iAccess for Maconomy

Install Guide
2018

Edited by

Anders Hessellund
Peter Enevoldsen

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This edition published September 2018.

Contents

Revision History

Date	Author	Notes
2017-08-11	AH	First draft of Extension documentation, aligned with iAccess 2.0.5
2017-03-10	AH	Aligned with iAccess 2.0
2016-11-10	CC	Included additional SPNEGO SSO browser setup instructions.
2016-11-10	AH	API changed from 2.0.0 to 3.0.0. Compatibility narrowed to 2.3GA. Updated Migration Guide and Troubleshooting Section.
2016-08-17	AH	Merged “Install Guide” and “Extension Manual” into a single document.
2016-02-19	PE	Extended the upgrade path with relation to removal of default preferences.
2016-02-07	AH	Aligned documentation with iAccess 1.3.
2016-01-20	AH	Aligned with both iAccess version 1.2.0 and 1.2.1.
2015-12-28	AH	First draft of iAccess 1.2 documentation. Installation notes not included yet.

Chapter 1

Introduction

The following document serves as an introduction to the iAccess for Maconomy product. The target audience is technical consultants and partners that need to install, extend, and maintain iAccess. In the first part, we will describe core concepts of the iAccess architecture. The second part describes how it can be installed using MConfig. The third part describes how it can be extended using the Maconomy Extender. In the fourth part, we will provide an overview of the current extension facilities. This part can be used as reference when working with iAccess extensions. Finally, the fifth part contains a migration guide, and a troubleshooting guide. Both of these guides should be particularly useful when upgrading an existing iAccess installation.

We also refer the reader to our Kona space, *iAccess for Maconomy*, where Product Management, and the Development Team will answer questions, and discuss feature requests for the product.

Chapter 2

iAccess Architecture

Briefly described, iAccess for Maconomy is an HTML5 web client. It is a lightweight user interface supplement to the existing Workspace Client. The backend is Maconomy, specifically the new RESTful web services exposed from Maconomy version 2.2 [3]. In this section, we will give a cursory overview of the technical architecture.

2.1 Technical Architecture

Figure 13 shows the high-level architecture of a Maconomy system with iAccess. This setup resembles the traditional Maconomy architecture with a few exceptions. In the following section, we will describe the core components involved and the purpose of each of those.

Maconomy 2.x server and database iAccess for Maconomy is available from Maconomy 2.2. It does not impose any specific requirements on the database, but it

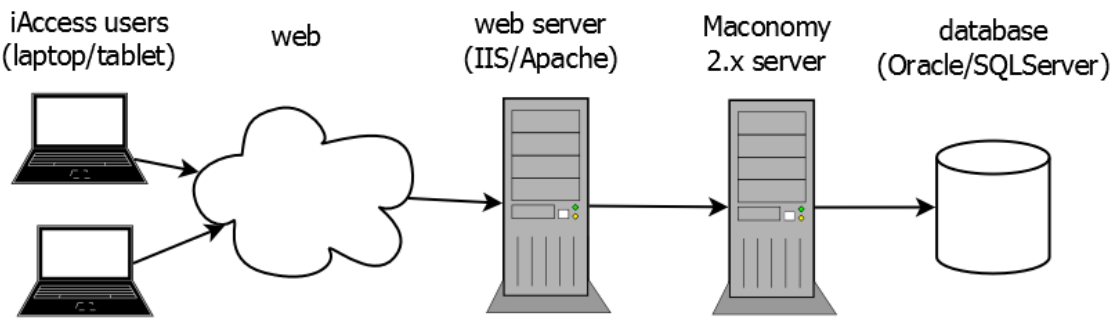

Figure 13: Architectural overview.

does require a Maconomy 2.2 server and corresponding RESTful API [3]. See the following section for more details on the required web services.

Web server (IIS/Apache) One or more web servers are required to serve both the static and dynamic content of iAccess. Static content such as HTML, JavaScript, CSS files, and so on are placed directly on the web server. Dynamic content such as specifications, files, and data are retrieved from the Maconomy server, but the web server in this case acts as proxy. Using the web server as proxy prevents cross-origin (CORS) issues on the client side. The web server is also required for encryption and compression of client-web server communication.

iAccess Clients The iAccess clients can be located both on the internal network or on the open Internet depending on the web server configuration and exposure. Furthermore, clients can run iAccess on different devices such as laptops with the main browsers (IE, Chrome, and Safari), as well as on iOS and Android tablets.

2.1.1 RESTful Web Service API

As mentioned previously, iAccess for Maconomy uses the RESTful Web Service API which was introduced in Maconomy 2.2. This is *not* the same thing as the existing MScript web services. Please see the RESTful Web Services documentation for more information [3]. For now, we will just briefly list the web service endpoints that iAccess uses and what they are used for:

/containers The *containers* endpoint delivers both metadata and data for the containers exposed by the Maconomy 2x server. Metadata include specifications of the names, actions, fields, and foreign keys exposed by different containers. Data include the actual filter-, card-, and table-data stored in the underlying database as well as information on which actions are enabled.

/filedrop The *filedrop* endpoint is used to upload files such as receipt attachments on expense sheets.

/configurations The *configurations* endpoint was introduced in Maconomy 2.2.2 and is used by iAccess 1.2 and onwards. This endpoint is used to retrieve JSON specifications from the Maconomy server, specifically the application specification (`application.json`) which configures iAccess. These specifications are the foundation of the iAccess extensibility model.

/auth The *auth* endpoint was introduced in Maconomy 2.2.4 and 2.3GA, and is used by iAccess 2 and onwards. This endpoint is used to obtain login tokens for 3rd party integrations such as Business Objects.

/environment The *environment* endpoint was introduced in Maconomy 2.2.5 and 2.3GA and is used by iAccess 2 and onwards. This endpoint is used to retrieve the end-user's environment variable, e.g., employee name and number, company info etc.

Chapter 3

Installing iAccess

This section describes the installation process for iAccess. Keep in mind that parts of the installation process (in particular, web server configuration) are specific to the individual installation. As such, this section can only offer general guidelines. In case of doubt, we recommend posting a question on our iAccess Kona Space.

3.1 The iAccess Manifest

To install iAccess, you need an installation of Maconomy and a suitable version of MConfig. The iAccess product is packaged in an archive file format called an FPU (*Flexible Packaging Unit*). An FPU contains the following components:

1. The Tools part which is a set of JavaScript, HTML and CSS files responsible for executing the iAccess Applications in the browser.
2. The embedded iAccess Applications each consisting of a set of JSON files describing workspaces, layouts, and other settings for iAccess against a particular range of Maconomy backends
3. The FPU manifest (`manifest.json`) which specifies various metadata about the FPU such as version info and compatibility constraints.

The required backend Maconomy version for a given iAccess is documented in the release notes for each iAccess release. You can also inspect the `manifest.json` file mentioned above. In the following `manifest.json` sample, we can see that this iAccess FPU is of version 2.0.5. The applications section lists the core Maconomy versions that this version of iAccess is compatible with. In this case, it is compatible with specifics service packs on 2.2 (internally called `17`), 2.3 (internally called `19`), and 20 (internally called `20`). Below, the metadata for the 2.2/17 application is expanded and it can be seen that iAccess 2.0.5 is specifically only compatible with service pack 5 of Maconomy 2.2 (internally called `17.0.105`).

```
{
  "manifest-version": 3,
  "marketingVersion": "iAccess for Maconomy 2.0",
  "fpuVersion": "2.0.5",
  "apiVersion": "8.0.0",
  "applications": {
    "17": {
      "requires": "^1.0.0",
      "src": "...",
      "backends": [
        {
          "tpu": { "versionRange": "17.0.105" },
          "apu": { "versionRange": "17.0.105" }
        }
      ]
    },
    "19": { ... },
    "20": { ... }
  }
}
```

Once the Maconomy system is installed and configured, MConfig can install the iAccess FPU on a web server of choice. MConfig will also install an iAccess Application matching the Maconomy backend on the Maconomy server, specifically in the `Web` folder in `MaconomyDir`. We currently support IIS and Apache (see release notes for the specific version requirements for these web server products). On the web server, you should also set up a web site with the installed iAccess `index.html` in the root. Once the MConfig installation has taken place, and the web site and proxy settings have been completed, iAccess is ready. Please observe that certain manual steps may be required for specific installations.

3.2 Security Considerations

While Deltek recommends the following procedures, ultimately each company is liable for its own security. The landscape evolves quickly, and each customer should continuously take internal measures to ensure its own security.

3.2.1 Regarding the use of HTTPS/TLS

Deltek best practice recommends that you configure web servers to use HTTPS (instead of HTTP). Using HTTPS/TLS encrypts your network traffic, making it difficult for anyone to access the credentials as they are passed to the web server. Using simple HTTP is tantamount to sending confidential information over the wire in clear text. The iAccess login page will display a warning message in case HTTP is used instead of HTTPS.

Figure 14: The Application Instance window.

3.2.2 Address Risk of Clickjacking

To reduce the likelihood of clickjacking, Deltek suggests you follow the OWASP guidelines to defend against clickjacking attacks. Based on the OWASP guideline, you can perform additional steps when configuring your webserver. See Additional Related Procedures for more details.

3.3 Prerequisites

The following are prerequisites to installing iAccess:

- Maconomy 2.2.6
- MConfig 8.17.2
- Extender 1.8.1
- RESTful Web Services is enabled in the Coupling Service
- iAccess downloaded from DSM, and iAccess FPU placed in the PUs folder (with the APU and TPU)
- If you are using Apache as the webserver, download the Apache binary package including OpenSSL, and install it from the following link: http://httpd.apache.org/
- Standard extensions are already installed

Additionally, this document assumes that you have already set up an application. For detailed instructions on setting up applications, see the Deltek Maconomy Installation Guide for your specific Maconomy version.

3.4 MConfig Installation

To begin installation with MConfig, complete the following steps:

Step 1 In the MConfig Main Window, double-click the application to open. The Application Instance window displays as shown in Figure 14.

Step 2 Click OSGi products. The OSGi Server Selection screen appears as shown in Figure 15.

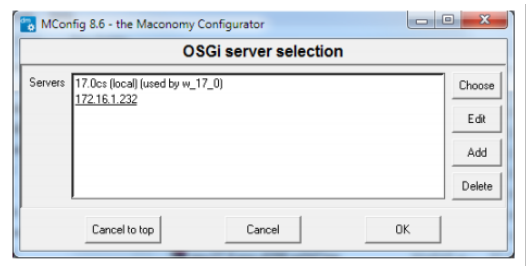

Figure 15: The OSGi Server Selection window.

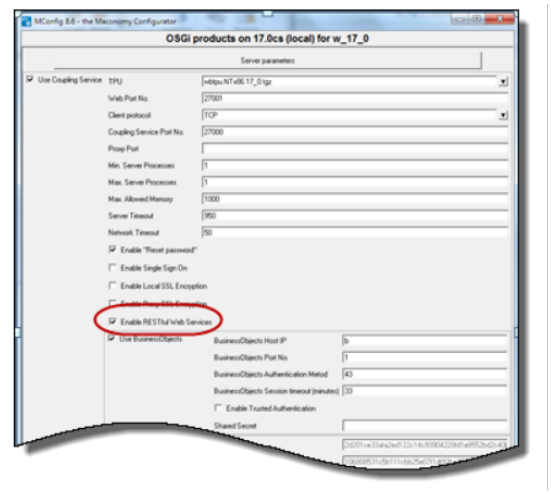

Figure 16: Coupling service selection.

Step 3 Select the Coupling Service to update as shown in Figure 16.

Step 4 Select the Enable RESTful Web Services check box as shown in Figure 17.

Step 5 Click OK to save, and click OK at the SSL warning to return to the Application Instance window. In the Application Instance window, click Web products as shown in Figure 18.

Note: While you can click OK at the SSL warning, Deltek recommends you follow the steps listed in the warning to ensure the security of your system.

Step 6 On the Web server selection screen, select the application to update. Select the iAccess check box as shown Figure 19. In the iAccess FPU field, select the relevant FPU from the drop-down list.

Step 7 Click Ok a couple of times to return to the main window, and click Next a couple of times, and then click Yes to complete the MConfig installation.

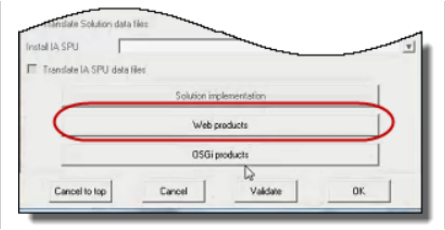

Figure 17: Enable RESTful Web Services.

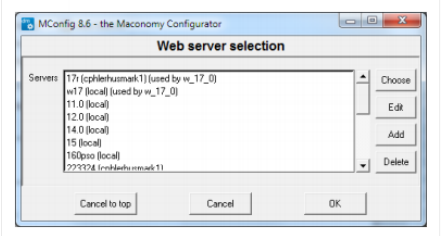

Figure 18: Select web products.

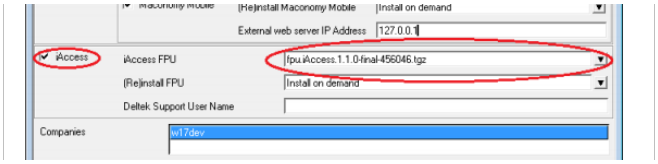

Figure 19: Web products window.

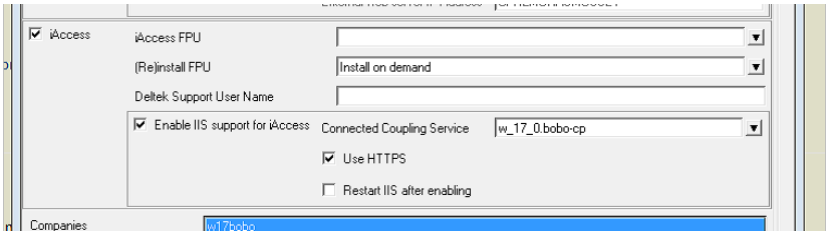

Figure 20: Enable IIS support using MConfig

3.5 Create a Website Using IIS

To create a website using IIS, you can enable IIS support automatically using MConfig, or perform the steps manually using IIS Manager.

Enabling IIS support with MConfig automatically completes the steps described under the manual installation. Use MConfig for the initial setup of an iAccess website using IIS. However, modifying the setup later should be done manually using IIS Manager.

3.5.1 Enable IIS Support Automatically Using MConfig

Automatic IIS configuration requires MConfig 8.12.4. Previous version will not perform a correct configuration of IIS due to a shortcoming in MConfig. To enable IIS support using MConfig, follow these steps (See Figure 20):

1. In MConfig, go to the Web Products window.
2. Select the Enable IIS support for iAccess check box and click OK.

Note: After you complete the initial installation with MConfig, you should check the setup in IIS Manager and possibly modify parameters, such as Web Server Port Number.

Note: If you enable IIS support automatically, MConfig also updates the IIS web.xml file with a routing rule that ensures login pages and other non-root URLs load properly.

3.5.2 Enable IIS Support Manually Using IIS Manager

After iAccess has been installed on your IIS web server, you can configure it using IIS Manager.

Add the Site

: Connect to your server in the Internet Information Services (IIS) Manager application and setup the iAccess site. The site should have the files shown in Figure 21 as root files.

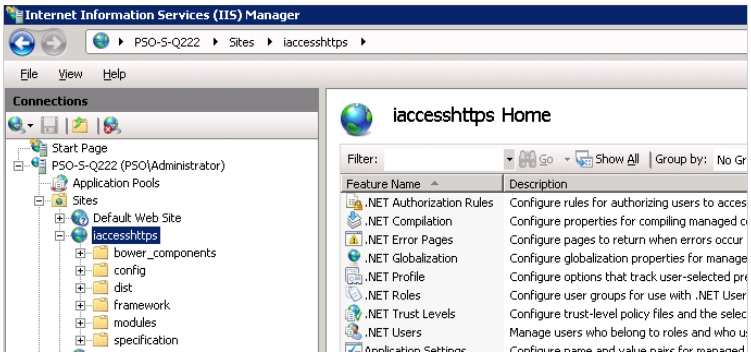

Figure 21: Add Site

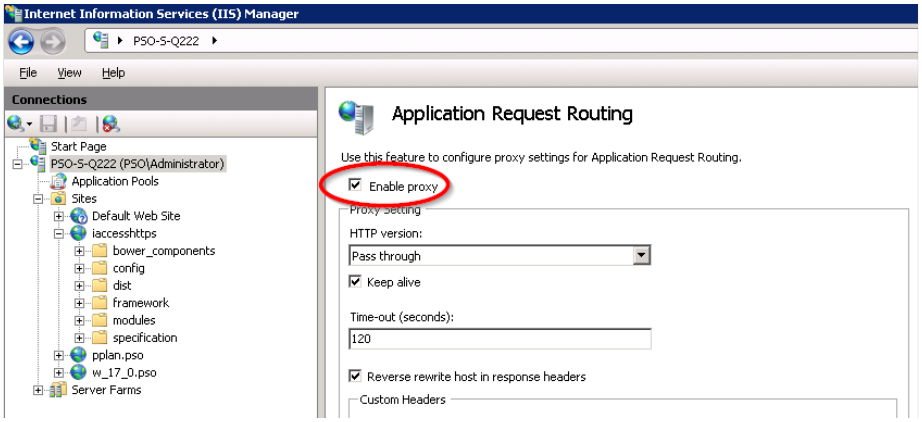

Figure 22: Enable Proxy

Add MIME Types

: Click the "MIME Types" and ensure that the MIME Types below are defined

```
.json application/json
.woff application/font-woff
.woff2 application/font-woff
```

In IIS 8.0 and up, the .woff extension exists by default but with a different type. Change it to *application/font-woff*.

Proxy Setup

1. Install Microsoft Application Request Routing for IIS ARR.
2. Restart IIS Manager.
3. In the *Application Request Routing* configuration, click *server proxy settings*.
4. Check *Enable proxy* as shown in Figure 22.

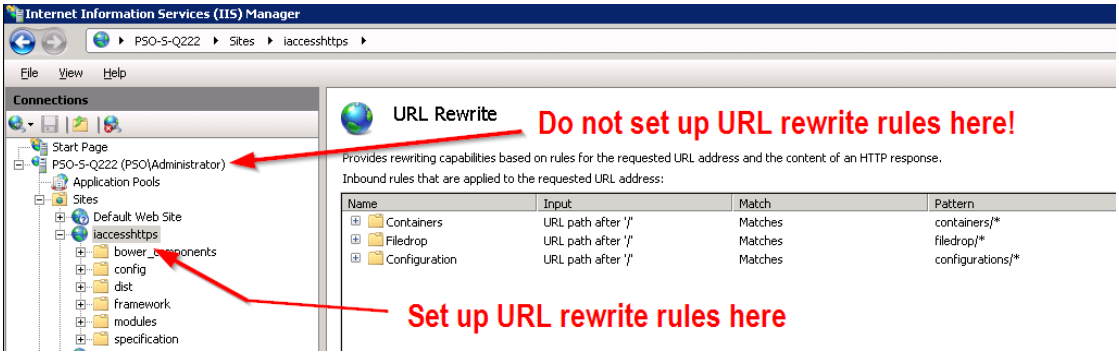

Figure 23: Add Proxy Rules

5. Open *URL Rewrite* to add proxy rules for the container, configurations and filedrop APIs. Note: This must be done on the local site, not globally as shown in Figure 23.

In IIS 8.0 and up, you will need to install the Web Platform Installer in order to install the ARR plugin.

Edit Routing Rules

To ensure that login pages (and other non-root URLs) load properly, open the IIS web.xml file and add the following rule *before* the other routing rules:

```
<rule name="DeepLinkingSupport" stopProcessing="false">
  <match url=".*" />
    <conditions logicalGrouping="MatchAll" trackAllCaptures="false">
      <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
      <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true"  ↩
    />
      <add input="{REQUEST_FILENAME}" pattern="containers*" negate="true" />
      <add input="{REQUEST_FILENAME}" pattern="filedrop*" negate="true" />
      <add input="{REQUEST_FILENAME}" pattern="configurations*" negate="true ↩
    " />
      <add input="{REQUEST_FILENAME}" pattern="auth*" negate="true" />
      <add input="{REQUEST_FILENAME}" pattern="environment*" negate="true"  ↩
    />
    </conditions>
  <action type="Rewrite" url="/" />
</rule>
```

Set up proxy for Container API

1. Click Add Rule...
2. Select Blank rule.

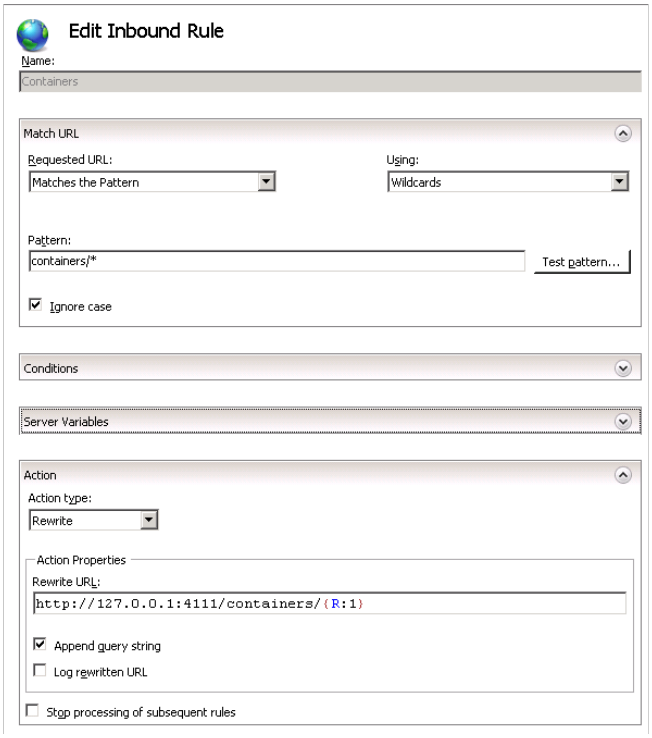

Figure 24: Container API

3. Fill out the rule as shown in Figure 24.

Make sure to choose the Coupling Service webport when setting this up, for example, 4111 in Figure 24. The host should be the ip or hostname of the coupling service, not necessarily 127.0.0.1. Here is an overview of the required parameters for the rule.

The rewrite URL *MUST* be HTTP rather than HTTPS. Otherwise the rewriting of response URLs will not work, and iAccess will not be able to make it past the login screen.

Match URL

```
Requested URL: Matches the Pattern
Using: Wilcards
Pattern: containers/*
Ignore case: checked
```

Action

```
Action type: Rewrite
Rewrite URL: http://<coupling-service-host>:<coupling-service-port>/ ↩
    containers/{R:1}
Append query string: checked
```

Set up proxy for Configurations API

1. Click Add Rule...
2. Select Blank rule.
3. Fill out the rule as shown in the section on setting up proxy for Containers, specifically with the following parameters:

Match URL

```
Requested URL: Matches the Pattern
Using: Wilcards
Pattern: configurations/*
Ignore case: checked
```

Action

```
Action type: Rewrite
Rewrite URL: http://<coupling-service-host>:<coupling-service-port>/ ↩
    configurations/{R:1}
Append query string: checked
```

Set up proxy for Filedrop API

1. Click Add Rule...
2. Select Blank rule.
3. Fill out the rule as shown in the section on setting up proxy for Containers, specifically with the following parameters:

Match URL

```
Requested URL: Matches the Pattern
Using: Wilcards
Pattern: filedrop/*
Ignore case: checked
```

Action

```
Action type: Rewrite
Rewrite URL: http://<coupling-service-host>:<coupling-service-port>/filedrop ↩
    /{R:1}
Append query string: checked
```

Set up proxy for Auth API

1. Click Add Rule...
2. Select Blank rule.
3. Fill out the rule as shown in the section on setting up proxy for Containers, specifically with the following parameters:

Match URL

```
Requested URL: Matches the Pattern
Using: Wilcards
Pattern: auth/*
Ignore case: checked
```

Action

```
Action type: Rewrite
Rewrite URL: http://<coupling-service-host>:<coupling-service-port>/auth/{R ↩
    :1}
Append query string: checked
```

Set up proxy for Environment API

1. Click Add Rule...
2. Select Blank rule.
3. Fill out the rule as shown in the section on setting up proxy for Containers, specifically with the following parameters:

Match URL

```
Requested URL: Matches the Pattern
Using: Wilcards
Pattern: environment/*
Ignore case: checked
```

Action

```
Action type: Rewrite
Rewrite URL: http://<coupling-service-host>:<coupling-service-port>/ ↩
    environment/{R:1}
Append query string: checked
```

Preserve the Host Header

Open a console with Administrative privileges, and navigate to

```
C:\Windows\System32\inetsrv
```

Enable preserveHostHeader by running the following command:

```
cd C:\Windows\System32\inetsrv
appcmd.exe set config -section:system.webServer/proxy /preserveHostHeader:" ↩
    True" /commit:apphost
```

Note: To preserve the spacing, copy the command and paste it in the command prompt.

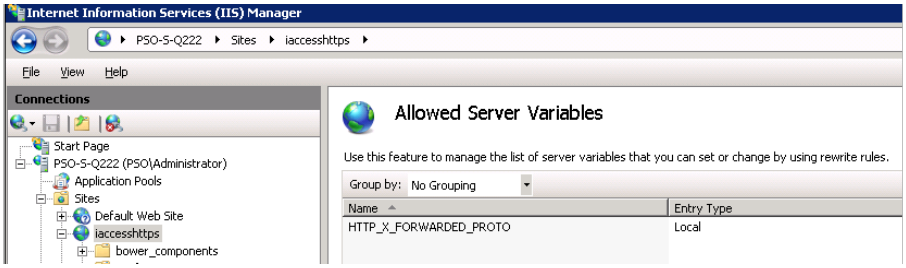

Figure 27: Add Server Variables

Restart the web server.

See *AppCmd* reference for more details.

3.5.3 Set Up HTTPS

Open the Server Variables screen by clicking *View Server Variables...* in the *URL Rewrite* screen.

In the Server Variables screen, click *Add...* and add the variable `HTTP_X_FORWARDED_PROTO` as shown in Figure 27.

In the URL rewrite rules (both containers, configurations, filedrop, auth, and environment) that proxies the web service, set the server variable `HTTP_X_FORWARDED_PROTO` to `https` as shown in Figure 28.

Restart the webserver.

Note: It is not possible to run both HTTP and HTTPS on the same IIS site.

3.6 Create a Website Using Apache

Here is a short guide to setting up iAccess on Apache (2.2 and 2.4)

3.6.1 Download Apache

Download the Apache 2.2 binary package including OpenSSL. Install it.

In `httpd.conf`, comment in the following modules:

```
LoadModule headers_module modules/mod_headers.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule ssl_module modules/mod_ssl.so
LoadModule rewrite_module modules/mod_rewrite.so
```

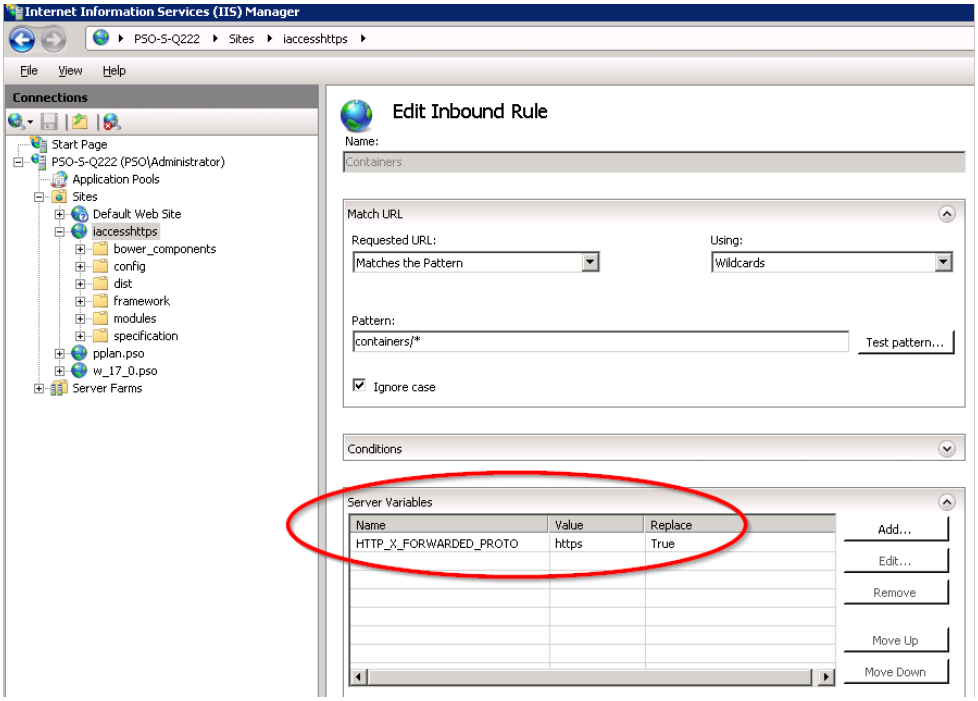

Figure 28: Setting up HTTPS

Comment in the inclusion: `Include conf/extra/httpd-vhosts.conf`

Comment out `#Listen 80` (we will use the `httpd-vhosts.conf` file instead)

3.6.2 Enable compression

If using Apache 2.2, comment in the following module: `LoadModule deflate_module modules/mod_deflate.so`

If using Apache 2.4, use this module: `LoadModule filter_module modules/mod_filter.so`

Configure the compression:

```
<IfModule deflate_module>
  AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css text/ ↩
    javascript   application/javascript
  SetOutputFilter DEFLATE
  DeflateCompressionLevel 5
</IfModule>
```

See Apache Deflation Module for more information.

3.6.3 Setup with SSL

Here is a template for setting up a virtual host that serves iAccess *with* SSL. Copy contents into the `httpd-vhosts.conf` file, and replace the variables with the desired values. Please observe that the line `Require all granted` below is only required for Apache 2.4.

```
Listen <port>
<VirtualHost *:<port>>
    ServerName <server-name>

    # Server iAccess files from installation directory
    DocumentRoot "<iAccess-installation-directory>"

    <Directory <iAccess-installation-directory>>
       Order deny,allow
       Allow from all
       AllowOverride All
       Require all granted
    </Directory>

    <Proxy *>
       Order deny,allow
       Allow from all
    </Proxy>

    ProxyRequests      Off
    ProxyPreserveHost  On
    # Signal to the coupling service that the originating protocol is HTTPS
    RequestHeader set X-Forwarded-Proto "https"

    # Proxy the web services from the coupling service
    ProxyPass  /containers                  http://<coupling-service-host>:< ↩
    coupling-service-web-port>/containers                retry=0
    ProxyPass  /filedrop                    http://<coupling-service-host>:< ↩
    coupling-service-web-port>/filedrop                  retry=0
    ProxyPass  /configurations              http://<coupling-service-host>:< ↩
    coupling-service-web-port>/configurations            retry=0
    ProxyPass  /auth                        http://<coupling-service-host>:< ↩
    coupling-service-web-port>/auth                      retry=0
    ProxyPass  /environment                 http://<coupling-service-host>:< ↩
    coupling-service-web-port>/environment               retry=0

    # Set up this virtual host to use SSL
    SSLEngine              On
    SSLProxyEngine         On
    SSLCertificateFile     <crt-file-location>
    SSLCertificateKeyFile  <key-file-location>
</VirtualHost>
```

Here is an example using the preceding template:

```
Listen 443
<VirtualHost *:443>
    ServerName techwebproject

    # Server iAccess files from installation directory
    DocumentRoot "C:/Maconomy/Webservers/w_20_0.101/htdocs/Maconomy/iAccess/ ↩
    w_20_0.101.w20/app"

    <Directory C:/Maconomy/Webservers/w_20_0.101/htdocs/Maconomy/iAccess/ ↩
    w_20_0.101.w20/app>
       Order deny,allow
       Allow from all
       AllowOverride All
       Require all granted
    </Directory>

    <Proxy *>
       Order deny,allow
       Allow from all
    </Proxy>

    ProxyRequests      Off
    ProxyPreserveHost  On
    # Signal to the coupling service that the originating protocol is HTTPS
    RequestHeader set X-Forwarded-Proto "https"

    # Proxy the web services from the coupling service
    ProxyPass  /containers              http://localhost:8085/containers    ↩
               retry=0
    ProxyPass  /filedrop                http://localhost:8085/filedrop      ↩
               retry=0
    ProxyPass  /configurations          http://localhost:8085/ ↩
    configurations          retry=0
    ProxyPass  /auth                    http://localhost:8085/auth          ↩
               retry=0
    ProxyPass  /environment             http://localhost:8085/environment   ↩
               retry=0

    # Set up this virtual host to use SSL
    SSLEngine              On
    SSLProxyEngine         On
    SSLCertificateFile     c:/sslkeys/server.crt
    SSLCertificateKeyFile  c:/sslkeys/server.key
</VirtualHost>
```

3.6.4 Edit Routing Rules

For more information, refer to Edit Routing Rules under the Enable IIS Support Manually Using IIS Manager section.

3.6.5 Verifying the setup

To quickly verify the setup, open a browser (such as Chrome) and navigate to the following URL:

```
https://<external-url-and-port>/containers/v1/<shortname>
```

For example:

```
https://acme.com/containers/v1/w20sp100
```

If the proxy configuration is correct, the browser should either download, or show an XML or JSON document that looks like the following:

```
<Endpoint xmlns="http://www.deltek.com/ns/webservices" shortname="m20sp100" ↩
    authenticated="false">
  <Versions>
    <TPU major="20" minor="0" sp="100" fix="0" beta=""/>
    <APU major="20" minor="0" patch="100" hotfix=""/>
  </Versions>
  <Languages> ... </Languages>
  ...
  <Links>
    <Link href="https://acme.com/containers/v1/w20sp100" rel="self"/>
  </Links>
</Endpoint>
```

3.7 Domain Login and Single Sign On

The domain login functionality in iAccess is based on Kerberos service tickets obtained through the SPNEGO authentication protocol. This protocol allows direct Single Sign On (SSO) when the user is running iAccess while already authenticated against the domain (that is, logged in to their computer via a domain account).

If the user is not authenticated against the domain, the browser typically prompts for domain credentials. Click **Cancel** in the browser login window and use the iAccess domain login page.

3.7.1 Browser Setup for Single Sign On

Refer to the instructions in this section to set up Single Sign On (SSO) for various browsers.

SSO Setup for Internet Explorer

For Internet Explorer (IE), you may need to add the iAccess server address to the Local intranet zone if it is not already in this zone, as IE does not permit Kerberos-based SSO for websites in the Internet zone.

More details are available in the "Client Side-Internet Explorer" section of the following Microsoft article about security zones in Internet Explorer:

`https://msdn.microsoft.com/en-us/library/ms995329.aspx`

SSO Setup for Chrome

You can choose one of two options:

- If using Windows, you can perform the setup required for Internet Explorer. Chrome can replicate the setup for IE.
- To configure Chrome to work with SSO using Kerberos authentication, follow the steps in the "Set Chrome policies for devices" guide (https://support.google.com/chrome/a/answer/187202?

The configurations should be done by IT administrators who want to set Chrome policies on their corporate-managed devices. The templates contain hundreds of available policies that can be set, but you should only focus on two of these, namely:

AuthNegotiateDelegateWhitelist (http://www.chromium.org/administrators/policy-list- 3#AuthNegotiateDelegateWhitelist)

and

AuthServerWhitelist (http://www.chromium.org/administrators/policy-list-3#AuthServerWhitelist).

The properties should be set to the domain you want to authenticate against, such as:

`*. example.com .`

SSO Setup for Chrome on Windows

After following the preceding guide from Google, you can set **AuthNegotiateDelegate-Whitelist** and **AuthServerWhitelist** as follows:

1. Navigate to Administrative Templates » Classic Administration Templates (ADM) » Google » Google Chrome » Policies for HTTP Authentication.
2. Click “Kerberos delegation server whitelist”.
3. Click Enabled.
4. In the Input field, enter the domain you want to authenticate against, such as “*. example.com ”.
5. Click Apply.
6. Click on “Authentication server whitelist”.
7. Click Enabled.
8. In the input field, enter the domain you want to authenticate against, such as “*. example.com ”.
9. Click Apply.
10. Open Chrome.
11. Check the values by navigating to the URL:

    `chrome://policy`

SSO Setup for Chrome on Mac

After following the preceding guide from Google, you should also read the Mac Quick Start guide from Google at:

`http://www.chromium.org/administrators/mac-quick-start`

If the “Workgroup Manager from Apple” is not available for your version of OS X, then you can set **AuthNegotiateDelegateWhitelist** and **AuthServerWhitelist** using one of two recipes.

By creating a com.google.Chrome.plist file:

1. Create com.google.Chrome.plist file with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"     "http://www. ↩
    apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>AuthNegotiateDelegateWhitelist</key>
    <string>*.example.com</string>
    <key>AuthNegotiateDelegateWhitelist</key>
```

```
    <string>*.example.com</string>
  </dict>
</plist>
```

2. Set the two string attributes to the domain you want to authenticate against.
3. Convert the com.google.Chrome.plist to the binary format by running the following command from the Terminal:

   ```
   plutil -convert binary1 com.google.Chrome.plist
   ```

4. Copy the file to "/Library/Managed Preferences/" by running the following command from the Terminal:

   ```
   sudo -s
   cp com.google.Chrome.plist /Library/Managed Preferences/<username>
   ```

5. Open Chrome.
6. Check the values by navigating to the following URL:

   ```
   chrome://policy
   ```

By using the "defaults" command:

1. Run the following commands:

   ```
   defaults write com.google.Chrome AuthServerWhitelist *.example.com
   defaults write com.google.Chrome AuthNegotiateDelegateWhitelist *. ↩
       example.com
   ```

2. Open Chrome.
3. Check the values by navigating to the following URL:

   ```
   chrome://policy
   ```

Or by using the following command:

```
defaults read com.google.Chrome
```

Note that using the "defaults" command only sets the "AuthServerWhitelist" and "AuthNegotiateDelegateWhitelist" Chrome properties for the current user.

SSO Setup for Safari

No setup is needed.

SSO Setup for iOS

Follow the steps in the following guide:

`https://samuelyates.wordpress.com/2013/10/11/kerberos-single-sign-on-in-ios-7/`

Remember to put your own servername on the URLPrefixMatches field. As the name implies, this has to contain a URL prefix. This could be "https://myserver.example.com:8080", so basically this will be set to the base server URL including an optional port number.

You can use Apple Configurator 2 to install the profile on a number of iPads:

`https://itunes.apple.com/us/app/apple-configurator-2/id1037126344`

SSO Setup for Firefox

Complete the following steps:

1. In the location bar, type: about:config.

This brings up the configuration page.

2. In the Filter:box, type: negotiate.

This restricts the listing to the configuration options you need.

3. Edit network.negotiate-auth.trusted-uris to the domain against which you want to authenticate. For instance: ".".example.com

3.8 Domain Controller Setup

The SPNEGO authentication protocol works by assuming the presence of a specific Service Principal Name (SPN) on the domain controller:

`HTTP/name.domain or HTTP/name`

where name and domain are the web server DSN name and domain respectively, as seen from the user's computer.

For example, if the user is opening iAccess using the internet address https://some-server.some-domain.com, then the browser expects one of the following SPNs to be present on the domain controller:

`HTTP/some-server or HTTP/some-server.some-domain.com.`

3.8.1 SPN Setup

It is a task for the domain administrator to ensure that these SPNs are created and associated with the existing domain account used for Maconomy SSO.

SSO with Active Directory

For Active Directory, associating SPNs with the existing domain account is done with the 'setspn' command.

To associate SPN with an existing domain account, complete the following step:

On a command line enter the following:

```
setspn -A HTTP/name account
setspn -A HTTP/name.domain account
```

where account is the name of the domain account used for Maconomy SSO.

Special Instructions for SPN Conflicts

If iAccess is installed on a web server that already hosts other web applications with SNEGO authentication, this causes a conflict on the SPN, as an SPN can only be associated with one domain account.

To resolve the issue, either install only one web application on each web server, or create multiple local DNS names for the web server, so that each web application can be accessed through different addresses and will map to different SPNs.

3.9 Maconomy Server Setup

Please refer to the Single Sign On with Kerberos section in the Deltek Maconomy System Administrator Guide.

3.10 Additional Related Procedures

3.10.1 Configure Web Server to Reduce Risk of Clickjacking

You can reduce the risk of clickjacking by performing an additional step when configuring your web server. This step applies to both Apache and IIS.

To configure your web server and reduce the risk of clickjacking, complete the following step:

Configure your web server to always reply with the following response headers:

```
Content-Security-Policy: frame-ancestors self
X-Frame-Options: SAMEORIGIN
```

These headers are then added to all responses.

3.10.2 Downloading Deltek Products using the Deltek Software Manager

You can use the Deltek Software Manager (DSM) to download complete Deltek products, hot fixes, and sub-releases. You can access DSM directly or through the Deltek Customer Care Connect site.

When you access DSM directly, you will be prompted to log on before you can access the application. If you access DSM from within the Deltek Customer Care site, you do not have to log on since you are already logged in to the Customer Care site.

Accessing DSM Directly

To access Deltek Software Manager directly, complete the following steps:

1. Launch Deltek Software Manager by taking one of the following actions:

   Click here: http://www.deltek.com/ On your desktop, click Start » Programs » Deltek » Maconomy iAccess » Deltek Software Manager.

2. In the Deltek Software Manager logon dialog box, enter your Deltek Customer Care User ID and Password, and click Logon.

3. To select the folder where you want to download Deltek products, click Settings above the right pane of Deltek Software Manager.

   Note: When you log on for the first time, Deltek Software Manager asks you to select a default folder where Deltek products are to be downloaded.

4. Use the Settings dialog box to specify the folder where you want to download Deltek products, and click OK.

   Note: You can change this folder anytime in the Settings dialog box.

5. In the left pane of Deltek Software Manager, expand the Deltek product that you want to download, if it is not already expanded.

   Note: If you clicked the link in step 1 to access DSM, the application automatically selects Maconomy iAccess for you.

6. Select the product type that you want to download. Your options are Complete, HotFix, and Sub-Release.
7. In the table, select the check box that corresponds to the Deltek product that you want to download. The right pane displays a message stating that the product has been added to the download queue. To view the items in the download queue, click View Download Queue at the bottom of the left pane.
8. Click Download at the bottom of the left pane. Deltek Software Manager downloads the product to the folder that you selected.

Accessing DSM from within the Customer Care Connect Site

To access Deltek Software Manager from within the Customer Care Connect site, complete the following steps:

1. In your Web browser, go to http://support.deltek.com.
2. Enter your Customer Care Connect Username and Password, and click Log In.
3. When the Customer Care Connect site displays, click the Product Downloads tab. You are automatically logged into Deltek Software Manager.
4. To select the folder where you want to download Deltek products, click Settings above the right pane of Deltek Software Manager.

   Note: When you log on for the first time, Deltek Software Manager asks you to select a default folder where Deltek products are to be downloaded.
5. Use the Settings dialog box to specify the folder where you want to download Deltek products, and click OK.

   Note: You can change this folder anytime in the Settings dialog box.
6. In the left pane of Deltek Software Manager, expand the Deltek product that you want to download, if it is not already expanded.
7. Select the product type that you want to download. Your options are Complete, HotFix, and Sub-Release.
8. In the table, select the check box that corresponds to the Deltek product that you want to download. The right pane displays a message stating that the product has been added to the download queue.

   Note: To view the items in the download queue, click View Download Queue at the bottom of the left pane.
9. Click Download at the bottom of the left pane. Deltek Software Manager downloads the product to the folder that you selected.

DSM Documentation and Troubleshooting

To view the online help for Deltek Software Manager, navigate to:

`https://dsm.deltek.com/DeltekSoftwareManager/Help/`

To view a tutorial on how to use Deltek Software Manager, navigate to:

`https://dsm.deltek.com/DeltekSoftwareManager/Tutorial/PubData/Engine/Default.htm?https%3A%2F%2Fdsm.deltek.com%2FDeltekSoftwareManager%2FTutorial%2FPubData%2F`

To view more information on troubleshooting Deltek Software Manager, navigate to:

`https://deltek.custhelp.com/app/answers/detail/a_id/52469`

Note: The preceding troubleshooting link only works if you are logged in to Deltek Customer Care Connect.

Chapter 4

Extending iAccess

Note to the Reader The Extension model has changed significantly from iAccess version 1 to version 2. This section describes key parts of the iAccess 2 extension facilities. However, it is not complete and should be considered *Work in Progress*. If you are unable to find what you are looking for here, we recommend posting a question to the Engineering team through our "iAccess for Maconomy" Kona space (https://www.kona.com/#!/projects/129727). We will reply there and take note of your questions, feedback, and comments for future iterations of this section.

This section describes how to make client-side iAccess extensions using built-in extension facilities as well as tooling provided by the Maconomy Extender. Client-side extensions cover authentication methods, preferred settings, menu customization, and updates to existing workspaces/addition of new workspaces. The primary goal of iAccess extensions is to customize the appearance of Maconomy functionality when leveraged over a Web user interface. If you need to make fundamental changes such as extending core business logic and adding new fields and actions, use the Java extension framework to create server-side extensions. While the two extension approaches can be used together, it is important to note that the purpose of client-side extensions is to change how Maconomy functionality is rendered, not how it works.

The iAccess runtime is essentially version-independent. It runs on different Maconomy backends and service pack ranges. On the other hand, the iAccess applications are tied to specific Maconomy backends or applications (that is, Application Packaging Units or *APUs*). If you install a specific iAccess FPU, it will look and act differently depending on which Maconomy version you are running in the backend. This approach is necessary since different Maconomy versions expose different containers, fields, actions, system parameters, and backend extension capabilities.

When developing new iAccess extensions or upgrading existing ones, you must take note of what the target Maconomy backend is because this restricts what containers, actions, fields, and so on, are available. The iAccess FPU contains different applications,

each compatible with a specific range of service packs on a given major Maconomy version. You must also look at the iAccess API version that the given iAccess FPU exposes. The API version determines the kinds of extension possibilities available and their properties.

Both the iAccess application version and the API version are specified in the root of the iAccess configuration files, specifically in the `application.json` file. In the following example, the iAccess application is called `17` and the current version is `1.0.2`. The FPU manifest describes the backend Maconomy versions compatible with this application. This sample configuration complies with version `8.0.0` of the iAccess runtime API.

```
{
  "api": "8.0.0",
  "application": {
    "name": "17",
    "version": "1.0.2"
  },
  "authentication": {
    "$ref": "Authentication:Authentication"
  },
  "platform": {
    "$ref": "Platform:Platform"
  },
  "shell": {
    "$ref": "Shell:Shell"
  },
  "workspace": {
    "$ref": "Workspace:Workspace"
  }
}
```

The remainder of this file consists of JSON references to the four main iAccess areas:

- Authentication - Configuration of available and preferred login methods.
- Platform - Configuration and tweaking of low-level iAccess runtime mechanics.
- Shell - Configuration of the sidebar menu, notifications, and preferences.
- Workspace - Configuration of workspaces and their layouts.

The following subsections describe these areas in greater detail.

A Note about iAccess Configuration Files

iAccess configurations use the JSON format, a standard notation similar to XML. During runtime, iAccess requests a single configuration file describing all extensions (standard or custom). This file is called `application.json` and is retrieved through the `configurations` end point in the REST API. In the iAccess source configuration files, the notation (see the following example)

```
{
  "$ref": "Authentication:Authentication"
}
```

is a heavily used modularization technique. The JSON object is replaced (or macro expanded) by the `Authentication.json` configuration file located in the `Authentication` folder.

When the iAccess configuration files are processed by the REST API, all JSON properties with the prefix `T$` are translated to the language utilized by the end-user. For example, if an iAccess source configuration file has the following contents:

```
{
    "T$title": "Hello World"
}
```

then a Spanish user receives the following content in his/her browser:

```
{
    "title": "Hola mundo"
}
```

The translation relies on standard Maconomy dictionaries. It only works if a dictionary containing the required terms exists in the chosen language.

4.1 Authentication

The *authentication* part is the first main area of iAccess extension. It specifies the authentication methods available to end-users when they log in to iAccess. How you configure authentication determines the screen end-users see when they first visit the iAccess web page. For a sample configuration, refer to the following text:

```
{
  "preferred": "maconomy",
  "methods": {
    "maconomy": {
      "enabled": true
    },
    "domain": {
      "enabled": true
    },
    "sso": {
      "enabled": true
    },
    "azure": {
      "enabled": true
    }
```

```
  }
}
```

Although this is probably the simplest area in terms of iAccess configuration, authentication is often hard to get right in actual installations. As an installation consultant, you must focus on setting up the right infrastructure and specifying a corresponding *preferred* authentication method. The preferred method determines how iAccess will redirect the browser for the user accessing the root URL (for example: `/`) If you specify *maconomy* as preferred, iAccess will redirect the browser to the `/maconomy` URL so the user can enter login credentials there.

iAccess currently supports four different authentication methods:

Maconomy A standard Maconomy login page where the user can enter his/her username and password.

Domain A domain login page where the user can enter domain, username, and password.

SSO Single Sign-On using Kerberos authentication and the SPNEGO protocol [4, 5] which takes the user past the login screen and directly into the application. If the login fails, the user is redirected to the domain login page as a fallback.

Azure Single Sign-On using Azure [1] which takes the user through the Azure login flow. This flow varies depending on the customer's setup. To force account selection during the Azure login flow, the user can opt to append `prompt=select_account` to the iAccess URL as an extra query parameter.

If you disable an authentication method, you also disable the iAccess URL associated with that login method. Deltek recommends that only required authentication methods are left enabled on production installations.

Subcontractor Login

It makes sense for a company with different types of users sharing an iAccess installation to employ more than one authentication method. For example, a company with domain login configured for its employees will need a different method for subcontractors who are not allowed on the domain. In this case, the company can provide Maconomy credentials for its subcontractors. The company can then choose domain login as its preferred authentication method on the system, but instruct its subcontractors to log in to iAccess using the `/maconomy` URL which leads directly to the Maconomy login page. The suggested configuration for this is as follows:

```
{
  "preferred": "domain",
  "methods": {
    "maconomy": {
      "enabled": true
    },
    "domain": {
```

```
      "enabled": true
    },
    "sso": {
      "enabled": false
    },
    "azure": {
      "enabled": false
    }
  }
}
```

4.2 Platform

The second main area of iAccess extension is the *platform* part which specifies various low-level iAccess behaviors. You can find the default configuration of the platform part in the `Platform.json` file:

```
{
  "usageTracking": {
    "$ref": "Usagetracking"
  },
  "containers": {
    "$ref": "Containers"
  }
}
```

Usage Tracking

The *usage tracking* part concerns the integration between iAccess and Google Analytics. This feature was also present in iAccess version 1 and is used to track page views, events, errors, and other usage data about end-users. You can acquire your own tracking ID directly from Google, and add it to the configuration so you can collect usage data about an iAccess installation. In the following excerpt, the Google Analytics tracking ID `UA-123456789-1` is added to the standard configuration:

```
{
  "enabled": true,
  "endUserOptIn": true,
  "trackingId": "UA-123456789-1"
}
```

The two mandatory properties, `enabled` and `enduserOptIn`, are used to (1) enable/disable tracking completely and (2) show/hide the cookie consent banner. If tracking is disabled, then iAccess still creates cookies in the browser but performs no actual tracking.

If the end-user opt-in is disabled, then users are not presented with a banner informing them of tracking.

Containers

The *containers* part is active and is a new feature introduced in iAccess version 2. The containers configuration allows consultants to supply additional metadata about REST API containers. This is necessary because the specification of containers that we get through the `containers` endpoint only describes a subset of the semantics of the backend. The configuration format follows a very simple convention: it is an object with properties matching the container names from the REST API. You can add information about each container's various aspects, and the iAccess runtime will then adapt to this new data.

Singleton Containers

Maconomy makes a special distinction between normal and *singleton* containers. Whereas normal containers represent collections of resources, singleton containers always represent a single resource. This leads to a slightly different navigation interface in the REST API. Only very few containers are singletons, and these are not specifically identified by the REST API. However, since some of these are very prominent for most users, iAccess allows the consultant to add information about singletons in the containers configuration.

```
{
  "TimeRegistration": {
    "card": {
      "singleton": {
        "DateVar": "card.datevar",
        "EmployeeNumberVar": "card.employeenumbervar"
      }
    }
  }
}
```

The preceding excerpt specifies *TimeRegistration* as a singleton container. *DateVar* and *EmployeeNumberVar* are used when navigating with this container. Singleton properties on the right side refer to internal names used by the REST API when navigating this container.

File Actions

Another feature of the containers configuration is the ability to add information about file actions. The REST API supports actions for uploading files. However, it does not

specify which ones are actually file actions. This information is important since the protocols for dealing with file actions are different from those for normal actions, both in the iAccess runtime and in terms of messages exchanged between the browser and the REST API.

```
{
  "ExpenseSheets": {
    "card": {
      "actions": {
        "AttachDocument": {
          "attachments": {
            "required": true,
            "allowMultipleAttachments": true
          }
        }
      }
    },
    "table": {
      "actions": {
        "AttachDocumentToLine": {
          "attachments": {
            "required": true,
            "allowMultipleAttachments": false
          }
        }
      }
    }
  }
}
```

The preceding configuration specifies that the *AttachDocument* action on the *ExpenseSheets* card pane and the *AttachDocumentToLine* action on the corresponding table pane are file actions. Both require an attached file, but they differ in the number of files they allow as attachments.

Action Parameters

Some Maconomy actions require certain variables to have a particular value when executed. The REST API does not specify these variables, so iAccess can add this information as so-called *parameters* in the container configuration. In the following example, the *ApproveAbsenceEntry* action on the card pane of the *AbsenceEntryOverview* container is configured to require four variables as parameters, namely: *EntryDateVar*, *ValidTillVar*, *NumberOfDaysVar*, and *ReasonVar*.

```
"AbsenceEntryOverview": {
  "card": {
    "actions": {
```

```
      "ApproveAbsenceEntry": {
        "parameters": {
          "EntryDateVar": "card.entrydatevar",
          "ValidTillVar": "card.validtillvar",
          "NumberOfDaysVar": "card.numberofdaysvar",
          "ReasonVar": "card.reasonvar"
        }
      }
    }
  }
}
```

Filter-As-Table Panes

Standard Maconomy containers have at least one of the following pane types: card, table, and filter. iAccess 2 includes a new special type called *Filter as Table* which is handled purely on the client-side. This pane type allows consultants to render a standard filter pane as a table pane. In general, filters are read-only whereas tables are editable. When using a filter as a table, the currently selected record is actually the corresponding card pane record. This makes the filter appear as an editable table, although it takes a bit longer to respond since iAccess has to load the corresponding card record whenever the user selects a filter row.

```
"AbsenceEntryOverview": {
  "filterAsTable": {
    "fields": {
      "RelatedEntryValidTill": {
        "immutable": true
      },
      "EmployeeName1": {
        "mapsTo": "EmployeeNameVar"
      }
    }
  }
}
```

To make a filter pane available as a table, you may need to add additional field specifications in the containers configuration file. In the preceding example, two changes are added. First, the *RelatedEntryValidTill* field is specified as immutable. This is a performance optimization which states that this field will not change its value when its card record is edited. This means that iAccess does not to need to reread the field when a user edits the card record. Second, the *EmployeeName1* field is mapped to the *EmployeeNameVar* field. This is necessary in cases where an underlying database column has different field names depending on whether it is shown in the filter or in the card. The `mapsTo` attribute allows iAccess to link these two fields.

4.3 Shell

The *shell* part is the third main area of iAccess extension, and this refers to the part of the application that surrounds the workspace area after the user logs in. The core configuration covers the sidebar menu as well as the documentation URL, settings, and notifications in the top-right corner.

```
{
  "menu": {
    "$ref": "Menu"
  },
  "documentationUrl": "https://help.deltek.com/Product/MaconomyiAccess/< ↩
    version>",
  "settings": {
    "$ref": "Settings"
  },
  "notifications": {
    "$ref": "Notifications"
  }
}
```

Menu

The sidebar menu is configured in the `Menu.json` file. The *workspace* part of the iAccess configuration contains the total list of available workspaces. The menu is a subset of these workspaces, ordered in groups. There is no limit to the number of groups nor the number of workspaces within a group. iAccess does not allow the use of nested groups, so avoid adding too many groups and workspaces to the menu. Doing so clutters the user interface and makes navigation difficult for the user.

```
{
  "defaultWorkspace": "WeeklyTimeSheets",
  "groups": [
    {
      "T$title": "Self Service",
      "items": [
        {
          "workspace": "WeeklyTimeSheets"
        },
        ...
      ]
    },
    ...
  ]
}
```

The `defaultWorkspace` property determines the workspace shown upon initial login. When the user selects a different workspace, iAccess saves this information locally and uses that workspace as the default view for the next login. You can disable this functionality using the `restoreLastWorkspace` property which has a default value of "true".

Groups have a `title` property and a list of items. Each item represents either a concrete workspace or a hyperlink. You can assign a title to a specific workspace. The `workspace` property refers to the internal name of the workspace as declared in the *workspace* part of the iAccess configuration. On the other hand, hyperlinks are composed of a title and a URL. Placing hyperlinks in the menu is convenient if access to third-party systems is required.

Groups, workspaces, and hyperlinks have an optional `condition` property which determines their visibility. To specify this property, you can use either an expression (for example, `true` or `false`), or an array of role names. If you use the latter, the group or item will only be displayed if the logged in user has one of those roles. The following example shows an excerpt of a menu where the *Employee Information* workspace is only visible to users with either the Project Manager or HR Manager role. When specifying a role name, use the the internal name (without spaces).

```
{
  "T$title": "Employee Information",
  "workspace": "EmployeeInformation"
  "condition": ['projectmanager', 'hrmanager']
}
```

Documentation URL

The documentation URL is a simple hyperlink to external documentation. It is rendered as a question mark in the top-right corner of the screen. By default, it points to the standard hosted help from Deltek. It is version-specific because the `<version>` placeholder is substituted with the iAccess version at runtime. Customers who prefer to use their own external documentation (such as a page on an internal Sharepoint site) can simply replace the URL here.

Settings

Settings concern iAccess language and formatting. Select the language used in iAccess from the dictionaries installed on the Maconomy server. To further filter the server list of languages, you can use the `couplingconfiguration.mcsl.xml` file found in the 'Definitions' folder on the server.

Aside from specifying the preferred language, you need to configure the `fixed` flag which states whether the user can change the language used. If you set the fixed flag to 'false',

the user can change the language on the login screen and in the settings dialog inside iAccess. If you set the fixed flag to 'true', make sure the preferred language is actually available from the server. In other words, the iAccess language configuration must be aligned with the set of dictionaries installed on the server and with any filters defined in the *MCSL* specification.

Default data formatting is location-specific. This format controls rendering of dates, as well as the decimal separator and digit grouping used. You can edit the preferred location and the settings for that location. You also have the option to assign formatting to a specific location by using the `fixed` property. The following example shows the `preferred` and `fixed` properties in use, as well as a single location configured with a date format, a decimal separator, and a digit grouping system.

```
{
  "preferred": "da_DK",
  "fixed": false,
  "available": {
    "da_DK": {
      "date": {
        "short": "dd/MM/y"
      },
      "symbol": {
        "decimal": ",",
        "group": "."
      }
    },
    ...
  }
}
```

You can also configure the optional `minutesThreshold` property, which determines whether a time entry is interpreted as minutes or hours. The default value is '10', which means that an entry of '10' is interpreted as 10 minutes, and an entry of '11' is interpreted as 11 hours.

Notifications

In the current iAccess version, only a hardcoded subset of notification types are available. This will change at a later stage. Each notification type is linked to a specific iAccess workspace. The current set of supported notification types and their corresponding iAccess workspaces are as follows:

Notification Type	Workspace
SubmitTimeSheet	Weekly Time Sheets
RejectedTimeSheetApprovalHierarchy	Weekly Time Sheets

Notification Type	Workspace
RejectedTimeSheetLines	Weekly Time Sheets
SubmitDailyTimeSheet	Daily Time Sheets
SubmitExpenseSheet	Expense Sheets
RejectedExpenseSheet	Expense Sheets
RejectedExpenseSheetLines	Expense Sheets
SubmitMileageSheet	Mileage Sheets
RejectedMileageSheet	Mileage Sheets
RejectedMileageSheetLines	Mileage Sheets
ApprovedAbsenceCalendarLines	Absence
RejectedAbsenceCalendarLines	Absence
SubmitAllowanceRequestAbsenceEntries	Absence
ApprovedAllowanceRequestAbsenceEntries	Absence
RejectedAllowanceRequestAbsenceEntries	Absence
ApproveAbsenceCalendarLines	Absence Approval
ApproveAllowanceRequestAbsenceEntries	Allowance Approval
RejectedPurchaseOrder	Purchase Orders

Use the `reloadInterval` property to configure how often iAccess retrieves the notifications computed on the server. You can also use the`recalculationInterval` property to configure how often iAccess should request a recalculation of a specific user's notifications. Both of these properties are specified in minutes. If you make the recalculation interval too short, you risk causing a serious system-wide performance degradation.

```
{
  "reloadInterval": 10,
  "recalculationInterval": 30
}
```

4.4 Workspace

The *workspace* part is the fourth main area of iAccess extension. It specifies the structure and layout for all workspaces available in iAccess. This part represents the majority of iAccess extension capabilities, since end-users spend most of their time in the workspaces. It has two components: a set of workspaces, and a set of global definitions.

```
{
  "workspaces": {
    "Absence": {
      "$ref": "EmployeeSelfService:AbsenceMgmt"
    },
    ...
  },
```

```
  "definitions": {
    "$ref": "GlobalDefinitions"
  }
}
```

In the `Workspace.json` file (as shown in the preceding example), the `workspaces` property is a map of all embedded workspaces, and includes the entire collection of workspaces. The sidebar menu is a subset of this collection.

The `definitions` property specifies a set of named styles that can be used across workspaces.

Each workspace has a unique name that serves as its internal identifier and an external, localizable title. The contents of a workspace has two main parts: the data bindings and the layout (as shown in the following example). The data bindings is a configuration of the Maconomy containers that provide data for the workspace. These containers are bound together in a tree, similar to how workspace definitions (*MWSL* specifications) are done for the Workspace Client. On the other hand, the layout is a configuration of how this tree of Maconomy containers is rendered to the end-user. This corresponds roughly to dialog layouts (*MDML* specifications) for the Workspace Client. The following subsections discuss data bindings and layouts in greater detail.

```
{
  "name": "ExpenseSheets",
  "T$title": "Expenses",
  "dataBindings": {
    ...
  },
  "layout": {
    ...
  }
}
```

Data Bindings

Copy to come.

Layouts

Copy to come.

Wizards

Copy to come.

Chapter 5

Miscellaneous

The following section contains a migration guide which describes how to migrate a customized iAccess installation. We also include a troubleshooting guide with a few tips about how to overcome typical installation issues.

5.1 Migration Guide

The following sections describes the steps needed for migrating from one specific version of iAccess to another. If the version that you are currently using is not mentioned here, or if you have received a special release or hotfix, please get in touch with the iAccess development team for further instructions.

Our long term goal in iAccess 2 is to make all migrations automated through tool support in the Maconomy Extender. This functionality is, however, not available yet and hence manual steps are always required. When migrating iAccess 2 extensions, it is important to notice that such migrations have two dimensions:

1. If the Maconomy backend has changed version then the standard iAccess configuration files will most likely be extracted from a different iAccess application. In that case, it is important to inspect the differences between the source iAccess application and the target one since fields, layout, workspaces, and available configurations may have changed. The easiest way to do this is to use a Diff tool to compare the old standard files with the new ones. These standard files are always placed in the `Web` folder in `MaconomyDir` on the Maconomy server.
2. Since iAccess is run in an agile fashion with frequent smaller releases, the configuration API will often change. The API version is specified in the manifest of the iAccess FPU. This version should match the one given in the `application.json` root configuration file. When the API changes, it is important to validate one's existing extensions in context of the target iAccess FPU. This can be done through

the validation facility in the Maconomy Extender.

5.1.1 From 1.x to 2.0

You need to redo all extensions from scratch. We are not delivering a migration tool at this point. Deltek recommends that you inspect the standard extensions delivered with iAccess 2.0, and then ask concrete questions in our "iAccess for Maconomy" Kona space (https://www.kona.com/#!/projects/129727). Engineering is actively monitoring this forum to ease the transition.

Please observe that for 2.0, two new rewrite rules are added for the two new REST endpoints that we rely on, specifically "auth" and "environment". On IIS, an additional rewrite rule is required as described in the "Edit Routing Rules" section. Also, you need to configure "woff2" as a supported MIME type.

5.1.2 From 1.2.x and 1.3.0-3 to 1.3.4

For upgrades from the 1.2.x series, perform the steps outlined in the following sections, and then proceed to the step given here. When migrating from 1.3.0-3 versions to 1.3.4, make sure you update all JSON references. You no longer need to include an iAccess namespace in the names of specification files. This means you should remove the `iaccess:`-prefix from JSON references. For example, change:

```
"screens": {
  "dm.dailytimesheets": {
    "$ref": "iaccess:dailytimesheets"
  }
```

to

```
"screens": {
  "dm.dailytimesheets": {
    "$ref": "dailytimesheets"
  }
```

5.1.3 From 1.2.0 and 1.2.1 to 1.2.2 and 1.3.0

Some of our core terminology has changed, as outlined in the following table. This means that you need to update the following keys in your configuration:

User Interface Concept	1.2.0 and 1.2.1 API	1.2.2 and 1.3 API
Default View	`defaultScreen`	`defaultView`
Views	`screens`	`views`

User Interface Concept	1.2.0 and 1.2.1 API	1.2.2 and 1.3 API
Leftnav	`sidebar`	`leftnav`

Since this is a *breaking change*, the API version has also changed. Configurations should state that they now rely on version `2.0.0` rather than `1.2.0`.

In 1.3, we added support for the Additional Table Fields extension point in expense and mileage sheets. This means you must merge the specifications from these views with the new defaults from the iAccess 1.3 FPU. You can use the Maconomy Extender to assist in this process.

We also added some new parts to the configuration. Integrate these changes by importing the latest specifications from a new FPU and merging these with existing customizations. Specifically, we introduced a `documentationUrl` under the `configuration` section in `application.json`. In this section, we also introduced references to two new specification files: `authentication.json` and `usagetracking.json`. Use the Maconomy Extender to add these files.

When migrating to `1.2.2` or to `1.3.0`, remove the preferences in the following listing from the `preferences.json` file. The defaults have changed and are not valid anymore. Customize the preferences as described in the configuration section.

```
"dateFormat": {
 "short": "M/d/yyyy"
},
"decimalSymbol": ",",
"digitGroupingSystem": ".",
"minutesThreshold": 10
```

5.1.4 From 1.1.x to 1.2.x

The major difference between the 1.x versions and 1.2.x is the introduction of the extensibility model. The table in the next section describes which configuration options from 1.2.x replace deprecated configuration options from 1.1.x.

Configuration of Leftnav

In version 1.1.x, you could configure which views were accessible via the leftnav by changing the `this.sidebarItems` array in `config.js`. This was done after installation on each individual web server.

In version 1.2.x, this kind of configuration is now a part of the central view configuration, and managed via the Maconomy Extender. You can access the configuration of each

view by following the links from `application.json` file. To show/hide a particular view, set the `enabled` attribute to either `true` or `false`.

```
{
  "name": "dm.dailytimesheets",
  "enabled": "true",
  ...
}
```

The mapping of view names between version 1.1.x and 1.2.x can be found in the following table:

1.1.x View Name	1.2.x View Name
inside.timesheets	dm.weeklytimesheets
inside.dailytimesheets	dm.dailytimesheets
inside.expensesheets.edit	dm.expensesheets
inside.mileagesheets.edit	dm.mileagesheets
inside.jobfavorites	dm.favoritemgmt
inside.absence.tabs	dm.absencemgmt

In version 1.1.x, you could specify the default leftnav tab in `config.js` with the `defaultSidebarItem` property. In version 1.2.x, you specify the default leftnav in the beginning of the `application.json` configuration as shown in the following example:

```
{
  "api": "1.2.0",
  "defaultScreen": "dm.weeklytimesheets",
  "screens": ...
}
```

Configuration of the Weekly Time Sheets View

In version 1.1.x, you could configure two properties of the time sheets' views: daily descriptions, and overtime specification.

In weekly timesheets, you could enable or disable daily descriptions in `config.js` by setting the `isDailyDescriptionsEnabled` property to either true or false. In version 1.2.x, you achieve this configuration by using the extension point `dm.additionalTableFields` described in a previous section.

Finally, in version 1.1.x, you could show or hide the overtime specification in weekly time sheets. In version 1.2.x, you achieve this by adding the `overtimetype` field to the table using the extension point `dm.additionalTableFields` described in a previous section.

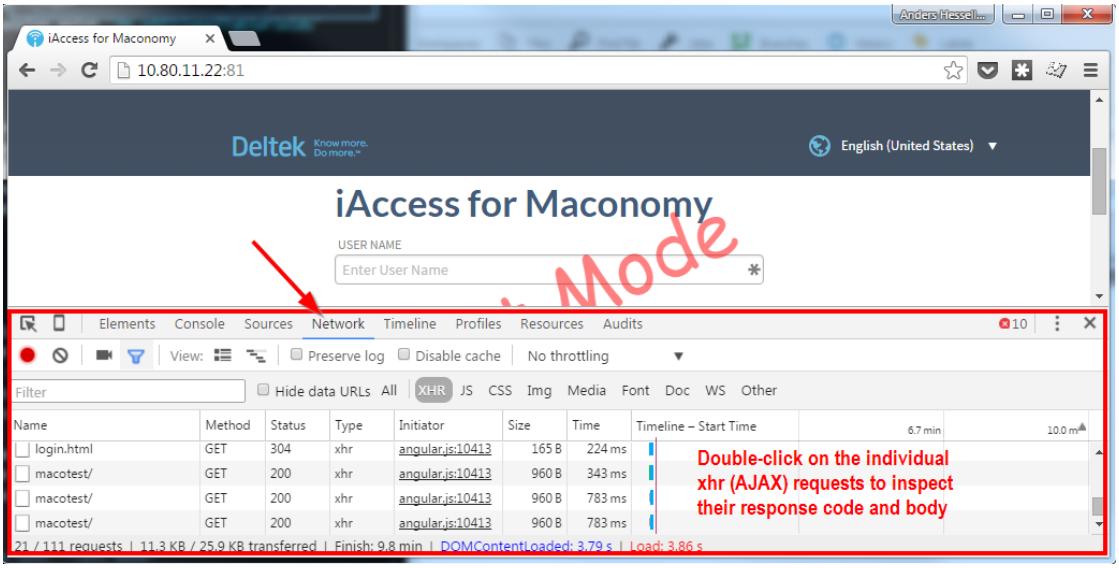

Figure 35: Use the Network Tab in your browser's Developer tools to debug failing requests and login problems.

Localization

In version 1.1.x, you localized a subset of the terms (for example, error messages) by placing a custom iAccess dictionary in the `i18n` folder on each web server. The 1.1.x version was only released with dictionaries for English and Danish. In version 1.2.x, all localization takes place on the Maconomy server through the existing localization engine. You customize translations by editing the traditional Maconomy dictionaries on the Maconomy server.

5.2 Troubleshooting Guide

Solutions to common installation issues are found in the Installing iAccess section. If your issue/problem is not listed there, the following section provides some additional clues to solve common issues. If you still cannot find a solution to your specific problem, please post a conversation in the *iAccess for Maconomy* Kona space or raise a support case through Customer Care to get your concrete issue resolved.

A piece of general advice for technical consultants: Always take a look at the requests that the browser issues when you are getting installation and/or network problems. In particular, the AJAX requests and error responses are often useful for uncovering installation and configuration errors. Figure 35 shows Developer Tools in Chrome where the Network Tab can be a very powerful tool to uncover installation and network problems.

"*Incompatible API versions...*"-Error

The iAccess specification format deployed on the Maconomy server is not compatible with the installed version of iAccess. This error usually occurs because either the `application.json` specification or the iAccess installed on a given web server have not been updated as part of a system upgrade. If the lowest number in the error message is the *required API version*, then you need to upgrade the iAccess version installed on the given web server. This requires the use of MConfig.

If the lowest number is the *loaded specification API version*, then update the specification deployed to the Maconomy server. This requires the Maconomy Extender.

"*Bad Request: Unable to connect to 'configurations' endpoint...*"-Error

When moving from version 1.1.x to 1.2.x, iAccess becomes dependent on a new webservice called *configurations*. This web service has to be available through the proxy configuration on the web server. This is similar to how the *containers* and *filedrop* web services are setup. See the Installing iAccess section for details.

Even if you have properly configured the *configurations* endpoint, you may still get the error on certain IIS installations. The problem can then be that some IIS installations do not allow the colon : character in URLs. To solve this, allow the colon : in the `web.config` file in the root of your web server [2]. Use its unicode encoded format `%u003a` in the configuration.

```
<configuration>
  <system.web>
  <!-- Default <,>,*,%,&,:,\,?
            or %u003c,%u003e,%u002a,%u0025,%u0026,%u003a,%u005c,%u003f -->
    <httpRuntime
       requestPathInvalidCharacters="%u003c,%u003e,%u002a,%u0025,%u0026,% ↩
    u005c,%u003f" />
  </system.web>
  <system.webServer>
    ...
  </system.webServer>
</configuration>
```

"*A%20Network%20Error%20Occurred*"-Window Opens

This error occurs when HTTPS has been partially or incorrectly configured on the web server. Double-check that the web server is configured according to the steps in the Installing iAccess section. This includes checking that the OSGi products in MConfig are configured correctly, and that HTTPS forwarding rules are set up on the web server.

Error 500 in the Browser on Apache 2.2 Installations

If you get a 500 error code in the browser and the following message in the Apache error log: `configuration error: couldn't perform authentication. AuthType not set!: /`, the cause is an incorrect `vhosts.conf` file. Specifically, make sure that the line `Require all granted` is not present. This problem occurs for Apache 2.2 only. Apache 2.4 requires this line.

Chapter 6

Figures

(27) Add Server Variables
(28) Setting up HTTPS
(29) iAccess in *Test Mode*
(30) Maconomy Extender 1.6 with a sample Maconomy Extender project
(31) Import iAccess specification from an FPU
(32) Locate the right iAccess FPU
(33) *Web*-folder with iAccess specifications
(34) Commit and push iAccess specifications
(35) Use the Network Tab in your browser's Developer tools to debug failing requests and gin problems.

Bibliography

[1] Microsoft Azure: Cloud Computing Platform Services. URL https://azure.microsoft.com/.

[2] How to make IIS allow colon sign in request url, February 2015. URL http://www.avantec.se/howto-make-iis-allow-colon-sign-in-request-url/.

[3] *Maconomy RESTful Web Services—Programmer's Guide*. Deltek Inc., September 2015.

[4] K. Jaganathan, L. Zhu, and J. Brezak. SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows. RFC 4559 (Informational), August 2006. URL https://tools.ietf.org/rfc/rfc4559.txt.

[5] Paul J. Leach, Karthik Jaganathan, Larry Zhu, and Wyllys Ingersoll. The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism. RFC 4178, October 2005. URL https://rfc-editor.org/rfc/rfc4178.txt.