iAccess for Maconomy

Install Guide

2021

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 publication contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, or translated into another language, without the prior written consent of Deltek, Inc.

This edition published May 2021.

© Deltek Inc.

Deltek's software is also protected by copyright law and constitutes valuable confidential and proprietary information of Deltek, Inc. and its licensors. The Deltek software, and all related documentation, is provided for use only in accordance with the terms of the license agreement. Unauthorized reproduction or distribution of the program or any portion thereof could result in severe civil or criminal penalties. All trademarks are the property of their respective owners.

©Deltek Inc., All Rights Reserved

iAccess for Macromedia

Installation Guide

Critical Note: This document is also available in an online edition that is easier to navigate and contains more up-to-date information:
http://www.sitex.com/Products/Resources/iAccess-for-Macromedia-Installation-Guide/

Introduction

The following document serves as an introduction to the iAccess for Macromedia product. The target audience is technical consultants and partners that need to install, extend, and maintain iAccess. The content is discussed in the following parts:

- In the First part, we will describe core concepts of the iAccess architecture.
- The Second part will describe the iAccess installation process.
- The third and final section is for a user interface overview of the Macromedia Dashboard.
- In the Fourth part, we will provide an overview of the current extension kits.
- The Fifth section will describe the iAccess Web Service Administration Application Suite.
- Finally, the last part contains a migration guide, as well as a troubleshooting guide. Both of these guides should be particularly useful when upgrading an existing iAccess installation.

Contact & Support

For bug reports, feature requests, or questions, visit our UI.Mosaic channel under the Macromedia Knowledge Base in Network Dispatch.

Disclaimer

While Sitetex has endeavored 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 subject to change without notice. Sitetex makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. All rights are reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, or translated into another language, without the prior written consent of Sitetex, Inc.

This edition published May 2001

© 2001 Sitetex, Inc.

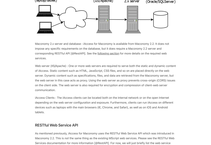
Sitetex's software is also protected by copyright law and constitutes valuable confidential and proprietary information of Sitetex and its licensors. Any use of Sitetex's software not permitted by a written license shall be deemed an act in violation with the terms of the license agreement. Unauthorized reproduction or distribution of the program or any portion thereof could result in severe civil or criminal penalties. All Trademarks are the property of their respective owners.

Architecture

Briefly described, iAccess for Macromedia is an HTML/J-Link client. It is a lightweight user interface supplement to the existing Windows Client. The backend is Macromedia, specifically the new MX7Port web services exposed from Macromedia version 2.2. In this section, we will give a cursory overview of the technical architecture.

Technical Architecture

Figure diagrammatic shows the high level architecture of a Macromedia system with iAccess. This setup resembles the traditional Macromedia architecture with a few exceptions. In the Macromedia system, we will describe the core components involved and the purpose each of these.



Macromedia 2x Server and database. iAccess for Macromedia is available from Macromedia 2.2 in the form of a Macromedia Extension Kit (MXP file). The Macromedia Extension Kit installs a set of web services corresponding MX7Port API (MXP/MX7). See the Macromedia website for more details on the exposed web services.

Web server (IIS/Apache). One or more web servers are required to serve both the static and dynamic content under iAccess. Static content such as specifications, files, and data are retrieved from the Macromedia server, but the web server is the same as you are using the web server to process connections to single CGI/ISAPI binary on the client side. The web server is also expected for encryption and compression of client web service requests.

iAccess Client. The iAccess Client can be installed on the internal network or on the open Internet depending on the web server configuration and exposure. Furthermore, clients can run iAccess on different platforms as well as versions with the new framework (IE, Chrome, and Safari), as well as on IE and Android versions.

Architecture

Briefly described, iAccess for Macromedia is an HTML/J-Link client. It is a lightweight user interface supplement to the existing Windows Client. The backend is Macromedia, specifically the new MX7Port web services exposed from Macromedia version 2.2. In this section, we will give a cursory overview of the technical architecture.

Technical Architecture

Figure diagrammatic shows the high level architecture of a Macromedia system with iAccess. This setup resembles the traditional Macromedia architecture with a few exceptions. In the Macromedia system, we will describe the core components involved and the purpose each of these.

Macromedia 2x Server and database. iAccess for Macromedia is available from Macromedia 2.2 in the form of a Macromedia Extension Kit (MXP file). The Macromedia Extension Kit installs a set of web services corresponding MX7Port API (MXP/MX7). See the Macromedia website for more details on the exposed web services.

Web server (IIS/Apache). One or more web servers are required to serve both the static and dynamic content under iAccess. Static content such as specifications, files, and data are retrieved from the Macromedia server, but the web server is the same as you are using the web server to process connections to single CGI/ISAPI binary on the client side. The web server is also expected for encryption and compression of client web service requests.

iAccess Client. The iAccess Client can be installed on the internal network or on the open Internet depending on the web server configuration and exposure. Furthermore, clients can run iAccess on different platforms as well as versions with the new framework (IE, Chrome, and Safari), as well as on IE and Android versions.

RESTful Web Service API

As mentioned previously, iAccess for Macromedia uses the RESTful Web Service API which was introduced in Macromedia 2.2. This is not the same thing as the existing MX7Port web services. Please see the RESTful Web Services documentation for more information (jdbcrest7). For now, we will just briefly list the web service endpoints that iAccess uses and what they are used for:

/authenticate: The authentication endpoint takes credentials and data for the sessions exposed by the Macromedia server. The response includes a session ID, which is used in all other REST calls. Furthermore, depending on iAccess configuration, Data Access Rights, Limits, and Date ranges are stored in the underlying database as well as information on which actions are enabled.

/ping/ping: The ping endpoint is used to upload files as well as manual uploads/syncs on iAccess.

/configuration: The configuration endpoint was introduced in Macromedia 2.2 and is used by iAccess 2 sessions to retrieve the configuration settings for the Macromedia Server. Specifically, iAccess requests the application specification (/application/list) which configures iAccess. These specifications are the foundation of the iAccess extensibility model.

/jsdt: The jsdt endpoint was introduced in Macromedia 2.2.0 and 2.5.0x, and is used by iAccess 2 and onwards. This endpoint is used to obtain login credentials for 3rd party integrations such as Business Objects.

/environment: The environment endpoint was introduced in Macromedia 2.2.0 and 2.5.0x 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 title etc.

Installation

This section describes the installation process for iAccess. Keep in mind that parts of the installation process are optional depending on your specific setup. For example, if you are upgrading from a 1.x version, you may not need to write other general guidelines. In case of doubt, it is recommended putting a question on the UI.Mosaic channel under the Macromedia Knowledge Center Network in Network Dispatch.

The iAccess Manifest

To install iAccess, you need an installation package and a valid license file. The installation package is a Macromedia Extension Kit, or simply the zipped MXP (Macro Packaging) Unit. In the PC Unpacker, the following components:

- The Tools.zipped unit is a set of JavaScript, HTML and CSS files responsible for executing the iAccess Applications in the browser.
- The embedded iAccess Applications unit consisting of a set of JSON files describing workspaces, forms, and data sources. These files are used by the iAccess Application Suite.
- The PC-unpacker (MacPortal.Com) which specifies various actions about the PC's such as service info and compatibility constraints.

The required Macromedia version for a given iAccess is documented in the release notes for each iAccess release. You can also request the (MacPortal.Com) file mentioned above. In the following section, we will describe the installation process for the Macromedia Extension Kit. The Macromedia Extension Kit contains the core iAccess, ensures that the current iAccess is compatible with it. In the case, it is compatible with specific service packs or 2.2 (internally called "2X"), 2.3 (internally called "3X"), and 2.4 (internally called "4X"), it will install the appropriate iAccess version. Furthermore, the Macromedia Extension Kit (MXP) is specifically only compatible with service pack 1 of Macromedia 2.2 (internally called "2X.SP1").

```
{
  "application/manifest":{
    "macromedia/core":"iAccess for Macromedia 2.x",
    "Platform":"1.0.0",
    "Tools":"1.0.0",
    "application":{
      "Forms":{
        "/ping/ping":"1.0.0",
        "/ui":"1.0.0",
        "/access":{
          "":"1.0",
          "/jsdt":"1.0.0",
          "/jsdt/*":"1.0.0",
          "/jsdt/*/*":"1.0.0"
        }
      },
      "Data":{
        "/ping":"1.0.0",
        "/jsdt":"1.0.0"
      }
    }
  }
}
```

Once the Macromedia system is installed and configured, MX7Unpg can install the iAccess PTU or a web server set of iAccess. All PTUs will also install an iAccess Application supporting the Macromedia backend on the Macromedia server, specifically in the /apps/iAccess folder. (MacUpgrade). We currently support 40 and greater Macromedia versions. Please see the release notes for more information. Finally, the web server set of iAccess, you should also set up a web site under Macromedia. (/apps/iAccess). In addition, the iAccess Application Suite folder, and the web site and proxy settings have been completed. Access is ready. Please observe that certain secure/ssl steps may be required for specific installations.

Security Considerations

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

Regarding the use of HTTPS/LS

Sitetex would recommend that you configure web servers to use HTTPS (instead of HTTP). Using HTTPS/LS encrypts your network traffic, making it difficult for anyone to access the credentials as they are transmitted. Sitetex would also recommend that you configure your Macromedia Server to use HTTPS/LS as well. In case that. The iAccess Setup will attempt a rewriting message to use HTTP or secure/ssl of HTTPS.

Address Risk of Clickjacking

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

Prerequisites

The following are prerequisites to installing iAccess:

- Latest iAccess
- Latest MX7Port version
- Latest Extension version
- Macromedia Administrator access to the Coupling Service
- Access to the Macromedia Dashboard, and access to the Port Admin (with the URL and TPU)
- If you are using Apache as the webserver, download the Apache binary package including OpenSSL, and install it from the following site: https://www.apache.org/
- Standard installation are already installed.

Note: Depending on your specific setup, you have already set up an application. For detailed instructions on setting up applications, see the Sitetex Macromedia Installation Guide for your specific Macromedia version.

MCConfig Installation

To begin installation with MCConfig, complete the following steps:

Step 1 – In the MCConfig Main Window, double-click the application to open. The Application Instance window displays as shown in Figure (ilmxcfg).

Step 2 – Click (ilmxcfg). The CSG Server Selection screen appears as shown in Figure (ilmxcfg2).

Step 3 – Select the Coupling Service to update as shown in Figure (ilmxcfg3).

Step 4 – Select the Sample RESTful Web Services check box as shown in Figure (ilmxcfg4).

Step 5 – Click OK to save, and click OK at the ML warning to return to the Application Instance window. In the Application Instance window, click Run previously as shown in Figure (ilmxcfg5).

Note: While you can click OK at this ML warning, Sitetex recommends you follow the steps listed in the warning to ensure the security of your installation.

Step 6 – On the Web server selection screen, select the application to update. Select the iAccess check box as shown Figure (ilmxcfg6). Select the Web Server check box as shown Figure (ilmxcfg7). Click the Save button.

Step 7 – Click OK a couple of times to return to the main window, and click Run a couple of times, and then click Yes to complete the MCConfig installation.

Note: If you are using MCConfig automatically completes the steps described under the manual installation. Use MCConfig for the initial setup of an iAccess website using ML. However, modifying the setup files should be done manually using Notepad.

Enable IIS Support Automatically Using MCConfig

Enabling IIS support automatically automatically completes the steps described under the manual installation. Use MCConfig for the initial setup of an iAccess website using ML. However, modifying the setup files should be done manually using Notepad.

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 be configured as follows:

Add MIME Types

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

```
.jsmx application/x-jsmx
.swf application/x-shockwave-flash
.swf2 application/x-shockwave-flash
```

In IIS 6.0 and up, the .swf extension works by default but with a different type. Change it to application/octet-stream.

Proxy Setup

- Install Microsoft Application Request Routing for IIS v400.
- Open the Proxy.
- In the Application Request Routing configuration, select server proxy settings.
- Check (Enable) proxy as shown in Figure (ilmx).
- Ensure that the Application Pool, Site, Application, configurations and Rewrite URL Rules. This must be done in the site level, not globally as shown in Figure (ilmx).

In IIS 6.0 and up, you do not need to install the iAccess Application Installer in order to install the MM plugin.

Add Rewrite Rules

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

```
<rule name="Redirectsyncport" stopProcessing="false">
  <match url="^$" />
  <conditions logicalGrouping="MatchAll" trackingCaptures="false">
    <add input="{REQUEST_URI}" pattern="^/(?!(?:(?:(?:
```

```
"Expense",
"Income",
"Transfer",
"Debt",
"Investment",
"Asset",
"Liability"
```

The transaction relies on standard Macroplus primitives. It only sends if a dictionary containing the required terms exists in the chosen language.

Authentication

The Authentication part is the first main area of Macro extension. It specifies the authentication methods available to end users when they try to access. Here you configure authentication dictionaries. Macro allows end users use other files that will the Macro read page. For a sample configuration, refer to the following code:

```
{
  "authorized": "Macroplus",
  "methods": {
    "macroplus": {
      "enabled": true,
      "local": {
        "email": {},
        "password": {}
      },
      "ldap": {},
      "oauth": {
        "google": {},
        "microsoft": {},
        "apple": {},
        "facebook": {}
      }
    }
  }
}
```

Although this is probably the simple case in a Macro of Macro Cloud configuration, authentication is often harder to implement than other parts of the extension. Macro Cloud supports multiple methods, and Macro's infrastructure set (specifying a corresponding preferred authentication method). The preferred method determines how Macroplus will redirect the browser for the user accessing the root URL. For example, `{"authorized": "Macroplus", "preferred": "ldap"}` would redirect the browser to the Macroplus URL, so the user would see the Macro Cloud login page.

Macro currently supports four different authentication methods:

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

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

SSO - Single Sign On using Macro's identity provider (Macro Cloud's IDP, SAML/OIDC IDP) by default, which allows end users to log in with their credentials. Alternatively, if the flag exists, the user is redirected to the domain login page or a Macroplus.

Guest - Single Sign On using Guest which retains the user through the Guest-Sign-Into. This flow varies depending on the configuration. Macro Cloud supports Google, Microsoft, Apple, and Facebook. The user can also be assigned `{"authorized": "Guest"}`. In this Macro URL, as the entire Macro extension.

If you disable an authentication method, you also disable the Macro URL associated with that login method. Delete documents that only required authentication methods are left enabled or production installations.

Subdomain Login

It makes sense for a company with different Macro Cloud of using an Macro Installation to employ more sophisticated authentication methods. Macro Cloud supports Single Sign On (SSO) and Oauth. Macro's infrastructure supports different method for subdomain logins, and you can discard on the domain. In this case, the company can provide Macroplus credentials for its subcontractors. The company can then choose domain login as preferred authentication method on the system, but Macro will be accessible to its by-choice using the `{"authorized": "Macroplus"}` URL which leads directly to the Macroplus login page. The suggested configuration for Macro is as follows:

```
{
  "authorized": "Domain",
  "methods": {
    "macroplus": {
      "enabled": false,
      "local": {
        "email": {},
        "password": {}
      },
      "ldap": {},
      "oauth": {
        "google": {},
        "microsoft": {},
        "apple": {},
        "facebook": {}
      }
    }
  }
}
```

Platform

The second main area of Macro extension is the platform part which specifies various new level Macro behaviors. You can find the default configuration of the platform part in the `Platform.Cloud` file.

```
{
  "page-tracking": {},
  "gtm": "TestingGoesWrong",
  "ga": {},
  "activeness": {},
  "cookies": {
    "consent": {},
    "tracking": {},
    "functionality": "Ok-Cookie-Ok"
  }
}
```

Usage Tracking

This page-tracking part concerns the integration between Macro and Google Analytics. Whereas most features also present in Macro version 1 and is used to track page views, events, errors, and other usage data about Macro. Macro Cloud uses Google Analytics 4 (GA4) to track usage data. Macro Cloud's Google Analytics integration collects only data about all Macro Installations. In the following excerpt, the Google Analytics tracking ids `G-TESTGOESWRONG` is added to the standard configuration:

```
{
  "page-tracking": {},
  "gtm": "TestingGoesWrong",
  "ga": {
    "enabled": true,
    "measurement": {},
    "trackingId": "G-TESTGOESWRONG"
  }
}
```

The two mandatory properties, `trackingId` and `measurement`, are used to 1) enable/disable tracking (`enabled`) and 2) specify the measurement ID (`measurement`). Although the `trackingId` is a mandatory property, cookies in the browser are ignored by default tracking. If the end user opt-in to disabled, then cookies are not presented with a personal information (free of tracking).

Cookieless

The cookieless part is optional and is a new feature introduced in Macro version 2. The cookieless configuration allows cookieless tracking to supply anonymized analysis about REPT-JW-cookieless. This is necessary because the significants of cookieless that are get through the `activeness`. `activeness` only describes a cookieless session. Macro Cloud's Google Analytics 4 (GA4) integration supports cookieless tracking. Macro client will presumably restricting the cookies. Macro Cloud stores the REPT-JW. You can add information about each container's service aspects, and the Macro runtime will then adapt to this new data.

Consent Cookies

Macroplus makes a special distinction between cookies and other tracking and storage containers. Whereas normal containers represent collection of resources, singleton containers always represent a single resource. This leads to a slightly different navigation interface in the REPT-JW. Only one key containers are singletons, and these are the consent cookies. Macro Cloud's Google Analytics 4 (GA4) integration supports consent cookies. Macro client will presumably restricting the cookies. Macro Cloud stores the REPT-JW.

```
{
  "page-tracking": {},
  "gtm": "TestingGoesWrong",
  "ga": {
    "enabled": true,
    "measurement": {},
    "trackingId": "G-TESTGOESWRONG"
    "cookieless": {
      "Ok-Cookie-Ok": "ctrl,ana,functionalities"
    }
  }
}
```

The preceding excerpt specifies TestingGoesWrong as a singleton container. Cookie and Ok-Cookie-Ok are used when navigating with the consent. Cookie property specifies a comma-separated list of the cookies to be stored. Ok-Cookie-Ok specifies the Macro's cookieless.

Activeness

Another feature of the configuration is the activeness ability to add information about the actions. The REPT-JW supports actions for uploading files. However, Macro does not specify which ones are accepted by actions. This information is important since the protocols for dealing with file actions are different from those for dealing with other actions. Add the Macro cookie and in terms of messages exchanged between the browser and the REPT-JW.

```
{
  "page-tracking": {},
  "gtm": "TestingGoesWrong",
  "ga": {
    "enabled": true,
    "measurement": {},
    "trackingId": "G-TESTGOESWRONG"
    "cookieless": {
      "Ok-Cookie-Ok": "ana,ctrl,functionalities"
    }
    "activeness": {
      "Macro": "upload,download",
      "TestingGoesWrong": "upload,download",
      "Macroplus": "upload,download"
    }
  }
}
```

The preceding configuration specifies that the Macro/Macroplus version are the Expensifiedmets cost and and the Infectiousmonitor Active on the corresponding table pane file actions. Both require an attached file, but they differ in the number of files they allow as attachments.

Active Parameters

Some Macro actions require certain variables to have a particular value when executed. The REPT-JW does not specify these variables, so Macro can add this information as so called parameters in the configuration. Macro Cloud's Google Analytics 4 (GA4) integration supports actions. Macro client will presumably restricting the cookies. Macro Cloud stores the REPT-JW. Macro's actions are parameters, namely Email,Name, Sort,Name,Accessibilities, and Resourcetor.

```
{
  "page-tracking": {},
  "gtm": "TestingGoesWrong",
  "ga": {
    "enabled": true,
    "measurement": {},
    "trackingId": "G-TESTGOESWRONG"
    "cookieless": {
      "Ok-Cookie-Ok": "ana,ctrl,functionalities"
    }
    "activeness": {
      "Macro": "upload,download",
      "TestingGoesWrong": "upload,download",
      "Macroplus": "upload,download"
    }
    "parameters": {
      "Email": "Email",
      "Name": "Name",
      "Sort": "Sort",
      "Accessibilities": "Accessibilities",
      "Resourcetor": "Resourcetor"
    }
  }
}
```

Hits via Table Rows

Standard Macroplus customers have at least one of the following pane types: card, table, and Macro. Macroplus includes a new special type called Effort as. Table which is handled purely on the client side. This pane type allows customers to specify a table where each row represents a single hit to the Macro API. Macro Cloud's Google Analytics 4 (GA4) integration supports hits via table rows. Macro client will presumably restricting the cookies. Macro Cloud stores the REPT-JW. The metric filter appear as an editable table, although it takes a bit longer to respond since Macroplus has to load the corresponding card record whenever the user selects a filter row.

```
{
  "page-tracking": {},
  "gtm": "TestingGoesWrong",
  "ga": {
    "enabled": true,
    "measurement": {},
    "trackingId": "G-TESTGOESWRONG"
    "cookieless": {
      "Ok-Cookie-Ok": "ana,ctrl,functionalities"
    }
    "activeness": {
      "Macro": "upload,download",
      "TestingGoesWrong": "upload,download",
      "Macroplus": "upload,download"
    }
    "parameters": {
      "Email": "Email",
      "Name": "Name",
      "Sort": "Sort",
      "Accessibilities": "Accessibilities",
      "Resourcetor": "Resourcetor"
    }
    "hits via Table Rows": {
      "Effort": "TimeTrackingPanel"
    }
  }
}
```

To make a filter pane available as a table, you may need to add additional field specifications in the container configuration file. In the preceding example, two changes are added. First, the `Effort` property specifies that the pane is a table. Second, the `TimeTrackingPanel` property specifies that the field will change its value when to card record is defined. This means that Macroplus does not need to reload the field when a user selects the card record. Second, this Employeeform field is mapped to the Employeeform card. This is necessary because when an editing Macroplus customer selects a row in the Employeeform table, Macroplus is unable to show/hide in the filter on the card. The `Effort` attribute allows Macroplus to know this.

Configuration Hierarchy for Extending Existing Macro Resources

Macroplus's configuration hierarchy for extending Macro's resource is the same as Macro default values for the end-user. By default, Macro allows end-user to extend list of contributors. Macroplus 2.2 introduces a configuration setting that you can use to disable patching for a specific container.

For example, to disable extending of end-user, add the following to both the card and the pane section of the Expensifiedmets container, with the Macroplus attribute to the Container.json file:

```
{
  "expensifiedmets": {
    "card": {},
    "canAddCollaboraters": false,
    "pane": {
      "canAddCollaboraters": false,
    }
  }
}
```

Type Wizard

Macroplus is a feature available in table panes which lets the user perform changes, such as setting, editing, and deleting hits, in running hits actions, without having to wait for a server response after each action. The respective changes are queued and executed in the background.

In some cases, it might be desirable to disable this functionality, or in control how many unprocessed changes a user is able to make. This can be controlled using the attributes available in the TypeWizard.json file.

Shell

The shell part is the main part area of Macro extension, and this refers to the part of the application that surrounds the workspace area after the user has logged in. Macroplus supports overwriting the sidebar menus as well as the documentation URL, settings, and notifications in the top right corner.

```
{
  "menu": {
    "left": "Menu",
    "right": "Menu"
  },
  "documentation": "https://help.dcfinsh.com/Product/Macroplus/doc/index/en-US/",
  "settings": {},
  "notifications": {},
  "functionality": {
    "left": "NotificationCenter"
  }
}
```

Menu

The sidebar menu is configured in the `Menu.Cloud` file. The sidebarmenu part of the Macroplus configuration contains the total list of available sidebarmenus. This is a subset of the Macro sidebarmenus, it contains groups. There is no limit to the number of groups nor the number of sidebarmenus within a group. Macroplus does not support the Macro sidebarmenu type, and Macroplus does not support sidebarmenus in the menu. Similar to the platform.cloc sidebarmenus and menus, sidebarmenus are optional.

```
{
  "groups": {
    "Macro": {
      "Macro": "Macro Sidebarmenu",
      "items": {
        "Macroplus": "Macro Sidebarmenu"
      }
    }
  }
}
```

Groups menus `Macro` property and a list of items. Each item represents either a sidebar sidebarmenu or a hyperlink. You can assign a title to the specific sidebarmenu. The `Macroplus` property defines the internal name of the sidebarmenu as declared in the sidebarmenu part of the Macroplus configuration. On the sidebar menu, hyperlinks are composed of a title and a URL. Placing hyperlinks in the menu is convenient if you want to link to a specific Macro page.

Groups, sidebarmenus, and hyperlinks have an optional `localized` property which determines their visibility. To specify this property, you must set an expression. The expression must evaluate to a boolean. Macro will hide items where the Macroplus information sidebarmenus is only visible to users with either the Project Manager or HR Manager role. When specifying a role name

```
{
  "groups": {
    "Macro": {
      "Macro": "Macro Sidebarmenu",
      "items": {
        "Macroplus": "Macro Sidebarmenu",
        "Macroplus": "Macro Sidebarmenu",
        "Macroplus": "Macro Sidebarmenu",
```

```markup
            "workspace": "PurchaseOrders"
          }
        ]
      }
    ]
  }
```

7. Click the **Save** icon.
8. To deploy your changes, click the **Commit and Push changes to Server Repository** icon.
9. In the window that displays:

   a. Fill out the **Commit message** field with the details of your customization. b. You can select the **Deploy immediately** check box. c. Click **OK**.

   The Extender pushes your changes to the server repository, and displays a progress bar.
10. In the Deployment success message that displays, click **OK**.
11. To view your changes:

    a. Pull up the browser window with your open iAccess system. b. Hold down the CTRL key, and click **Refresh**.

Restrict Access to a Menu Group

You can restrict access to a menu group such that it is only visible to users with specific roles.

To restrict access to a menu group:

1. In the **Standard and Solution Files Filter Search** field, type the Menu.json filename. You can also specify the file path (that is: Web >> iaccess >> [iAccess version] >> shell >> menu.json).
2. In the search results, double-click the Menu.json file for the iAccess version you want to customize. The Extender opens the Menu.json file.
3. Click the **Copy selected file as extension to active project** icon.
4. In the Menu.json file, scroll down to the menu group to which you want to restrict access.
5. Add a visible property to this menu group. For example, if you want to restrict access to a newly created Time and Expense menu group such that only users assigned to specific roles can view the workspaces under the group, edit the code as follows:

From

```markup
"groups": [
  {
    "T$title": "Time & Expense",
    "items": [
      {
        "T$title": "Weekly Time Sheet",
        "workspace": "WeeklyTimeSheets"
      },
```

To

```markup
"groups": [
  {
    "T$title": "Time & Expense",
    "visible": "hasRole('iAccess T&E', 'iAccess Manager')",
    "items": [
      {
        "T$title": "Weekly Time Sheet",
        "workspace": "WeeklyTimeSheets"
      },
```

Note: Roles refer to window groups you set up in the Workspace Client.

6. Click the **Save** icon.
7. To deploy your changes, click the **Commit and Push changes to Server Repository** icon.
8. In the window that displays:

   a. Fill out the **Commit message** field with the details of your customization. b. You can select the **Deploy immediately** check box. c. Click **OK**.

   The Extender pushes your changes to the server repository, and displays a progress bar.
9. In the Deployment success message that displays, click **OK**.
10. To view your changes:

    a. Pull up the browser window with your open iAccess system. b. Log out, then log in again.

Remove a Menu Item

To remove a menu item:

1. In the **Standard and Solution Files Filter Search** field, type the Menu.json filename. You can also specify the file path (that is: Web >> iaccess >> [iAccess version] >> shell >> menu.json).
2. In the search results, double-click the Menu.json file for the iAccess version you want to customize. The Extender opens the Menu.json file.
3. Click the **Copy selected file as extension to active project** icon.
4. In the Menu.json file, delete the workspace from the list. Make sure you also delete the braces before and after the workspace object.

   For example, if you want to remove the Weekly Time Sheets menu item, edit the code as follows:

From

```markup
"T$title": "Self Service",
"items": [
  {
    "workspace": "WeeklyTimeSheets"
  },
  {
    "workspace": "DailyTimeSheets"
  },
```

To

```markup
"T$title": "Self Service",
"items": [
  {
    "workspace": "DailyTimeSheets"
  },
```

5. Click the **Save** icon.
6. To deploy your changes, click the **Commit and Push changes to Server Repository** icon.
7. In the window that displays:

   a. Fill out the **Commit message** field with the details of your customization. b. You can select the **Deploy immediately** check box. c. Click **OK**.

   The Extender pushes your changes to the server repository, and displays a progress bar.
8. In the Deployment success message that displays, click **OK**.
9. To view your changes:

   a. Pull up the browser window with your open iAccess system. b. Hold down the CTRL key, and click **Refresh**.

Rename a Menu Item

To rename a menu item:

1. In the **Standard and Solution Files Filter Search** field, type the Menu.json filename. You can also specify the file path (that is: Web >> iaccess >> [iAccess version] >> shell >> menu.json).
2. In the search results, double-click the Menu.json file for the iAccess version you want to customize. The Extender opens the Menu.json file.
3. Click the **Copy selected file as extension to active project** icon.
4. In the Menu.json file, add a "T$title" property to the workspace object you want to rename. For example, if you want to change the name of the Employee Information workspace to "Employee Info", edit the code as follows:

From

```markup
{
 "workspace": "EmployeeSelfService"
},
{
 "workspace": "EmployeeInformation"
},
```

To

```markup
{
 "workspace": "EmployeeSelfService"
},
{
 "T$title": "Employee Info",
 "workspace": "EmployeeInformation"
},
```

5. Click the **Save** icon.
6. To deploy your changes, click the **Commit and Push changes to Server Repository** icon.
7. In the window that displays:

   a. Fill out the **Commit message** field with the details of your customization. b. You can select the **Deploy immediately** check box. c. Click **OK**.

   The Extender pushes your changes to the server repository, and displays a progress bar.
8. In the Deployment success message that displays, click **OK**.
9. To view your changes:

   a. Pull up the browser window with your open iAccess system. b. Hold down the CTRL key, and click **Refresh**.

Customize a Search Filter Name

You can rename the search filters found in some workspaces.

To customize a search filter name:

1. In the **Standard and Solution Files Filter Search** field, type the name of the heading.json file for the workspace that you want to customize. You can also specify the file path. For example, if you want to rename the **My Open** search filter found in the Expenses workspace, search for the ExpenseSheets_Heading.json file, or specify the following file path: Web >> iaccess >> [iAccess version] >> workspace >> EmployeeSelfService >> ExpenseSheets >> ExpenseSheets_Heading.json.
2. In the search results, double-click the heading.json file for the iAccess version you want to customize. The Extender opens the heading.json file.
3. Click the **Copy selected file as extension to active project** icon.
4. In the heading.json file, scroll to the "selection" property. Under this, the "options" property lists the available search filters for a workspace.
5. Edit the "T$title" property for the search filter you want to rename. For example, if you want to rename the **My Open** search filter, edit the code as follows:

From

```markup
"options": {
  "MyOpenExpenseSheets": {
    "T$title": "My Open",
    "restriction": "not FullyApproved and EmployeeNumber = employeeNumber()"
  }
```

To

```markup
"options": {
  "MyOpenExpenseSheets": {
    "T$title": "My Open Expenses",
    "restriction": "not FullyApproved and EmployeeNumber = employeeNumber()"
  }
```

6. Click the **Save** icon.
7. To deploy your changes, click the **Commit and Push changes to Server Repository** icon.
8. In the window that displays:

   a. Fill out the **Commit message** field with the details of your customization. b. You can select the **Deploy immediately** check box. c. Click **OK**.

   The Extender pushes your changes to the server repository, and displays a progress bar.
9. In the Deployment success message that displays, click **OK**.
10. To view your changes:

    a. Pull up the browser window with your open iAccess system. b. Hold down the CTRL key, and click **Refresh**.

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.

```markup
{
  "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.

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

Data Bindings

Copy to come.

Layouts

Copy to come.

Wizards

Copy to come.

Customization Procedures

Rename a Card Action

iAccess has several actions available in the card part of each workspace. You can customize the names of most of these actions.

To rename a card action:

1. In the **Standard and Solution Files Filter Search** field, type the name of the ActionBar.json file that corresponds to the workspace that you want to customize. You can also specify the file path. For example, if you want to rename the **Submit** action in the card part of the Expenses workspace, search for the ExpenseSheets_ActionBar.json file, or specify the following file path: Web >> iaccess >> [iAccess version] >> workspace >> EmployeeSelfService >> ExpenseSheets >> ExpenseSheets_ActionBar.json.
2. In the search results, double-click the ActionBar.json file for the iAccess version you want to customize. The Extender opens the ActionBar.json file.
3. Click the **Copy selected file as extension to active project** icon.
4. In the ActionBar.json file, scroll to the instance of the "actions" property that contains the action you want to rename. This property lists the actions that you are allowed to customize.
5. Edit the "T$title" property for the action you want to rename. For example, if you want to rename the **Submit** action, edit the code as follows:

From

```markup
"actions": [
  {
    "T$title": "Submit",
    "source": "SubmitExpenseSheet"
  },
```

To

```markup
"actions": [
  {
    "T$title": "Submit My Expense",
    "source": "SubmitExpenseSheet"
  },
```

6. Click the **Save** icon.
7. To deploy your changes, click the **Commit and Push changes to Server Repository** icon.
8. In the window that displays:

   a. Fill out the **Commit message** field with the details of your customization. b. You can select the **Deploy immediately** check box. c. Click **OK**.

   The Extender pushes your changes to the server repository, and displays a progress bar.
9. In the Deployment success message that displays, click **OK**.
10. To view your changes:

    a. Pull up the browser window with your open iAccess system. b. Hold down the CTRL key, and click **Refresh**.

Rename a Field

To rename a field:

1. In the **Standard and Solution Files Filter Search** field, type the name of .json file for the part of the workspace that contains the field you want to rename. You can also specify the file path. For example, if you want to rename the **Total Amount** field found in the card part of the Expenses workspace, search for the ExpenseSheets_Card_Row.json file, or specify the following file path: Web >> iaccess >> [iAccess version] >> workspace >> EmployeeSelfService >> ExpenseSheets >> ExpenseSheets_Card_Row.json.
2. In the search results, double-click the .json file for the iAccess version you want to customize. The Extender opens the .json file.
3. Click the **Copy selected file as extension to active project** icon.
4. In the .json file, scroll down to the "column" property for the column that contains the field you want to rename.
5. Edit the "T$title" property for the field you want to rename. For example, to change the name of the **Total Amount** field to **Full Amount**, edit the code as follows:

From

```markup
"unitField": {
  "T$title": "Total Amount",
  "source": "AmountBase",
```

To

```markup
"unitField": {
  "T$title": "Full Amount",
  "source": "AmountBase",
```

6. Click the **Save** icon.
7. To deploy your changes, click the **Commit and Push changes to Server Repository** icon.
8. In the window that displays:

   a. Fill out the **Commit message** field with the details of your customization. b. You can select the **Deploy immediately** check box. c. Click **OK**.

   The Extender pushes your changes to the server repository, and displays a progress bar.
9. In the Deployment success message that displays, click **OK**.
10. To view your changes:

    a. Pull up the browser window with your open iAccess system. b. Hold down the CTRL key, and click **Refresh**.

API Reference

The following section gives an overview of all parts of the configuration API. Each configuration part consists of a JSON object with a set of properties.

Application

The application name and version.

Properties: version

version

The version of this particular application. This version number is NOT related to core Maconomy versions.

Type	Required	Default Value
string	true	*n/a*

AuthenticationType

Representation of available methods of logging in to iAccess.

Configuration API

The core application configuration is split into four main areas: authentication, platform, shell, and workspace. The entire configuration has an API version used to guarantee compatibility and enforce upgrades. Finally, the configuration includes a dictionary of the static terms embedded in iAccess. These terms are localized before the client receives the configuration from the REST API.

Properties: api, application, authentication, platform, shell, workspace

api

The version of the configuration format that this representation conforms to. Versions follow the SemVer standard.

Type	Required	Default Value
string	true	*n/a*

application

The application name and version.

Type	Required	Default Value
Application	true	*n/a*

authentication

Authentication determines preferred login mechanism as well as general configuration of the login flow.

Type	Required	Default Value
IAuthenticationConfiguration	true	*n/a*

platform

General configuration that spans the entire application

Type	Required	Default Value
IPlatformConfiguration	true	*n/a*

shell

The application shell covers the menu, documentation, notifications and auxiliary features such as settings, change password and about information.

Type	Required	Default Value
IShellConfiguration	true	*n/a*

workspace

Configuration of general properties that span all workspaces as well as configuration of individual workspaces in terms of data bindings and layouts.

Type	Required	Default Value
IWorkspaceConfiguration	true	*n/a*

Domain

No description available

Properties: enabled

enabled

No description available

Type	Required	Default Value
boolean	true	*n/a*

EmployeeLookup

Configuration properties related to looking up employees for mentioning in messages.

Properties: restriction, limit

restriction

An optional restriction to be applied when searching for employees. For example: `"not Blocked"`.

Type	Required	Default Value
string	false	*n/a*

limit

The maximum number of employees that will be loaded for the employee search. Default: `100000`

Type	Required	Default Value
number	false	n/a

Formatting

Configure formatting rules for data types.

Properties: amount, autointervals, boolean, date, enum, integer, real, string, time, timeduration, pp:color, pp:datetime, pp:double, pp:float, pp:guid, pp:integer, pp:string

amount

No description available

Type	Required	Default Value
AmountFormatting	false	n/a

autointervals

No description available

Type	Required	Default Value
AutoIntervalsFormatting	false	n/a

boolean

No description available

Type	Required	Default Value
BooleanFormatting	false	n/a

date

No description available

Type	Required	Default Value
DateFormatting	false	n/a

enum

No description available

Type	Required	Default Value
EnumFormatting	false	n/a

integer

No description available

Type	Required	Default Value
IntegerFormatting	false	n/a

real

No description available

Type	Required	Default Value
RealFormatting	false	n/a

string

No description available

Type	Required	Default Value
StringFormatting	false	n/a

time

No description available

Type	Required	Default Value
TimeFormatting	false	n/a

timeduration

No description available

Type	Required	Default Value
TimeDurationFormatting	false	n/a

pp:color

No description available

Type	Required	Default Value
PlusplusServerColorFormatting	false	n/a

pp:datetime

No description available

Type	Required	Default Value
PlusplusServerDateTimeFormatting	false	n/a

pp:double

No description available

Type	Required	Default Value
PlusplusServerDoubleFormatting	false	n/a

pp:float

No description available

Type	Required	Default Value
PlusplusServerFloatFormatting	false	n/a

pp:guid

No description available

Type	Required	Default Value
PlusplusServerGuidFormatting	false	n/a

pp:integer

No description available

Type	Required	Default Value
PlusplusServerIntegerFormatting	false	n/a

pp:string

No description available

Type	Required	Default Value
PlusplusServerStringFormatting	false	n/a

IAmountFormatting

No description available

Properties: short, decimalPlaces

short

No description available

Type	Required	Default Value
unknown	false	n/a

decimalPlaces

No description available

Type	Required	Default Value
number	false	n/a

IAuthenticationConfiguration

Set up authentication schemes.

Properties: preferred, methods

preferred

The preferred method of performing login. This method determines the entry route when arriving from the bars, default if URL.

Type	Required	Default Value
AuthenticationType	true	n/a

methods

Used to disable/enable individual login methods.

Type	Required	Default Value
Methods	true	n/a

IAutoTimestampFormatting

No description available

IAvailableFormats

No description available

IBooleanFormatting

No description available

IConversationConfiguration

Configuration of the conversation functionality.

Properties: enabled, retrieveinterval, emptyresultsdelay, conversationTypes

enabled

Determines if the conversation functionality is enabled in Access. For example: `TRUE`.

Type	Required	Default Value
string	true	n/a

retrieveinterval

The retrieve interval in seconds. The list of unread messages in the message center component is automatically refreshed once per this time interval. For example: `300`, indicates that refresh happens every minute.

Type	Required	Default Value
number	true	n/a

emptyresultsdelay

Configuration properties related to loading up employees for mentioning in messages.

Type	Required	Default Value
TimeIntervalLeaf	false	n/a

conversationTypes

A mapping between conversation types and workspaces supporting these conversations. This mapping is used by the message center to open a workspace associated with a conversation, and to navigate to a record to which the conversation is attached.

Type	Required	Default Value
string → ConversationType	true	n/a

IConversationType

No description available

Properties: workspace, parameters

workspace

The name of the target workspace for the conversation type.

Type	Required	Default Value
string	true	n/a

parameters

A map of uri parameters for the workspace linking to primary keys of the object attached to the conversation.

Type	Required	Default Value
string → string	false	n/a

IDateFormatting

No description available

IEnumFormatting

No description available

IEnumerationConfiguration

No description available

Properties: limit

limit

The maximum number of resources popups that should be loaded from the backend. Default: `10000`

Type	Required	Default Value
number	false	n/a

IFormats

No description available

Properties: audiences, fixed, available

audiences

No description available

Type	Required	Default Value
string	false	n/a

fixed

No description available

Type	Required	Default Value
boolean	false	n/a

available

No description available

Type	Required	Default Value
AvailableFormats	false	n/a

IGlobalDefinitions

Definitions are used to share common reticulate concepts and items across different workspaces and areas of Access.

Properties: ids, infobubbles, icons, styles, tables, searchLayouts

ids

Predefined GID that can be referred and used across all workspaces.

Type	Required	Default Value
string → unknown	true	n/a

infobubbles

Info bubbles that can be used across all workspaces.

Type	Required	Default Value
string → IRecursiveInfobubbleDefinition	true	n/a

icons

Mapping between symbolic names to pixel width.

Type	Required	Default Value
Icons	true	n/a

styles

Reusable style definitions that can be applied across all workspaces.

Type	Required	Default Value
string → unknown	true	n/a

colors

Reusable colors used across different workspaces.

Type	Required	Default Value
string → string	true	n/a

tables

Global configuration for tables.*

Type	Required	Default Value
TableProperties	true	n/a

searchLayouts

Layouts used for performing different kinds of searches across all workspaces.

Type	Required	Default Value
IGlobalSearchLayouts	true	n/a

IGlobalSearchLayouts

Represents a hierarchy of search layout configurations. Search layouts are defined per search container where a search will be performed. For example, the field `CONTAINER` on a time sheet line is associated with the search container `PROJECTMANAGER`. From the `DEFAULTS` hierarchy. Therefore, for searches started from this field, the search inherits the the `PROJECTMANAGER` container with the value.

Properties: containers

containers

No description available

Type	Required	Default Value
array [unknown]	true	n/a

IIntegerFormatting

No description available

Properties: groupingseparator, short, decimalPlaces

zeroSuppression

True if zero values should be suppressed.

Type	Required	Default Value
boolean	false	n/a

short

No description available

Type	Required	Default Value
unknown	false	n/a

decimalPlaces

No description available

Type	Required	Default Value

Type	Required	Default Value
number	false	n/a

ILanguageConfiguration

No description available.

Properties: [supported](#), [fixed](#)

preferred

The preferred locale, e.g. 'de_DE'

Type	Required	Default Value
string	true	n/a

fixed

True if the language is fixed and cannot be changed by the end-user. This removes the language selector from the both login screen and settings dialog.

Type	Required	Default Value
boolean	true	n/a

IMenu

The menu provides access to the workspaces that are available to the logged in user. It menu consists of workspaces and groups. The workspaces may be different and in a different order in menu groups and items depending on the user's privileges and preferences.

Properties: [workspaceWithoutGroups](#), [defaultWorkspace](#), [groups](#)

workspaceWithoutGroups

Workspaces that are not contained in groups.

Type	Required	Default Value
array [IWorkspaceSummary]	false	n/a

defaultWorkspace

A prioritized list of candidates (default or startup workspaces). On startup, this list is evaluated and the first candidate where the condition is satisfied is chosen as default for the logged in user.

Type	Required	Default Value
array [unknown]	false	n/a

groups

The set of menu groups available in the menu.

Type	Required	Default Value
array [IMenuGroup]	true	n/a

IMenuGroup

A group of workspaces grouped under a common title.

Properties: [name](#), [expanded](#), [visible](#), [title](#)

items

The workspaces or links contained in this menu group.

Type	Required	Default Value
array [unknown]	false	n/a

expanded

Determines whether a menu group is expanded or collapsed on login.

Type	Required	Default Value
unknown	false	n/a

visible

A visibility conditate determining whether this layout part is shown.

Type	Required	Default Value
string	false	n/a

title

An mandatory, localizable title.

Type	Required	Default Value
string	true	n/a

INotificationConfiguration

Represents the configuration of notification reload and recalculation intervals.

Properties: [recalculation](#), [types](#)

recalculation

The recalculation settings determine when the first recalculation of the end-user's notifications occur and what the time-lapse between subsequent recalculations is, all values are specified in minutes.

Type	Required	Default Value
INotificationRecalculation	true	n/a

types

Supported notification types as specified in the WWL files.

Type	Required	Default Value
array < INotificationType >	true	n/a

INotificationType

No description available.

Properties: [workspace](#), [parameters](#)

workspace

The name of the workspace

Type	Required	Default Value
string	true	n/a

parameters

A map of all parameters for the workspace linking to the Focus key to question.

Type	Required	Default Value
dictionary < string, string >	false	n/a

IPeoplePlannerColorFormatting

No description available.

IPeoplePlannerDateTimeFormatting

No description available.

IPeoplePlannerDoubleFormatting

No description available.

IPeoplePlannerFloatFormatting

No description available.

IPeoplePlannerGuidFormatting

No description available.

IPeoplePlannerIntegerFormatting

No description available.

IPeoplePlannerStringFormatting

No description available.

IPlatformConfiguration

No description available.

Properties: [usageTracking](#), [containers](#), [traceThread](#), [language](#), [enums](#), [maconomyContainerWebServiceVersion](#)

usageTracking

No description available.

Type	Required	Default Value
IUsageTrackingConfiguration	true	n/a

containers

Additional configuration for containers on a Maconomy installation.

Type	Required	Default Value
unknown	false	n/a

traceThread

Configuration of the logs sheet functionality.

Type	Required	Default Value
ITraceThreadConfiguration	false	n/a

language

The language configuration determines default language and whether the end user can change language.

Type	Required	Default Value
ILanguageConfiguration	true	n/a

enums

Configuration of enumerations.

Type	Required	Default Value
IEnumerationConfiguration	false	n/a

maconomyContainerWebServiceVersion

The version number of the Maconomy Container Web Service REST API to use. If not specified, then version two will be used.

Type	Required	Default Value
number	false	n/a

IRealFormatting

No description available.

Properties: [zeroSuppression](#), [short](#), [decimalPlaces](#)

zeroSuppression

True if zero values should be suppressed.

Type	Required	Default Value
boolean	false	n/a

short

No description available.

Type	Required	Default Value
unknown	false	n/a

decimalPlaces

No description available.

Type	Required	Default Value
number	false	n/a

IReusableInfoBubbleDefinition

A definition of a reusable info bubble.

Properties: [parameters](#), [width](#), [position](#), [title](#), [subTitle](#), [rows](#), [name](#), [label](#)

parameters

No description available.

Type	Required	Default Value
IReusableInfoBubbleParameters	true	n/a

width

You can optionally specify the width of the info bubble. In 100 px (ex: 100) px text. 60 pct (ex: 60) pct. 60 pct or 400 px The default behavior is: 1 column is rendered as 2 columns is rendered as 3 columns is rendered 4+

Type	Required	Default Value
string	false	n/a

position

An optional position of the info bubble. Right is default.

Type	Required	Default Value
string	false	n/a

title

The title or heading of the info bubble

Type	Required	Default Value
unknown	false	n/a

subTitle

The sub heading of the info bubble

Type	Required	Default Value
unknown	false	n/a

rows

The rows or content of the info bubble

Type	Required	Default Value
array [unknown]	true	n/a

parse

The parse used to resolve unqualified references in the scope. If unspecified, the parent scope is inherited.

Type	Required	Default Value
string	false	n/a

label

A frame or identifier to give this component. This may be used when the HTML is generated, when hyperlinks are generated as well as other processes such automated testing. Please use a value that is complete. Alphanumeric, dash and underscore characters only. Avoid using a dot '.' in the name, it will be converted to dashes which clash on the original value as you type. Values must not begin with 'dm-' since this is reserved for internal use.

Type	Required	Default Value
string	false	n/a

IReusableInfoBubbleParameters

Defines the parameters for the reusable info bubble.

Properties: [fields](#), [forkeyKeys](#)

fields

No description available.

Type	Required	Default Value
array [string]	true	n/a

forkeyKeys

No description available.

Type	Required	Default Value
array [string]	false	n/a

ISettings

Settings represents formats, language, and other end-user configurable properties.

Properties: [formats](#), [minutesThreshold](#), [menuSearch](#), [formatting](#), [userInactivityInterval](#)

formats

The formats configuration determines the data formatting in a way which can be changed by the end-user.

Type	Required	Default Value
IFormats	true	n/a

minutesThreshold

The minute threshold determines when a time entry is interpreted as minutes or as hours. Default is '32' which means that an entry of '32' will be interpreted as 32 minutes, and an entry of '77' will be interpreted as 1.3 hours.

Type	Required	Default Value
number	false	n/a

menuSearch

The menuSearch configuration determines if the search component is enabled in the menu.

Type	Required	Default Value
boolean	true	n/a

formatting

Configure formatting rules for data types.

Type	Required	Default Value
IformattingRule	true	n/a

userInactivityInterval

The user inactivity interval in minutes. Access will perform certain automated operations, such as periodically refreshing communication or notifications, only when the user is active. If the inactivity interval exceeds the number of minutes specified in this property, then such automatic refresh will be paused.

Type	Required	Default Value
number	false	n/a

IShellConfiguration

The application shell includes the menu, documentation, notifications and auxiliary features such as settings, change password and about information.

Properties: [menu](#), [documentationUrl](#), [settings](#), [notifications](#), [permissions](#)

menu

No description available.

Type	Required	Default Value
IMenu	true	n/a

documentationUrl

The URL used for header help. Can be a complete string with variables such as:

Type	Required	Default Value
string	true	n/a

settings

No description available

Type	Required	Default Value
Settings	true	n/a

notifications

No description available

Type	Required	Default Value
NotificationConfiguration	true	n/a

conversations

No description available

Type	Required	Default Value
ConversationConfiguration	true	n/a

IStringFormatting

No description available

ITableProperties

Common table properties.

Properties: name, displayNameFormat, columns, columnChooser, sortOrder, groupBy, aggregations, pivot, initialDensity, enableFilterBar, enableGlobalSearch, rowNumbers, enableCellEditing, enableRowEditing, showRowGroupFooter, showGrandTotal, rowPerPageOptions, defaultRowsPerPage, cellSelection, undoRedoLimit, clipboard, exportFormats, conditionalFormatting, toolbar, rowHeight, headerHeight, footerHeight, rowStyleRules, columnStyleRules, enableAnimations, enableRowDragging, enableColumnDragging, enableMultiSelectMode, enableSelectionCheckbox, selectionMode, selectionCheckboxPosition, enableCopyOnSelect, enableSelectionAggregation, selectionAggregations, enableFocusMode, enableFreezeRows, enableFreezeColumns, enableSelectionPersistence, enableRowExpansion, rowExpansionMode, rowExpansionTemplate, enableMasterDetail, enableTreeMode, treeDataSource, treeDataFieldMapping, enableTreeSelectionCascade, enableTreeLazyLoading, enableTreeDragDrop, enableTreeSortMode, enableTreeFilterMode, enableTreeAutoFit, enableTreeExpandAnimation, enableTreeCollapseAnimation, enableTreeIndentGuides, enableTreeNodeCount, enableTreeNodeIcons, enableTreeNodeCheckbox, enableTreeNodeCheckboxCascade, enableTreeNodeCheckboxIndeterminate, enableTreeNodeCheckboxLeafOnly, enableTreeNodeCheckboxPropagation, enableTreeNodeCheckboxPropagationOnly, enableTreeNodeCheckboxPropagationOnlyLeaf, enableTreeNodeCheckboxPropagationOnlyLeafOnly, enableTreeNodeCheckboxPropagationOnlyLeafOnly, enableTreeNodeCheckboxPropagationOnlyLeafOnly

```
    "font-weight": "bold",
    "Text-align": "right"
  }
},
```

To

```
"columns": [
  {
    "ref": "EntryText",
    "text": {
      "T$title": "Total"
    },
    "css": {
      "border-right": "1px solid transparent",
      "font-weight": "bold",
      "Text-align": "right"
    }
  },
```

From 2.1 to 2.2

Update Foreign Key Definitions

If you created iAccess custom layouts specifically for Maconomy 2.4.0 and are upgrading to iAccess 2.2 (which requires you to run Maconomy 2.4.1 in the backend), you need to update your customizations according to the new naming conventions.

Update your custom usage of named references (that is, foreign key definitions) according to the following list of changed names:

Dialog: Approve Expense Sheets By Employee

```
Old: "Table_ExpenseSheetHeader"
New: "Table_CopyFromExpenseSheetNumber_ExpenseSheetHeader"

Old: "ApprovalRelationInstanceKey_ExpenseSheetHeader"
New: "Table_ExpenseSheetHeader"

Old: "ExpenseSheetHeaderEmployeeNumberVar_Employee"
New: "Table_Employee"

Old: "SuperiorEmployee_Employee"
New: "Table_SuperiorEmployee_Employee"

Old: "SecretaryEmployee_Employee"
New: "Table_SecretaryEmployee_Employee"

Old: "TutorEmployee_Employee"
New: "Table_TutorEmployee_Employee"

Old: "ApprovalGroupInstanceKey_ApprovalGroup"
New: "Table_ApprovalGroup"
```

Dialog: Approve Purchase Order Lines By Employee

```
Old: "ApprovalLine_PurchaseOrderLine"
New: "Table_PurchaseOrderLine"
```

Introduction of Action Groups

Previous to this release, all actions were shown in a drop-down list called "Other actions". Actions are now shown on a line together with "Save" and "Revert". The following example:

```
{
  ...
  "actions": [
    {
      "title": "Submit",
      "source": "SubmitTimeSheet"
    },
    {
      "title": "Submit",
      "source": "SubmitTimeSheet"
    }
  ]
  ...
}
```

Renders actions as:

```
Save | Revert | Submit | Reopen
```

You can also make action groups. This means you can have one or more groups of actions next to the row of action buttons.

```
{
  ...
  "actions": [
    {
      "title": "Submit",
      "source": "SubmitTimeSheet"
    },
    {
      "title": "Submit",
      "source": "SubmitTimeSheet"
    },
    "actions": [
      {
        "title": "Print",
        "source": "PrintTimeSheet"
      },
      {
        "title": "Copy From",
        "source": "CopyTimeSheet"
      }
    ],
    "title": "Even more actions",
    "actions": [
      {
        "title": "Extra action"
        ...
      },
      {
        "title": "Extra action 2"
        ...
      }
    ]
  ]
  ...
}
```

Renders actions as:

```
Save | Revert | Submit | Reopen | Other actions | Even more actions

                                 Print           Extra action
                                 Copy From       Extra action 2
```

If you leave out "title" for an action group, the group is labelled "Other actions" by default.

From 2.1 to 2.1.1

Reusable info-bubbles

You need to manually update all custom reusable info-bubbles.

The format for the **parameters** atribute, found in the info-bubble definition, is updated.

From:

```
{
  "parameters": [
    fieldName1,
    fieldName2,
    ...
  ],
  ...
}
```

To:

```
{
  "parameters": {
    "fields": [
      fieldName1,
      fieldName2,
      ...
    ]
  },
  ...
}
```

From 2.0.x to 2.1

iAccess 2.1 includes the parameterized workspace panes feature, a functionality that introduces a breaking change in workspace and container configurations.

Workspace Definition

You need to manually update all custom workspaces.

The format for the **parameters** attribute-found under the **container** definition in the dataBindings section of the workspace-is updated

From:

```
{
  urlParameterName: fieldName,
  ...
}
```

To:

```
{
  fieldName: parameterObject,
  ...
}
```

where `parameterObject` can be a URL parameter definition or an expression parameter definition.

1. URL parameter

```
{
  "urlParameter": name
}
```

where `name` is a string with the name of the parameter that will be used in the URL.

2. Expression parameter

```
 {
  "expression": expression
 }
```

where `expression` is a string with the Expression Language expression providing the value for the parameter. The expression may refer to fields in any pane of the workspace, except for the pane where this parameter is defined as well as this pane's descendants.

For example, the `WeeklyTimeSheets.json` file contained the following lines:

```
{
  "date": "DateVar",
  "EmployeeNumber": "EmployeeNumberVar"
}
```

After the update, this is now converted to:

```
{
  "DateVar": {
    "urlParameter": "date"
  },
  "EmployeeNumberVar": {
    "urlParameter": "EmployeeNumber"
  }
}
```

Layout Definition

Some ***breaking changes*** are applied to the layout definitions.

1. Label is renamed to text. This means that all instances of `label` elements must be changed to `text` elements.

```
{
  "label": {
    "template": "Some string"
    ...
  }
  ...
}
```

must be changed to

```
{
  "text": {
    "template": "Some string"
    ...
  }
  ...
}
```

2. Action rows have been removed. Previously, you could add a row of actions anywhere in the layout. Now, you can only add actions to the top of the layout or in a tab.

A `record` element can now be added to the top of the layout. This can be thought of as the heading defining the shown entity. In the following example, the `title` defines week information for the time sheet, and the `subTitle` defines employee information. The `status` shows the status of the time sheet, and the `actionBar` contains the actions.

```
"layout": {
  ...
  "record": {
    "pane": "TimeRegistrationCard",
    "T$title": "Week ^{WeekNumberVar}^{PartVar}: ^{PeriodStartVar} - ^{PeriodEndVar}",
    "subTitle": {
      "reference": {
        "description": "EmployeeNameVar",
        "key": "EmployeeNumberVar"
      }
    },
    "status": {
      "source": "SelectedTimeSheetStatusVar",
      ...
    },
    "actionBar": {
      ...
      "actions": [
        {
          "source": "SubmitTimeSheet",
          ...
        }
        ...
      ]
    }
  }
  ...
}
```

Actions can be added to the tabs as shown in the following example:

```
{
  ...
  "tabs": [
    {
      "T$title": "Expense Sheets",
      ...
      "actions": [
        {
          "source": "ApproveExpenseSheetsByEmployeeCard.ApproveAll"
        }
      ]
      ...
    }
  ]
  ...
}
```

Container Definition

This functionality also introduces a breaking change in the container configuration (`Containers.json`).

The `singleton` attribute under the `card` pane configuration is now replaced by three attributes:

- `singletonIdentifiers` - This maps field names in the card pane to REST API identifiers, such as card.datevar.
- `stateParameters` - This lists the fields in the card pane of this container that do not persist on the server side, and thus need to be maintained on the client side.
- `navigationParameters` - This lists the fields in the card pane of this container that control the internal navigation in the card and the table. These fields cannot be changed at the same time as other fields in the card or table panes.

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 UI-iAccess channel under the Maconomy Knowledge Center team in Microsoft Teams. Engineering is actively monitoring that channel 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.

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"
  }
```

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`
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
```

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.

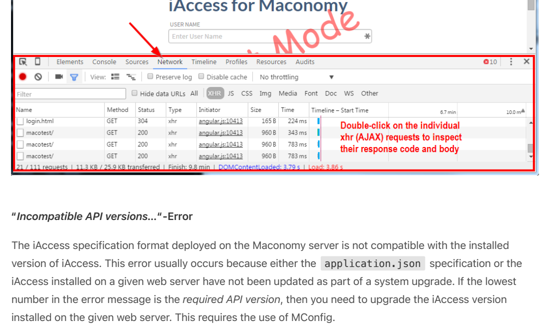
Localization

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.

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 start a conversation in our UI-iAccess channel under the Maconomy Knowledge Center team in Microsoft Teams. You can also opt to 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 @chrome 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 [@IISColons]. 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.