
  - 4.3 Troubleshooting issues with themes ..... 25
- 5 Adding content to forms** ..... **26**
  - 5.1 Working with the custom hooks file ..... 27
    - 5.1.1 The loadHook function ..... 27
    - 5.1.2 Working with values ..... 29
    - 5.1.3 Adding content ..... 30
  - 5.2 Finding data references ..... 31
    - 5.2.1 Viewing the response from the API ..... 33
    - 5.2.2 Logging data elements to the browser console ..... 35
  - 5.3 Example custom hooks file ..... 37
    - 5.3.1 Cascading style sheet ..... 37
    - 5.3.2 Adding elements ..... 37
    - 5.3.3 Escaping values ..... 37

# 1 Introduction

This document contains instructions on how to customize the MyID Operator Client.

This includes:

- Adding custom links to the database.

These external web links are controlled by the **Edit Roles** feature of MyID; you can provide different links to different users and operators depending on their roles. You can also supply parameters to the web link from the database to provide information about the logged-in user, and the external page can obtain an access token that allows you to authenticate to the MyID Core API.

See section [2, Adding links to the database](#).

- Adding custom links to the configuration file

These external web links are provided to all users, independent of their role in MyID. You can also provide an authentication token.

See section [3, Adding links to the configuration file](#).

**Note:** You can combine both methods of adding links; you can add links to the configuration file that are available to all users, and add links to the database that are available depending on the roles of the users.

- Setting a custom theme

You can specify the colors used for the MyID Operator Client; this color theme is also passed through to MyID Desktop or the Self-Service App when you launch these applications from the MyID Operator Client.

See section [4, Customizing the theme](#).

- Adding content to forms.

You can add your own custom content to forms (for example, View Person). Your content is displayed in a custom panel on the form, and can include HTML, CSS, and JavaScript.

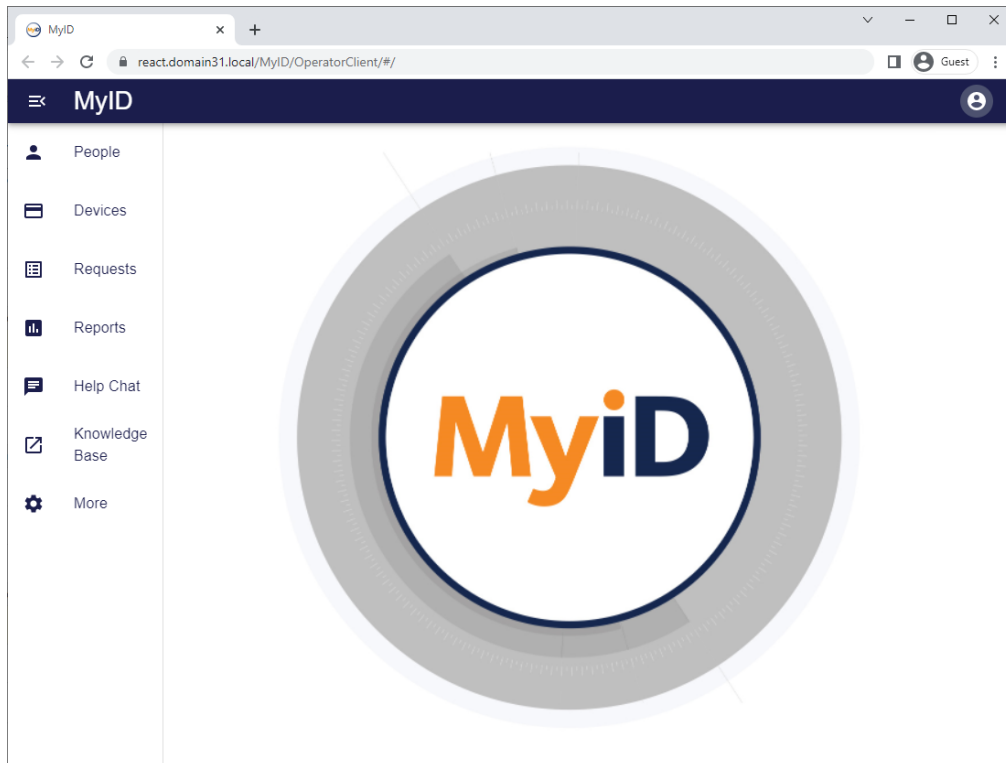
See section [5, Adding content to forms](#).

## 1.1 Change history

| Version    | Description   |
|------------|---|
| TLK2025-01 | First version.  |
| TLK2025-02 | Added details of providing a banner image.<br>Added details of adding content to forms. |

## 2 Adding links to the database

You can update the MyID database to add custom links to external web pages. These links appear in the category list in the MyID Operator Client, below the standard categories and above the More category that contains links to administrative workflows in MyID Desktop.



**Note:** Launching external links does not add any audit or system event information to the MyID database. However, if your external web page interacts with MyID (for example, using the access token to access the MyID Core API) this action will be audited as usual.

## 2.1 Use cases

You may want to use this feature for the following:

- Adding a direct link to your intranet, helpdesk, or other systems.
- Creating an email link to create an email with the recipient and subject filled in.
- Linking to custom pages that use the MyID Core API; you can pass the access token from the MyID Operator Client to your custom pages so your user does not have to log in again.
- Adding a direct link to a MyID report with pre-defined search criteria; for example, a list of requests pending approval.

You can control access to these external links using the **Edit Roles** workflow; for example, you can give your end users access to a helpline, and your operators access to an internal knowledge base.

You can include dynamic data in the external links; for example, you can pass the user's logon name to the external system so that it knows who is accessing it. You can also include an authentication token in the submission to the external link, allowing that system to authenticate to the MyID Core API or the MyID Client Service API.

To add or remove links, you use stored procedures in the MyID database, making the process quick and easy to use.

## 2.2 Adding a link

To add a link to an external site, you must run a stored procedure in the MyID database.

The syntax is:

```
EXEC sp_AddUILink
    @Name=' [Operation Name] ',
    @Href=' [Operation Link] ',
    @Icon=' [Operation Icon] ',
    @PassthroughAuth=[TRUE/FALSE];
```

where:

- @Name – the name of the link you want to add to the MyID Operator Client. This name is used to refer to the link if you want to remove it, and as its label on-screen and in the **Edit Roles** workflow. Must be unique.

See section 2.3, *Configuring access to a link* for details of using the **Edit Roles** workflow.

- @Href – the link you want to add. Provide a full URL; you can use any protocol supported by your browser, including `mailto:` links; for example:

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

```
mailto:support@mydomain.com?Subject=Request for support
```

You can also provide substitution codes that take information from the database and add it to the URL; see section 2.5, *Providing substitution codes* for details.

- @Icon – the icon you want to use for the link.  
See section 2.6, *Available icons* for a list of the icons you can use.
- @PassthroughAuth – whether MyID sends the current authentication token to the external site.

Set to `TRUE` to send the authentication token, or `FALSE` not to send the token. This parameter is optional, and defaults to `FALSE` if you do not provide it.

See section 2.7, *Obtaining the authentication token* for details of working with passthrough authentication.

For example:

```
EXEC sp_AddUILink
    @Name='Knowledge Base',
    @Href='https://myserver.example.com/kbase/',
    @Icon='Launch',
    @PassthroughAuth=FALSE;
```

The script provides a report on its actions, including the name of the link, its ID, and the reference to use if you want to change the translation of the link text:

```
Operation added with name Knowledge Base and ID 310000
```

```
To allocate permissions, use the Edit Roles workflow. The operation will be
in the Links section with name Knowledge Base
```

```
To add translations for this operation, use the namespace
links.external.310000.desc
```

For information about translations, see section 2.8, *Translating link text*.

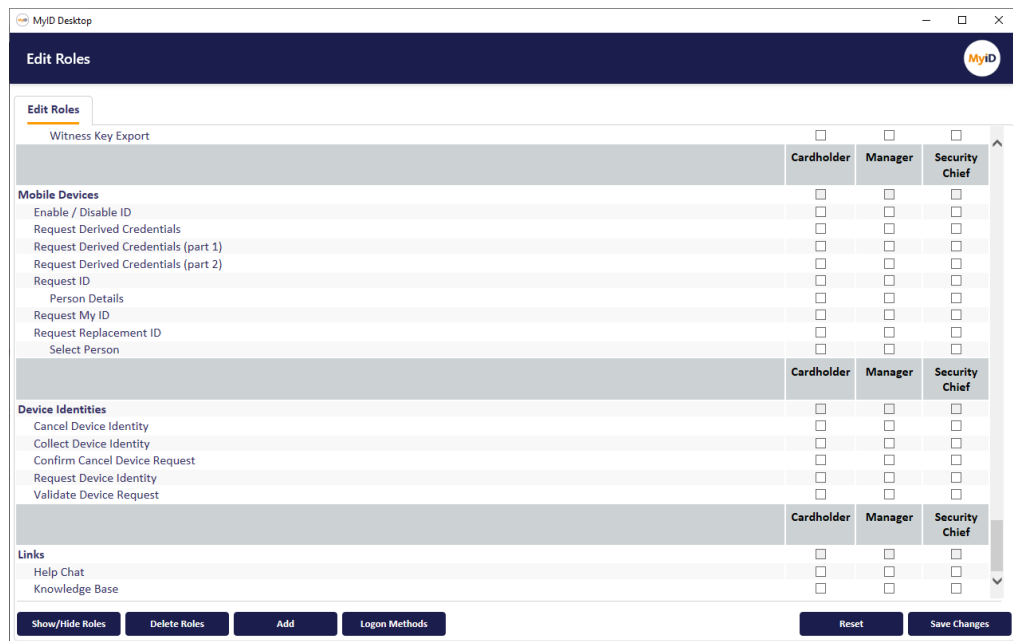


### 2.3 Configuring access to a link

Access to links is controlled through the standard MyID **Edit Roles** workflow. By default, newly-added links are not available to any user; you must use **Edit Roles** to make them available to the people you want to be able to use them.

To configure access to a link:

1. In the MyID Operator Client, from the **More** category, select **Configuration Settings** then **Edit Roles**.
2. Scroll to the **Links** section.



3. For each role you want to be able to access the link, select the option.  
You can use the **Show/Hide Roles** button to display the roles you want to work with.  
For more information about the **Edit Roles** workflow, see the *Administration Guide*.
4. Click **Save Changes**, then **Finish** to close the MyID Desktop window.
5. If you have made any changes to your own role, sign out and sign in again to see the links.

## 2.4 Removing a link

If you want to remove a link, you can use another stored procedure in the MyID database.

The syntax is:

```
EXEC sp_RemoveUILink
    @Name='[Operation Name]';
```

where:

- @Name – The name of the link you want to remove.

For example:

```
EXEC sp_RemoveUILink
    @Name='Knowledge Base';
```

The link is removed from the database, and any permissions associated with the link are also removed. If you subsequently add another link of the same name, you must set up permissions again.

## 2.5 Providing substitution codes

You can include in the link any content from the `vPeopleUserAccounts` view in the MyID database that relates to the logged-on user. Add the name of the field you want to use in double square brackets:

```
[[field name]]
```

and the value for the logged-on user is substituted when you click on the link.

For example:

```
EXEC sp_AddUILink
    @Name='User Query',
    @Href='https://myserver.example.com?user=[[LogonName]]',
    @Icon='Launch',
    @PassthroughAuth=FALSE;
```

In this example, the link is:

```
https://myserver.example.com?user=[[LogonName]]
```

If you are logged on with a `LogonName` of "Jane Smith", the link to the external website becomes:

```
https://myserver.example.com/?user=Jane%20Smith
```

















You can parse the query string in your web page; for example:

```
<html>
  <head>
    <title>onload test</title>
    <script>
      function load() {
        const queryString = window.location.search;
        const urlParams = new URLSearchParams(queryString);
        const query = urlParams.get('user');
        document.getElementById("user").innerText = query;
      }
      window.onload = load;
    </script>
  </head>
  <body>
    <p>Query requested for <span id="user">Not Set</span>.</p>
  </body>
</html>
```

## 2.6 Available icons

You can use the following icons for the external links:

- Launch
- Home
- CreditCard
- CardMembership
- VerifiedUser
- GetApp
- ListAlt
- Assessment
- Person
- Settings
- ContactMail
- MobileFriendly
- Laptop
- Chat
- Notifications
- Storage

- |   |                |   |               |
|---|----------------|---|---------------|
|    | Launch         |    | VerifiedUser  |
|    | Home           |    | GetApp        |
|    | CreditCard     |    | ListAlt       |
|    | CardMembership |    | Assessment    |
|    | Person         |    | Laptop        |
|    | Settings       |    | Chat          |
|  | ContactMail    |  | Notifications |
|  | MobileFriendly |  | Storage       |

## 2.7 Obtaining the authentication token

If you are using passthrough authentication, MyID sends the current authentication token to the external web page. You can extract this access token and use it to provide authentication; for example, you can use it to call the MyID Core API with the credentials of the logged-on user.

**Important:** MyID passes the access token to an external page only if you are connecting through https.

To obtain the token, your external page must pass a message back to the opening page to say that it has loaded; your external page must then set up a listener to wait for the response that contains the token.

For example:

```
<html>
<head>
<script>
window.addEventListener("load", () => {
  window.opener.postMessage({loaded:true}, "*")
  document.getElementById("status").innerText = "loaded";
});
window.addEventListener("message", event => {
  document.getElementById("status").innerText = "message received - " +
  new Date();
  if (event.data?.token) document.getElementById("token").innerText =
  event.data.token;
});
</script>
</head>
<body>
Status: <span id="status">Not Set</span><br/>
Token: <span id="token">Not Set</span>
</body>
</html>
```

See the *Calling the API from an external system* section of the *MyID Core API* guide for more details about using the access token to call the API.

## 2.8 Translating link text

By default, the links use the name that you provide when you add them to the database. If you want to change the name of the link, or to provide alternative names in different languages, you can use the MyID translation system to do so.

When you add a link to the database, the stored procedure reports its namespace; for example:

```
To add translations for this operation, use the namespace
links.external.310000.desc
```

You can use this namespace code to add a string to the dictionary resource files.

For more information, see the *Translating Resources* guide in the *Translation* package. For information about obtaining this document, contact customer support quoting reference SUP-138.

## 3 Adding links to the configuration file

As an alternative to adding links to the database, you can edit a configuration file to add new links to the MyID Operator Client.

This method has the following differences:

- You do not have to set up role access to see the links; all links are automatically available to all users.
- You cannot control access to the links using roles.
- You cannot use substitution codes.
- You cannot provide translations for the links.

If you need to use substitution codes, provide different links to different users, or translate the links, you must add the external links to the database instead; see section 2, [Adding links to the database](#).

### 3.1 Enabling external links

To enable external links, you must set the `MyID:LoadExtraApiLinks` option, which is stored in the `appSettings.json` file for the `rest.core` web service, to `true`. To ensure that the setting is not overwritten by subsequent updates to the MyID server, you are recommended to make this change to the `appSettings.Production.json` override file instead.

To enable external links:

1. Open the following file in a text editor:

```
appSettings.Production.json
```

This file is located in the following folder:

```
C:\Program Files\Intercede\MyID\rest.core\
```

If the file does not exist, create a new text file with that name.

2. Add the following information to the file:

```
{
  "MyID": {
    "LoadExtraApiLinks": true,
  }
}
```

3. Save the file.
4. In IIS, recycle the app pool used by the `rest.core` web service.

If you want to disable this feature, set the `MyID:LoadExtraApiLinks` option to `false`.

## 3.2 Adding a link

To add a link, you must create a file called `ExtraAPILinks.json` in the same folder as the `appSettings.Production.json` file for the `rest.core` web service, and provide the details for the link you want to add.

To add a link:

1. Open the following file in a text editor:

```
ExtraAPILinks.json
```

This file is located in the following folder:

```
C:\Program Files\Intercede\MyID\rest.core\
```

If the file does not exist, create a new text file with that name.

2. Add one or more links, using the following format:

```
[
  {
    "op": "[operation ID]",
    "cat": "[type of link]",
    "desc": "[link description]",
    "entity": "[icon code]",
    "href": "[url]"
  },
]
```

Each link has the following properties:

- `op` – the operation ID. This must be a unique value.
- `cat` – set this to one of the following:
  - `external` – the link is to an external website, but does not pass an access token.
  - `externalWithToken` – the link is to an external website, and will pass an access token.

See section [2.7, Obtaining the authentication token](#) for details of obtaining access tokens.

- `desc` – the label used for the link on screen.
- `entity` – a code for the icon.

See section [2.6, Available icons](#) for a list of the icons you can use.

- `href` – the link you want to add.

Provide a full URL; you can use any protocol supported by your browser, including `mailto:` links; for example:

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

```
mailto:support@mydomain.com?Subject=Request for support
```

3. Save the file.

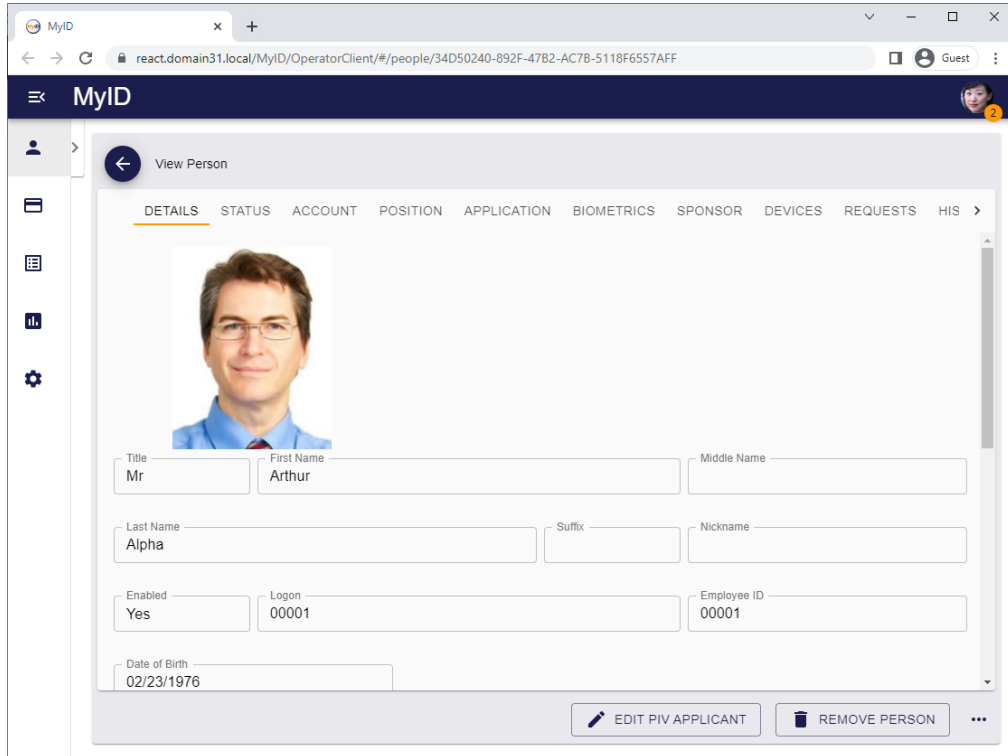
For example:

```
[
  {
    "op": "1",
    "cat": "external",
    "desc": "Knowledge Base",
    "entity": "Launch",
    "href": "https://myserver.example.com/kbase/"
  },
  {
    "op": "2",
    "cat": "externalWithToken",
    "desc": "Help Chat",
    "entity": "Chat",
    "href": "https://myserver.example.com/chat/"
  },
  {
    "op": "3",
    "cat": "external",
    "desc": "Email",
    "entity": "ContactMail",
    "href": "mailto:support@mydomain.com?Subject=Request for support"
  },
]
```

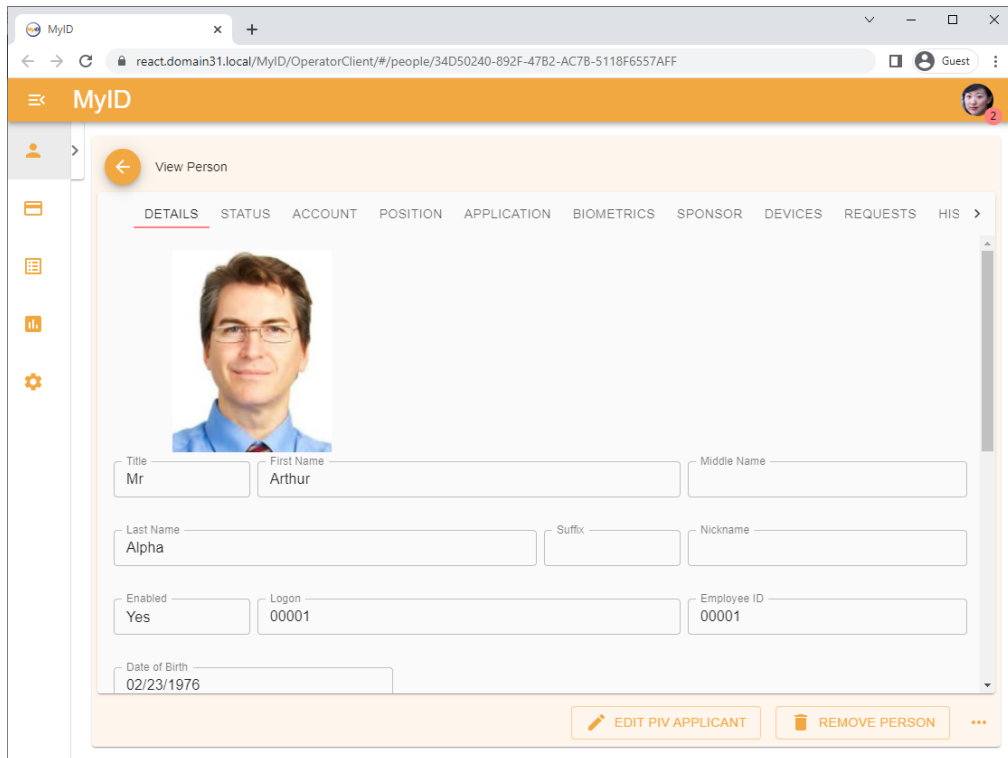


## 4 Customizing the theme

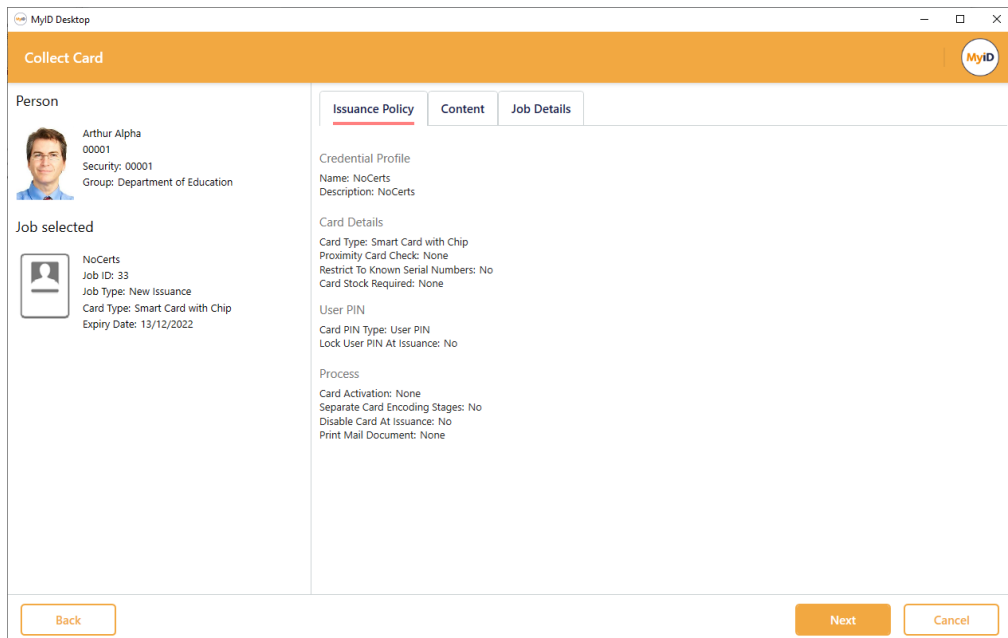
By default, the MyID Operator Client uses a blue and gray theme.



You can customize this theme to use whichever colors match your corporate identity; for example:



In addition, when you launch a workflow in MyID Desktop or the Self-Service App, the theme is passed to the application:



**Note:** You can apply a theme to the MyID Windows clients (MyID Desktop, the Self-Service App, the Self-Service Kiosk, and the MyID Client Service) locally on the client PC; you can also change the logos using this mechanism. See the *Customizing the MyID Windows Clients* guide for details.

## 4.1 Theme limitations

There are some limitations when working with themes.

### 4.1.1 Legacy workflows in MyID Desktop

Some workflows in MyID use legacy web-based technology; for these workflows, the outer MyID Desktop window uses the theme passed by the MyID Operator Client, but the buttons and other on-screen elements may retain their original colors:

MyID Desktop

Change Security Phrases

MyID

**Security Phrases**

Please choose 2 memorable security phrases from the lists below and provide answers that are only known to yourself. These questions will be used to provide access to future MyID sessions.

For security reasons, you must provide answers to ALL displayed questions in order to make changes.

Password: \*

Mother's maiden name?

Answer 1:

Answer 2:

Confirm 1:

Confirm 2:

Note \*\* denotes the default security phrase.

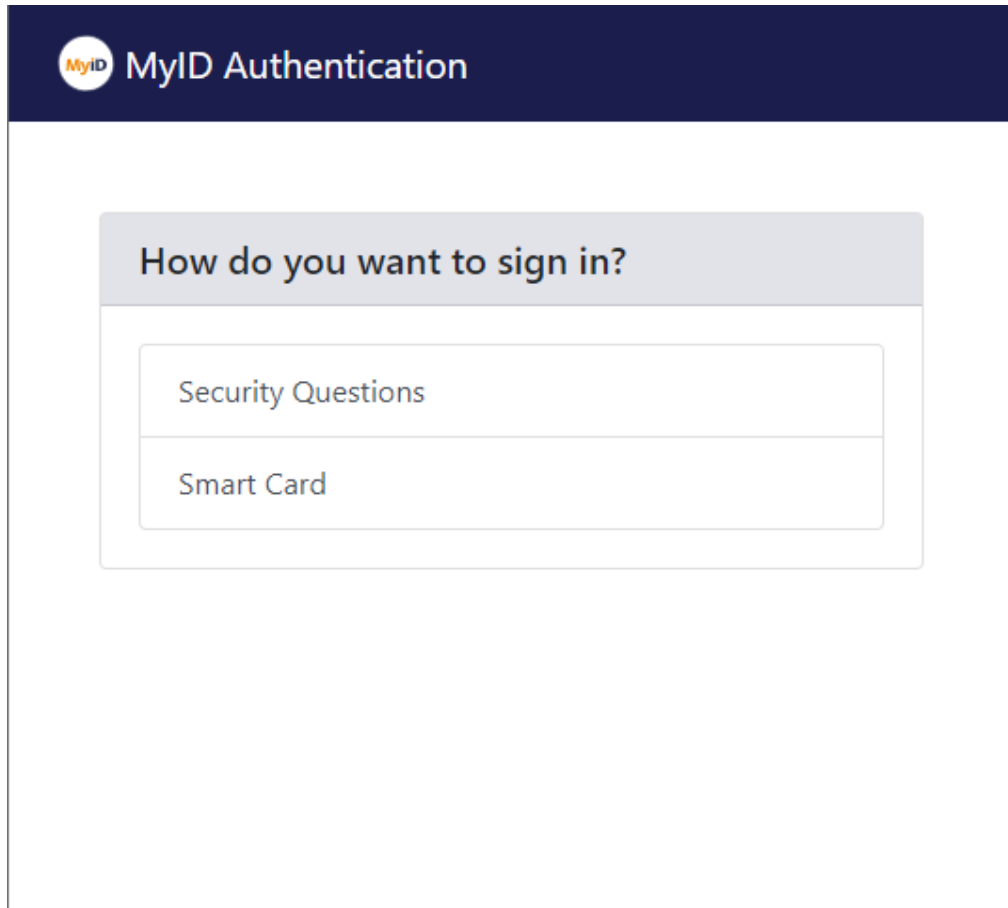
Show characters

Save Cancel

In the screenshot of the **Change Security Phrases** workflow above, the **Save** and **Cancel** buttons are blue instead of the orange specified by the theme; however, the title bar uses the setting from the theme.

### 4.1.2 Operator Client authentication

The authentication window for the MyID Operator Client does not currently support customizing the theme.



## 4.2 Setting the theme

The theme is controlled by the presence of a file called `theme.json` in the following folder on the MyID web server:

```
C:\Program Files\Intercede\MyID\OperatorClient\custom\
```

By default, this file does not exist, and the MyID Operator Client uses the default settings for the themes.

However, there are sample `theme.json` files provided in subfolders of the `custom` folder that you can use as a starting point for your own theme.

The format is as follows:

```
{
  "palette": {
    "common": {
      "black": "rgba(0, 0, 0, 1)",
      "white": "rgba(255, 255, 255, 1)"
    },
    "background": {
      "paper": "rgba(255, 255, 255, 1)",
      "default": "rgba(250, 250, 250, 1)"
    },
    "primary": {
      "light": "rgba(76, 81, 111, 1)",
      "main": "rgba(27, 29, 76, 1)",
      "dark": "rgba(0, 0, 29, 1)",
      "contrastText": "rgba(255, 255, 255, 1)"
    },
    "secondary": {
      "light": "rgba(255, 202, 71, 1)",
      "main": "rgba(255, 153, 0, 1)",
      "dark": "rgba(198, 106, 0, 1)",
      "contrastText": "rgba(255, 255, 255, 1)"
    },
    "error": {
      "light": "rgba(229, 115, 115, 1)",
      "main": "rgba(244, 67, 54, 1)",
      "dark": "rgba(211, 47, 47, 1)",
      "contrastText": "rgba(255, 255, 255, 1)"
    },
    "text": {
      "primary": "rgba(0, 0, 0, 1)",
      "secondary": "rgba(115, 115, 115, 1)",
      "disabled": "rgba(0, 0, 125, 0.38)",
      "hint": "rgba(255, 0, 0, 0.3)"
    }
  }
}
```

For example:

- `"primary": "main"` is used for the title bar at the top of the screen, category text, buttons, and the background of the self-service menu.
- `"secondary": "main"` is used for the selection markers for tabs, selection markers for report items, and for the badge displaying the number of updates on the self-service menu.
- `"text": "primary"` is used for form names, tab names, and field contents.
- `"text": "secondary"` is used for field labels.

You can edit this file in any text editor, or use a variety of online tools to generate your own Material UI theme.

Note, however, that some theme editors output RGB hex color codes; for example:

```
"main": "#8080FF"
```

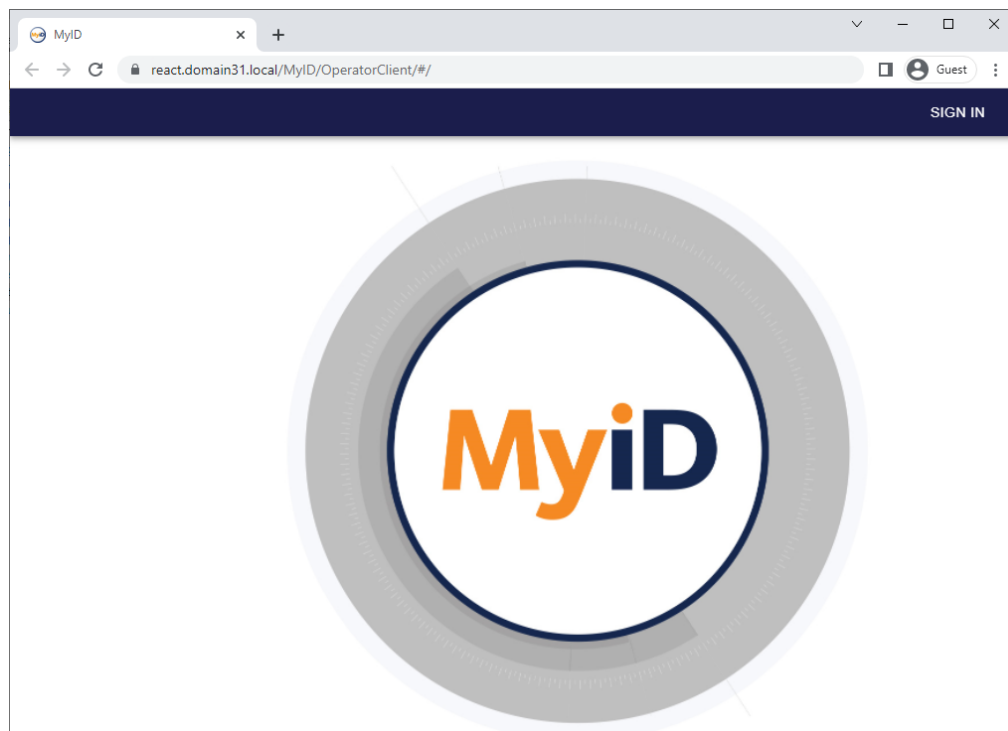
The MyID Operator Client requires RGBA color codes; for example:

```
"main": "rgba(128, 128, 255, 1)"
```

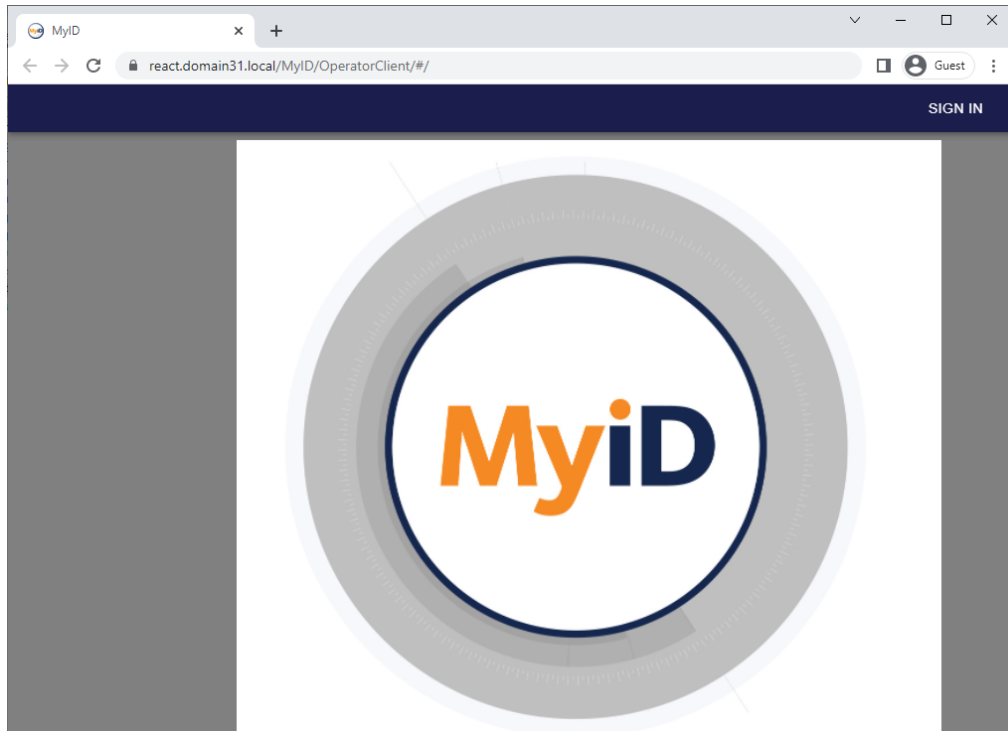
Therefore, you must check the output of any theme editing tools, and convert the RGB hex codes to RGBA if necessary.

#### 4.2.1 Changing the MyID logo

The MyID logo is designed to work on a white background color.



If you change the "background": "paper" value, the logo may look out of place:



To remedy this, you can replace the logo with an edited image.

The logo file is `Welcome-BG.jpg` and is located in the following folder:

`C:\Program Files\Intercede\MyID\OperatorClient`

Make sure you replace the logo with an image of the same dimensions and format.

## 4.2.2 Adding a banner image

The banner area at the top of the screen (containing the **Sign In** link) is by default a solid color. You can use an image instead of a solid color.

To do so, create an image file called:

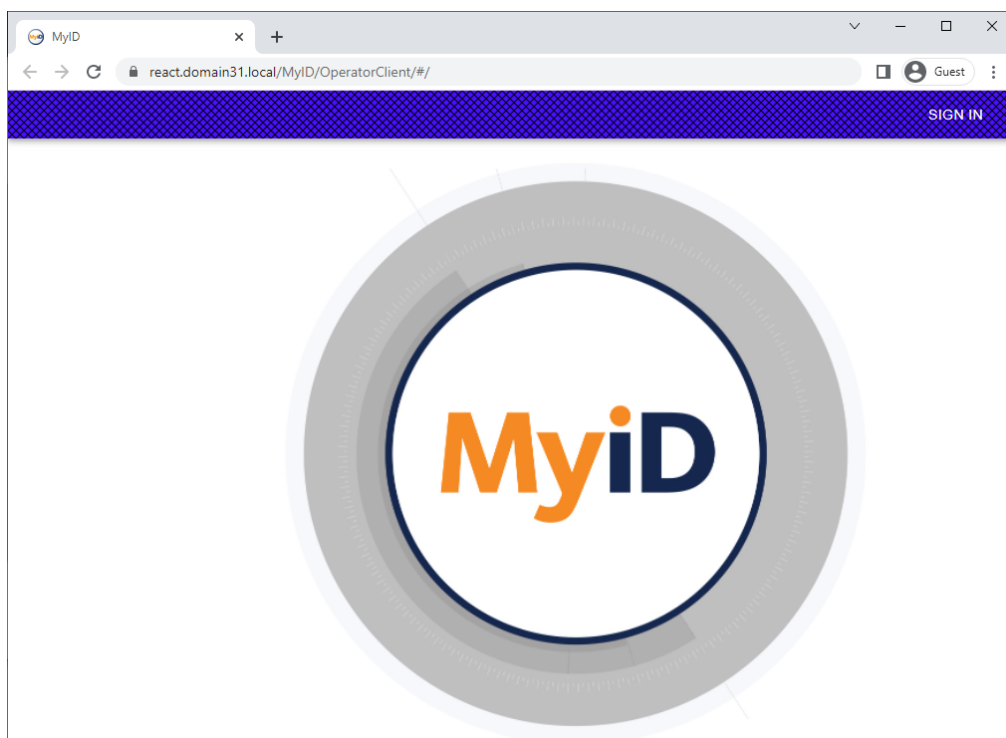
Appbar-BG.png

and place it in the following folder:

C:\Program Files\Intercede\MyID\OperatorClient

The banner area is 48 pixels high. If the image is too tall, it is cropped. If the image is too short, it is tiled vertically.

The banner area extends to the full length of the window. If the image is not wide enough, it is tiled horizontally.



You are recommended to create a banner image 48 pixels high and as wide as the widest window your clients are likely to use.



### 4.3 Troubleshooting issues with themes

If you have experience problems when attempting to apply a theme to the MyID Operator Client, try the following:

- If the MyID Operator Client does not use the theme, refresh the browser with CTRL+F5.
- Check the format of the `theme.json` file. Invalid theme files are ignored.
- Check that the color codes used are RGBA, and not RGB hex codes; that is, the theme file contains entries similar to:

```
"main": "rgba(128, 128, 255, 1)"
```

and not:

```
"main": "#8080FF"
```

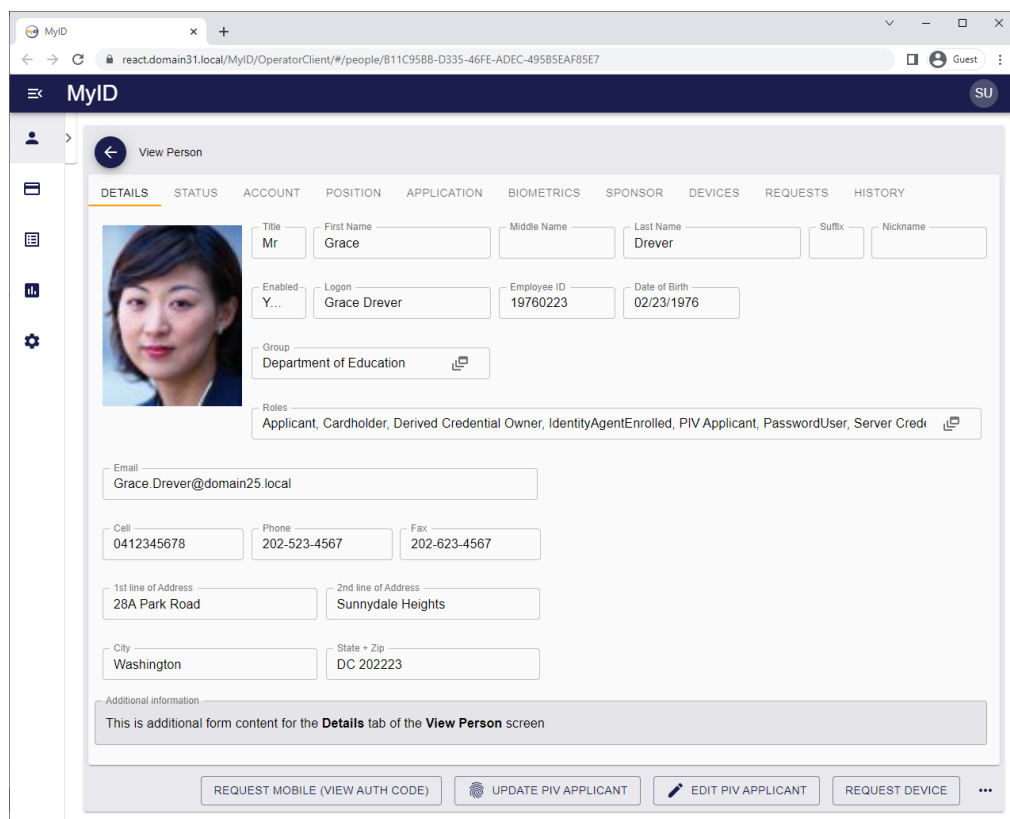
- If MyID Desktop or the Self-Service App do not use the themes applied to the MyID Operator Client, ensure that your client software is up to date, including the MyID Client Service. This feature was introduced with the client software provided with MyID 12.4:
  - MyID Desktop DSK-3.13.1000.1
  - MyID Client Service MCS-1.7.1000.1
  - Self-Service App SSP-3.13.1000.1
- The MyID Operator Client passes the theme to the MyID Client Service the first time it launches it. If you shut down the MyID Client Service then restart it, the MyID Operator Client does not pass the theme again, which means that MyID Desktop or the Self-Service App will not use the theme. Refresh the browser with CTRL+F5 and the MyID Operator Client will pass the theme to the MyID Client Service (and on to MyID Desktop or the Self-Service App) again.

## 5 Adding content to forms

You can add your own custom content to forms. This content can include HTML, CSS, and JavaScript, and is displayed in an **Additional Information** panel at the bottom of the form.

You can include any information from MyID that is available to the form (for example, details from other tabs) and you can pull additional information from external sources.

Note, however, that adding additional information to these forms does not allow you to store this information in the MyID database. If you want to add more information to MyID records, you must use Project Designer; contact your account manager for details.



## 5.1 Working with the custom hooks file

You can control the customization of the forms within the MyID Operator Client using a custom hooks file.

The hooks file does not exist by default; you must create it yourself. Alternatively, you can use the example file as a basis for your own customizations; see section [5.3, Example custom hooks file](#).

The filename is:

```
hooks.js
```

Copy the file into the following folder:

```
C:\Program Files\Intercede\MyID\OperatorClient\custom
```

### 5.1.1 The loadHook function

This JavaScript file must contain a function, the syntax of which is:

```
function loadHook(op, values)
```

where:

- `op` – the ID of the current operation.
- `values` – a block of JSON containing all the available values from the form.

The MyID Operator Client calls this function when it opens the form. Edit this function to return a block of JSON containing the HTML, CSS, and JavaScript you want to include on your customized form. You use the same function for every tab on every form; you must include the identifier for the tabs you want to customize within the function.

For example:

```
function loadHook(op, values) {
  return {
    "additionalHtml": {
      "forms.people.viewperson.details":
        "This is the <b>Details</b> tab",
      "forms.people.viewperson.status":
        "This is the <b>Status</b> tab"
    }
  };
}
```

This example returns the following JSON:

```
{
  "additionalHtml": {
    "forms.people.viewperson.details": "This is the <b>Details</b> tab",
    "forms.people.viewperson.status": "This is the <b>Status</b> tab"
  }
}
```

The MyID Operator Client uses the tab identifiers to display the content in the appropriate place.

For details of how to determine the tab identifiers you want to use, see section [5.2, Finding data references](#).

### 5.1.2 Working with values

You can use information from the values passed in to the `loadHook` function.

To determine what values are available, see section 5.2.2, *Logging data elements to the browser console*.

For example, if you wanted to add a link to the **Details** page of the View Person screen that you could click to email the person, you could add a reference to `values.Email` to the `hooks.js` file:

```
function loadHook(op, values) {
  return {
    "additionalHtml": {
      "forms.people.viewperson.details":
        "Email address: <a href='mailto:" + values.Email + "'>" + values.Email
    + "</a>"
      }
    };
}
```

**Note:** This is a simplified example. You are strongly recommended to escape any values you add to the form to prevent attacks; you can use the sample escape function provided in the example file; see section 5.3.3, *Escaping values*.

Additional information

Email address: [susan.smith@example.com](mailto:susan.smith@example.com)

### 5.1.3 Adding content

You can add HTML, CSS, and JavaScript; whatever you provide in the return JSON is used as the content for a `div` within the **Additional Information** area of the appropriate tab of the appropriate form. This content can also access any additional JavaScript functions that you include in the `hooks.js` file.

For example:

```
function loadHook(op, values) {
  return {
    "additionalHtml": {
      "forms.people.viewperson.details":
        "<style type='text/css'> .Label {font-weight: bold;} </style>" +
        "This is the <span class='Label' onclick=popupAlert();>Details</span>
tab",
      "forms.people.viewperson.status":
        "This is the <b>Status</b> tab"
    }
  };
}

function popupAlert() {
  alert('This is a test.');
```

which produces the following content on the form:

```
<div id="forms.people.viewperson.details"><style type="text/css"> .Label
{font-weight: bold;} </style>This is the
<span class="Label" onclick="popupAlert();">Details</span> tab</div>
```

This produces the following display:

Additional information

This is the **Details** tab

When you click the **Details** text, the `onclick` event calls the `popupAlert()` function.

react.domain31.local says

This is a test.

OK

## 5.2 Finding data references

To work with the custom hooks file, you must know the reference for the tab on the form you want to customize.

**Note:** You can customize data form tabs (that is, tabs with data entry forms; for example, the **Details** tab); you cannot customize report tabs (that is, tabs with a list of items; for example, the **Devices** tab).

For example:

| Screen             | Tab         | Reference                                 |
|--------------------|-------------|---|
| View Request       | Request     | forms.requests.viewrequest.request        |
| View Audit         | Audit       | forms.audits.viewaudit.audit              |
| View Device        | Details     | forms.devices.viewdevice.details          |
| View Person        | Details     | forms.people.viewperson.details           |
|                    | Status      | forms.people.viewperson.status            |
|                    | Account     | forms.people.viewperson.account           |
|                    | Position    | forms.people.viewperson.position          |
|                    | Application | forms.people.viewperson.application       |
|                    | Biometrics  | forms.people.viewperson.biometrics        |
|                    | Sponsor     | forms.people.viewperson.sponsor           |
| Edit Person        | Details     | forms.people.editperson.details           |
|                    | Account     | forms.people.editperson.account           |
| Edit PIV Applicant | Details     | forms.people.editpivapplicant.details     |
|                    | Status      | forms.people.editpivapplicant.status      |
|                    | Account     | forms.people.editpivapplicant.account     |
|                    | Position    | forms.people.editpivapplicant.position    |
|                    | Sponsor     | forms.people.editpivapplicant.sponsor     |
|                    | Application | forms.people.editpivapplicant.application |
|                    | Biometrics  | forms.people.editpivapplicant.biometrics  |

The above table includes screens and tabs that are available only on PIV systems, and is an example only.

This section provides instructions for finding the references for the tabs on your system.

If you want to access any data from the form, you must also know the reference for the data element; for example:

| Screen      | Tab     | Field       | Reference |
|-------------|---------|-------------|-----------|
| View Person | Details | Title       | Title     |
|             |         | First Name  | FirstName |
|             |         | Middle Name | Initial   |
|             |         | Last Name   | Surname   |
|             |         | Suffix      | XuPIV4    |
|             |         | Nickname    | Xu49      |

The above table includes fields that are available only on PIV systems, and is an example only.

This section provides instructions for finding the references for the fields on your system.



### 5.2.1 Viewing the response from the API

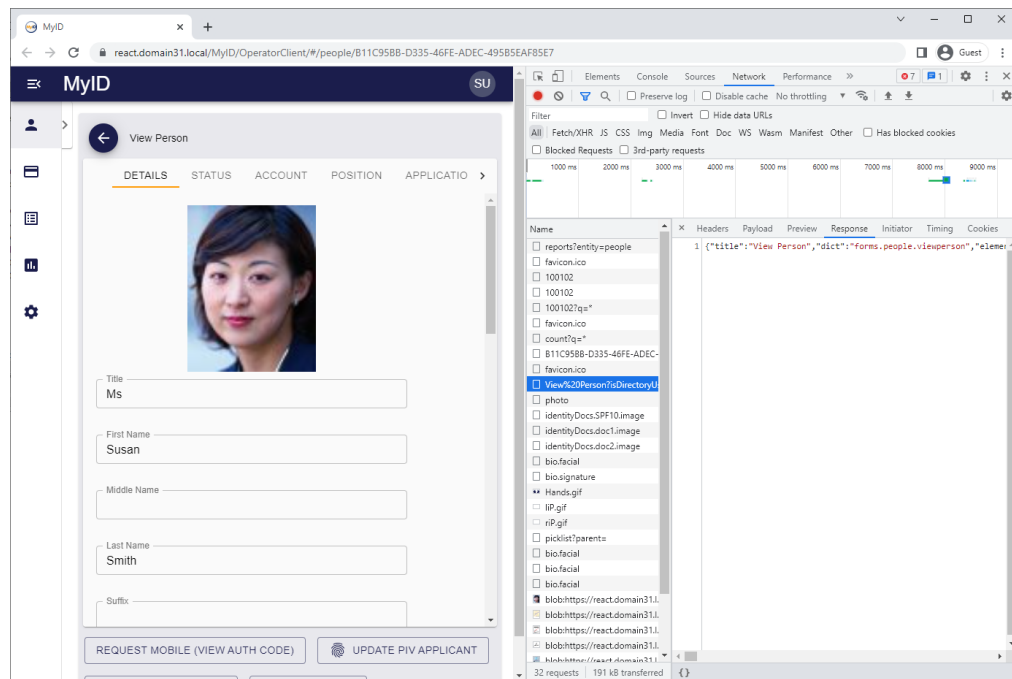
You can obtain a JSON data structure containing all the information you need by viewing the response to the forms API call; for example, for View Person:

```
rest.core/api/forms/View%20Person
```

1. Open a MyID Operator Client window in your browser.
2. Open the browser's network log.  
For example, in Chrome, press F12, then select the **Network** tab.
3. Log in to the MyID Operator Client.

**Note:** The response from the API is returned the first time you open the form, so you are recommended to start a new session.

4. Open the form you want to customize.  
For example, search for a person, then open the View Person form.
5. In the network log, locate the `/api/forms` call to the API, then open the Response.



The response is a block of JSON that contains all of the elements available on the form.

You are recommended to use a JSON formatter to make the data easier to read.

To find the ID of a tab, look for the relevant `dict` attribute. For example:

```
"label": "Details",
"dict": "forms.people.viewperson.details",
```

This shows that the ID of the **Details** tab on the View Person form is:

```
forms.people.viewperson.details
```

To find the ID of a field on the form, look for the relevant `name` attribute. For example:

```
"field":{
  "name":"Email",
  "component":"inputBox",
  "type":"text",
  "label":"Email",
  ...
}
```

This shows that the ID of the **Email** field on the form is:

Email

You can access this in the custom area using:

```
values.Email
```

**Note:** Extended attributes use codes such as `Xu32` which may not be immediately obvious. The `label` element helps you match the reference to the on-screen display. You can also search the *Lifecycle API* guide in the MyID documentation set for assistance, particularly for PIV attributes. In addition, some extended attributes may have been added during the customization of your system.

## 5.2.2 Logging data elements to the browser console

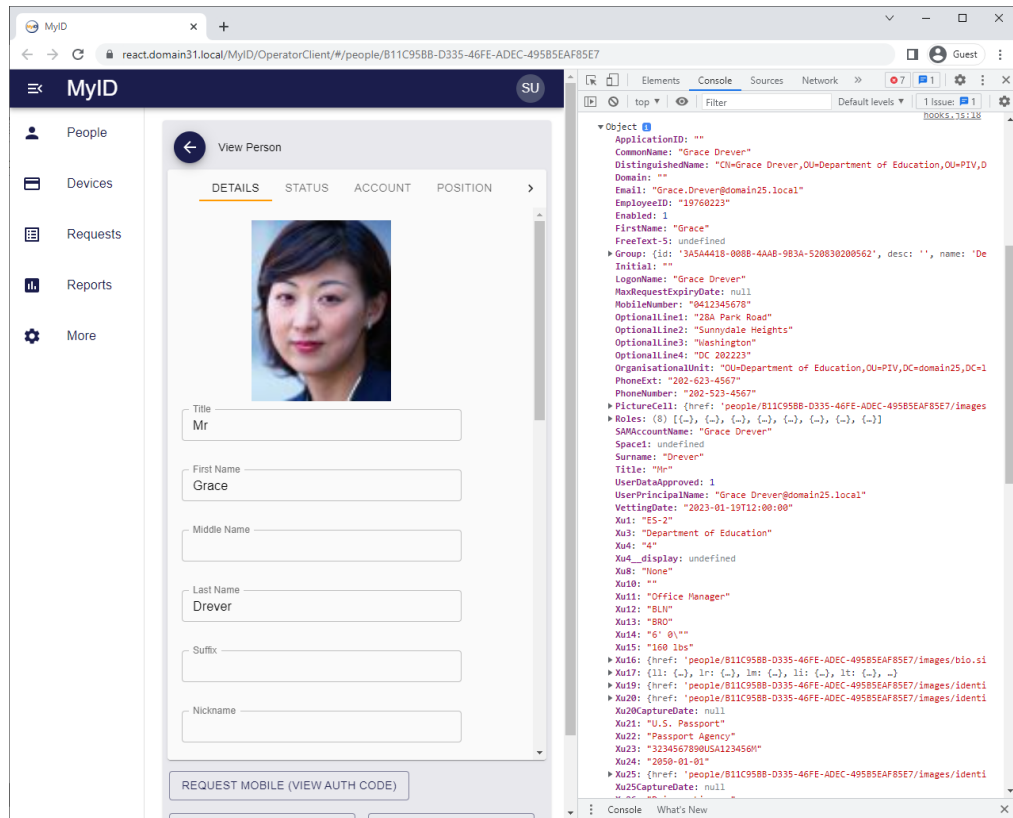
To determine what data references are available, you can also write the data elements to the browser console. This method does not provide the references for the tabs, but does provide a simpler list of the data elements, and shows you the current values for each of those data elements.

1. Create a `hooks.js` file with the following content:

```
function loadHook(op, values) {  
  console.log(values);  
}
```

**Note:** You can add the console statement to a `hooks.js` file that already contains additional statements; make sure the console statement is the first line within the `loadHook` function.

2. Copy this file into the following folder on the MyID web server:  
C:\Program Files\Intercede\MyID\OperatorClient\custom
3. Open a MyID Operator Client window in your browser, and navigate to the screen you want to customize.  
For example, open the View Person screen.
4. Open the browser console.  
For example, in Chrome, press F12, then select the **Console** tab.



The console lists all of the data elements available on the form and their current values.

## 5.3 Example custom hooks file

The MyID Integration Toolkit contains a sample `hooks.js` file that you can use as the basis of your own customizations. The MyID Integration Toolkit is available on the Intercede Customer Portal:

[forums.intercede.com/documentation/](https://forums.intercede.com/documentation/)

### 5.3.1 Cascading style sheet

The sample file contains the following function that allows you to add a simple style sheet to your customizations:

```
addStyle()
```

### 5.3.2 Adding elements

The sample file contains several functions that make it easy to add elements to your customizations. For example:

```
addTextBoxWith({ id, value, disabled })
```

This function adds a text box with a particular `id` and `value`, and can be enabled or disabled.

### 5.3.3 Escaping values

The `escapeXML` function in the example file prevents special characters from being incorporated in your custom area and potentially providing a vector to attack your system. You are recommended to use this function to escape any values you include.

The function escapes the following characters:

```
& < > ' "
```

For example:

```
function loadHook(op, values) {
  return {
    additionalHtml: {
      "forms.people.viewperson.details":
        "Email address: <a href='mailto:" + escapeXML(values.Email) + "'>" +
        escapeXML(values.Email) + "</a>"
    }
  };
}

function escapeXML(input) {
  if (input) {
    return input
      .replace(/&(?!#|amp;)/g, "&amp;")
      .replace(/</g, "&lt;")
      .replace(/>/g, "&gt;")
      .replace(/'/g, "&apos;")
      .replace(/"/g, "&quot;");
  }
  return "";
}
```