Module Suite User Manual

AnswerModules Sagl
AnswerModules Sagl
Copyright © 2013-2022 AnswerModules Sagl

Module Suite User Manual

About this guide

- Audience and objective 22
- Prerequisites 22

Release Notes

Module Suite 3.0.0 23

- Version 3.0.0 (Generoso) - Release notes 23
  - Module Suite Compatibility Matrix 23
  - Major Changes in version 3.0.0 24
    - IDEs 24
      - Filtering 25
      - Remote snippets repositories 26
      - Concurrent Script Editing 26
    - Content Script 27
    - Administration 27
    - Beautiful WebForms 27
      - New V5 library 27
    - New widgets for library V4 28
    - Smart Pages 28
      - Commands definition cache 28
      - Actions definition cache 28
      - Overrides optimization 28
        - How OM is created ? 30

- All Enhancements in version 3.0.0 32
- Issues Resolved in version 3.0.0 33

Module Suite 2.9.0 35

- Version 2.9.0 (Ceresio) - Release notes 35
  - Module Suite Compatibility Matrix 36
  - Major Changes in version 2.9.0 36
    - Content Script 36
      - Extension for Core Share (NEW) 36
      - Extension for OAuth Services (NEW) 36
      - Extended logging functionality 37
      - Other improvements 37
    - Beautiful WebForms 37
      - Improved SmartUI compatibility for widgets. 37
    - Smart Pages 37
      - "CSSmartMenu" has become "CSSmartView" 37
      - Global revision of Smart Pages widgets 43
      - New Smart Pages widgets 43
      - Added support for flexbox on Smart Pages used as Smart View tiles. 44
      - Revised Tree Widget 44
  - All Enhancements in version 2.9.0 48
  - Issues Resolved in version 2.9.0 49

Module Suite 2.8.0 52

- Version 2.8.0 - Release notes 52
  - Module Suite Compatibility Matrix 53
  - All Enhancements in version 2.8.0 53
  - Issues Resolved in version 2.8.0 54

Module Suite 2.7.0 56

- Version 2.7.0 - Release notes 56
  - Module Suite Compatibiliy Matrix 56
  - Major Changes in version 2.7.0 57
    - Extension Distributed Agent (NEW) 57
    - Smart Pages 57
  - All Enhancements in version 2.7.0 57
  - Issues Resolved in version 2.7.0 58

Module Suite 2.6.0 60

- Version 2.6.0 - Release notes 61
  - Module Suite Compatibiliy Matrix 61
  - Major Changes in version 2.6.0 61
    - Content Script 62
    - Beautiful WebForms 62
      - Form Builder 62
      - Extension for Workflow 62
      - Extension SFTP (NEW) 62
    - Smart Pages (NEW) 62
  - All Enhancements in version 2.6.0 63
  - Issues Resolved in version 2.6.0 63

Module Suite 2.5.0 64

- Version 2.5.0 - Release notes 65
  - Module Suite Compatibiliy Matrix 65
  - Major Changes in version 2.5.0 SP1 65
  - Major Changes in version 2.5.0 65
    - Content Script 66
      - Extension Engeenering (NEW) 66
  - All Enhancements in version 2.5.0 66

- Issues Resolved in version 2.5.0 66

Module Suite 2.4.0 67

- Version 2.4.0 - Release notes 68
  - Module Suite Compatibility Matrix 68
  - Major Changes in version 2.4.0 68
    - Beautiful WebForms 69
      - Form Builder 69
    - Content Script 69
      - Extension Package for Blazon (NEW) 69
      - Extension Package for the integration with S3 by AWS (NEW) 70
      - Extension Package for xECM (NEW) 70
      - Extension Package for Office documents 70
  - All Enhancements in version 2.4.0 71
  - Issues Resolved in version 2.4.0 71

Module Suite 2.3.0 72

- Version 2.3.0 - Release notes 72
  - Major Changes in version 2.3.0 73
    - Beautiful WebForms Form Builder 73
      - Enhanced support for Internationalization 73
      - Inline FormTemplate Manipulation 73
    - Content Script 73
      - Auditable and indexable 73
      - Scheduling and Callbacks 73
  - All Enhancements in version 2.3.0 73
  - Issues Resolved in version 2.3.0 75

Module Suite 2.2.0 76

- Version 2.2.0 - Release notes 76
  - Major Changes in version 2.2.0 77
    - License 77
    - Beautiful WebForms Form Builder 77
      - CHEH Snippets 77
      - Widget Visibility 77
      - Buttons' Icons and Colors 78
      - Inline FormTemplate Manipulation 78
      - New And Updated Widgets 78
      - Field default value 78
      - OnLoad script returns JSON Data 78
    - New Content Script APIs 78
      - DocBuilder 78
      - Callback Scripts 78
  - All Enhancements in version 2.2.0 79
  - Issues Resolved in version 2.2.0 80

Module Suite 2.1.0 81

- Version 2.1.0 - Release notes 81
  - Major Changes in version 2.1.0 82
    - License 82
    - Beautiful WebForms library of widgets 82
    - Beautiful WebForms Studio 82
    - New Content Script APIs 82
      - Web-Services API extension pack 82
  - All Enhancements in version 2.1.0 83
  - Issues Resolved in version 2.1.0 83

Module Suite 2.0.0 85

- Version 2.0.0 - Release notes 85
  - Major Changes in version 2.0.0 86
    - Support for Content Server 16 86
    - Completely renewed development environments and editors 86
    - Full revamp of Beautiful WebForms widgets and templates libraries 87
    - New Content Script APIs 87
      - Records Management API extension pack 87
      - Physical Objects API extension pack 87
      - LDAP integration API extension pack 87
      - SQL extension pack 87
    - Content Script PDF API improvements 88
    - Third party dependencies upgrade 88
    - Weblingo override functionality 88
    - Cross-script referencing 88
    - Save views as Widgets 88
    - Workflow Query builder 88
  - All Enhancements in version 2.0.0 88
  - Issues Resolved in version 2.0.0 90
  - Important Notes when updating Module Suite to version 2.0.0 92
    - Installing the new libraries 93
      - Upgrade procedure for CSFormSnippets 93
      - Upgrade procedure for CSFormTEMPLATEs 94
      - Custom Form Templates and form widgets 94

Previous releases 96

- Previous releases - Release notes 96

Architecture

Module Suite 97

- Beautiful WebForms 97
- Content Script 97
- Smart Pages 98
- Script Console 98
- Module Suite default extensions 98
  - Content Script Extension For Workflows 98
  - Content Script Extension For WebReports 99
  - Module Suite Extension For ClassicUI 99

Module Suite Extensions 99

- ModuleSuite Extension For DocuSign 99
- ModuleSuite Extension For ESign 100

Applicative Layers 100

Requirements, links and dependencies 100

- Supported Content Server versions 101
- Dependencies 101

Modules layouts 101

- Content Script 102
  - amlib 102
  - csscripts 102
  - library 102
  - override 103
- Beautiful WebForms 104

- Script console 104
  - Script Console main configuration file 105

Installation and Upgrade

Prerequisites 106

Installing the Suite 106

- Installation procedure 107
  - Deploy Content Server Modules 107
  - Install Content Server Modules 114
  - Activate and Configure 115

Installing the Suite on Unix 117

- Installation procedure 117
  - Deploy Content Server Modules 117
  - Install Content Server Modules 119
  - Activate and Configure 120

Installing Content Script 122

- Installation procedure (Windows) 122

Installing Beautiful WebForms 132

- Installation procedure (Windows) 132

Installing Smart Pages (f.k.a. Module Suite Extension for SmartUI) 135

- Prerequisites 136
- Installation procedure 136
  - Installing the Module Suite extension for SmartUI 136
  - Deploying Beautiful WebForms static resources 137

- Importing the SmartUI Extension library objects 137

Installing on a clustered environment 138

- Installation approach 138
  - Deployment on the primary node 139
- Installation procedure 139

Installing Script Console 140

- Installation procedure 140
- Configure Script Console 145

Installing Extension Packages 149

- Installation procedure 149
- Rendition Extension Package 153
  - What is it? 153
  - Install the third party rendition engine 153
    - wkhtmltopdf 153
      - Installation 153
        - Configuration 154
    - rend 155
      - Installation (Windows) 155
      - Installation (Unix) 155
    - Configuration 156
- Content Script Extension for SAP 158
  - What is it? 158
  - Extension setup 158
    - Installing the Content Script Extension for SAP 159
    - Installation validation 161
    - Configuration options 161

Installing Extension for DocuSign 162

- Prerequisites 162
- Installation procedure 163
  - Installing the Content Script Extension for DocuSign 163
  - Installing the Script Console Extension for DocuSign (OPTIONAL) 166
  - Configuration 169
- Admin dashboard 170

Applying HotFixes 172

- Hotfixes deployment 173

Upgrading Module Suite 174

- Upgrading from a previous version 174
  - Upgrading the primary node 174
    - How the library upgrade works 175
  - Upgrading a secondary node 176

Uninstalling Module Suite 177

- Uninstallation procedure 177

Content Script

Content Server object 180

- Creating a Content Script 180
  - Object's properties 181
    - Static variables 181
    - Scheduling 182
    - Impersonate 183
    - Icon Selection 183

Content Script editor 184

- Shortcuts 186
- Top Bar controls (DEVELOPER) 187
- Top Bar controls (ADMINISTRATOR) 188
- Auto-completion 189
- Code Validation
- Versions tab
- Code Snippet library
- Online Help

Language basics 190

- Statements 191
- Basic Control Structures 192
  - Flow control: if – else 192
  - Flow control: if - else if - else 192
  - Flow control: inline if - else 192
  - Flow control: switch 192
  - Looping: while 193
  - Looping: for 193
- Operators 193
- Methods and Service Parameters 194
- Properties and Fields 195
- Comments 195
- Closures 195
- Content Script programming valuable resources 196

Writing and executing scripts 196

- API Services 197
  - Content Script API Service 197

- Content Script API Objects 197
- Execution context 200
  - Request variables 201
  - Support variables 202
  - Support objects 203
- Base API 204
- Script's execution 206
  - Script's output 207
    - HTML (default) 207
    - JSON 207
    - XML 208
    - Files 208
      - Managed resources 209
  - Redirection 210
  - HTTP Code 210
- Advanced programming 210
  - Templating 210
    - Content Script velocity macros 210
  - OScript serialized data structures 213
  - Optimizing your scripts 213
    - Behaviors 213
      - BehaviorHelper 214
      - Default Behaviours 214

Working with workflows 215

- Content Script Workflow Steps 216
  - Content Script Package 216
  - Content Script Workflow Step 216
  - Workflow routing 218

Managing events (callbacks) 219

- Synchronous and Asynchronous callbacks 220
- InterruptCallbackException - transaction roll-backed 222

Extending REST APIs 223

- Extending REST APIs:CSServices 223
  - Basic REST service 223
  - Behaviour based REST services 224
    - Service example 224

Extending Content Script 226

- Create a Custom Service 226
  - Content Script SDK setup 227
  - content-script-services.xml – Service description file 234

Content Script extension for SAP 234

- Content Script Extension for SAP 234
  - Using the extension 234
    - Function execution results 235
  - SAP service APIs 237
    - API Objects 237
      - SapField 237
      - SapFunction
      - SapStructure
      - SapTable

Extension: Classic UI 238

- Customize an object's functions menu: CSMenu 238
- Customize a space's add-items menu: CSAddItems 240
- Customize a space's buttons bar: CSMultiButtons 242

- Customize a space's displayed columns: CSBrowseViewColumns 244
  - Default Columns 247
- Customize a space content view: CSBrowseView 249
- Create a custom column backed by Content Script: CSDataSources 251

Beautiful WebForms

Content Server object 254

- Creating a Beautiful WebForms View 254
- Understanding the view object 255

Form builder 256

- Layout 256
- Shortcuts 257
- Top Bar controls (DESIGNER) 258
- Top Bar controls (DEVELOPER) 260

Building views 261

- Understanding the grid system 261
- Understanding the Beautiful WebForms request life-cycle 263
- How incoming requests are processed 263
  - Lifecycle schema 264
  - Custom Logic Execution Hooks (CLEH) 265
    - Managing form fields values 266
    - Adding and removing values from multivalue fields 268
    - Form actions 269
      - Standard form actions 269
    - Custom form actions 271

- Attaching Custom information and data to a Beautiful WebForms view 273
  - ViewParams 273
  - ViewParams variables 274
  - Form Components that make use of 'viewParams' values. 275
- The widgets library 275
  - The widget configuration panel 276
- Beautiful WebForms View Templates 277
- Customize the way validation error messages are rendered 278
  - Display errors in Smart View 280

Widgets 281

- Beautiful WebForms Widgets 281
  - Model and Template 282
  - Static Resources Management 287
  - Widgets libraries 289
    - Widget Library V1 289
    - Widget Library V2 290
    - Widget Library V3 290
    - Widget Library V4 291

Extending BWF 292

- Content Script Volume 293
  - CSServices 293
  - CSFormTemplates 294
  - CSFormSnippets 295

Embed into SmartUI 297

- Embed into Smart View 297
  - Why? 297
  - Create an embeddable WebForms 297

- How to publish a Webform into a Smart View perspective 298
- ModuleSuite Smart Pages is installed 298
- ModuleSuite Smart Pages is not installed 299

Update view library 300

- Beautiful Webforms views updater 300
  - What is it? 300
  - Tool setup 300
  - Tool usage 302

Extension: Mobile WebForms 304

- What is it? 304
  - AppWorks Mobile Application 304
  - Module Suite based extension for REST APIs 305
  - Mobile WebForms Application Builder 305
- Mobile WebForms setup 305
- Using the tool 306
  - Creating the form 307
  - Implementing the Content Script end-point 307
  - Building the OpenText AppWorks Gateway Application 308

Extension: Remote WebForms 310

- What is it? 311
- Extension setup 311
- Create remote package 313
  - Using forms.createExPackage API 313
  - Using Beautiful Webforms Studio 314
- How to deploy a Beautiful WebForms remote form package 316
- Synchronize form data back to Content Server 316
  - Remote data pack files are produced on Script Console and sent over to Content Server 316

- Form data are submitted directly from Script Console 319

Smart Pages

Working with Smart Pages 322

- Basic concepts 322
  - Module Suite Tiles in the Widget Library 323
    - Tile Configuration 323
    - Tile: Content Script Result 324
    - Tile: Content Script Tile Chart 325
    - Tile: Content Script Tile Tiles 328
    - Tile: Content Script Tile Links 334
    - Tile: Content Script Tile Tree 337
    - Tile: Content Script Node Table 339
    - Embedding Beautiful WebForms views in SmartUI 345
  - Icon reference cheat sheet 345
    - Iconset Color codes 345
    - All icons 346
  - Smart Pages 348
    - Smart View overrides - general concepts 348
      - How OM is created ? 351
    - Overrides 352
      - CSSmartView:Columns 352
      - CSSmartView:Actions 354
      - CSSmartView:Commands 356

Script Console

Working with Script Console 359

- Execution modes 359
  - Command Line Shell Mode 359
  - Script Interpreter Mode 365
  - Server Mode 366
- Script repositories 366
- Script Console Internal scheduler configuration file 366

Extension for DocuSign

Working with DocuSign 368

- Creating a signing Envelope 368
  - EXAMPLE: Creating a simple envelope 368
  - EXAMPLE: Creating an envelope using a predefined template 369
- Embedded recipients 370
  - EXAMPLE: Get a pre-authenticated signing URL for an OTCS internal user 370
- Envelope status update and signed document synch back 371
  - EXAMPLE: Poll DocuSign for Envelope updates and synch back documents 371

How to

Content Script: Retrive information 373

- Nodes 373
  - Getting Content Server nodes 373
    - Getting a node given its ID 374
    - Get a list of nodes given their IDs 375
    - Get Volumes 375
    - Get Nodes By Path 375
- Users and Groups 376
  - Getting Content Server Users and Groups 376
    - Get current User 376
    - Get by member ID 376
    - Get member by the name 377
    - Get members by ID 377
- Permissions 377
  - Getting Content Server Node Permissions 377
- Categories 379
  - Getting Node Categories 379
- Classification 379
- Executing SQL queries 380
  - Execute a simple SQL query 380
  - Execute a SQL query with pagination 381
- Working with Forms 381
  - Retrive submitted data 383

Content Script: Create objects 385

- Coming soon... 386

Training Center 386

- What is it? 386
- Training Center setup 386
- Using the tool 387

Adminisration

Admnistrative pages 389

- Base Configuration 389
- Select default IP address 390
- Enable / Disable Module Suite features 391
- Logging administration 393
- Manage API Services 394
- Scheduling 395
- Manage Callbacks 396

Content Script Volume 396

- CSSystem 398
- CSFormTemplates 398
- CSHTMLTemplates 398
- CSFormSnippets 399
- CSScriptSnippets 399

Snippets and Widgets library 399

- Module Suite components and widgets library 399
  - Import and upgrade tool 400
    - Load a Library's manifest file 401
    - Analysing the incoming changes and the current Library version 401
  - Perform the initial library import 403

Tags

About this guide

Audience and objective¶

Module Suite is a collection of solutions that extend the capabilities of OpenText Content Suite and can be successfully deployed to cover a wide range of tasks, from very simple automation operations to more complex and complete applications.

This guide is structured to target those who intend to create, deploy, use, and maintain applications using Content Script, Beautiful WebForms or Smart Pages, and/or want to have a deeper understanding of the possibilities and what can be achieved with the solutions. It is also intended to help the administrators of systems that deploy Module Suite Components.

Prerequisites¶

The majority of this manual has been designed to be accessible to anyone familiar with the basic end-user features of OpenText Content Server. Readers are expected to be comfortable with creating items, navigating workspaces and searching for items. Although not essential, the following knowledge is beneficial:

- OpenText Content Server Knowledge Fundamentals
- Familiarity with the basics of HTML
- Ability to create simple LiveReports or WebReports
- Knowledge of the DTree view from the OpenText Content Suite schema

Copyright © 2013-2022 AnswerModules Sagl

Release Notes

Version 3.0.0 (Generoso) - Release notes¶

Release Date	End of AMP(*)	End of Life
2021-06-30	2024-06-30	2025-06-39

(*) Active Maintenance Period

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Modules Suite version 3.0.0.

This guide

The information presented in the on-line guide are mostly non-version specific. AnswerModules team does its best to ensure that, where necessary, is made clear that the information presented is only applicable to specific versions, however if you are looking for this version-specific documentation, you can find it here *(http://developer.answermodules.com/manuals/3.0.0)*

Script Console Installer

The Script Console installer has been temporarily removed from the Module Suite master installer. It will be reinstated in the next minor release.

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Module Suite Compatibility Matrix¶

OpenText Content Server	MS 3.0.0
Content Suite 16.2 EP6	X
Content Suite 16.2 EP7	X
Content Suite 20.2	X
Content Suite 20.3	X
Content Suite 20.4	X

Copyright © 2013-2022 AnswerModules Sagl

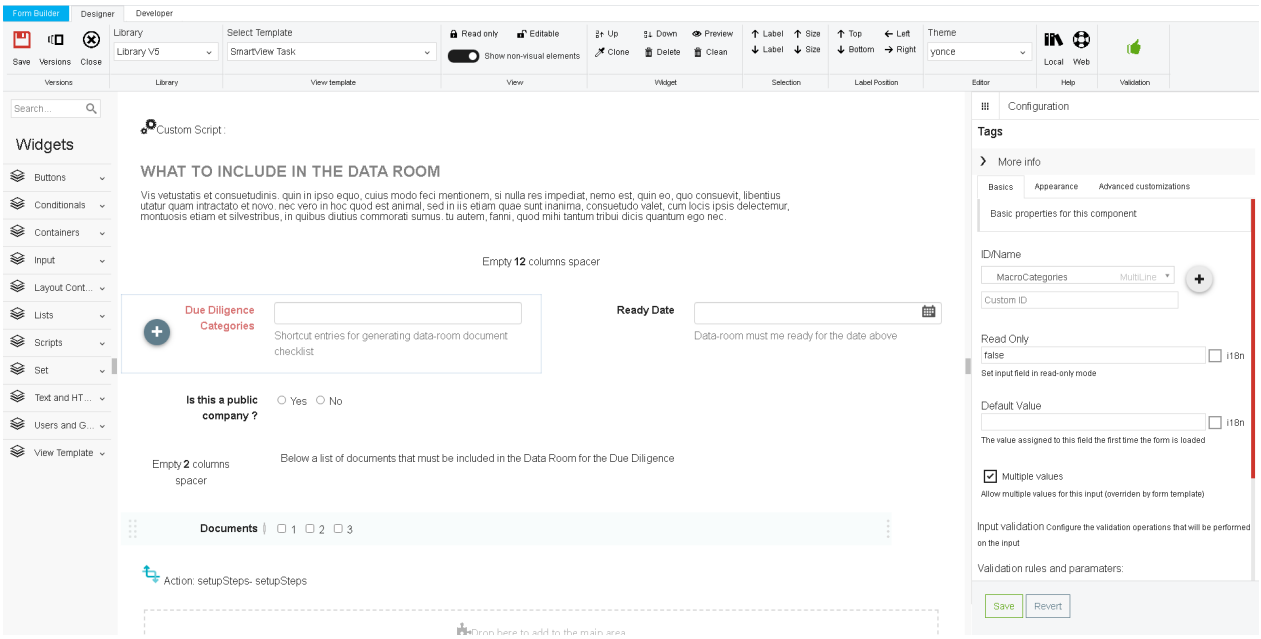
OpenText Content Server	MS 3.0.0
Content Suite 21.1	X
Content Suite 21.2	X
Content Suite 21.3	X
Content Suite 21.4	X

Major Changes in version 3.0.0¶

IDEs¶

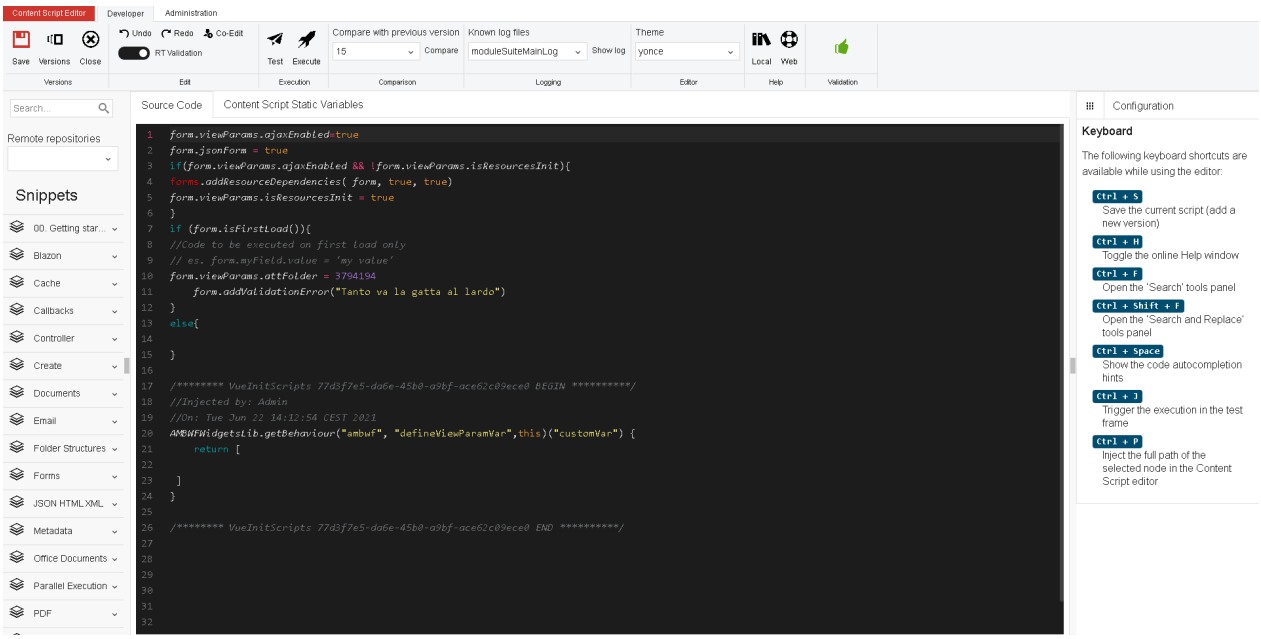
All the Module Suite's IDEs have been deeply revised. Among the new functionalities introduced: filtering for snippets and widgets, editor theme selector, log level rapid switch for Content Script Editor, remote repositories for Content Script snippets, Content Script Co-edit (Beta)

Beautiful WebForms

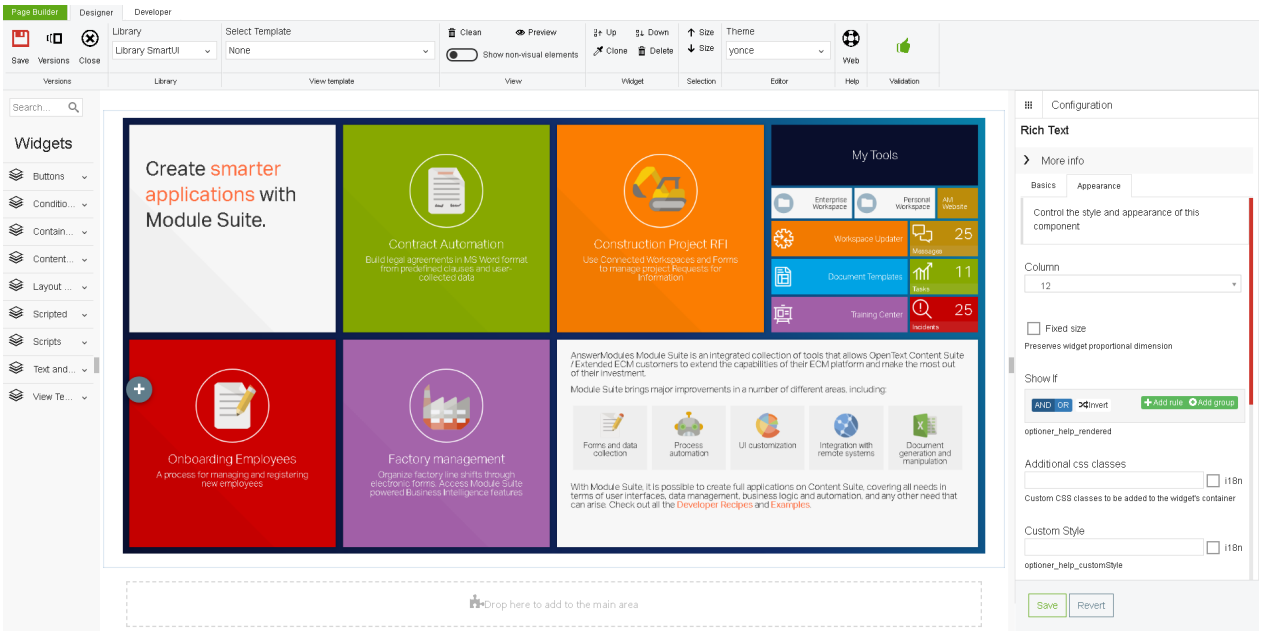


Content Script Editor

Copyright © 2013-2022 AnswerModules Sagl



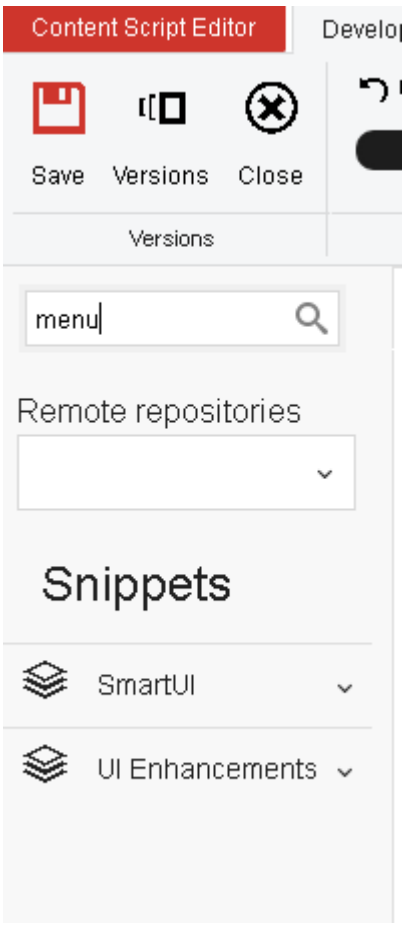
Page Builder



Filtering¶

A new filtering feature has been added to all IDEs to make it easier to select the appropriate widget or snippet in large libraries.

Copyright © 2013-2022 AnswerModules Sagl



Remote snippets repositories¶

You can now retrieve Content Script snippets from remote repositories. This allows you to maintain an enterprise KB related to Content Script (in the form of a local Snippets repository or leverage Snippet repositories offered by third-party vendors. To register a new repository you need to add a custom option in Base Configuration having the form: `amcs.msrepo[n].url =Label|repoUrl` where `n` is a number between 0 and 10.

```
amcs.msrepo0.url=Sandbox|https://developer.answermodules.com/resources/repos/sandbox
```

Concurrent Script Editing¶

Module Suite 3.0 features an experimental functionality that allows several developers to simultaneously collaborate on the editing of the same script. The functionality leverages WebRTC to establish a peer-to-peer direct connection among developers. The developer's browser will connect to the specified signaling server to find other peers. A password can be specified to encrypt all communication on the signaling server even if no sensitive information (WebRTC connection information, shared data) is shared over the signaling server.

Copyright © 2013-2022 AnswerModules Sagl

Content Script¶

Updated of all major dependencies to their latest releases. New APIs for creating and manipulating OTEmail objects and OT Pulse comments. Improvements to performances related to the retrieval of information from the database.

Administration¶

New performances tuning options available in the Module Suite base configuration.

Beautiful WebForms¶

New V5 library¶

Module Suite 3.0 introduces a new widget libary based on reactive components (Vue.js *(https://vuejs.org/)*). With this library, the already powerful engine, used to perform server-side rendering of forms' views is complemented by a reactive framework operating directly in the user's browser. When a form's view is composed using this library, the data model that is normally used in server-side rendering (form) is also serialized into a JSON object on the user's browser. This client side "model" feeds a reactive application developed with Vue.js *(https://vuejs.org/)*. Thanks to this new approach we open up the possibility of performing numerous manipulations of the data model directly on the user's client.(i.e. it is no longer necessary to perform a client-server round-trip to manipulate the data-model), which do not longer require to update (totally or partially) the page containing the view. To support and facilitate the manipulation of the data model on the user's browser, the concept of `action`, already in use for server-side manipulation of the data model, has been extended and revised. When an action is now triggered the frameworks looks for its implementation first in a client-side registry, and only if it is not found proceeds invoking the server-side business logic (CLEH). The implementation of a client-side action is pretty simple and can leverage a dedicated javascript API, whose main methods are:

```
form //represents the form object (as in CLEH scripts)
form.validate() //Triggers form validation
form.getFieldReference(index, fieldName) //Access the input widget associate to a specific form's fi
                                         //fieldName is the field's path in the form (e.g. MySet:MyF
                                         //index represent the set row
form.viewParams // The viewParams variable as in CLEH scripts
                // e.g. form.viewParam.vmVar
form.submitForm(withValidation) //Submits the form eventually triggering the form's validation first
form.getFieldValues(fieldName) //Retrives the list of values for the given form's field
form.getViewParamsValue(viewParamName) //Retrieves the value associated to the given viewParams's va
                                       //The main difference between form.viewParams.myVar and form.
                                       //is that if myVar contains an object having the following st
                                       // {ajax:{url:"https://some.service.com/endpoint", data:[]}}
                                       //the API form.getViewParamsValue('myVar') automatically fetc
                                       //remote service and caches the result in the objects 'data' p
form.setViewParamsValue(variable, value) //Set the value of a viewParams variable
form.setFieldValues(fieldName, values) //Set the values for the given field
form.setFieldReadOnly(fieldName, values) //Set the field as read-only or editable
form.addField(fieldName, index) //Adds an instance to the specified field
```

Copyright © 2013-2022 AnswerModules Sagl

```
form.removeField(fieldName, index) //Remove an instance to the specified field
form.addConstraint(fieldName,contraint, configuration) //Adds the specified validation constraint to
form.removeConstraint(fieldName, contraint) //Removes the specified validation constraint to the giv
```

CLEH scripts

If an action is triggered but it can not be found among the registered client side actions, we assume it is a server side action and the CLEH script is executed allowing server side manipulation of the data-model

New widgets for library V4¶

Added new widgets in library V4

Smart Pages¶

Commands definition cache¶

It is now possible to cache (using the distributed memcache) the result of the execution of the scripts stored under "CSSmartView:Commands" used to load the definitions of the additional commands you want to be available in Smart View pages. The scripts outcome is cached on a per-user basis. To enable the caching set to true the "amcs.amsui.volumeCache" parameter in Base Configuration. To programmatically clean the cache use the amsui.clearCache() API.

Actions definition cache¶

It is now possible to cache (using the distributed memcache) the list of scripts under "CSSmartView:Actions" used for lazy loading additional commands in the Smart View pages. The scripts list is cached on a per-user basis. To enable the caching set to true the "amcs.amsui.volumeCache" parameter in Base Configuration. To programmatically clean the cache use the amsui.clearCache() API.

Overrides optimization¶

The internal mechanisms related to how the customizations are applied to the menus and the columns of the browsing pages of the Smart View interface have been deeply revised. The content of the Overrides folder is now used to compute an Override Map (OM), specific to your repository, having the following structure:

```
OM = [
    "globals": [            (1)
        540588
    ],
    "type": [               (2)
        "144": [            (3)
            548066
        ]
    ],
    "tenants": [            (4)
```

Copyright © 2013-2022 AnswerModules Sagl

```
        "497147": [          (5)
            "globals": [     (6)
                548169
            ],
            "type": [        (7)
                "144": [     (8)
                    496932
                ]
            ],
            "ids": [         (9)
                "496931": [ (10)
                    545972
                ]
            ]
        ]
    ]
]
```

where:

- (1) identifies a list of scripts to be always executed
- (2) a list of scripts to be executed only if the current space has at least one node having of the identified type (3)
- (4) scripts to be considered only if the current space is descendant of the specified tenant (5) (a space identified by its DataID)
- (5) is a "tenant" configuration
- (6) identifies a list of scripts that must always be executed if the current space is descendant of the specified tenant (5)
- (7) a list of scripts to be executed only if the current space has at least one node having of the identified type (8) and is descendant of the specified tenant (5)
- (9) a list of scripts to be executed only if the current space has at least one node having of the identified id (10) and is descendant of the specified tenant (5)
- scripts in the OM are executed in the following order (1), (2), (6), (7), (10).

Given the above example and imagining that all the scripts in (3) (8) and (10) return the list ["comm_one","comm_two"], the resulting AOM will contain:

```
(3) AOM = [
                ...
                "S144":[commands:["comm_one","comm_two"]],
                ...
            ]
    (8) AOM = [
                ...
                "S144":[commands:["comm_one","comm_two"]],
                ...
            ]
    (10) AOM = [
                ...
                "D496931":[commands:["comm_one","comm_two"]],
                ...
            ]
- scripts in (1), (6), (10) MUST return a Map having entries of the form:
    "SXXXX":[
        commands:["comm_one", "comm_two",...],
        columns: [ //Optional
                    col_name:"col value", //value can be HTML
```

Copyright © 2013-2022 AnswerModules Sagl

```
                    ...
                    ]
    ]
    where XXXX is a valid SubType
    or
    "DYYYY":[
        commands:["comm_one", "comm_two",...],
        columns: [  //Optional
                    col_name:"col value", //value can be HTML
                    ...
                    ]
    ]
```

where YYYY is a valid node's ID.

OM is to be considered a "static" information in productive environments and as such, to guarantee optimal performances, the framework should be allowed to cached it by setting to "true" the " amcs.amsui.volumeCache" parameter int the base configuration.

When a user changes the current space, the OM is evaluated by the framework against the users' permissions and the actual override map (AOM) associated to the space is determined. AOM is determined by executing the relevant scripts in OM in the order described above. The AOM has the following form:

```
AOM = [
    "S144":[                                                                    (1)
            commands:["comm_one", "comm_two",...], //list of commands' command_key  (2)
            columns: [                                                          (3)
                        col_name:"col value", //value can be HTML
                        ...
                    ]
           ],
    "D1234":[                                                                   (4)
            commands:["comm_one", "comm_two",...], //list of commands' command_key
            columns: [
                        col_name:"col value", //value can be HTML
                        ...
                    ]
           ]
    ...
]
```

where: (1) represents commands and columns to be associated to all the nodes having the identified subtype, (3) can be omitted, (4) represents commands and columns to be associated a specific node (identified by its id), (4) takes precedence over (1).

How OM is created ?¶

In order to determine the OM, the content of the "Overrides" folder is evaluated following the logic below:

```
[
    "globals":[             (1)
        540588
    ],
    "type": [               (2)
```

Copyright © 2013-2022 AnswerModules Sagl

```
        "144": [            (3)
            548066
        ]
    ],
    "tenants": [            (4)
        "497147": [         (5)
            "globals": [    (6)
                548169
            ],
            "type": [       (7)
                "144": [    (8)
                    496932
                ]
            },
            "ids": [        (9)
                "496931": [ (10)
                    545972
                ]
            ]
        ]
    ]
]
```

- (1) Contains the list of scripts objects stored directly under "Overrides"
- (2) For each direct subfolder of "Overrides" that has a name starting by the letter "S" an entry is created in "type" map (2). The key of such entry is the target subtype (as specified in the subfolder's name) while the value is the list of scripts contained the aforementioned subfolder.
- (4) For each direct subfolder of "Overrides" that has a name starting by the letter "D" an entry is created in "tenants" map (2). The key of such entry is the tenant's DataID (as specified in the subfolder's name) while the value is the tenant OM configuration.
- (5) For each "tenant" subfolder a sub-Override Map is created (SOM). The structure of SOM is identical to the one of OM with the only difference that subfolders of a tenant subfolder having a name starting with the letter "D" are used in SOM for creating entries in the "ids" map.

Below an exemplar content of the Overrides folder

Name	ID	SubType
Overrides	00001	AnsTemplateFolder
- GlobaScript	00002	Content Script
- S144	00003	Content AnsTemplateFolder
- - Document Script	00004	Content Script
- D1234	00005	AnsTemplateFolder
- - S0	00006	AnsTemplateFolder
- - - Folder Script	00007	Content Script
- - D5678	00008	AnsTemplateFolder
- - - Node Script	00009	Content Script

Copyright © 2013-2022 AnswerModules Sagl

and the resulting OM

```
[
    "globals":[
        00002
    ],
    "type": [
        "144": [
            00004
        ]
    ],
    "tenants": [
        "1234": [
            "globals": [ ],
            "type": [
                "0": [
                    00007
                ]
            },
            "ids": [
                "5678": [
                    00009
                ]
            ]
        ]
    }
]
```

All Enhancements in version 3.0.0¶

ID	Scope	Description
#001130	Smart Pages	Add redirect and Smart View navigation capabilities to Smart Pages Controller script
#001119	Smart Pages	Added Iterator widget to Smart Page
#001120	Smart Pages	Added Include SmartPage widget to Smart Page
#001122	Beautiful Webforms	Two new uses cases for ADN
#001015	Module Suite	Content Script Performances improvements
#001097	Beautiful Webforms	Graphical request: item reference popup style with Smart View template
#001052	Smart Pages	Unable to access Content Script and some components with X-Content-Type-Options HTTP Header
#000990	Beautiful Webforms	Add 'Advanced customizations' configuration tab to the 'Custom Action Button' widget
#000672	Content Script	Getting nodes when a parent is a associated volume
#000993	Extension - Docx	Improved support for OpenDope custom XML Parts

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#000624	Content Script	Being able of creating EMAIL object (subtype 733)
#000714	Content Script	Content-Disposition handler in Content Script
#000700	Beautiful Webforms	Retrieve Pulse comments

Issues Resolved in version 3.0.0¶

ID	Scope	Description
#001090	Online Documentation	Review a little detail in Workflow routing page
#001060	Content Script	Problem with AmWorkID and AMSubWorkID with form is status of a workflow
#001103	Smart Pages	Issue on the buttons of the Buttons Group widget (Smart Page)
#001104	Beautiful Webforms	Issue on the buttons of the Buttons Group widget
#001080	Online Documentation	Rend page: missing property and problem with Linux instruction (or in the package)
#001037	Content Script	Content Script: managecallbackso.cs is used and fails on an environment based on PostgreSQL DB
#001108	Online Documentation	Docx issue with Office 365 document
#001053	Content Script	managecallbacksm.cs script fail on a case sensive DB
#000891	Beautiful Webforms	Inconsistent behavior for check-boxes when used with Widget Space Content
#001040	Beautiful Webforms	Regression 029: form server side object is not correctly initialized if some field has default value
#000994	Beautiful Webforms	ADN DropDown widget is not working
#001016	Beautiful Webforms	No error message when validation is in OnLoad or on PreSubmit
#000642	Beautiful Webforms	Unable to access API documentation for Remote WebForms feature form.amRemotePack
#001041	Content Script	Regression 029: nodes loaded through getChildren(Fast) APIs are not properly initialized when versionables

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#000944	Content Script	Document generated with a merge is corrupted if there are comments in the documents
#001034	Smart Pages	Form with Wysiwyg widget on Smartpages: dropdown menu and pop up for insert object are not showed properly
#001030	Smart Pages	Two small anomalies with Content Scripts in Smart UI: error in move operation and no way to see permissions
#001025	Content Script	Error checking attributes starting from a shortcut
#000985	Beautiful Webforms	Space Content: the uploaded document has random string in name
#001065	Beautiful Webforms	Radio selection reset after document upload
#001043	Content Script	Regression on patch 029: JDBC API is not working
#001094	Beautiful Webforms	Default in Modal Container
#001109	Smart Pages	CSSmartUIService is unable to deserialize page model if model.data contains classes that have been defined within a Script
#001056	Content Script	Regression on patch 029: timeout putting a value in cache
#000644	Beautiful Webforms	It is not possible to save an empty content script
#001028	Extension - xECM	Missing 'Inline Guide' for xecm extension
#001029	Extension - xECM	Wrong parameters type of editor autocomplete of the 'AddRole' method of the 'xecm' extension
#000939	Smart Pages	Erroneous behavior when selecting rows in Smart Pages Datatable widget
#000521	Beautiful Webforms	Source Code editor within Form Builder is initialized with wrong code when a new empty BWF view is created
#000998	Beautiful Webforms	Minor error in panel container
#001095	Beautiful Webforms	Scroll relocator: if added to a page there is a JS error
#001121	Beautiful Webforms	Error getting menu from a document
#000905	Beautiful Webforms	Datatable widget doesn't support client side actions (like pagination, search and sorting)

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#000983	Beautiful Webforms	Multiple input field overlap date picket
#001038	Smart Pages	Missing search on columns in Node Table Table Tile
#001019	Beautiful Webforms	Existing Datatables widgets have data loading issues after applying hoftix_2.9.0_001
#000886	Smart Pages	Toggle Preview not available on Smart Page
#001000	Beautiful Webforms	Plus button not clickable on FireFox
#000957	Smart Pages	Widget Nodes table - Error on selecting nodes
#000971	Beautiful Webforms	Select from list widget ignore the selected value when it is in a tab
#000953	Beautiful Webforms	Workflow comment added many times with SmartView Template when Tab Action Buttons widget is used
#0001051	Content Script	Real fields in categories are assigned Float values if accessed through GCSPrimitiveAttribute
#000995	Beautiful Webforms	Model properties are not updated for widgets in layout containers.
#000991	Beautiful Webforms	Make library update more robust
#001013	Module Suite	ScriptManager Initialization invalidates Session Cache
#000980	Smart Pages	Custom columns created with new CSSmartView:Columns functionality not showing in Smart Views

Version 2.9.0 (Ceresio) - Release notes¶

Release Date	End of AMP(*)	End of Life
2020-12-21	2023-12-21	2024-12-21

(*) Active Maintenance Period

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Modules Suite version 2.9.0.

This guide

Copyright © 2013-2022 AnswerModules Sagl

The information presented in the on-line guide are mostly non-version specific. AnswerModules team does its best to ensure that, where necessary, is made clear that the information presented is only applicable to specific versions, however if you are looking for this version-specific documentation, you can find it here *(http://developer.answermodules.com/manuals/2.9.0)*

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Module Suite Compatibility Matrix¶

OpenText Content Server	MS 2.9.0
Content Suite 16.2 EP6	X
Content Suite 16.2 EP7	X
Content Suite 20.2	X
Content Suite 20.3	X
Content Suite 20.4	X
Content Suite 21.1	X

Major Changes in version 2.9.0¶

Content Script¶

Extension for Core Share (NEW)¶

Programmatically manage sharing of content through Core Share

Extension for OAuth Services (NEW)¶

Manage OAuth2 authentication flow(s) in Content Script

Copyright © 2013-2022 AnswerModules Sagl

```
//Get accesstoken and redirect the user on this same script if authorization
//is required
token =  oauth.getAccessToken("default", "${url}/runcs/${self.ID}", [:])
if(!token.accessToken && token.accessTokenUrl){
    redirect token.accessTokenUrl
    return
}

rest = csws.getHttpBuilder("https://api.zoom.us/v2/users")
result = rest.get(){
    request.headers['Authorization'] = "Bearer ${token.accessToken}"
}
out << result
```

Extended logging functionality¶

- Added Content Script API to initialize separate Content Script log appenders.
- Additional log files can be accessed directly from the Content Script Editor.

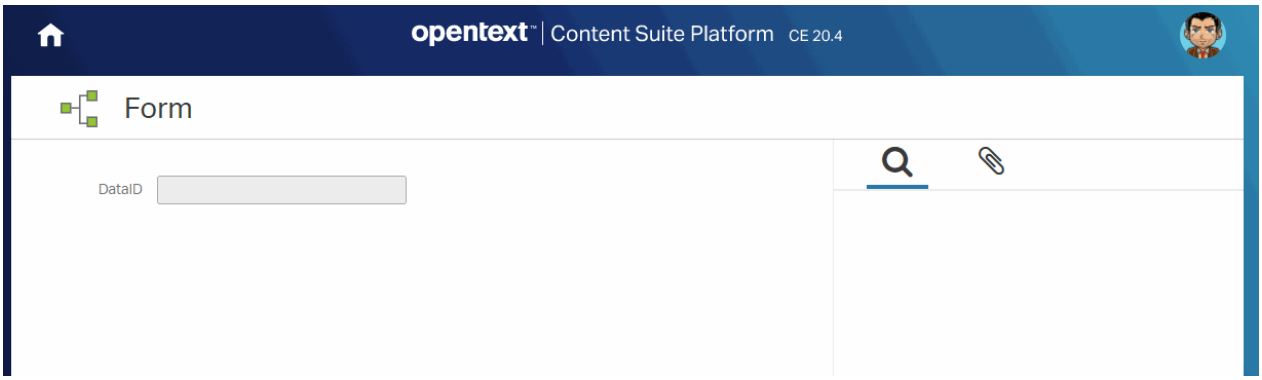
Other improvements¶

- Map/Reduce framework support has been optimized.
- 50+ New APIs added across different endpoints.

Beautiful WebForms¶

Improved SmartUI compatibility for widgets.¶

- ItemReference Popup: now supports SmartUI variant for selection popup and contextual menu.



Smart Pages¶

"CSSmartMenu" has become "CSSmartView"¶

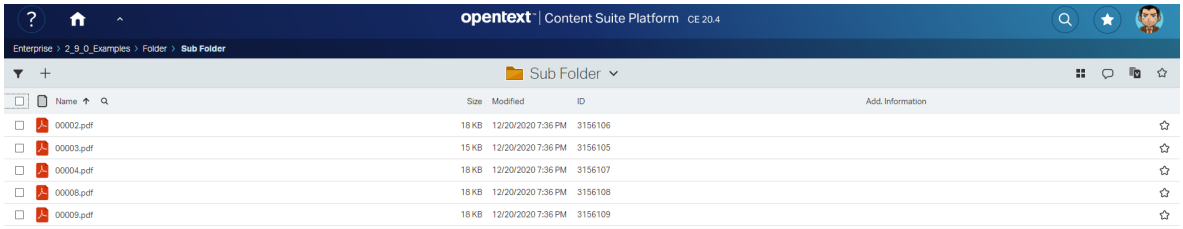
CSSmartMenu, the folder within the Content Script volume that allowed you to manage menu extensions for SmartView browsing views, has been renamed to **CSSmartView**. This change

Copyright © 2013-2022 AnswerModules Sagl

reflects the fact that, as of this release, it will allow to control numerous new customizations to various SmartView features, and not only limited to the menus.

- **CSSmartView:Columns**: it's now possible to add/remove columns from/to browsing views using Content Scripts stored in the aforementioned folder. E.g.

  Example

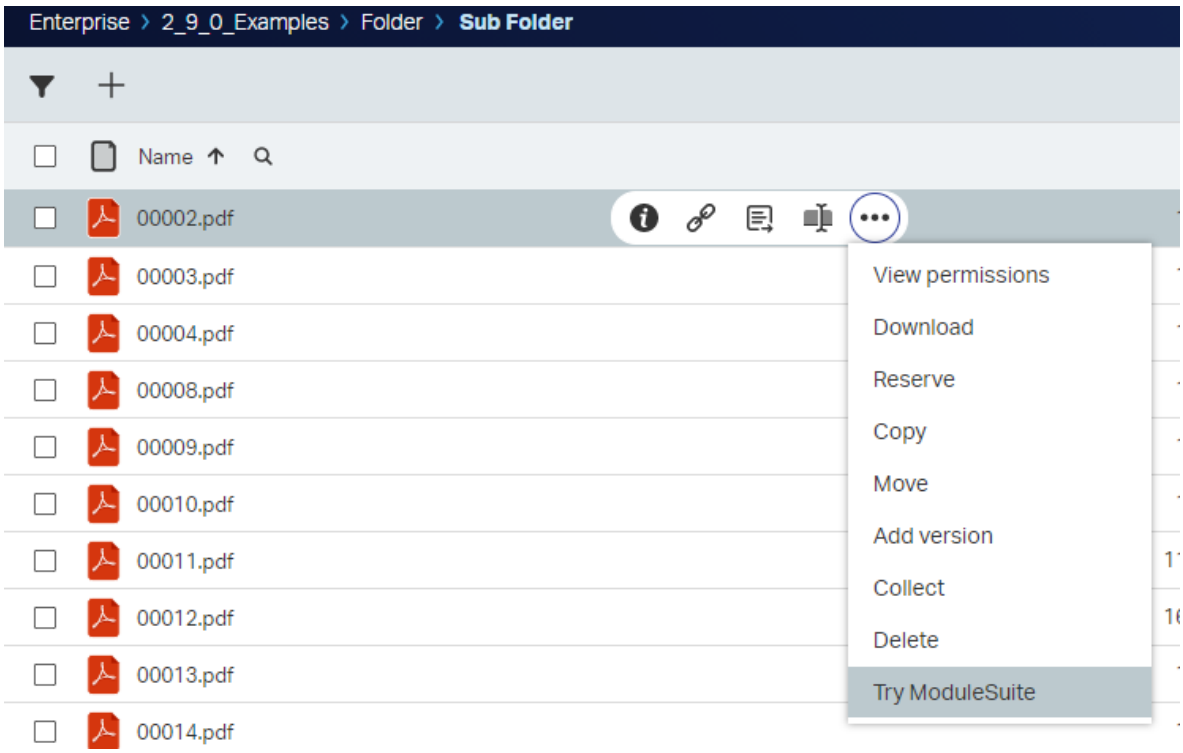
  Script

```
//In the execution context of this script:
// - nodesColumns ( a map that associates nodes' ids with their columns definitions). Tipicall
// - nodes: the list of nodes records. Tipically contains a single item.
// - req: the original REST request record
// - envelope: the current REST API call envelope

nodesColumns[3156087]?.add([type:43200, data_type:43200, name:"Add. Information", sort_key:"ty

//Must return the revised nodeColumns
return nodesColumns
```

- **CSSmartView:Actions**: it's now possible to add custom actions to a node's menu lazy loaded set of actions . E.g.

  Example

Copyright © 2013-2022 AnswerModules Sagl



Script

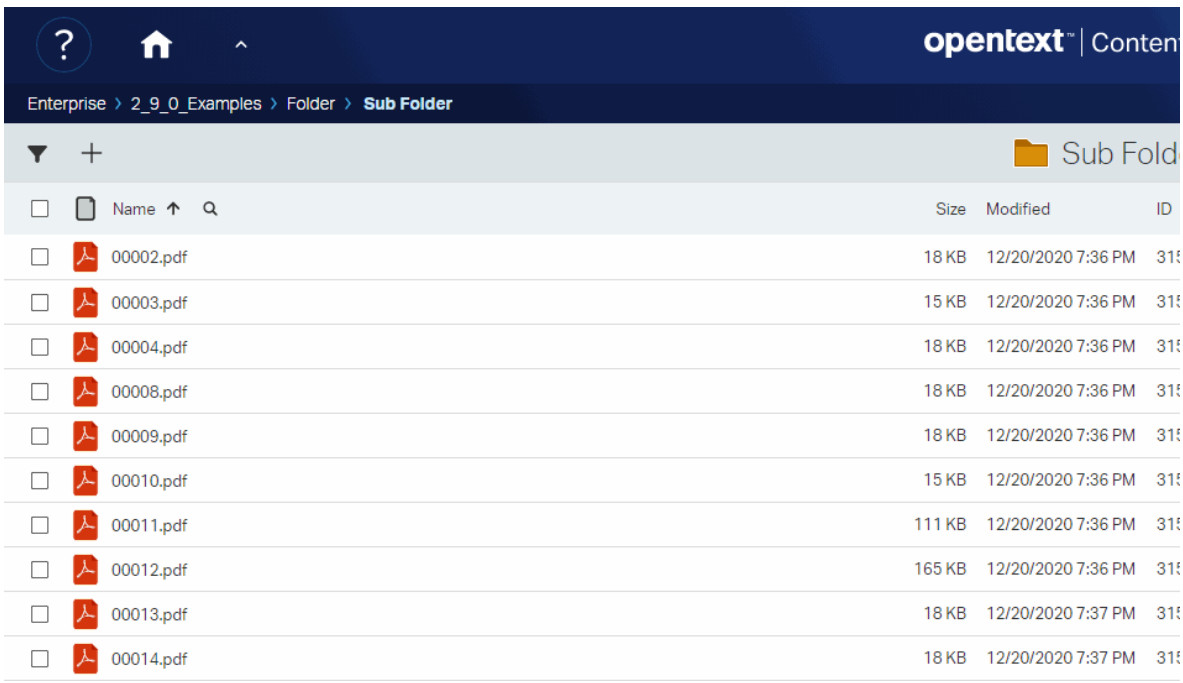
```
/**
This script receives the following variables in the execution context:

- actions: a map that associates the node id to the list of available actions
E.g.
    "12345": {
        "data": {
            "Classify": {
                "content_type": "application/x-www-form-urlencoded",
                "method": "POST",
                "name": "Add RM Classification",
                "href": "/api/v2/nodes/2891606/rmclassifications",
                "body": "{\"displayPrompt\":false,\"enabled\":false,\"inheritfrom\":false,\"ma
                "form_href": ""
            },
            "initiatedocumentworkflow": {
                "content_type": "",
                "method": "",
                "name": "",
                "href": "",
                "body": "initiate_in_smartview",
                "form_href": "",
                "wfList": [

                ]
            },
            "zipanddownload": {
                "content_type": "",
                "method": "POST",
                "name": "Zip and Download",
                "href": "/api/v2/zipanddownload",
                "body": "",
                "form_href": ""
            },
            "RemoveClassification": {
                "content_type": "application/x-www-form-urlencoded",
                "method": "POST",
```

Copyright © 2013-2022 AnswerModules Sagl

```
                "name": "Remove Classificfation",
                "href": "/api/v2/nodes/2891606/rmclassifications",
                "body": "",
                "form_href": ""
            }
        },
        "map": {
            "default_action": "open"
        },
        "order": [
            "initiatedocumentworkflow",
            "Classify",
            "RemoveClassification",
            "zipanddownload"
        ]
    }
}

- req: the current HTTP request
- envelope: the REST API request's envelope

By changing the support variable "actions" you can make visible actions defined by scripts in

**/

actions[3156106].data["am_release"] = [
    body:"am_release"
]
actions[3156106].order.add("3156106")
```

- **CSSmartView:Commands**: it's now possible to define multiple commands in the same script and group them in the same sub-menu. E.g.

Example



Script

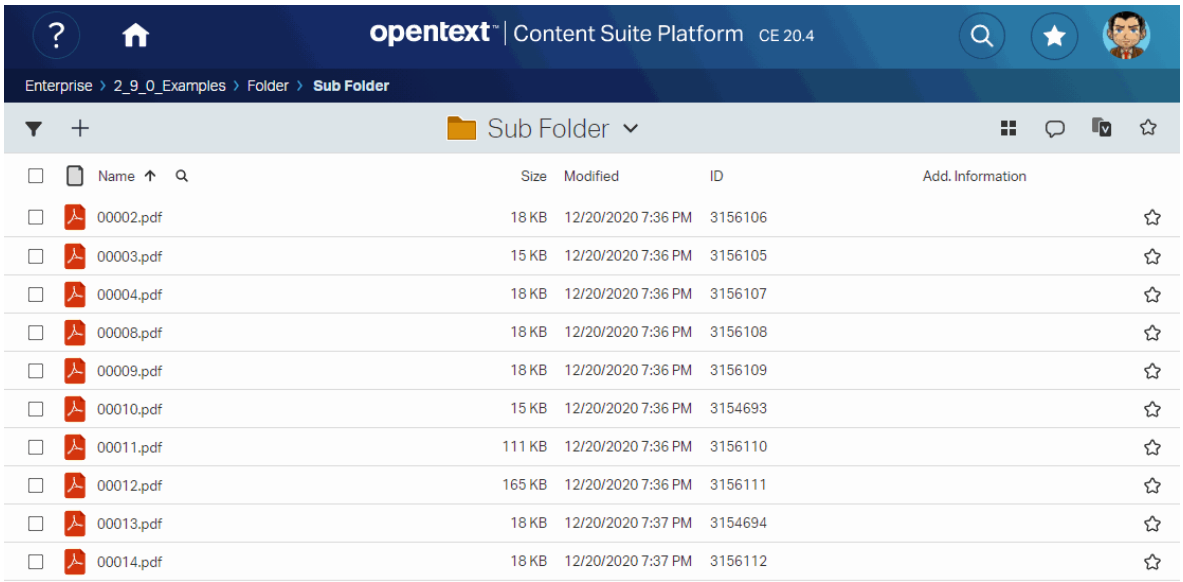
Copyright © 2013-2022 AnswerModules Sagl

```
//Commands scripts can now return a list
return [
    [
        am:[
            exec:[
                mode:"group"// (1) This command will act as our flyout
            ]
        ]
        ,scope: "multiple"
        ,group: "info"
        ,flyout: "am_group" // (2) This command will act as our flyout
        ,baricon: null
        ,icon: null
        ,name: "Try Module Suite"
        ,command_key: "am_group"
        ,signature: "am_group"
    ],
    [
        am:[

            confirmation:[
                required:false,
                title:"",
                message:""
            ],
            panel:[
                width:40,
                cssClass:"",
                slides:[
                    [
                        title:"",
                        script:null
                    ]
                ]
            ],
            key:[
                code: 83
                ,message:""
                ,nogui:false
            ],
            exec:[
                mode:"script"
                ,script: 2644067
                ,params:[

                ]
                ,refresh_on_success:true
                ,on_success_action:""
                ,newtab:false
                ,url:""
            ]
        ]
        ,baricon: null
        ,icon: null
        ,name: "Content Script"
        ,command_key: "am_content_script"
        ,signature: "am_content_script"
        ,scope: "multiple"
        ,flyout:"am_group"
        ,selfBlockOnly: false
    ]
    ...
]
```

- **CSSmartView:Commands**: Content Script scripts executed as commands can now return execution information to the caller. E.g.

Copyright © 2013-2022 AnswerModules Sagl

Example

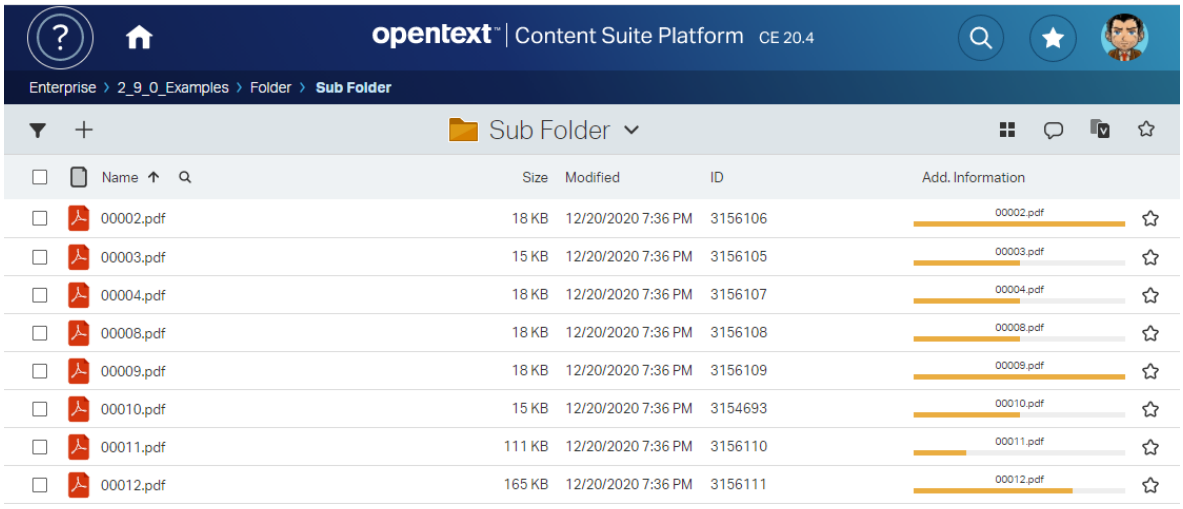


Script

```
//Script code...
//Once done...notify caller
json([message:[type:'success', text:"Get the Module Suite. You won't need anything else.", det
```

- **CSSmartView:Override**: It is now also possible to enhance the information associated with nodes with column information injected via Module Suite. E.g.

Example



Script

```
def drawStatusBar = { node ->

    def statusList = ['Draft', 'Under Revision', 'Approved', 'Published']
    def numSteps = statusList.size()
```

Copyright © 2013-2022 AnswerModules Sagl

```
    def currStep = new Random().nextInt(statusList.size())
    def currStepName = node.name

    def stepStyle = "height:100%; width:calc(100% / ${numSteps}); float:left; background-color

    def stepsHtml = ""

    (currStep + 1).times{
        stepsHtml += """<span style="${stepStyle}"></span>"""
    }


    return """
    <div style="text-align:center; font-size:.75em">${currStepName}</div>
    <div style="margin:3px 0; padding:0; height:5px; background-color:#eee;">${stepsHtml}</div
}



retVal =  nodes.collect{
    [
        ("D${it.dataid}" as String):[ //The object returned MUST be made of simple types (no G
            commands:["am_group", "am_bwf"],
            columns:[
                // Column defined in CSSmartView:Columns as nodesColumns[3156087]?.add([type:4
                // columns of type 43200 can be used to inject HTML
                _am_info:drawStatusBar(it)
            ]
        ]
    ]
}
return retVal
```

Global revision of Smart Pages widgets¶

- Smart Pages widgets have been reviewed in terms of both styling and structure.
- Smart Pages CSS is now better isolated from Beautiful WebForms CSS. The base CSS class for Smart Pages has been changed from "am-smartui" to "am-smartpage". The "am-smartui" class is now reserved for Beautiful WebForms. NOTE: This may cause breaking changes in custom rules and page templates based on the legacy class.

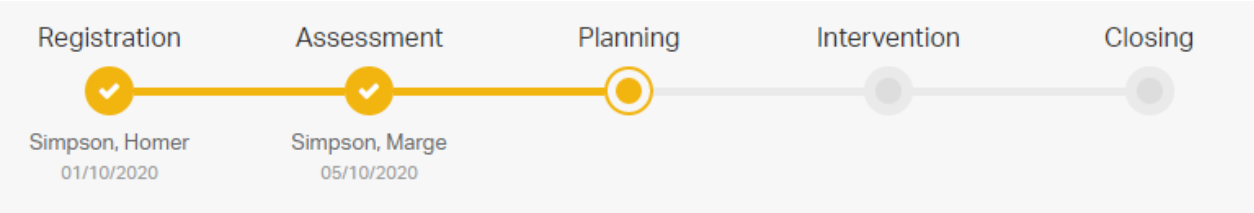
New Smart Pages widgets¶

The following widgets have been added to the Smart Pages Editor, and can now be used to build Smart Pages:

- Button Container: a container-type widget meant to hold regular buttons. Can be configured to display as a button-group.

- Link Button: a button widget that will open a configured url.
- Step indicator: a widget to display a process status and execution details.

Copyright © 2013-2022 AnswerModules Sagl



Added support for flexbox on Smart Pages used as Smart View tiles.¶

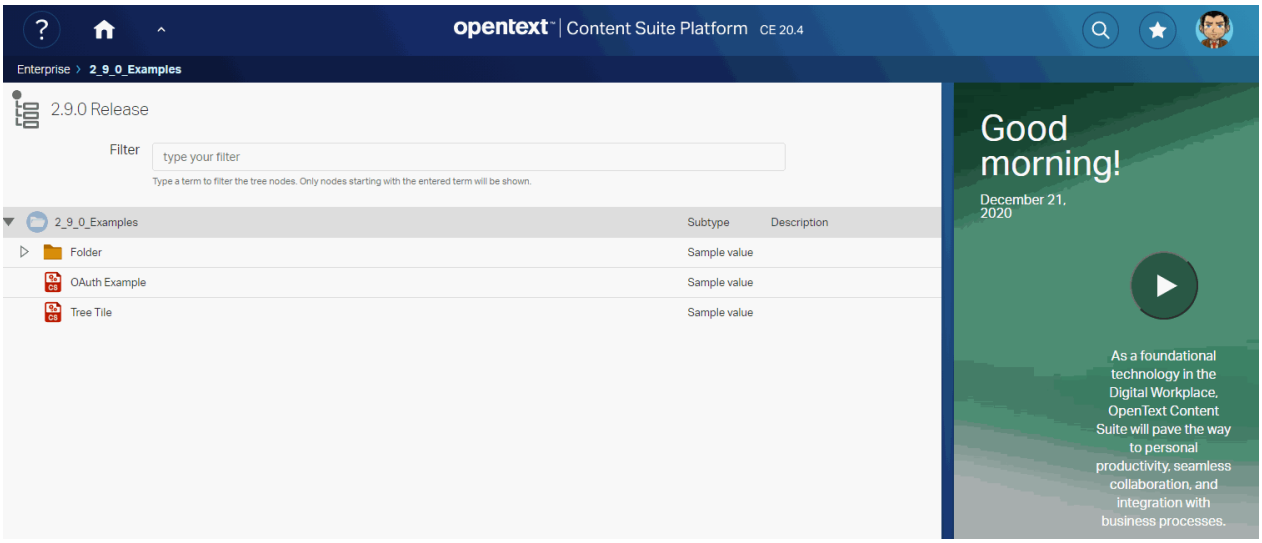
Add the CSS helper class "am-page-container-flex" as a custom class within the Smart View Tile configuration to enable flex support on Layout containers and panels within the included Smart Page. This will allow to create tiles that better occupy all the available vertical space.

Revised Tree Widget¶

Tree Widget has been revised and enhanced with new functionalities.

- It is now possible to enable the standard Smart UI function menu on tree nodes.
- It is now easier to bind tree nodes to Smart UI actions.
- Helper CSS classes have been added to support adding extra columns to the tree nodes.
- Default tree look & feel is now more similar to Smart UI style.

Example



Script

```
def rootID = params.uiParentID ?: (self.parent.ID as String)


if( params.widgetConfig ){

    if(rootID?.isLong()){

        def node = docman.getNodeFast( rootID as Long )
        json( [ id          : 'treeWColumns',
            widgetConfig : [
```

Copyright © 2013-2022 AnswerModules Sagl

```
                //node_tag           : "DIV", // Custom node tag
                tileLayoutClasses  : "",
                tileContentClasses : "",
                treeId             : "test",
                reloadCommands     : ["updateTree"],
                root               : "${rootID}",
                plugins            : [ "wholerow"],
                theme              : [ 'name'       : 'proton',
                                       'responsive' : true
                                       ],
                grid : [
                    columns : [
                        [width : 50, header : "Nodes"],
                        [width : 30, header : "Actions", value : "icon"]
                    ]
                ],

                html               : """
<style type="text/css">
</style>

<form class="binf-form-horizontal">
<div class="binf-form-group">
    <label for="filter" class="binf-control-label binf-col-xs-2" style="padding-top:7.5px">Filte
    <div class="binf-col-xs-8">
    <input id="filter" name="filter" placeholder="type your filter" type="text" aria-describedby
    <span  style="margin-left:0px" id="filterHelpBlock" class="binf-help-block">Type a term to f
    </div>
</div>
</form>


<script>
    csui.onReady2([
        'csui/lib/underscore',
        'csui/lib/backbone',
        'csui/lib/jquery',
        'csui/lib/radio'
        ],
        function(_,Backbone, jQ, Radio){

            var amChannel = Radio.channel("ampagenotify");

            amChannel.on("printConsole", function(params){
                console.log("GOT request "+JSON.stringify(params));
            });

            amChannel.on("smartPage_action", function(action,param){
                console.log("GOT Page Action request. Action: "+action+ " parameter: "+param);
            });

            jQ("#filter").on("blur", function(){
                amChannel.trigger("updateTree",{'term':jQ(this).val()})
            })


        });
</script>"""
            ]
            ] )
        return
    }
}



def getNodeContent = { node ->

    def content
```

Copyright © 2013-2022 AnswerModules Sagl

```
    def text = node.name

    content = """<span class="jstree-anchor-cols">
                    <span class="jstree-col-main">${text} <span class="csui-table-cell-name-
                    <span class="jstree-col-2">
                        <span>Sample value</span>
                    </span>
                    <span class="jstree-col-4">
                        <span>${node.comment}</span>
                    </span>
                </span>"""
    return content

}

def getHeaderRow = { node ->

    return """<span class="jstree-anchor-cols">
                <span class="jstree-col-main">${node.name}</span>
                <span class="jstree-col-2">Subtype</span>
                <span class="jstree-col-4">Description</span>
                <!--<span class="z-treenode-actions">Actions</span>-->
            </span>"""

}


/* Utility function to fetch all children of a node. */

def fetchChildNodes = { spaceNode, Boolean shouldExpand ->

    def data = spaceNode.childrenFast.collect{ node ->

        retVal = [
            name     : node.name,
            id       : (node.isContainer)?"${node.ID}_node":"${node.ID}_doc",
            text     : getNodeContent(node),
            icon     : "${node.webClass}",
            children : node.isContainer && node.childCount > 0,
            state : [
                opened : shouldExpand
            ]
        ]

        if( !node.isContainer ){
            retVal.a_attr = [
                            "data-toggle"    : 'command',
                            "data-am-action" : 'Download,Delete,Properties,am_zoom',
                            "data-am-params" : node.ID
                            ]
        } else {
            retVal.action = 'navigate'
        }
        return retVal

    }?.sort{ it.name }

    return data

}


/* Utility function to fetch children of a node in "paged" fashion. */

def fetchChildNodesPage = { spaceNode, Boolean shouldExpand, Integer pageNumber, Integer pageSiz

    nodePage           = spaceNode.getChildrenPage()
    nodePage.pageSize  = pageSize
```

Copyright © 2013-2022 AnswerModules Sagl

```
    nodePage.pageNumber = pageNumber

    nodes = docman.listNodesByPage( nodePage, "name", false, false, false, false)

    def data = nodes.collect{ node->

        retVal =[
            name     : node.name,
            id       : (node.isContainer)?"${node.ID}_node":"${node.ID}_doc",
            icon     : "csui-icon ${node.webClass}",
            text     : getNodeContent(node),
            children : node.isContainer && node.childCount > 0,
            state    : [
                opened : shouldExpand
            ]
        ]

        if( !node.isContainer ){
            retVal.a_attr = [
                            "data-toggle"    : 'command',
                            "data-am-action" : 'Download,Delete,Properties,am_zoom',
                            "data-am-params" : node.ID
                            ]
        }

        return retVal
    }?.sort{ it.name }

    return data

}



// MAIN CODE

if( !(rootID.split("_")[-1] in ["page", "node", "doc"])){

    // This is the root node. The Outline is closed by default

    def node = docman.getNodeFast( rootID as Long )

    docman.getChildrenFast(node)

    data = [
        [
            name     : node.name,
            icon     : "csui-icon cs_folder_root",
            id       : "${node.ID}_node",
            text     : getHeaderRow(node),
            children : fetchChildNodes(node, false),
            state : [
                opened : true
            ],
            action : 'navigate'
        ]
    ]

} else {

    data = []

    Boolean shouldExpand = false
    Integer pageSize     = 5

    def idElements     = params.uiParentID.split("_")

    Long space         = idElements[0] as Long
    String spaceType   = ( idElements.size() > 1) ? idElements[-1] : 'node'
```

Copyright © 2013-2022 AnswerModules Sagl

```
        Integer pageNumber = (spaceType == 'page') ? ( idElements[1] as Integer ) : null
        def spaceNode      = docman.getNodeFast( space )

        if( spaceType == 'page' ){
            data = fetchChildNodesPage( spaceNode, shouldExpand, pageNumber, pageSize )

        } else if( idElements[-1] == "node" ){
            if( spaceNode.childCount > pageSize ){

                // Paginate children list if it is bigger than 'pageSize'

                Integer numTotalPages = Math.ceil( spaceNode.childCount / pageSize) //spaceNode.chil

                numTotalPages.times{ pageIndex ->

                    def children = true

                    if( pageIndex == 0 ){

                        // Pre-expand the first page
                        children = fetchChildNodesPage( spaceNode, shouldExpand, 1, pageSize )
                    }

                    data.add([
                        name     : "${space}_${pageIndex + 1}_page",
                        id       : "${space}_${pageIndex + 1}_page",
                        icon     : "cs_vfolder",
                        text     : "${(pageIndex * pageSize) + 1} ... ${(pageIndex + 1) * pageSize }"
                        children : children,
                        state    : [
                            opened : shouldExpand
                        ]
                    ])
                }

            } else {

                data = fetchChildNodes( spaceNode, shouldExpand )
            }
        }
    }

    if(params.term){
        //This should be consider just an example of a possible filtering solution. Since we are not
        //as the last operation is not impacting performances very much.
        filter = { list, term ->
            list.removeAll{ it.children == false && ! it.name.toUpperCase().startsWith( term.toUpper
            list.each{ listElement ->
                if(listElement.children && listElement.children instanceof List){
                    filter(listElement.children, term)
                }

            }
        }
        filter(data, params.term)
    }

    json( data )
```

All Enhancements in version 2.9.0¶

ID	Scope	Description
#000936	Beautiful Webforms	Add internationalization support also to widget

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#000916	Beautiful Webforms	Add and remove button for multifield in modal popup
#000876	Beautiful Webforms	Allow feedback from actions performed in embedded forms to show in standard Smart UI feedback panel
#000908	Smart Pages	Smart menu: additional menu command in SmartUI override only at the first level
#000928	Beautiful Webforms	Item Reference Pop Up browse is in classic view
#000871	Content Script	When changing the default CS.log location from the Opentext.ini file the change does not take effect
#000931	Content Script	Missing documentation for xecm extension xecm.updateRole(..) API
#000955	Extension - FTP	[FTP api] Sending documents in binary mode
#000942	Smart Pages	More flexibility is required regarding the logic to use to show a custom action in a menu
#000440	Beautiful Webforms	Improve robustness of Jquery Interdependencies widget
#000947	Module Suite	Re-import of a Content Script is not supported
#000949	Beautiful Webforms	In a set; delete link is missing for the first row
#000910	Smart Pages	CSSmartMenu not displayed on Results page

Issues Resolved in version 2.9.0¶

ID	Scope	Description
#000951	Beautiful Webforms	Click on a submit button display "saving" and there is no way to change the language
#000932	Beautiful Webforms	Customizing search string in Smart DropDown
#000906	Beautiful Webforms	Datatable widget: if a language file is applied; all the words are translated properly excluding the search box of the columns
#000917	Beautiful Webforms	Search/Clear buttons overlapped for Item reference Popup
#000904	Beautiful Webforms	Minor CSS issue in User by login widget

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#000915	Beautiful Webforms	Graphical issue in multifield: plus and minus button are in strange position
#000894	Beautiful Webforms	Issue in installation of BWF updated with case sensitive database
#000898	Beautiful Webforms	Smartlookup behavior used by Smart Dropdown DB lookup widgets does not support PostgreSQL
#000805	Beautiful Webforms	Pattern validation rule is truncated if the model contains parentheses
#000885	Smart Pages	Issue Smart View Custom Menu in execution classic mode
#000891	Beautiful Webforms	Inconsistent behavior for check-boxes when used with Widget Space Content
#000881	Beautiful Webforms	The Item Reference Popup Widget in Library V3 does not render correctly
#000883	Beautiful Webforms	Currency Widget anomaly validation on IE
#000923	Beautiful Webforms	Smart ViewTask template not displayed correctly on CS 20.3
#000913	Beautiful Webforms	Item reference Popup: with v4 library the context menu is the one of Smart UI
#000719	Beautiful Webforms	User By login doesn't show the values if is located at the bottom of the page
#000924	Beautiful Webforms	Usability issue with DropDown and page scroll
#000751	Beautiful Webforms	Item Reference Popup doesn't work properly
#000874	Extension - ZIP	Error returning zip resource on Linux
#000892	Beautiful Webforms	XSS security vulnerability
#000946	Module Suite	Content Script execution audit track flag has been associated to the wrong bit
#000927	Beautiful Webforms	Visualization issues with Add Delete Button widget
#000840	Beautiful Webforms	OnChangeAction doen't work with V3:SmartView Template - registerInitWidgetCallback

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#000922	Smart Pages	Low performance in Nodes table tile
#000919	Content Script	Issue method grantFullControl on Add Major Version
#000903	Content Script	fileName null in the CSVersion object
#000680	Content Script	Accessing rendition content on CSVersion result in wrong content
#000925	Content Script	Content Script Scheduling administration does not work with PostgreSQL databases
#000896	Beautiful Webforms	Issue in Manage Callbacks: on Linux box an error is returned trying to listing the callbacks
#000957	Smart Pages	Widget Nodes table - Error on selecting nodes
#000902	Beautiful Webforms	Issue widget "go" anchor on library V4
#000958	Content Script	Method distagent.mapReduce doesn't work correctly
#000938	Module Suite	Library 2.7 and 2.8 included in installation packages
#000921	Smart Pages	GoTo option for action button is not working
#000890	Beautiful Webforms	Alert javascript when editing a document in the SmartView Task template on IE
#000940	Beautiful Webforms	Button label text outside the button
#000937	Content Script	Unable to retrieve classifications for an email (subtype 749)
#000962	Beautiful Webforms	Server Side validation for Smart DropDown: the field with error is not highlighted
#000909	Beautiful Webforms	Panel Arrow wrong direction
#000918	Beautiful Webforms	JS Conditional container behavior with multiple field
#000964	Content Script	Trace files in REST API call
#000933	Smart Pages	Expand tile button not working Smart UI when there is a parameter in the URL

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#000929	Content Script	Erratic problems related to script execution are recorded in complex applications that make massive use of the runCS API.
#000930	Content Script	ContentScript hasTemplate API might rise an error at startup
#000884	Beautiful Webforms	Issue Wysing Editor the copied image is duplicated
#000888	Script Console	Issue load configuration with database PostgreSQL
#000954	Script Console	Regression in RenderForm.cs
#000920	Module Suite	Regression in cache.putForUser API
#000882	Smart Pages	Custom Menus are not displayed in "Node Browsing Table" when the widget is associated to a Virtual Folder
#000965	Content Script	xecm.createWorkspace doesn't work if a multiple attribute of a category is set
#000866	Extension - eSign	when executing the esign.addESignNatureToFormStep(form; "Approve and Sign") the module returns an error
#000952	Content Script	Error in some methods for Physical Object
#000934	Script Console	Script Console slow calling script with RunCS
#000887	Script Console	Log not working in Script Console
#000854	Script Console	The first attempt to authenticate to the script console always fails with a 403 error
#000970	Beautiful WebForms	BWF Studio fails to export a form (for remote usage) if a validity date has not been specified

Version 2.8.0 - Release notes¶

Release Date	End of AMP(*)	End of Life
2020-07-16	2023-07-16	2024-07-16

(*) Active Maintenance Period

Copyright © 2013-2022 AnswerModules Sagl

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Modules Suite version 2.8.0.

This guide

The information presented in the on-line guide are mostly non-version specific. AnswerModules team does its best to ensure that, where necessary, is made clear that the information presented is only applicable to specific versions, however if you are looking for this version-specific documentation, you can find it here *(http://developer.answermodules.com/manuals/2.8.0)*

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Module Suite Compatibility Matrix¶

OpenText Content Server	MS 2.8.0
Content Suite 16.2 EP6	X
Content Suite 16.2 EP7	X
Content Suite 20.2	X
Content Suite 20.3	X

All Enhancements in version 2.8.0¶

ID	Scope	Description
#000865	Script Console	The Script Console Installer does not have the ZIP Extension among the available extensions
#000814	Beautiful Webforms	Error widget does not show "go" anchors if errors are triggered in javascript validation
#000863	Beautiful Webforms	Extend BWF view updater support to PostgreSQL DBMS
#000833	Module Suite	Now you can add a custom value in the library search filter of the Import / Upgrade Tool; use * to search all libraries
#000861	Content Script	Extend support of jdbc 'otcs' datasource to PostgreSQL and SAP Hana dbms
#000868	Module Suite	Unable to access Content Script and some components with X-Content-Type-Options HTTP Header

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#000867	Smart Pages	Add content awareness to Content Script Nodes Table Smart UI widget
#000810	Smart Pages	Smart UI Commands can now open a Smart Page or Content Script result in a side panel
#000813	Module Suite	CSSynchEvents might fail if "Node Cache" is enabled
#000824	Module Suite	"Node Cache" feature is preventing "startWorkflow" API from working properly

Issues Resolved in version 2.8.0¶

ID	Scope	Description
#00006	Extension - Blazon	Bloazon ExtPack returns the wrong file when OCR or PDF/A convertion are enabled
#00044	Script Console	Script Console returns 500 Error after a period of inactivity
#00018	Module Suite	Missing Workflow Attachments and Workflow Comments
#00069	Beautiful Webforms	Issue widget Debug
#00037	Beautiful Webforms	Submit multiple times
#00058	Content Script	getuserbyname REST API endpoint returns error on Postgresql DBMS
#00000	Beautiful Webforms	Javascript error raised from the "Error" widget if server side validation fails on an hidden field
#00001	Beautiful Webforms	Currency widget resets value upon loading on Internet Explorer 11
#00022	Content Script	Current user context is not restored when a subscript (impersonating an other user) trigger the execution of a CSSE Script
#00002	Beautiful Webforms	Phone widget always shows validation error on Internet Explorer 11
#00003	Beautiful Webforms	Error widget "go" anchor not working if error is associated to Currency widget
#00004	Beautiful Webforms	Space Content widget drop area not working on Internet Explorer 11
#00025	Beautiful Webforms	CSTemplate:V3:Smart View Embeddable doesn't work properly in OTCS 20.2

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#00026	Beautiful Webforms	velocity.tools.generic.LogTool error
#00027	Beautiful Webforms	Wrong search box position when dropdown opens above
#00028	Content Script	Trace files in REST API call when a document is returned
#00029	Beautiful Webforms	ReadOnly of the User by login widget does not work
#00030	Module Suite	Regression: Compilation of scripts using classes defined in other scripts might fail
#00035	Beautiful Webforms	Incorrect alignment of the form fields readonly with the Opentext V4 template
#00039	Beautiful Webforms	Panel Layout issue 'is collapsible'
#00047	Content Script	Content Script content cached
#00048	Smart Pages	Teams Integration
#00049	Smart Pages	SmartUI Menu on CS 16.2.8
#00050	Beautiful Webforms	Anomaly validates fields with multiple lines
#00051	Content Script	Error creating a Business Workspace template
#00052	Beautiful Webforms	Anomaly in the 'Tile Links' snippet code
#00056	Beautiful Webforms	Currency Widget anomaly when the "Clean format" is not used
#00057	Smart Pages	Issue SMART UI expand
#00062	Smart Pages	Scrolling Does not Work on Content Script Expand Screen
#00064	Smart Pages	SmartUI ModuleSuite Tile Widgets not aligning
#00070	Smart Pages	Custom Smart UI Menu Commands does not works for IE browser
#00042	Script Console	Script Console login timeout on startup
#00019	Beautiful Webforms	Additional validation constraints are not enforced for fields that are not visible when the webform is loaded for the first time

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#00072	Smart Pages	Parameters are not properly managed when an action is invoked by a Smart View Custom Menu Item
#00073	Beautiful Webforms	Missing workflow step info on form object when using Beautiful Webforms in workflow steps
#00020	Smart Pages	The Smart UI command that opens a panel causes the loading indicator to activate; closing the panel has no effect
#00012	Beautiful Webforms	Smart Dropdown component initialization fails if associated form field value contains a single quote
#00011	Beautiful Webforms	The "Select *" widgets of the V3 library do not send any value (their value is reset when performing any action)

Version 2.7.0 - Release notes¶

Release Date	End of AMP(*)	End of Life
2020-04-17	2023-04-17	2024-04-17

(*) Active Maintenance Period

This guide

The information presented in the on-line guide are mostly non-version specific. AnswerModules team does its best to ensure that, where necessary, is made clear that the information presented is only applicable to specific versions, however if you are looking for this version-specific documentation, you can find it here *(http://developer.answermodules.com/manuals/2.7.0)*

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Modules Suite version 2.7.0.

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Module Suite Compatibiliy Matrix¶

OpenText Content Server	MS 2.7.0
Content Suite 16.2 EP6	X
Content Suite 16.2 EP7	X

Copyright © 2013-2022 AnswerModules Sagl

OpenText Content Server	MS 2.7.0
Content Suite 20.2	X

Major Changes in version 2.7.0¶

- Introduced the new concept of layered configuration. It's now possible to replicate the structure of the Content Script Volume in a Content Script Volume Folder to isolate customization that are related to specific applications. Module Suite will resolve the resulting multi-level configuration structure.
- Introduced the possibility to add new command in SmartUI by defining Content Script objects in the **CSSmartMenu** folder of the Content Script Volume.
- Introduced the possibility to schedule a job that leverages the Map-Reduce framework implemented by the Distributed Agent. With this new feature, developer will be able to process much larger amount of data, reducing the impact over the system.
- Updated API Guides and in-line documentation.

Extension Distributed Agent (NEW)¶

A new service "distagent" has been introduced for managing script scheduling and supporting the usage of OTCS' Distributed Agent framework.

Smart Pages¶

It's now possible to include multiple WebForms in a single SmartPage. Several improvements in Module Suite tiles.

All Enhancements in version 2.7.0¶

ID	Scope	Description
#0000781	Beautiful Webforms	PDF Viewer widget now supports Content Suite Viewer
#0000780	Extension - PDF	API for easy retrieval of pdf text annotations (comments)
#0000777	Smart Pages	It is now possible to dynamically configure additional Smart UI commands through Module Suite
#0000776	Module Suite	Updated Java core dependencies
#0000768	Module Suite	Enhanced compatibility with xECM for Engineering
#0000767	Module Suite	Enhance Content Script license configuration field in Base Configuration

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000764	Smart Pages	PDF Preview Content Script
#0000733	Smart Pages	It's now possible to associate a custom CSS class to Content Script Result Tile
#0000720	Content Script	Trim custom parameters in the Base configuration
#0000713	Beautiful Webforms	actionParams handler in the submit action in the SmartViewTask template
#0000689	Content Script	New xlsx API to get Style ID used in a specific cell
#0000688	Module Suite	increase the contrast of the flag in the upgrade library tool
#0000572	Module Suite	It is now possible to change Duration, StardDate and DueDate of a Workflow Task
#0000340	Beautiful Webforms	Replicate Content Script Volume structure for applications
#0000186	Content Script	Content Script Scheduling configuration revision
#0000123	Content Script	Improvements in CSVersion

Issues Resolved in version 2.7.0¶

ID	Scope	Description
#0000795	It's no longer possible to change logging level as "script by script" bases	
#0000794	Module Suite	Posgresql minor compatibility issues
#0000792	Module Suite	Xml Import might generate trace files on 16.2.9->16.2.11
#0000791	Module Suite	It's no longer possible to change a script logging level
#0000790	Module Suite	The in-line guide contains outdated screen shots
#0000789	Beautiful Webforms	The method 'overrideFieldValidation' of form's fields is not working when the form is loaded for the first time
#0000788	Smart Pages	Smart Pages "Expand" button widget not working
#0000787	Smart Pages	Setting custom "Tile CSS classes" on Content Script tiles is overridden if "Should load widget configuration" flag is set
#0000779	Extension - SQL	Any input parameter that begins with the pound sign is interpreted as a filter

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000775	Content Script	Test Content Script command doesn't work with Classic Link Behavior Smart View
#0000774	Beautiful Webforms	Itemreference Popup does not receive focus in case of error
#0000772	Module Suite	Managecallback dashboard does not return callback associated to the node's subtype
#0000771	Content Script	xECM fails in creating a CWS when script is scheduled
#0000770	Content Script	CSEvents scripts are executed twice
#0000769	Extension - xECM	Incorrect mapping of user data when loading users for workspace roles
#0000766	Beautiful Webforms	Select from ViewParams widget in Library V4 does not save selection value
#0000765	Beautiful Webforms	API method forms.addResourceDependencies(...) fails to load dependecies if view names are specified
#0000763	Smart Pages	Preview Icon on SmartUI WF Task Form
#0000762	Module Suite	MS log does not work on OT EP8 (16.2.11)
#0000761	Content Script	Conflict between Enterprise Library Extension and Advanced Version Control API
#0000760	Content Script	Base Configuration secret fields data is lost when reloading and saving configuration
#0000759	Content Script	Content Script SQL APIs return wrong values for numeric values that are outside of the Integer range
#0000755	Beautiful Webforms	Several issues with SmartPages SmartUI widgets. Reload commands not properly managed.
#0000748	Extension - Rendition	Command line placeholders cause exception in rend.genericRendition(...) API
#0000745	Script Console	Missing lib dependencies for OpenJDK compatibility
#0000744	Extension - ZIP	Regression in extract method of CSCompressedResource
#0000743	Script Console	Script Console Installer is extracting dependencies files in the wrong location
#0000742	Extension - ZIP	Regression on method listContent of CSCompressedResource
#0000741	Content Script	docman.getNodeByNickname(..) API throws an exception if a node with that nickname does not exist.

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000739	Smart Pages	Include WebForm widget's configuration does not accept templating expression
#0000737	Smart Pages	Rich Text rendering issues
#0000736	Smart Pages	Title and SubTitle widget error in configuration panel
#0000735	Smart Pages	Image Widget configuration problems
#0000734	Smart Pages	Error in loading widget configuration: Datasource is called twice with widgetConfig=true
#0000732	Module Suite	Itemreference service does not return result if params.term is empty (
#0000731	Smart Pages	Error in Smart Page rendering if Controller script is not initialized
#0000730	Beautiful Webforms	Readonly mode is not properly handled by input widgets on BWF library V2 views on Module Suite 2.6
#0000726	Beautiful Webforms	Uploading file in Space content widget clear the radio button values in some circumstances
#0000724	Beautiful Webforms	Toggle preview on Beautiful Webforms doesn't work
#0000723	Beautiful Webforms	Preview tab and attachment tab selectors not properly rendered on SmartView Task view template
#0000722	Beautiful Webforms	Beautiful WebForms views are no longer rendering errors raised from OTCS (Oscript) upon submissions (e.g. TKL valid values)
#0000711	Beautiful Webforms	Currency field doesn't save value if in the view there is a masking script
#0000699	Beautiful Webforms	docman.getNodeAuditDataPage(CSNode node) doesn't work properly in specific circumstances
#0000695	Beautiful Webforms	The month back command on the datepicker widget does not work properly when there is also a space content widget
#0000684	Content Script	Sending email with O365 not working
#0000682	Content Script	getClassifications of a Connected Workspace returns an empty list
#0000671	Content Script	Unable to get attachmentList email java.lang.NullPointerException when the attachment is a msg file

Copyright © 2013-2022 AnswerModules Sagl

Version 2.6.0 - Release notes¶

Release Date	End of AMP(*)	End of Life
2019-12-06	2022-12-06	2023-12-06

(*) Active Maintenance Period

This guide

The information presented in the on-line guide are mostly non-version specific. AnswerModules team does its best to ensure that, where necessary, is made clear that the information presented is only applicable to specific versions, however if you are looking for this version-specific documentation, you can find it here *(http://developer.answermodules.com/manuals/2.6.0)*

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Modules Suite version 2.6.0.

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Module Suite Compatibiliy Matrix¶

OpenText Content Server	MS 2.6.0
Content Server 10.0.x	
Content Server 10.5.x	
Content Suite 16	
Content Suite 16 EP2-EP5	
Content Suite 16 EP6	
Content Suite 16 EP7	X

Major Changes in version 2.6.0¶

- Introduced the **Smart Pages** module. **Smart Pages** is a brand new module that aims to simplify the creation of good-looking functional user interfaces, both as a standalone solution and as part of the Smart View perspectives.

Copyright © 2013-2022 AnswerModules Sagl

Smart Pages features a WYSIWYG drag-and-drop editor, similar to the one already available for Beautiful WebForms, to enable business users to autonomously tailor their Smart View experience.

- Introduced the possibility to limit the APIs a developer can utilize in Content Script objects. Content Script script engine can now be configured so that scripts are executed in a controlled container:*Sandbox* where only *whitelisted* APIs are usable. Whenever a developer uses an unauthorized API, at the time the script is executed, the engine will return an error containing information about the forbidden API and how to add exceptions to the *preamble* script if using this API is inevitable . The use of prohibited APIs may be authorized by super-users on the basis of what is specified in the *preamble* of the script.

Content Script¶

Process Builder API
: A new API for defining workflow maps. This new API greatly reduces the effort required to set up a Workflow application, while opening up a brand new spectrum of possibilities.

Beautiful WebForms¶

Form Builder¶

- Introduced a brand new widget library (V4). The new library makes extensive use of modern AMD framework for Javascript resources loading. All the widgets of the new library are SmartView ready and can be safely utilized in WebForms published in SmartView tiles (see. Module Suite SmartView Extension)

Extension for Workflow¶

It's now possibile to execute a Content Script associated to a Workflow Map as a Workflow *Event Script*.

Extension SFTP (NEW)¶

A new extension package that allows you to establish SFTP connections to remote servers.

Smart Pages (NEW)¶

Smart Pages has superseded the Module Suite extension of the Smart View. Previously available Smart View tiles, part of the AnswerModules's library have been ported and improved.

- Content Script Result: **Content Script Result** tile accepts now, as a datasource, both a Content Script and a Smart Page object. **Content Script Result** tile supports now the

Copyright © 2013-2022 AnswerModules Sagl

expanding behaviour, a second Content Script or Smart Page datasource can be associated to the *expanded* version of the tile.

- It's now possibile to configure AnswerModules' Smart View Tiles in order to pre-fetch their configuration using a separate call to the associated Content Script datasource endpoint. The Content Script datasource endpoint is in this case called with an additional *widgetConfig* parameter.

All Enhancements in version 2.6.0¶

ID	Scope	Description
#0000668	Module Suite	Module Suite 2.5 backward compatible with OTCS 16.0.12 (2019-03)
#0000683	Content Script	Change default attrSourceType in docman.moveNode() from Merge to Original
#0000660	Extension - Rendition	Limit the possibility to execute arbitrary drop in
#0000633	Module Suite	Introduced security checks to prevent the developers from accidentally damaging the system
#0000092	Beautiful Webforms	Add possibility to bind form field "editable" flag to a variable in binding
#0000566	Content Script	Enable CS to be used as EventScripts in WF
#0000669	Module Suite	Remove Manage Content Script Extension Packages function from Administration pages
#0000646	Content Script	Request to support SFTP
#0000656	Beautiful Webforms	Added support for WYSIWYG editors in FormBuilder widgets configuration panel
#0000047	Content Script	It's now possible to set create custom properties of type hidden in Base Configuration
#0000658	Extension - Docx	Is now possible to force Excel to update internal formulas upon file opening

Issues Resolved in version 2.6.0¶

ID	Scope	Description
#0000639	Beautiful Webforms	onChangeAction set on Smart DropDown causes a reloads loop
#0000687	Extension - eSign	reset ESign value in the workflow step

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000379	Beautiful Webforms	Submit Button in new Beautiful WebForms view does not allow to select icon
#0000070	Beautiful Webforms	Opening a Beautiful Form View with Form Builder gives error 500 if Form Template is empty
#0000185	Content Script	Typo in error message for RunAs functionality
#0000467	Beautiful	Webforms Checkbox widgets are not initialized correctly when creating a new view
#0000528	Extension - Docx	Error logs when reading core docx properties that have not been set
#0000626	Beautiful Webforms	Clear button hides when there are more lines in the CS Modernizzed Classic UI template
#0000628	Beautiful Webforms	User icons in the UserByLogin widget are not displayed correctly in V3 Smart View and Smart View Task template
#0000667	Module Suite	Error URL in online Documentation Import and Upgrade tool
#0000638	Beautiful Webforms	Space Content does not allow adding new documents with IE
#0000651	Module Suite	Errors in the links in the Extension for ClassicUI online guide
#0000685	Content Script	unScheduleContentScript (CSNode node) does not work
#0000650	Content Script	Querybuilder does not find archived workflows
#0000671	Content Script	Unable to get attachmentList email java.lang.NullPointerException when the attachment is a msg file
#0000692	Beautiful Webforms	Drop area widget fails to initialize
#0000657	Content Script	Accessing node content randomly returns an empty file
#0000665	Content Script	Issue with DAPI.GetNode cache causes wrong node data to be loaded in specific circumstances
#0000666	Extension - PDF	Errors deleting temp files after PDF manipulations
#0000659	Extension - Docx	POI library compatibility issue with OTCS 16.2.8

Copyright © 2013-2022 AnswerModules Sagl

Version 2.5.0 - Release notes¶

Release Date	End of AMP(*)	End of Life
2019-05-23	2022-05-23	2023-05-23

(*) Active Maintenance Period

This guide

The information presented in the on-line guide are mostly non-version specific. AnswerModules team does its best to ensure that, where necessary, is made clear that the information presented is only applicable to specific versions, however if you are looking for this version-specific documentation, you can find it here *(http://developer.answermodules.com/manuals/2.5.0)*

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Modules Suite version 2.5.0.

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Module Suite Compatibiliy Matrix¶

OpenText Content Server	MS 2.5.0
Content Server 10.0.x	
Content Server 10.5.x	
Content Suite 16	
Content Suite 16 EP2-EP5	
Content Suite 16 EP6	X

Major Changes in version 2.5.0 SP1¶

- Introduced support for PostgreSQL and SAP Hana Databases

Major Changes in version 2.5.0¶

OpenText Content Suite 16 EP6

Copyright © 2013-2022 AnswerModules Sagl

Content Suite 16 EP6 introduces some major breaking changes on the internal Content Server application. Changes have been applied on the application's layout as well as on the Java related-components of the application. These changes have been reflected on Module Suite leading to the decision of making the latest version of our product only available on OpenText Content Suite Platform 16 EP6.

- Upgraded numerous third party dependencies
- Introduced support to OpenJK11
- Improved, by over 15% Beautiful WebForms programmatic rendering performances
- Introduced a new logging feature that allows the creation of **isolated** log files
- Introduced new HTML to PDF rendition engine based on Google Puppeteer can be utilized in place of Wkhtmltopdf
- Improved usability of Module Suite IDEs

Content Script¶

Extension Engeenering (NEW)¶

Programmatically manage creation for CAD documents

All Enhancements in version 2.5.0¶

ID	Scope	Description
#0000640	Content Script	Self support variable in Content Script is now always available
#0000620	Module Suite	Remove link to legacy upgrade component library tool from Base Configuration
#0000629	Beautiful Webforms	New API to get PDF size
#0000614	Content Script	Extend PDF Watermark color support to include hexadecimal colors
#0000621	Beautiful Webforms	Server side validation errors are not properly displayed unless the velocity macro is overridden by an Errors widget

Issues Resolved in version 2.5.0¶

ID	Scope	Description
#0000641	Content Script	Fix issue with #csmenu macro that was preventing the macro from working properly
#0000604	Module Suite	Error in online documentation for Smart Dropdown fields
#0000623	Content Script	Content Script scheduling weekdays are incorrectly mapped

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000630	Beautiful Webforms	Date field (in a Set field) not properly initialized when loaded through CS API. Time information is missing.
#0000636	Beautiful Webforms	Phones widget validation return error if the field is empty
#0000615	Content Script	Synch callbacks are randomly not executed immediately after services restart
#0000616	Content Script	Code search shortcut CTRL + F does not work
#0000622	Beautiful Webforms	The following error is reported in the MS master log file ERROR [rendering] Left side ($field.getValidationStatus().size())
#0000619	Content Script	"template" service is not able to load resource bundles from OTCS when used in a Content Script
#0000618	Beautiful Webforms	Included subviews are not updated upon modification
#0000617	Beautiful Webforms	Submit Button with Params widget is no longer passing parameter value in library version 2.3
#0000601	Content Script	Form SET data is lost when using workflow.startWorkflow(..) API
#0000600	Content Script	Set properties on Word Documents may fail to update existing properties
#0000599	Extension - Docx	Cell style not preserved on cell value update
#0000603	Content Script	Nickname is cleared from node when node.update() is invoked
#0000611	Beautiful Webforms	Pattern validation rule is invalid if the pattern contains commas
#0000610	Beautiful Webforms	Server side validation for "Alpha" and "Alphanumeric" rules not working as expected
#0000612	Beautiful Webforms	Field Length validation not working on Multiline fields when form is loaded using forms.getFormInfo(..) API

Copyright © 2013-2022 AnswerModules Sagl

Version 2.4.0 - Release notes¶

Release Date	End of AMP(*)	End of Life
2018-11-09	2021-11-09	2022-11-09

(*) Active Maintenance Period

This guide

The information presented in the on-line guide are mostly non-version specific. AnswerModules team does its best to ensure that, where necessary, is made clear that the information presented is only applicable to specific versions, however if you are looking for this version-specific documentation, you can find it here *(http://developer.answermodules.com/manuals/2.4.0)*

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Modules Suite version 2.4.0.

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Module Suite Compatibility Matrix¶

OpenText Content Server	MS 2.4.0
Content Server 10.0.x	
Content Server 10.5.x	X
Content Suite 16	X
Content Suite 16 EP2-EP5	X
Content Suite 16 EP6	

Major Changes in version 2.4.0¶

- Dropped support for JRE version 7

JRE 7 no longer supported

Starting with update 2015-09 *(https://knowledge.opentext.com/knowledge/cs.dll/info/62452805)* OpenText Content Server is shipped with a JRE version 8. Starting from Module Suite version 2.4 support for JRE 7 is discontinued.

Copyright © 2013-2022 AnswerModules Sagl

Customers willing to install Module Suite 2.4 are invited to verify that this requirement (JRE >=8) is satisfied by their Content Suite platform.

- Upgraded numerous third party dependencies
- Improved isolation of Java components with respect to the environments
- Improved, by over 30%, the modules overall performances
- Introduced new APIs to support large PDF documents manipulation without encountering excessive system memory consumption related issues
- Introduced the possibility to assign a custom priority to Content Scripts scheduled using the DA framework
- Cache service now supports the direct caching of ServiceContextAware objects (e.g. CSNode, CSUser, CSGroup, CSFolder...) objects

Beautiful WebForms¶

- Significant reduction, approximately 50%, of the View's footprint on the database
- Introduced numerous new widgets
- Introduced the concept of Layout: widgets can now contain additional widgets and therefore be utilized to better organize the form's page

Form Builder¶

- Introduced a brand new widget library (V3). The new library makes extensive use of modern AMD framework for Javascript resources loading. All the widgets of the new library are SmartView ready and can be safely utilized in WebForms published in SmartView tiles (see. Module Suite SmartView Extension)
- Introduced 'in line text editor' functionality in the FormBuilder. Form field labels and static text can be edited directly from the SmartEditor (no need to open the configuration panel)
- Improved functionality for repositioning widgets in the View.
- Simplified the resizing of widgets and labels. Any widget or label dimensions can now be determined through simple actions performed on the same, within the SmartEditor.
- Improved the usability of the configuration panel. Configuration options are now organized in tabs, form's fields can now be "searched" by name or type.
- Introduced a new View preview feature directly in the editor.
- Introduced the functionality to roll-back widget configuration changes during an editing session.

Content Script¶

Extension Package for Blazon (NEW)¶

Programmatically manage rendition jobs on Blazon

Copyright © 2013-2022 AnswerModules Sagl

Extension Package for the integration with S3 by AWS (NEW)¶

Extension Package for xECM (NEW)¶

Utilize Content Script to programmatically create and administer Connected Workspaces

Extension Package for Office documents¶

Several enhancements have been introduced:

- It is now possible to change the width of the columns in an Excel file
- It is now possible to merge Word files
- Improved support for the systematic creation/editing of Word files

Copyright © 2013-2022 AnswerModules Sagl

All Enhancements in version 2.4.0¶

ID	Scope	Description
#0000598	Content Script	It's now possibile to embed resources into CSEmail (e.g. a logo image)
#0000597	Extension - Docx	It's now possibile to merge multiple Word documents togheter
#0000596	Extension - Docx	It's now possibile to change the size of Excel columns
#0000594	Module Suite	Cache service now supports the direct caching of CSNode objects
#0000593	Content Script	Objects extending ServiceContextAwareObject are now Serializabile. Before to attempt to serialize them call the detach method to
#0000592	Content Script	CSNode's metadata are now always accessible no matter which method as been used to load the node
#0000588	Content Script	It's now possibile to change the priority for scheduled tasks
#0000585	Extension - AmGui	AddItems override is not working on Compound Documents
#0000570	Content Script	Add support form Windows authentication in CSWS
#0000562	Module Suite	REST APIs: Being able to set StatusCode on success, Content Type on error
#0000590	Module Suite	CSMultiButtons in collections

Issues Resolved in version 2.4.0¶

ID	Scope	Description
#0000589	Beautiful Webforms	CreateForm method does not work when submission mechanism = WORKFLOW
#0000583	Beautiful Webforms	Multiple collapsible Panel Containers collapse wrong panel when clicked
#0000582	Beautiful Webforms	Passing an empty string as value for a field of type "Date", results in wrong value stored in the form

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000578	Beautiful Webforms	Under undetermined circumstances getTemplate function causes a trace file's generation
#0000576	Content Script	pat162000373_CS64_WIN breaks setting major/minor version and num of versions to keep through CS
#0000574	Script Console	Memory leak in CommandLauncherJob if the scheduled script performs actions related to RBWF
#0000568	Content Script	It's not possible to programmatically start a WF having WebReport WP enabled
#0000595	Beautiful Webforms	Validation of Views' source code is not working and a lot of error messages are generated within the logfile
#0000591	Content Script	It's not possible to update a CSUser without providing a new password for the same
#0000584	Beautiful Webforms	PrimitiveField contains a debugging instruction which is logging at error level
#0000581	Extension - Docx	xlsx.getAllWorksheets() API returns an empty list
#0000579	Content Script	Unable to send on a CS Workflow task configured to run in background if a previous run of the same returned an error
#0000577	Content Script	Saving script fails if script body is empty
#0000575	Content Script	Templating service error occurs when using Escape XML utility

Version 2.3.0 - Release notes¶

Release Date	End of AMP(*)	End of Life
2017-12-22	2020-12-22	2021-12-22

(*) Active Maintenance Period

This guide

The information presented in the on-line guide are mostly non-version specific. AnswerModules team does its best to ensure that, where necessary, is made clear that the information presented is only applicable to specific versions, however if you are looking for this version-specific documentation, you can find it here *(http://developer.answermodules.com/manuals/2.3.0)*

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Module Suite version 2.3.0.

Copyright © 2013-2022 AnswerModules Sagl

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

PDF Version

Major Changes in version 2.3.0¶

Beautiful WebForms Form Builder¶

FormBuilder widgets library has been deeply revised in order to further simplify WebForm views creation.

Enhanced support for Internationalization¶

The support for forms' views internationalization has been improved. It's now possible to mark labels and other widgets' properties for being inserted into localization files. Localization files can now even be stored directly on Content Server.

Inline FormTemplate Manipulation¶

FormBuilder now supports the inline creation of fields of type “Set”.

Content Script¶

Auditable and indexable¶

Content Script objects are now indexable (upon proper configuration in the Base Configuration).

Content Script execution is now auditable (upon proper configuration in the Base Configuration).

Scheduling and Callbacks¶

Several performance enhancement related to script scheduling and both for synchronous and asynchronous callbacks.

All Enhancements in version 2.3.0¶

ID	Scope	Description
#0000541	Content Script	Determine if a category needs to be upgraded

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000545	Extension - AdLib	Support overriding of job ticket output folder path
#0000540	Content Script	It's now possibile to reduce the DA Load related to CSEvents management
#0000539	Content Script	It's now possibile to disable CSEvents (preventing the system to record the event in the DA framework)
#0000538	Beautiful Webforms	FormBuilder now supports Set creation
#0000523	Beautiful Webforms	Form Builder should be using the "advanceMode" flag to decide whether to display the visual builder or the source code editor
#0000524	Beautiful Webforms	Form Builder layout should be preserved for BWF views edited with source code editor
#0000522	Beautiful Webforms	Form Builder should be initialized with last available Form Builder layout when opening a manually edited view
#0000495	Module Suite	Being able to override WebNodeActions using CSVolume scripts
#0000527	Content Script	Improve UI for importing and upgrading ModuleSuite Library
#0000532	Content Script	Search service API behavior changes with OTCS 16.2 release
#0000518	Module Suite	Content Script workflow steps can now be executed through the Workflow agent
#0000525	Content Script	New API to access valid values list for Category popup fields
#0000535	Module Suite	Optimized FormView footprint on database
#0000533	Content Script	Autocompletion provides now additional information regarding existing APIs
#0000512	Extension - Docx	Initial support for PPTX files
#0000530	Beautiful Webforms	Is now possible to use SWF to create generic purposes forms
#0000511	Beautiful Webforms	Smart Drop Down DB Lookup - javascript performs too many unnecessary async calls
#0000499	Content Script	Make Content Script indexable
#0000500	Extension - Docx	Being able to update SpreadSheet and single cell formulas
#0000497	Content Script	Being able to execute a LiveReport through the "Fast" interface for SQL

Copyright © 2013-2022 AnswerModules Sagl

Issues Resolved in version 2.3.0¶

ID	Scope	Description
#0000548	Beautiful Webforms	The "showtime" flag is not correctly handled in Date fields for forms loaded with forms.getWorkFlowForm(..) API
#0000542	Content Script	Issues in creating new binders with CSSynchEvents enabled
#0000547	Content Script	Error setting FromDate and ToDate in Physical Object Template
#0000516	Beautiful Webforms	Checkbox component not correctly initialized when creating a new BWF view
#0000537	Beautiful Webforms	the formToMap method of the FormService might rise exceptions parsing dates
#0000519	Content Script	Default value of Content Script static variables causes "Compilation Failed" warning to be added at every save
#0000504	Beautiful Webforms	Currency Field doesn't trigger onChangeAction in IE
#0000492	Beautiful Webforms	Javascript init script for Datatable widget is missing
#0000496	Beautiful Webforms	ADN Widget's javascript init function has the "support" and "context" variables hard-coded
#0000526	Beautiful Webforms	Request parameters are not passed to Form when executed through nickname
#0000509	Content Script	Regression in GCSCategory constructor derived from #0000490
#0000513	Beautiful Webforms	"Item reference Popup" component search error after update to Module Suite 2.2
#0000534	Content Script	Content Script editor automatic code validation needs resource optimization
#0000529	Beautiful Webforms	Error exporting form view with 'None' value selected in the 'Select Template' option in Form Builder
#0000536	Module Suite	Content Script scheduling is not working properly on 16 and 16.2 (unable to un-schedule)
#0000517	Beautiful Webforms	Minor visual issues in top bar buttons in Form Builder and Content Script Editor
#0000520	Content Script	"Test" button in Content Script Editor saves the current code but executes the previous version

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000510	Content Script	Managecallbacks script might fail with case-sensitive databases
#0000490	Content Script	GCSCategory is now CSNode aware, thus is possible to drive inheritance for sub nodes
#0000488	Content Script	docman.getNodesInContainer(..) API fails if target node contains a virtual folder
#0000491	Content Script	Virtual Folder nodes are not managed properly on CS16
#0000506	Content Script	NextUrl parameter is not passed to the execution context when executing a script using the open command
#0000508	Content Script	docman.rhRequest(..) API calls fail if Trusted Referring Websites are set in Content Server Security Parameters Admin page
#0000505	Beautiful Webforms	Form field "removeField(..)" API performs a wrong check on index
#0000507	Beautiful Webforms	Change event handlers are triggered twice
#0000503	Content Script	Execution of runCS instruction fails in callback-scripts caused by a NPE relative to CSVARS
#0000501	Beautiful Webforms	FormBuilder might fail to load on huge views
#0000494	Content Script	createRendableForm(..) API returns invalid content if IIS Application Request Routing is configured (for Brava)
#0000498	Content Script	OScript deserialization might fail for RecArrays

Version 2.2.0 - Release notes¶

Release Date	End of AMP(*)	End of Life
2017-05-15	2020-05-15	2021-05-15

(*) Active Maintenance Period

End of Life

This release has reached the end of its life and is no longer maintained or supported.

Copyright © 2013-2022 AnswerModules Sagl

This guide

The information presented in the on-line guide are mostly non-version specific. AnswerModules team does its best to ensure that, where necessary, is made clear that the information presented is only applicable to specific versions, however if you are looking for this version-specific documentation, you can find it here *(http://developer.answermodules.com/manuals/2.2.0)*

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Module Suite version 2.2.0.

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

PDF Version

Major Changes in version 2.2.0¶

License¶

Whenever the number of licensed seats are exceeded, the Module Suite's licensing manager starts logging a warning message for every operation performed.

Beautiful WebForms Form Builder¶

FormBuilder has been deeply revised in order to further simplify WebForm views creation.

CHEH Snippets¶

The feature that allows a Form designer to inject Content Script snippets into CLEH scripts (OnLoad, PreSubmit, OnSubmit) now depends on the widget's configuration itself. In other words, any change applied to the widget's configuration will trigger an equivalent change into the injected piece of Content Script.

This behavior can be disabled either through the "Auto-inject code" switch, displayed at the very top of any widget's configuration panel or deleting the synchronization hash injected as part of the header of all the CLEH snippets.

Widget Visibility¶

A simplified widget-visibility rule builder can now be used to determine when a widget should be displayed in the form.

Copyright © 2013-2022 AnswerModules Sagl

Buttons' Icons and Colors¶

Easy to use icon-color selectors have been introduced for button-widgets.

Inline FormTemplate Manipulation¶

Add functionality that allows Form Template manipulation (add new fields) directly from within the FormBuilder.

New And Updated Widgets¶

Currency, Include SmartUI Widget, Chart, DocuSign, Handsontable, DropDown DB Lookup, Set ViewParams Variable, View Template Logo, Smart DropDown DB Lookup, Include Script Result, Redirect to URL, Workflow Comments, Custom Action Button, Submit Button With Param

Field default value¶

Add functionality that allows setting the default value displayed by the form widgets

OnLoad script returns JSON Data¶

CLEH script management has been modified in order to allow OnLoad script to return JSON data, thus to be used for implementing Ajax-enabled backend services for BWF-widgets (e.g. Handsontable widget).

New Content Script APIs¶

New Content Script APIs and API extension packages have been released. Collaboration APIs have been extensively revised and simplified. New extension packages include:

DocBuilder¶

Allow developers to programmatically produce PDF and Word files. The new APIs are accessed through the new "*docbuilder*" service endpoint, available upon installation of the extension package.

Callback Scripts¶

Synchronous Callback Scripts are now executed in isolated context (a separate execution context for each script). To switch back to the previous implementation set the 3rd bit of the "amcs.core.debugEnabled" configuration bitmask to one (e.g. amcs.core.debugEnabled=4).

Copyright © 2013-2022 AnswerModules Sagl

All Enhancements in version 2.2.0¶

ID	Scope	Description
#0000431	Content Script	Templating service is wrapping CS Context in Templating Context when using subviews in form views
#0000440	Beautiful WebForms	Improved robustness of Jquery Interdependencies widget
#0000474	Beautiful WebForms	Improved accuracy for Form to PDF rendering through HTML to PDF rendition
#0000466	Beautiful WebForms	User by Login revised in order to avoid preload of all users if no filter is selected
#0000460	Content Script	The Callback-Scripts management class has been refactored in order to switch from a single execution context mode to a fully separated set of contexts
#0000465	Beautiful WebForms	Enhance visibility of CSS grid columns
#0000445	Mail Service	Mail Service - Added support for Receipt Request and other SMTP headers
#0000446	Mail Service	Mail Service - Added support for BCC addresses
#0000485	GUI Service	SQLQueryRowProvider features methods for executing paginated SQL Queries
#0000484	Content Script	SqlService ext-pack features methods for executing paginated SQL Queries
#0000469	Beautiful WebForms	CLEH widgets snippets are now standard CS snippets, evaluated using widget's configuration
#0000478	Content Script	Major enhancement for Collaboration service
#0000444	Beautiful WebForms	New widget for currencies
#0000463	Content Script	The information related with the original user ID and username are now available in the script Execution Context
#0000475	Content Script	Improved performances of Templating service - producePDF(..) API

Copyright © 2013-2022 AnswerModules Sagl

Issues Resolved in version 2.2.0¶

ID	Scope	Description
#0000429	Beautiful WebForms	am_printFix function in am_init file is not working properly (Form to PDF)
#0000439	Beautiful WebForms	Jquery interdependencies widget does not support binding to Radio Basic widget
#0000457	Beautiful WebForms	JS Conditional Container v1 escapes "operator" configuration causing a javascript error in client.
#0000438	Beautiful WebForms	User by Login widget not working correctly on environments with case sensitive database
#0000459	Module Suite	Wrong documentation in Base Configuration for amcs.core.callbackSynchEventsEnabled flag
#0000473	Beautiful WebForms	am_CssViewDependecies and am_JsViewDependecies variables in the viewParams map are overridden whenever a view invoke a CLEH action (the information regarding Form's static resources is lost)
#0000476	Beautiful WebForms	PDFPreview tool not working on OTCS 16.0.3
#0000468	Beautiful WebForms	Default Submit Button widget is not initialized correctly when creating a new view
#0000442	Content Script	Email setCharset(..) API has wrong help text
#0000441	Content Script	Email "distribute" functionality sends same notification multiple times to the same recipients
#0000481	Content Script	Docman copyNode(..) API adds a new version to copied node as default behavior
#0000482	Content Script	Docman moveNode(..) API adds a new version to node as default behavior
#0000483	Content Script	Docman copyNode(..) API only copies the last version of the node as default behavior
#0000437	Content Script	Reading excel cell values for spreadsheet columns after AA returns null values
#0000452	Content Script	Enabling Content Script scheduling with default configuration fails
#0000428	Content Script	Content Script autocomplete function fails if script csvars is empty
#0000454	GUI Service	Performance issues due to a regression in version 2.1

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000464	Content Script	Regression in ServiceWrapperFactory
#0000453	Content Script	Issues using CSSearchQueryBuilder when search slices have been renamed
#0000479	Content Script	It's now possibile to proper cast a CSNode to its subclasses (e.g. node as CSDocument)
#0000477	Beautiful WebForms	Empty value for amSaveValues property of form objects returned by getFormInfo method
#0000472	ESign	Unable to reject on multi-user step
#0000471	Beautiful WebForms	Mapping Script widget attached to Smart Dropdown clear fields on reload if AJAX initialization is disabled on Smart Dropdown
#0000462	Content Script	Callback scripts: NodeCreate, NodeCreatePre, NodeUpdate are not managing rollback properly
#0000461	Content Script	Wrong parent-callback event registered on NodeCreatePre event
#0000451	Content Script	Request parameters are not passed to Content Script when executed through nickname
#0000472	ESign	ESign Service is not working properly with ModuleSuite version > 2.1
#0000448	Beautiful WebForms	ESign Widget is not working properly with ModuleSuite version > 2.1
#0000447	Beautiful WebForms	amSaveValues are not updated on the basis of the form object

Version 2.1.0 - Release notes¶

Release Date	End of AMP(*)	End of Life
2016-11-16	2019-11-16	2020-11-16

(*) Active Maintenance Period

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Module Suite version 2.1.0.

End of Life

This release has reached the end of its life and is no longer maintained or supported.

Copyright © 2013-2022 AnswerModules Sagl

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

PDF Version

Major Changes in version 2.1.0¶

License¶

Module Suite's activation license now keeps in consideration: the system's fingerprint and the number of purchased seats.

Beautiful WebForms library of widgets¶

The Beautiful WebForms library of input widgets has been significantly improved and simplified. The library comprises now more than ninety elements. The widgets/view template architecture has been revised in order to speed up and favor the development of new input widgets by customers and partners. Widgets' external dependencies can now be easily tracked allowing developer to focus only on core functionalities.

Beautiful WebForms Studio¶

Beautiful WebForm Studio wizard is now shipped with Module Suite. The Studio can be accessed directly from the Content Script Volumes (CSVolume:CSTools:Beautiful WebForms Studio).

New Content Script APIs¶

New Content Script APIs and API extension packages have been released. New extension packages include:

Web-Services API extension pack¶

Content Script now features a complete set of APIs for consuming both REST and SOAP web services. The new APIs are accessed through the new "*csws*" service endpoint, available upon installation of the extension package.

Copyright © 2013-2022 AnswerModules Sagl

All Enhancements in version 2.1.0¶

ID	Scope	Description
#0000427	Script Console	It is now possible to export a Tomcat 8 compatible configuration using exportWar.cs script
#0000413	Beautiful Webforms	Introduce a simpler way of managing the loading of static dependencies for form views
#0000423	Beautiful Webforms	PDF web viewer tool based on pdf.js
#0000418	Beautiful Webforms	New validation constraint: Content Script (remote invocation)
#0000410	Beautiful Webforms	Support for TKL and ADN TKL form fields
#0000414	Content Script	Is now possible to execute a Content Script programmatically. New API: docman.runContentScript()
#0000377	Content Script	Add possibility to un-assign task
#0000380	Content Script	New API: CSUser.getDepartmentGroup()
#0000381	Content Script	New API: users.getGroup(Long groupID)
#0000382	Content Script	Add library snippet for Smart Drop Down backend service
#0000388	Beautiful Webforms	When loading a submitted form with date fields with the "show time" flag active seconds information is lost.
#0000392	Beautiful Webforms	Add support for form page repositioning to first error when a client side validation error occurs.
#0000393	Beautiful Webforms	Read-only checkbox layout should be similar to active checkbox

Issues Resolved in version 2.1.0¶

ID	Scope	Description
#0000422	Beautiful Webforms	Checkbox widgets set to "readonly" lose the selected value on page reload
#0000421	Beautiful Webforms	Radio components in forms using BWF library V1 not working after upgrade from version 1.7.0 to 2.0
#0000426	Beautiful Webforms	Multiple collapsible Panel Containers included in sub-views collapse wrong panel when clicked
#0000425	Extension - LDAP	LDAP Service fails to initialize profile with 'secure' parameter set to 'false'

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000419	Beautiful Webforms	Submit button clicked after the execution of an action which returns a file keep triggering the same action
#0000411	Beautiful Webforms	Regression in version 2.0.0: V1 am_init.js file has been overridden
#0000416	Beautiful Webforms	OnChange Script BWF snippet does not work with fields with multiplicity higher than one
#0000412	Content Script	Sql Service ignores parameters of type GString
#0000383	Beautiful Webforms	OnChangeAction snippet does not trigger with Radio Basic components
#0000389	Beautiful Webforms	If a redirect instruction is used in the AfterSubmit script of a form workflow step the workflow's step won't be completed
#0000385	Beautiful Webforms	When loading a submitted form with date fields with the "show time" flag active time information is lost
#0000394	Extension - Docx	Field updating on Office documents fails when there are properties with spaces in the property name
#0000408	Content Script	Inline API Docs panel in Content Script Editor minimizes after clicking on any link
#0000395	Extension - Docx	Field updating on Office documents fails when there are properties with long property name
#0000397	Content Script	Outdated configuration details for callbacks in Content Script inline documentation
#0000399	Beautiful Webforms	Selecting left or right position for labels in components in a Grid (Bravo) container causes a wrong positioning of the labels
#0000400	Extension - Docx	Error creating an empty XLSX spreadsheet
#0000398	Content Script	Error with template.producePDF(...) when a File is passed as the template.
#0000401	Content Script	Accessing admin pages generates a trace file when you have an error coming from underlying DBMS
#0000404	Module Suite	CSCategory getAttributes returns just the name for attributes inside a set
#0000403	Content Script	Not able to rollback a transaction using nodecallbacks

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000407	Beautiful Webforms	Form fields requireness is not properly managed when retrieving form object with forms.getFormInfo API
#0000405	Content Script	Disable Weblingo overwrite without having to rise an Exception
#0000406	Content Script	template.producePDF(..) API causes a resource leak
#0000386	Beautiful Webforms	Wrong values in submitted checkbox and user fields when using the forms.submitForm(..) API
#0000387	Content Script	It's no longer possible to use a Content Script service as a Content Server standard REST
	Content Script	Impersonate ObjectFactory is not working (only Admin can use the feature)
#0000390	Content Script	ResourceManager rises an NPE when amcs.core.tempFilePath has not been specified

Version 2.0.0 - Release notes¶

Release Date	End of AMP(*)	End of Life
2016-07-22	2019-07-22	2020-07-22

(*) Active Maintenance Period

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Module Suite version 2.0.0.

End of Life

This release has reached the end of its life and is no longer maintained or supported.

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

PDF Version

Copyright © 2013-2022 AnswerModules Sagl

Major Changes in version 2.0.0¶

Support for Content Server 16¶

Module Suite 2.0.0 now includes support for Content Server version 16.

Content Script objects are available and executable form the new Smart UI.

Content Scripts can be used to build Smart UI tiles, and are available within the Perspective Builder when creating new Smart UI perspectives.

Currently supported platforms include:

- Content Server 10.0.x
- Content Server 10.5.x
- Content Server 16

Completely renewed development environments and editors¶

All Module Suite integrated developments environments, including

- Beautiful WebForms Form Builder,
- Content Script Editor,
- Template Editor,
- Content Script Snippet Editor,
- Form Widget Editor

have been completely renewed, and ergonomics have been greatly improved.

Beautiful WebForms Form Builder now features additional controls for multiple selection when positioning, editing or deleting widgets in the working area, as well as support for the new functionalities of the widgets in the library. Important functionalities (template selection, associated Content Script editing) have been made directly accessible from the main editor panel.

Content Script Editor content assist functionalities have been improved.

Copyright © 2013-2022 AnswerModules Sagl

Full revamp of Beautiful WebForms widgets and templates libraries¶

Form Widget in the Beautiful WebForms component library now feature

- Component libraries are now versioned in super-libraries. Module Suite 2.0.0 ships with library version 2. Version 1 corresponds to widgets available in Module Suite 1.7. Users are free to create their own library versions.
- All widgets have been refactored to provide a clean separation between presentation and dynamic behavior (to ensure compatibility with new CS16 Smart UI)
- Complex widgets which require backend scripting code now feature the possibility to automatically inject the custom code (reduces need to manually edit scripts)
- All widgets have seen their customization capabilities improved and extended.

New Content Script APIs¶

New Content Script APIs and API extension packages have been released, for a total of 400+ new documented APIs. New extension packages include:

Records Management API extension pack¶

Content Script now features a complete set of APIs for Content Server Records Management. The new APIs are accessed through the new "*recman*" service endpoint, available upon installation of the extension package.

Physical Objects API extension pack¶

Content Script now features a complete set of APIs for Content Server Physical Objects. The new APIs are accessed through the new "*physobj*" service endpoint, available upon installation of the extension package.

LDAP integration API extension pack¶

Content Script now features a basic set of APIs that allow to perform queries on any LDAP server. The new APIs are accessed through the new "*ldap*" service endpoint, available upon installation of the extension package.

SQL extension pack¶

Content Script now features a new API to perform database queries against the Content Server database, as well as over external databases. The new API provides support for query parameter bindings and SQL expression templates, and can be accessed through the new "*sql*" service endpoint, available upon installation of the extension package.

Copyright © 2013-2022 AnswerModules Sagl

Content Script PDF API improvements¶

A new set of APIs is now available within the "*pdf*" service endpoint. The new APIs allow to interact with PDF form documents, by extracting form information and data, automatically filling in form fields, and finalizing a PDF form.

Third party dependencies upgrade¶

Most third party dependencies included in the Module Suite release have been updated to the latest stable release. A complete list of the dependencies can be found within each module's installation folder.

Weblingo override functionality¶

A new advanced functionality, which allows administrators to configure overriding of any Content Server weblingo file, is now available as part of Content Script version 2.0.0.

Cross-script referencing¶

Any Content Script object that has been assigned a nickname within a specific namespace ("CSxxxx") can now be referenced directly by nickname within a second Content Script. This functionality allows, for example, to create function libraries without the overhead of executing the second script every time.

Save views as Widgets¶

Views created with the Beautiful WebForms builder can now be saved as a custom "composite" widgets and reused through the usual drag & drop process within another view. When such a composite widget is dropped within a view, it will be split in its basic parts, and each part will be configurable separately.

Composite widgets can be managed within the widget library as standard Beautiful WebForms widgets.

Workflow Query builder¶

A new workflow query builder is available as part of the "workflow" service APIs. The builder is intended to simplify the interaction with the workflow search service.

All Enhancements in version 2.0.0¶

ID	Scope	Description
#0000306	Module Suite	Module Suite objects are now fully compatible with Content Server transport

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000307	Script Console	It's now possible to schedule jobs while the console is offline
#0000322	Extension - Forms	Form Package attachments are now available in Remoted Workflow Forms and submitted forms (seq)
#0000332	Beautiful Webforms	Workflow Comments form snippet now includes CLEH configuration documentation
#0000333	Content Script	Content Script objects are now restricted objects by default
#0000334	Content Script	GCSSetAttribute and GCSPrimitiveAttribute now support each
#0000338	Content Script	Third party javascript libraries have been updated to latest stable version.
#0000339	Beautiful Webforms	Third party javascript libraries have been updated to latest stable version.
#0000342	Content Script	It is now possible to set a group as group leader using the setLeader(..) API on the CSGroup object.
#0000344	Beautiful Webforms	It's now possibile to convert form fields values to Dates
#0000345	Beautiful Webforms	Javascripts and Css resources can now be organized and included in Form Templates respecting dependencies.
#0000346	Beautiful Webforms	Form widget can now specify their own static dependencies (js, css)
#0000347	Content Script	It's now possible to create ANSTemplateFolder objects programmatically
#0000348	Beautiful Webforms	It's now possible to organize form widgets and form templates in libraries
#0000349	Extension - eSign	It's now possible to set the URL where to redirect navigation after having performed the sign step
#0000350	Script Console	Introduced the possibility to customize embedded jetty behaviour with init script
#0000351	Content Script	Enabled the possibility to control the mimetype of files returned as attachment
#0000352	Content Script	Optimize the CSResource file names.
#0000353	Content Script	Optimize the usage of "self" shortcut
#0000358	Module Suite	Initialization of logging system is now performed programmatically

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000359	Module Suite	CSNodes can now be exported in JSON (format compatible with REST APIs)
#0000360	Content Script	Is now possible to retrieve the full path for a node
#0000367	Module Suite	Is now possible to build queries on workflows data programmatially
#0000372	Extension - Forms	Script Console now supports forms built with Beautiful WebForms libraries V2
#0000374	Script Console	Added authentication layer on Script Console web interfaces
#0000375	Script Console	Session cookies are now enabled by default in application server configuration

Issues Resolved in version 2.0.0¶

ID	Scope	Description
#0000305	Module Suite	Content Script TemplateFolder webnode naming conflict
#0000308	Script Console	The script used to demonstrate the Console's internal scheduling features contains an error.
#0000309	Script Console	Java Classloader conflict running console as an standalone ApplicationServer application
#0000311	Content Script	SQL queries used to implement CSVolume caching have issues with case sensitive database installation
#0000312	Content Script	RunCS is triggering unnecessary script recompilation that might impact performances
#0000313	Content Script	Error when accessing property priority of CSWorkflowAssignedTask
#0000314	Content Script	SSL Error occurs when using RHOmnia proxy under https
#0000315	Extension - Docx	importMetadata API in xlsx service does not handle exceptions correctly
#0000316	Extension - Docx	importMetadata API in xlsx service attempts to process lines with empty dataID column
#0000317	Content Script	Class loading issues on Member Services (Content Server 10.0)
#0000318	Content Script	Unexpected error may arise when executing a custom REST service

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000319	Content Script	Content Script callback scripts are sometimes invoked twice
#0000320	Extension - Forms	Remote WebForms submission fails silently when application server and script console installation paths are on different drives
#0000324	Extension - Forms	View switching is not possible within Remote WebForms OnLoad scripts
#0000325	Extension - Forms	Checkbox widgets in Remote webforms do not handle events correctly
#0000326	Beautiful Webforms	Beautiful WebForms online documentation incorrectly references a non-existing "form.setValidationError(..)" API
#0000327	Beautiful Webforms	Workflow attachments folder identifier is not available in the form object when used in a user step
#0000328	Beautiful Webforms	The "Forms.createRendableForm(..)" API calls fail if target form object is not in the current script's context scope
#0000329	Content Script	Service calls fail when executed from different thread spawned with Thread.start
#0000330	Beautiful Webforms	Nesting multiple "Include Subview" widgets causes invalid form markup
#0000335	Content Script	Unable to send email to multiple recipients
#0000336	Content Script	The "asCSNode(..)" API does not behave in the same way as the docman API when retrieving nodes by path.
#0000337	Content Script	Unable to assign "long" values to category integer attributes
#0000341	Beautiful Webforms	SignaturePad component fails to initialize signature from pre-populated field value
#0000343	Content Script	Download link promoted function for regular documents disappears after installing Content Script
#0000354	Extension - Forms	Several improvements to FormBuilder and BWF Widgets
#0000355	Beautiful Webforms	Form service fails on "updateWorkflowForms(..)" if the form contains a set attribute
#0000356	Content Script	Not able to perform update or instert through runSql method if report ext is not installed (Content Server 10.0)

Copyright © 2013-2022 AnswerModules Sagl

ID	Scope	Description
#0000357	Extension - PDF	Temp resources are not cleaned when extracting pages from a PDF (both as PDFs and JPGs)
#0000361	Content Script	Cron expression is not persisted when scheduling a script in advanced mode
#0000362	Module Suite	Unable to change owner of document inside Project
#0000363	Module Suite	Form set fields are not always updated within workflows
#0000364	Script Console	Script Console "loadConfig" command fails if there are disabled services in the Content Server Base configuration that is being exported.
#0000365	Script Console	Placeholder expressions in exported Base Configuration are not valid for Script Console
#0000366	Module Suite	REST API Content Script always returns 200 OK Code
#0000368	Module Suite	Node features are empty if CSNode has been loaded with "fast" API variant
#0000369	Module Suite	It's no longer possible to send an email to multiple recipients specified in to or cc
#0000370	Script Console	Wrong error page format when running on external application server
#0000371	Extension - Forms	Wrong download attachment links in sample Remote WebForms form list page
#0000373	Extension - Forms	Potential security issue with content of form hidden fields
#0000376	Extension - Forms	Form attachments are present in the "views" list in the form.amRemotePack

Important Notes when updating Module Suite to version 2.0.0¶

Module Suite version 2.0.0 introduces a few paradigm shifts, mostly oriented to set the basis for a better support of the new Smart UI available with Content Server 16.

Noteworthy changes mainly involve Beautiful WebForms, and specifically the ways in which the libraries of widgets and templates are organized.

Module Suite version 2.0.0 introduces the concept of “*library versions*” for form widgets and form templates. When creating or editing a Beautiful WebForms view, editors must select which

Copyright © 2013-2022 AnswerModules Sagl

version of the widget library to use. The form templates should be chosen consequently (e.g. always use templates V2 with form views built using widget library V2).

Module Suite version 2.0.0 ships with two library versions:

- **V1**: this library matches the former Module Suite 1.7.0 widget library. It has been included for backward-compatibility, and can be safely discarded for new installations.
- **V2**: upgraded version specific to Module Suite 2.0.0. Objects within this library feature:
  - a clearer separation between widget presentation and dynamic behavior. Inline scripting has been removed from all widgets.
  - support for new editor features (label positioning, etc.)
  - support for dynamic dependency evaluation (third party or custom .js and .css inclusions)

Installing the new libraries¶

This procedure is only useful if there are preexisting Beautiful Webforms views on the target system. It can be safely ignored for vanilla installations.

After performing the standard module setup, the suggested procedure to upgrade the existing libraries related to Beautiful WebForms (*CSFormSnippets* and *CSFormTemplates*, located within the Content Script Volume) is described hereafter.

Upgrade procedure for CSFormSnippets¶

The *CSFormSnippets* library can be upgraded using the standard library upgrade procedure.

1. From the Module Suite Base Configuration, in the "*Manage component library*" section, choose the "*Upgrade*" option.
2. In the library selection window, select the *csformsnippets.lib* option and click "**upgrade**".
3. Upon completion, in the root of the Content Script Volume there will be 2 distinct folders related to *CSFormSnippets*, as shown below. At this point the new library is available for Beautiful WebForms.
4. Cleanup. The folder named "*_CSFormSnippets_BCK_yyyyMMdd_HHmm*" is a backup folder containing the previously installed library. It can be safely exported and/or removed. In case any of the standard widgets was customized, patched or otherwise modified, or new custom widgets were added within the standard library, make sure that you transfer any relevant changes to the new libraries before deleting the old version.

Copyright © 2013-2022 AnswerModules Sagl

Upgrade procedure for CSFormTEMPLATEs¶

Upgrade procedure for CSFormTemplates is slightly different, due to the fact that templates are referenced by object dataID within existing forms. In order to preserve functionality of existing forms, it is recommended to perform the upgrade as follows.

1. From the Module Suite Base Configuration, in the "*Manage component library*" section, choose the "*Upgrade*" option.
2. In the library selection window, select the *csformtemplates.lib* option and click "upgrade".
3. Upon completion, in the root of the Content Script Volume there will be 2 distinct folders related to CSFormTemplates:
    1. CSFormTemplates : the new template library
    2. *_CSFormTemplates_BCK_yyyyMMdd_HHmm* : a backup folder containing the previously installed template library.
4. The root of the "*CSFormTemplates*" folder will contain the following:
    1. a set of templates, which correspond to library V1 (Module Suite 1.7.0). These should correspond 1 to 1 with the templates contained in your backup folder: *_CSFormTemplates_BCK_yyyyMMdd_HHmm*
    2. a folder named V2, which contains new version 2 templates.
5. In the *CSFormTemplates* folder, delete all the templates (*DO NOT DELETE the V2 folder*) as shown in the next image.
6. Navigate to the *_CSFormTemplates_BCK_yyyyMMdd_HHmm* folder and MOVE all the old templates to the new *CSFormTemplates* folder (in this way, the dataIDs of the old templates will be preserved). The now empty *_CSFormTemplates_BCK_yyyyMMdd_HHmm* can be safely deleted.

Custom Form Templates and form widgets¶

Beautiful WebForms allows users to create custom libraries of form widgets, as well as new form templates. It is recommended to organize such custom elements within the Content Script Volume in dedicated, separate containers, in order to avoid issues when upgrading the standard libraries.

Copyright © 2013-2022 AnswerModules Sagl

Since version 1.7.0, the recommended structure was:

- Content Script Volume
  - CSFormSnippets
    - *<Customer name or custom component family name>*
      - *<custom widget A>*
      - *<custom widget B>*
  - *<Customer name>*
    - CSFormTemplates
      - *<custom template A>*
      - *<custom template B>*

With version 2.0.0, the recommended structure is extended to include the concept of library version.

- Content Script Volume
  - CSFormSnippets
    - V2
      - *<Customer name or custom component family name>*
        - *<custom component A>*
        - *<custom component B>*
  - *<Customer name>*
    - CSFormTemplates
      - V2
        - *<custom template A>*
        - *<custom template B>*

Custom widgets

Custom Beautiful WebForms widgets created prior to version 2.0.0 must be updated to be compatible with Beautiful WebForms Editor version 2.0.0. The procedure is straightforward and does not strictly require to perform any

Copyright © 2013-2022 AnswerModules Sagl

changes to the widgets' code. In order to update the existing widgets, perform the following steps for each one of them:

1. open the component editor
2. save the component

Saving the component will trigger recompiling, which is enough to ensure compatibility in the new Module Suite version.

Previous releases - Release notes¶

END OF LIFE

All versions of Module Suite prior to Module Suite 2.0 have reached the end of their life, they are no longer developed or supported.

The present document contains information regarding product enhancements, fixed issues and known issues related to AnswerModules Module Suite version previous of 2.2.0.

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, AnswerModules accepts no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

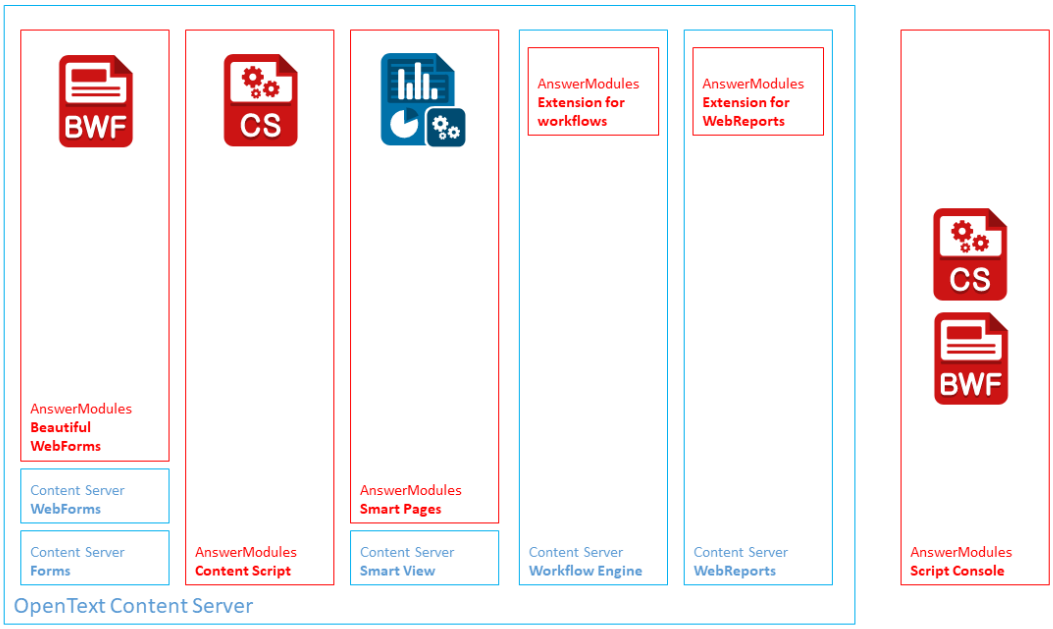
PDF Version

Copyright © 2013-2022 AnswerModules Sagl

Architecture

Module Suite

Module Suite for Content Server by AnswerModules is a comprehensive framework that includes various innovative solutions and extensions modules for OpenText Content Server.



Beautiful WebForms¶

The Beautiful WebForms Framework is an enhancement to the standard OpenText WebForms module that provides developers with all the required tools to create and manage next generation form based applications on Content Server. The module significantly contributes in delivering to the application's end users a modern, comfortable and ergonomic usage experience while at the same time lowering overall development and maintenance costs.

Content Script¶

Content Script is the first genuine scripting engine for OpenText Content Server. Content Script enables the creation of a new type of executable script object, capable of both automating actions that can be performed through the standard Content Server UI, as well as creating custom interfaces, consoles, reports, and more.

Copyright © 2013-2022 AnswerModules Sagl

Note

Content Script API and API Extension Packages (CSEPs)

One of the most powerful features of Content Script lies in the fact that within the Content Script code it is possible to interact with Content Server itself and with external services or data sources through a set of service APIs. The API layer is engineered for extensibility, and new APIs are released periodically to enable the most various tasks. Also, thanks to the Content Script SDK, Modules Suite owners and developers can create their own extensions. CSEP can be enabled and disabled dynamically from within the administrative pages of Content Server.

Smart Pages¶

Smart Pages is a solution that allows developers to leverage the Content Script template engine's capabilities to create UI elements of any sort by adopting a rigorous MVC *(https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller)* design pattern. Smart Pages have been primarily engineered to be utilized in the context of Smart View applications, where they can be useful for creating Smart View perspective tiles. Smart Pages replaces the Module Suite View extension for Smart View which has been discontinued at version 1.8.

Script Console¶

Unlike the Content Script Module and Beautiful WebForms Module, which are standard extension modules and live inside OpenText Content Server, the Content Script Console is a standalone, multi-platform (Unix, Windows) environment for the execution of Content Scripts and Beautiful WebForms. As such, it is executed separately from Content Server, potentially on different physical environments (such as an Administrator's own workstation or a server in a network DMZ), but retains the capability of interacting with one or more Content Server environments.

Module Suite default extensions¶

Module Suite comes out-of-the box with a set of extensions that enable new usage scenarios for core Content Server modules.

Content Script Extension For Workflows¶

The Content Script Extension for Workflows allows you to add Content Script Steps to new or existing Content Server Workflow Maps.

Content Script Steps are automatic steps that will execute the associated Content Script when triggered. The execution outcome will be interpreted by the step itself in order to route the Workflow to the next step. It is possible to build expressions that check for successful execution, execution errors or that interpret the outcome of the script.

Copyright © 2013-2022 AnswerModules Sagl

The usage of Content Script Steps can reduce to a minimum the need for custom Event Trigger Scripts.

Content Script Extension For WebReports¶

The Content Script Extension for WebReports improves standard WebReports functionality by introducing new usage scenarios, such as:

- the possibility to use a Content Script as a Data Source for WebReports
- the possibility to execute WebReports from within a Content Script
- the possibility to execute Content Scripts from within a WebReport thanks to a custom subtag

Module Suite Extension For ClassicUI¶

The Module Suite Extension for CalssicUI is a simple and fast way to enhance the OpenText Content Server user experience.

This powerful tool gives the possibility to manage: - An objects menu options - Manage default and custom columns at run-time - Redesign guis by embedding fancy widgets - Customize the way items are being created in the system - Dynamically create forms without having to write HTML code - Easily perform massive operations

Module Suite Extensions

ModuleSuite Extensions enhance the capabilities of existing standard Content Server Modules, if they are installed on the systems.

ModuleSuite Extension For DocuSign¶

ModuleSuite Extension For DocuSign has been developed in order to dramatically simplify the integration between OpenText Content Server and the DocuSign® signing platform. These integration solutions are based on AnswerModules' core solution, Module Suite, and thanks to their outstanding flexibility can be utilized to implement all sorts of use-case scenarios.

Most common usage scenarios

- Manually starting a DocuSign® signing workflow directly within the Content Server UI in order to have a set of Content Server documents signed by a group of Content Server users
- Manually starting a DocuSign® signing workflow directly within the Content Server UI in order to have a set of Content Server documents signed by a group of external users;

Copyright © 2013-2022 AnswerModules Sagl

- Managing one or more DocuSign® signing workflows, each one involving both Content Server users and non-Content Server users, as part of the execution of a Content Server internal workflow

ModuleSuite Extension For ESign¶

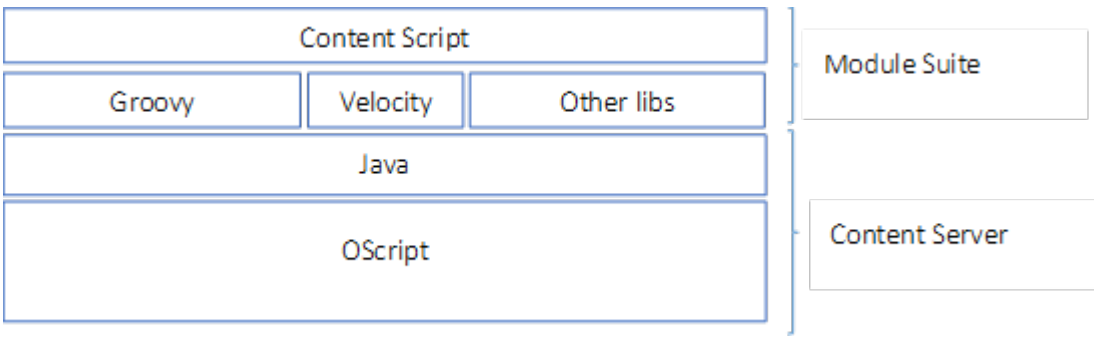
ModuleSuite Extension For ESign allows for Beautiful WebForms to be used as the signing step in a signature workflow.

Applicative Layers

One of the main reasons that brought to the creation of the Module Suite was the need to improve Content Server's capability of integration with other systems. For this very reason, on top of an OScript Layer that implements most of the Content Script Core functionalities, we developed an integration layer that makes use of the Content Server embedded Java Virtual Machine.

Content Script was developed with a language grammar and syntax fully compatible with Groovy, the well-known Scripting language for Java, in order to speed up development and most importantly open Content Server to a wider range of developers than the reduced OScript developers' community.

On the other hand, being OScript's grammar very similar to Groovy's, OScript developers should easily find their way with the Content Script language.



Note

In recent years, more and more functionalities of Content Server have been making use of the embedded Java Virtual Machine. Nevertheless, the standard level of isolation of these components has not yet been significantly improved. It is still up to system administrators and developers to manually assure the absence of conflicts in the system when new Java libraries become necessary. Module Suite comes with a higher level of isolation and implements its own additional libraries management

Copyright © 2013-2022 AnswerModules Sagl

Requirements, links and dependencies

Supported Content Server versions¶

The Modules Suite currently supports the following Content Server versions:

- 10.0.x (up to version 2.1)
- 10.5.x
- 16.0.x
- 16.2.x

Dependencies¶

Module or Component	Included In	Depends On
Content Script	-	-
Beautiful WebForms	-	Content Script
Script Console	-	Content Script
Remote Beautiful WebForms	Script Console	Beautiful WebForms
Module Suite Extension For WebReports	Content Script	WebReports
Module Suite Extension for Workflows	Content Script	
Module Suite Extension for Classic UI	AMGUI Ext.Pack	Content Script
Module Suite Extension for SmartUI	AnswerModules Module Suite for SmartUI	Content Script
Module Suite Extension for ESign	AnswerModules Content Script eSign ExtPack	Content Script, Beautiful Webforms, ESign
Module Suite Extension for DocuSign	AnswerModules Module Suite extension for DocuSign	Content Script, Beautiful Webforms, Script Console

Modules layouts

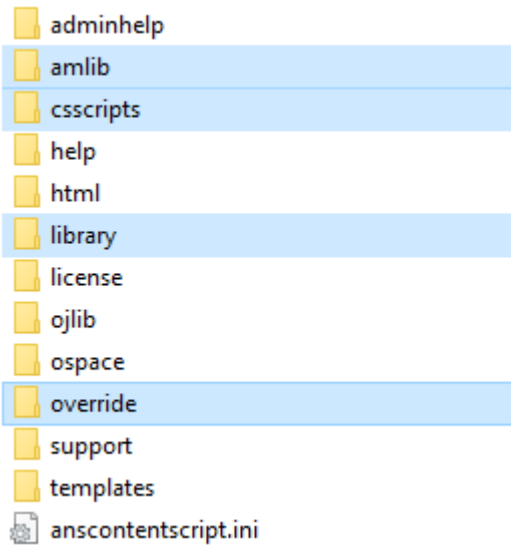
Module Suite's modules present a peculiar layout that differentiate them from most of the Content Server's modules you might have worked with. Knowing the modules' internal structure

Copyright © 2013-2022 AnswerModules Sagl

is of primary importance when it comes to: upgrading, maintaining or extending your Module Suite instance.

Content Script¶

Content Script features a set of layout differences in respect to standard Content Server modules. In the following paragraphs each one of these differences is discussed in details.



amlib¶

The "**amlib**" directory contains all the core libraries of the Content Script Java Layer. It is also used to deploy and manage Content Script Extension packages. If a Content Script API Service (CSAS) , made available from a CSEP, needs to load its own Java libraries, then they will be deployed in a sub-directory of the amlib directory having the same name of the Content Script API Service identifier. This way, two different Content Script API Services can load two different version of the same Java library.

csscripts¶

Content Script scripts can be used and also invoked directly from OScript. Scripts under this folder can be executed as part of OScript scripts or functions. Some of them are used to implement Module Suite's administrative pages.

> The Content Script OScript APIs are not covered by this training manual.

library¶

Module Suite's components behaviour and functionalities can be modified and extended by manipulating the content of the **Content Script Volume** (a Content Server's Volume created when installing the Content Script module).

Copyright © 2013-2022 AnswerModules Sagl

The purpose of most of the structure and content of the Content Script Volume can be easily understood by simply navigating the volume thanks to the "convention over configuration" paradigm that has been adopted. That means that most of the time, simply creating the right Content Script, Template Folder or Template in the right place will be enough to activate a specific feature. The default configuration (i.e. the default Content Script Volume's structure) should be imported as part of the installation procedure of the Content Script module.

In the next sections we will refer to specific locations in the Content Script Volume content as "**Component Library**" or simply "**Library**". This directory contains the default initial version of the Library and will be used later on to manage Library's backups and upgrades. The Library can always be imported, exported or upgraded directly from the Module Suite's administrative pages.

override¶

Content Script can be used to deeply customize the Content Server standard UI through a non-disruptive (applying non-permanent modifications) functionality that allows developers to override the standard result of a Content Server **weblingo** file evaluation with the result of a Content Script execution.

Weblingo override functionality is controlled by XML configuration files to be placed in the "*override*" folder in the anscontentscript module.

```
<?xml version="1.0" encoding="UTF-8"?>
<override>
    <active>false</active>
    <target>
        <![CDATA[E:\OTHOME\module\webattribute_10_5_0\html\attrstring.html]]>
    </target>
    <!-- Content Script ID -->
    <script>ID</script>
    <!-- BEFORE, AFTER, CUSTOM -->
    <mode>CUSTOM</mode>
    <!-- Optional Script's Parameters -->
    <params>
        <entry>
            <key>key</key>
            <value>value</value>
        </entry>
    </params>
</override>
```

Within the folder, you should find a sample XML configuration file that should be quite self-explanatory. The XML file points to a Content Script object, identified by *dataID*, which implements the functionality.

Setting the "*active*" flag to "*true*" will activate the override.

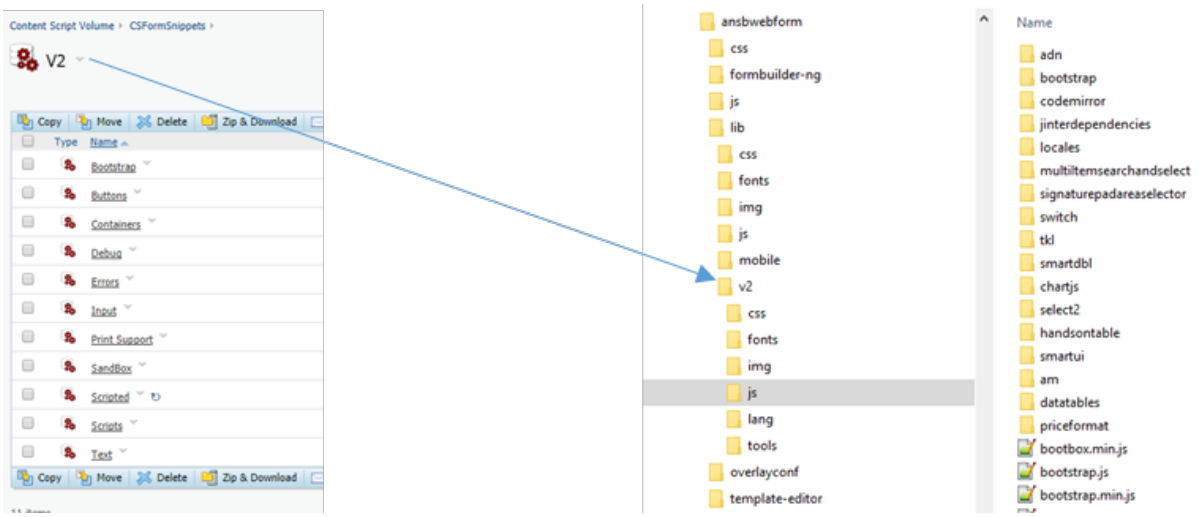
> Please note that this is a very low-level functionality and such might have a significant impact on users' experience ***use it with caution***. The feature requires a restart every time the configuration is changed.

Copyright © 2013-2022 AnswerModules Sagl

Beautiful WebForms¶

The most relevant aspects of the module's internal structure for the Beautiful WebForms module are related with the "**support**" directory. Beautiful WebForms default View Templates make use of several JavaScript libraries: they have been selected, written and optimized to work together with View Templates.

In particular, the Beautiful WebForms' unique validation framework makes use of the libraries stored under the "**js**" directory. The recommended way to load these libraries is to make use of the Velocity macros expressly designed to load them



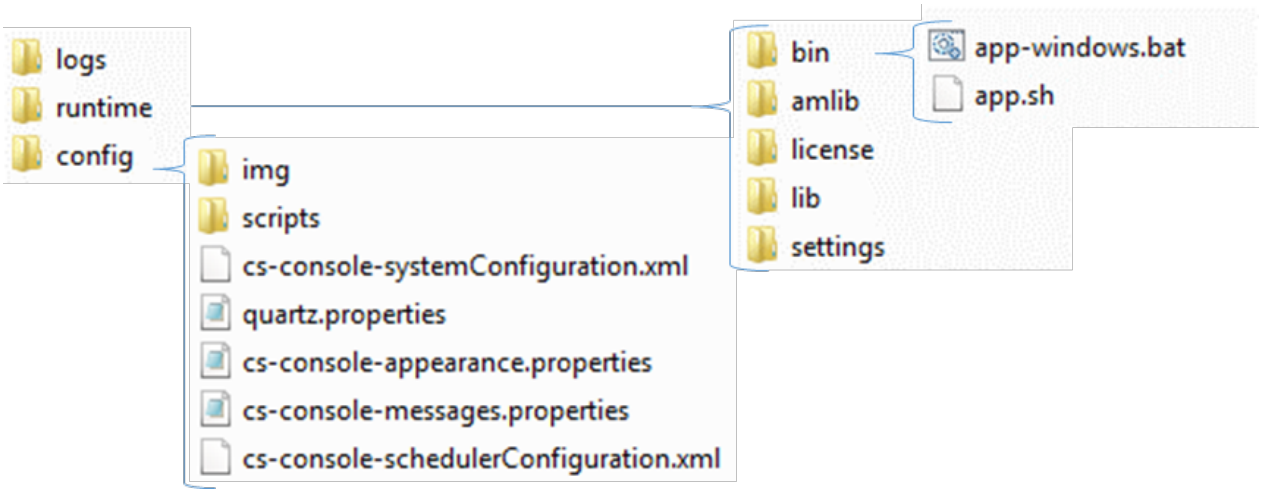
> Starting with version 2.0 the module's static resources have been deeply revised and re-organized. They are now structured in a way that reflects the way Beautiful WebForms' widgets are organized in the Content Script volume. Beautiful WebForms' widget are in fact now organized into libraries.

Script console¶

The Script Console internal structure reflects its ability to connect to multiple Content Server Instances and to organize Content Script scripts in multiple repositories.

> Since version 1.7.0, the Script Console runtime and configuration folders are all stored under the same installation path. The Script Console installation folder will appear as shown in figure here below:

Copyright © 2013-2022 AnswerModules Sagl



Script Console main configuration file¶

The Script Console main configuration file (**cs-console-systemConfiguration.xml**) is stored under the **config** directory. As the naming of the file tells us, it is an XML based configuration file, intended to include general configuration parameters of the Script Console as well as specific settings related to the Content Server system to which the Script Console can be connected.

The configuration file is automatically modified by specific actions performed on/through the Console (such as registering a new target Content Server system) or can be edited manually by the administrators.

Copyright © 2013-2022 AnswerModules Sagl

Installation and Upgrade

Prerequisites

Prerequisites¶

The following section describes the step-by-step procedure that will lead to the installation of the Module Suite on a single-machine Windows based Content Server environment.

The following guide assumes that the following are available:

- The required Module Suite installers for Windows

Before proceeding with the installation, make sure that the installer version matches the OpenText Content Server target system version.

E.g.:

- ***module-suite-2.7.0-OTCS162.exe*** is the Windows installer for OpenText Content Server 16.2.X;
- ***module-suite-2.6.0-OTCS162.exe*** is the Windows installer for OpenText Content Server 16.2.X;
- ***module-suite-2.5.0-OTCS162.exe*** is the Windows installer for OpenText Content Server 16.2.X;
- ***module-suite-2.4.0-OTCS16.exe*** is the Windows installer for OpenText Content Server 16.0.X;
- ***module-amcontentscript-2.3.0-OTCS105.exe*** is the Windows installer for OpenText Content Server 10.5.X;
- ***module-amcontentscript-2.2.0-OTCS10.exe*** is the Windows installer for OpenText Content Server 10.0.X;

Tip

Hotfixes and patches are continuously published on the AnswerModules Support Portal. Check the availability of applicable patches when starting a new installation.

E.g. https://support.answermodules.com/portal/kb/articles/2-2-0-patches-and-hotfixes-for-content-server-16 *(https://support.answermodules.com/portal/kb/articles/2-2-0-patches-and-hotfixes-for-content-server-16)*

- A valid AnswerModules license key

An activation key is only required starting from version 1.7.0 of the Module Suite.

Important

Starting from version 2.0.0 activation keys are bound to system's fingerprint.

Copyright © 2013-2022 AnswerModules Sagl

Installing the Suite

Installation procedure¶

We will refer to the Content Server installation directory as **%OTCS_HOME%**

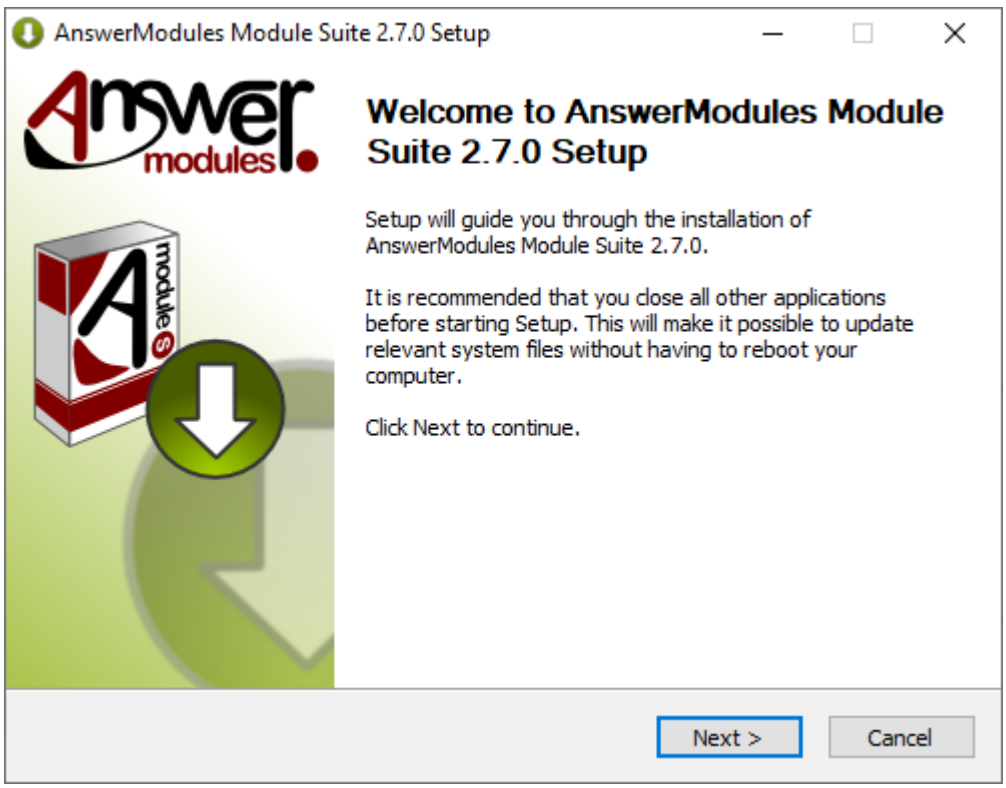
Deploy Content Server Modules¶

- Stop the Content Server
- Run the **Module Suite Master Installer** and unpack: Content Script, Beautiful WebForms, Smart Pages modules and all the desired Module Suite Extension packages.

Step-by-step procedure

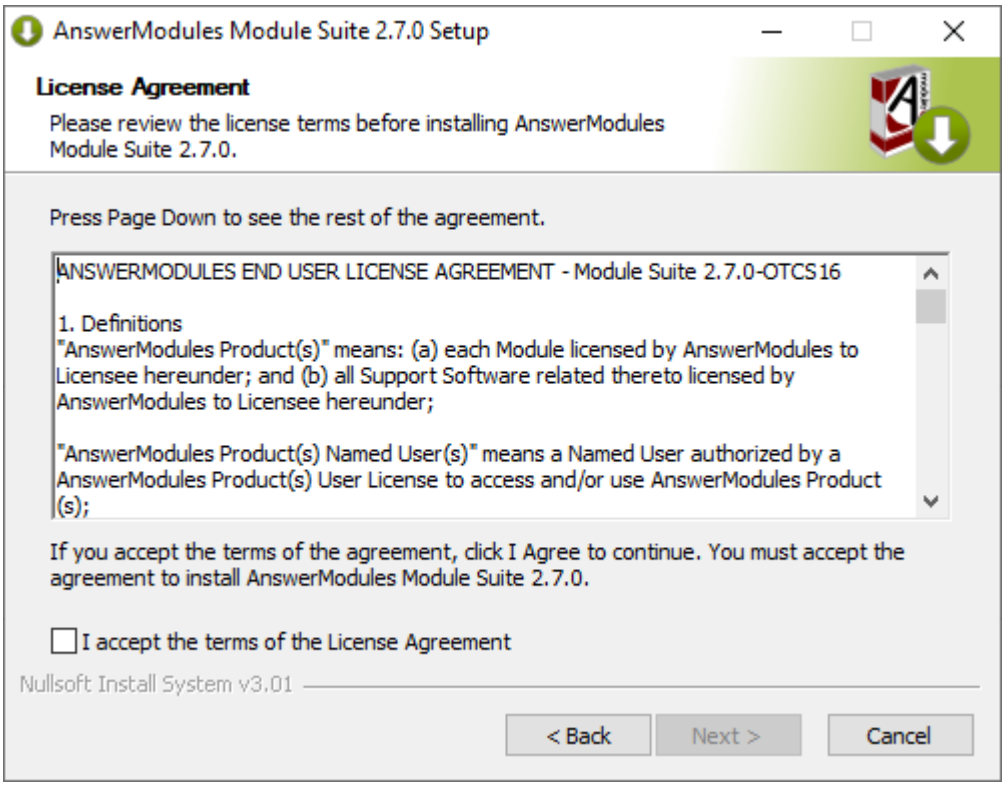
The following screens will guide you through the deployment of Module Suite modules.

1. Welcome Screen: Select “Next” when ready to start the installation.

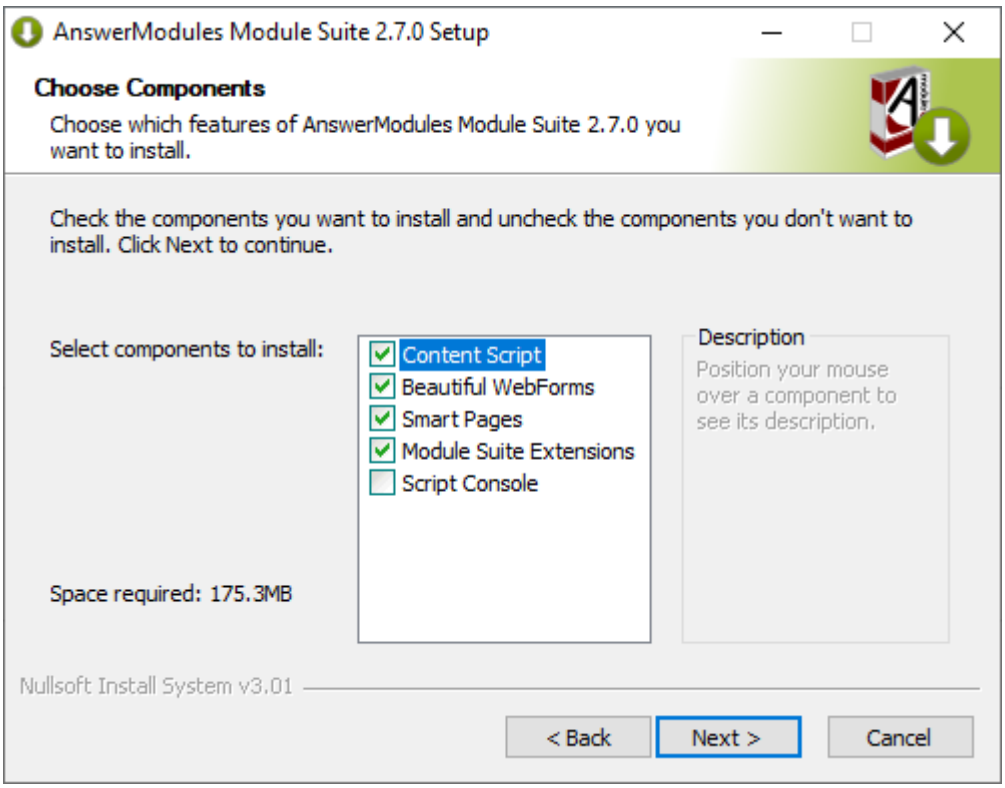


2. EULA Screen: Acceptance of the end-user license agreement is mandatory for proceeding with the installation
   A copy of the agreement will be available, upon installation, in:
   %OTCS_HOME%/module/amcontentscript_X_Y_Z/license/EULA
   Select “Next” when ready.

Copyright © 2013-2022 AnswerModules Sagl

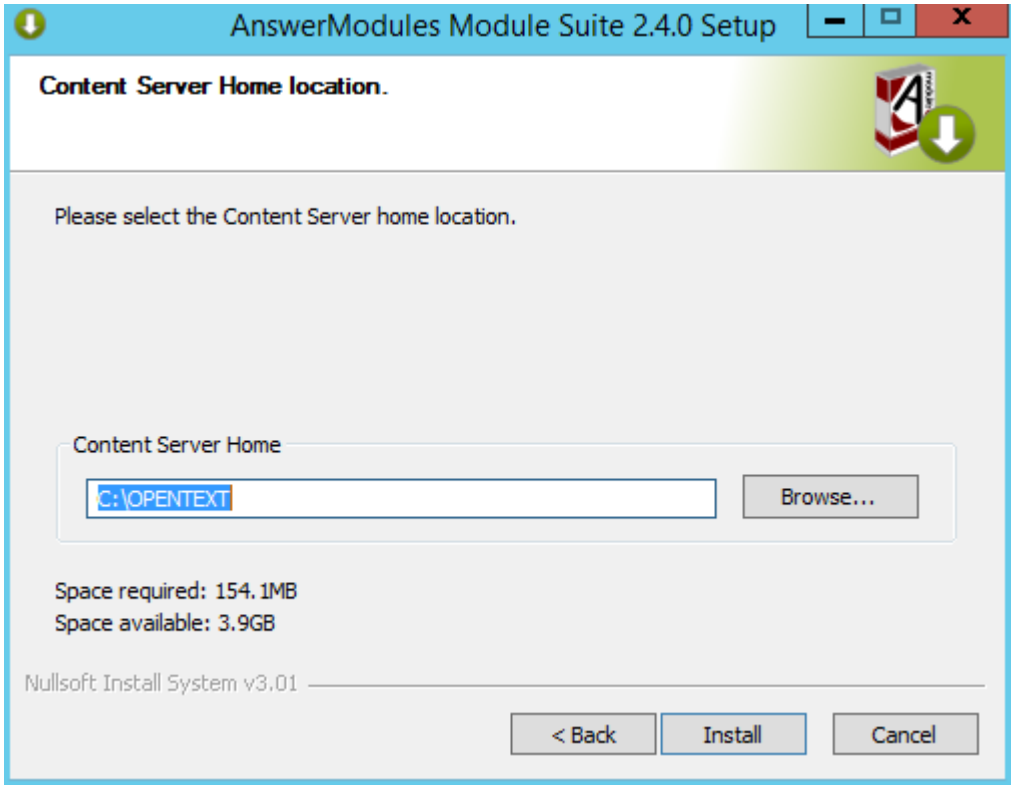


3. Components selection: Script console is unselected by default because it is not a Content Server module. A standard Module Suite installation does not require this component to be installed.
Select “Next” when ready.

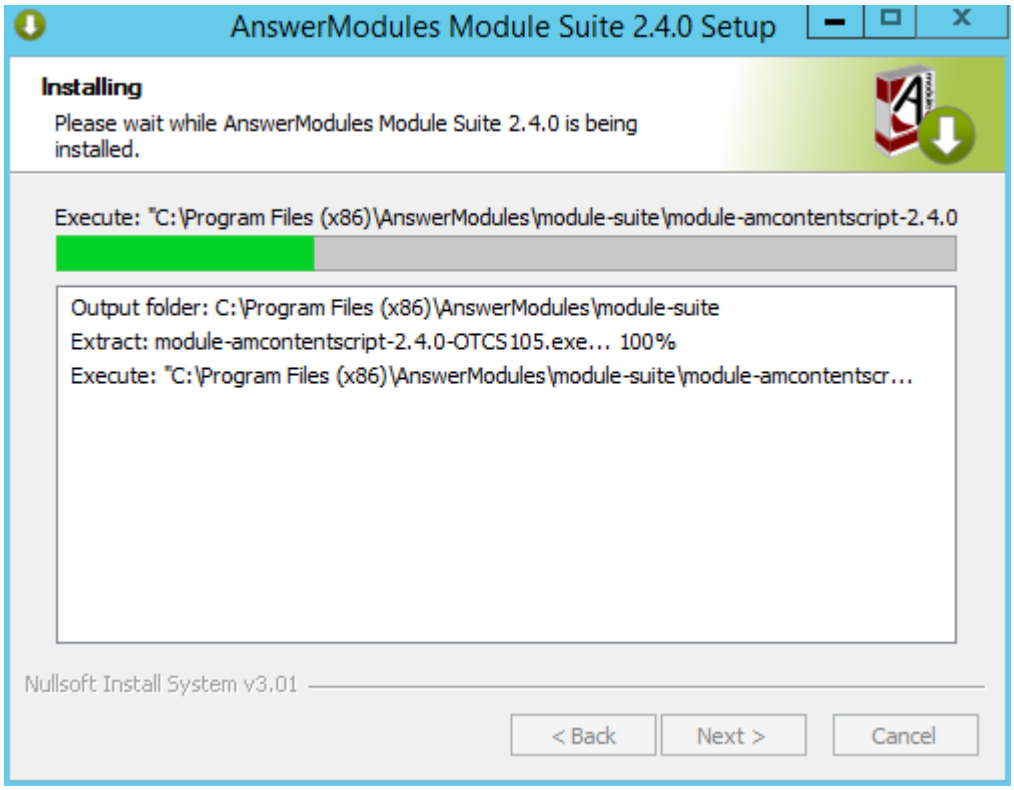


Copyright © 2013-2022 AnswerModules Sagl

4. OTHOME selection: The installer will prompt you for the location where Content Server is installed. Browse to your **OTCS_HOME** and select “Next” when ready to start the installation.

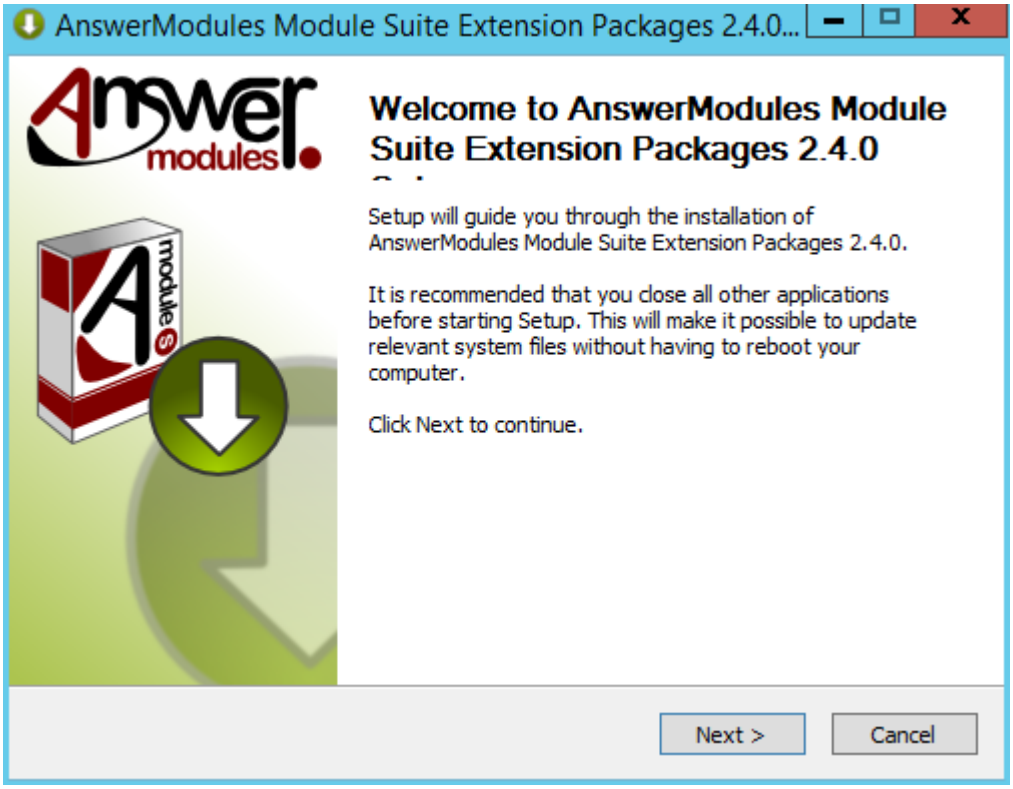


5. Automatic import of Content Server dependencies: The installer will automatically attempt to load a few libraries from Content Server.
   In case of failure, a warning message could appear during this phase of the installation. In such case, the operation must be performed manually

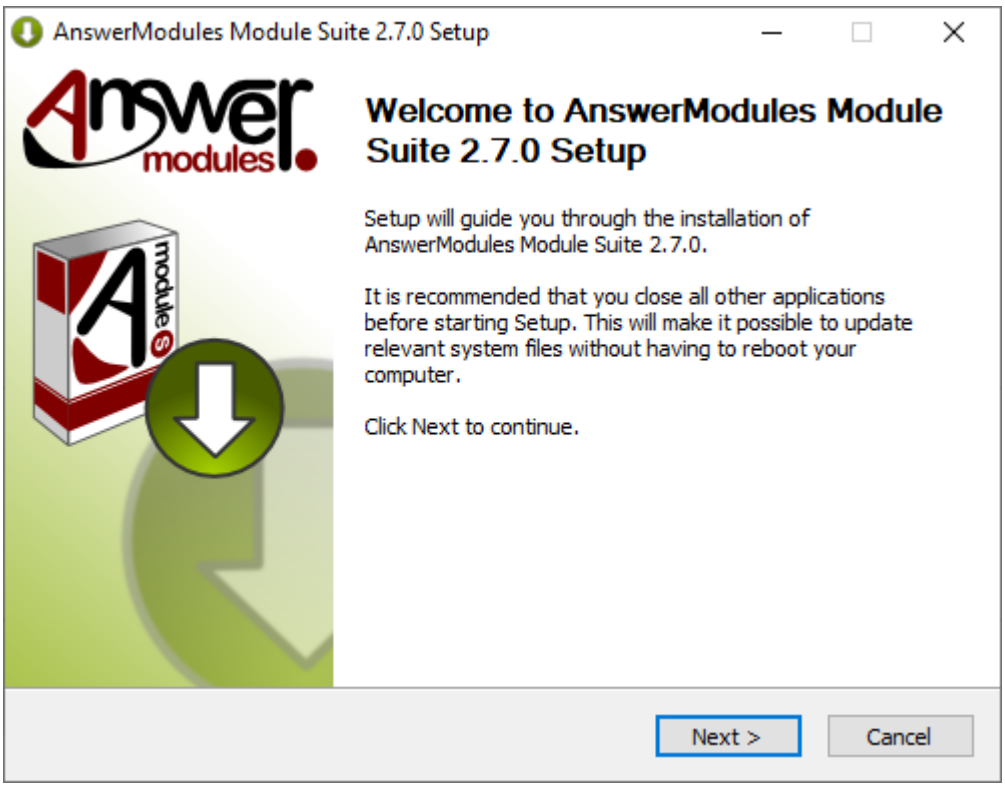


Copyright © 2013-2022 AnswerModules Sagl

6. Module Suite Extension Packages: Select “Next” when ready.

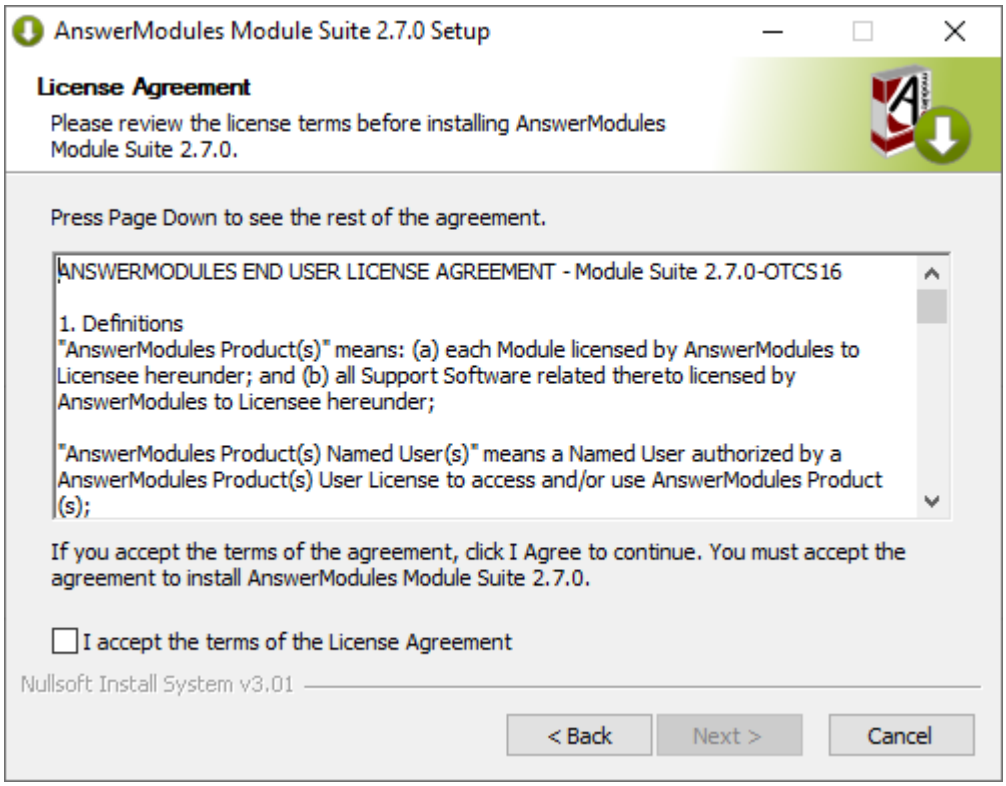


7. Welcome Screen: Select “Next” when ready to start the installation.



8. EULA Screen: Acceptance of the end-user license agreement is mandatory to proceed with the installation
   A copy of the agreement will be available, upon installation, in:
   %OTCS_HOME%/module/amcontentscript_X_Y_Z/license/EULA Accepting the End User Agreement is mandatory to proceed with the installation.
   Select “Next” when ready.

Copyright © 2013-2022 AnswerModules Sagl

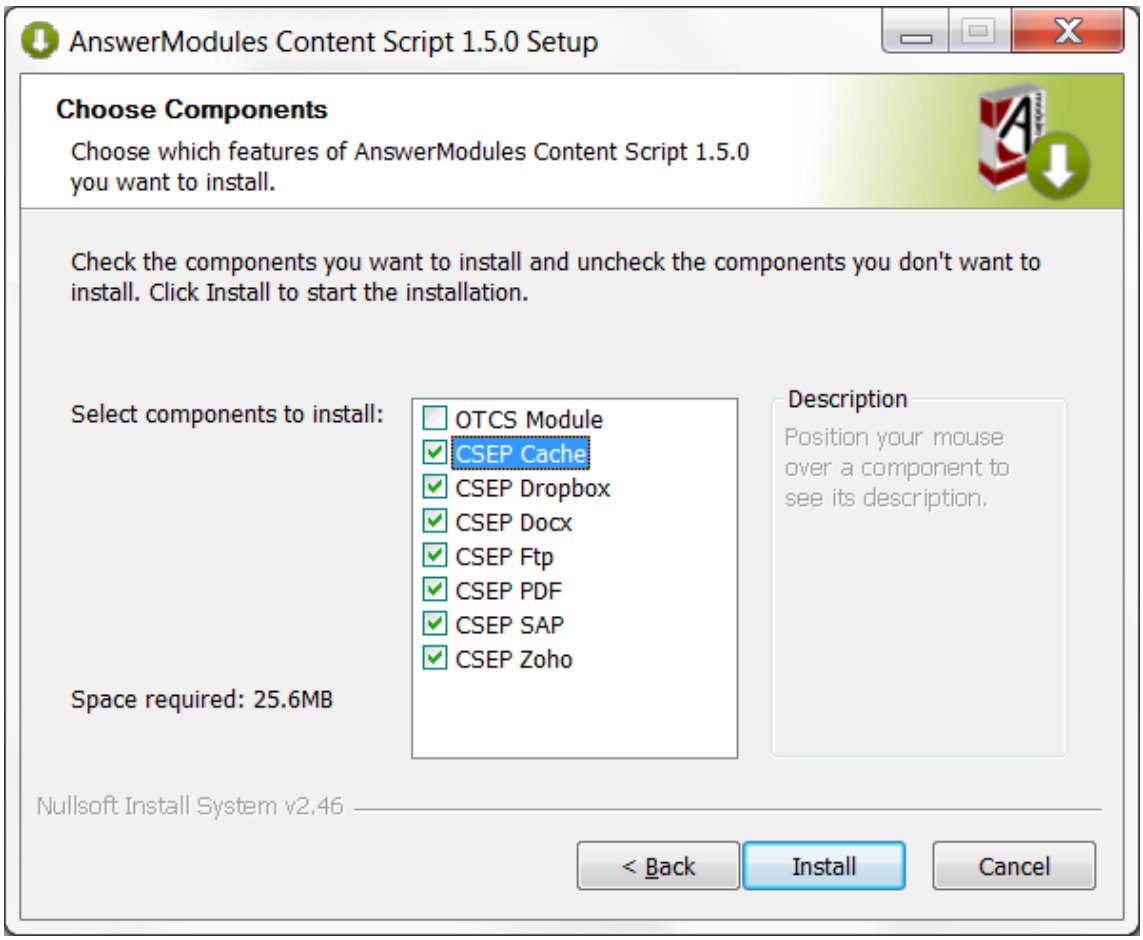


9. Components selection:Unselect the *OTCS Module* component. Select all of the extension components that are to be installed
   Select “Install” when ready.

CSEP SAP* - DO NOT INSTALL THEM UNLESS YOU WANT TO CONFIGURE THEM

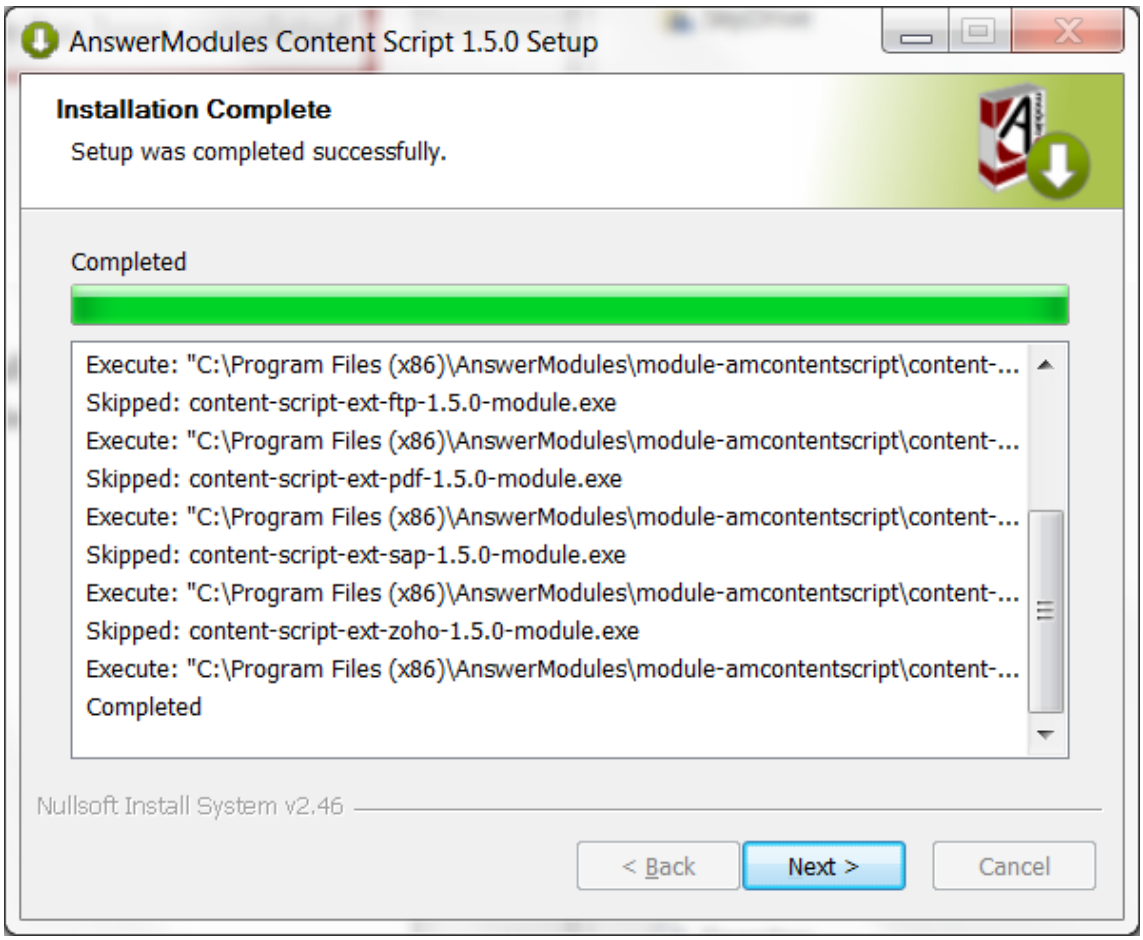
This extension package requires the SAP™ JCo library *(https://support.sap.com/en/product/connectors/JCo.html)* to be available in the extension repository `<OTHOME>/module/anscontentscript_x_y_z/amlib/sap` and is certified for use with SAP™ JCo version (3.0.6) when used on OpenText Extended ECM and version (3.0.10) when used on CSP. SAP™ JCo library *(https://support.sap.com/en/product/connectors/JCo.html)* can be downloaded from SAP™ website. More on this extensions *(/installation/extpacks/#content-script-extension-for-sap)*.

Copyright © 2013-2022 AnswerModules Sagl

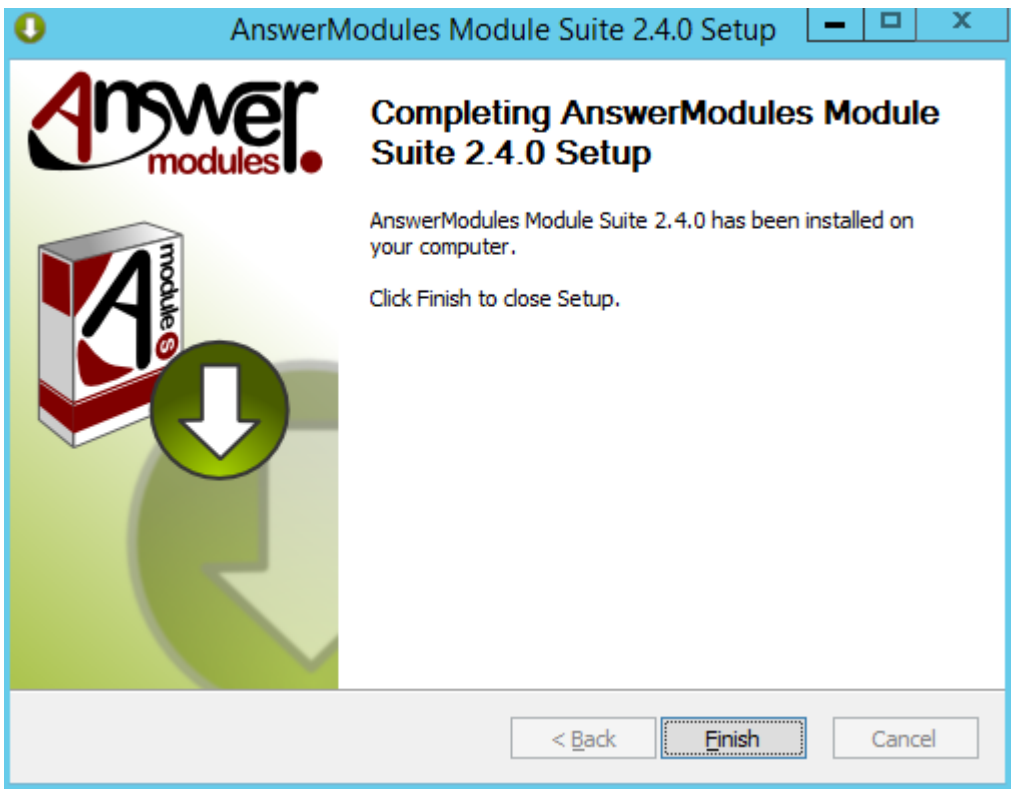


10. Installation: The extension packages are automatically installed.
Select “Next” when the procedure is complete.

Copyright © 2013-2022 AnswerModules Sagl



11. Installation completed: Select “Finish” and return to the installation checklist to finalize the module setup.



Copyright © 2013-2022 AnswerModules Sagl

What to do if the installer raises the error: Unable to automatically extract...

Some Content Script extension packages require two Java libraries that are specific to the target Content Server environment and are not distributed with the module.
The required library files are:

- csapi.jar
- service-api-X.X.XX.jar

and can be found in the web app located in:

- %OTCS_HOME%\webservices\java\webapps\cws.war (on CS 16.X)
- %OTCS_HOME%\webservices\java\webapps\cws.war (on CS 10.5.X)

To retrieve the files:

- copy the file named XXX.war to a temporary folder
- rename the file XXX.war in XXX.
- extract the zip archive contents locate the files in the WEB-INF/lib folder

Once the files have been located, copy them to the folder:

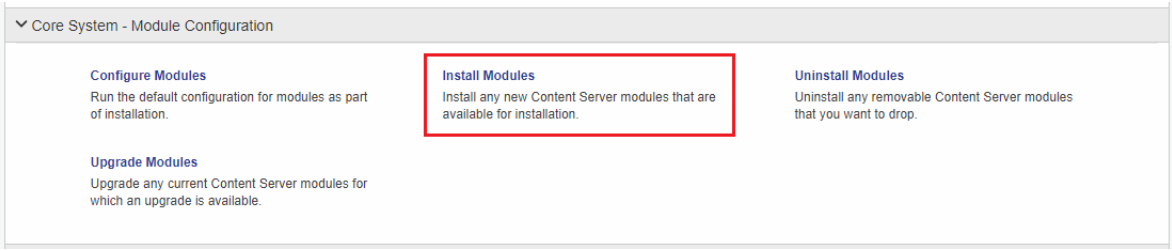
%OTCS_HOME\staging\anscontentscript_x_y_z\amlib

Staging

At this point, the Modules have been deployed in the Content Server Staging folder and is available for installing it through the Content Server administration pages.

Install Content Server Modules¶

- Start Content Server
- Login as Administrator and access the Module administration panel
- Access the Content Server Admin pages > **Core System - Module Configuration** > **Install Modules**



- From the available modules, select **“AnswerModules Content Script X.Y.Z”**
- Follow the installation steps and restart Content Server when prompted.
- From the Administration Home, access the Module administration panel
- Select **“Install Modules”**

Copyright © 2013-2022 AnswerModules Sagl

- From the available modules, select **“Answer Modules - Beautiful Web Forms X.Y.Z”**
- Follow the installation steps and restart Content Server when prompted.
- From the Administration Home, access the Module administration panel
- Select **“Install Modules”**
- From the available modules, select **“Answer Modules - Smart Pages X.Y.Z”**
- Follow the installation steps and restart Content Server when prompted.

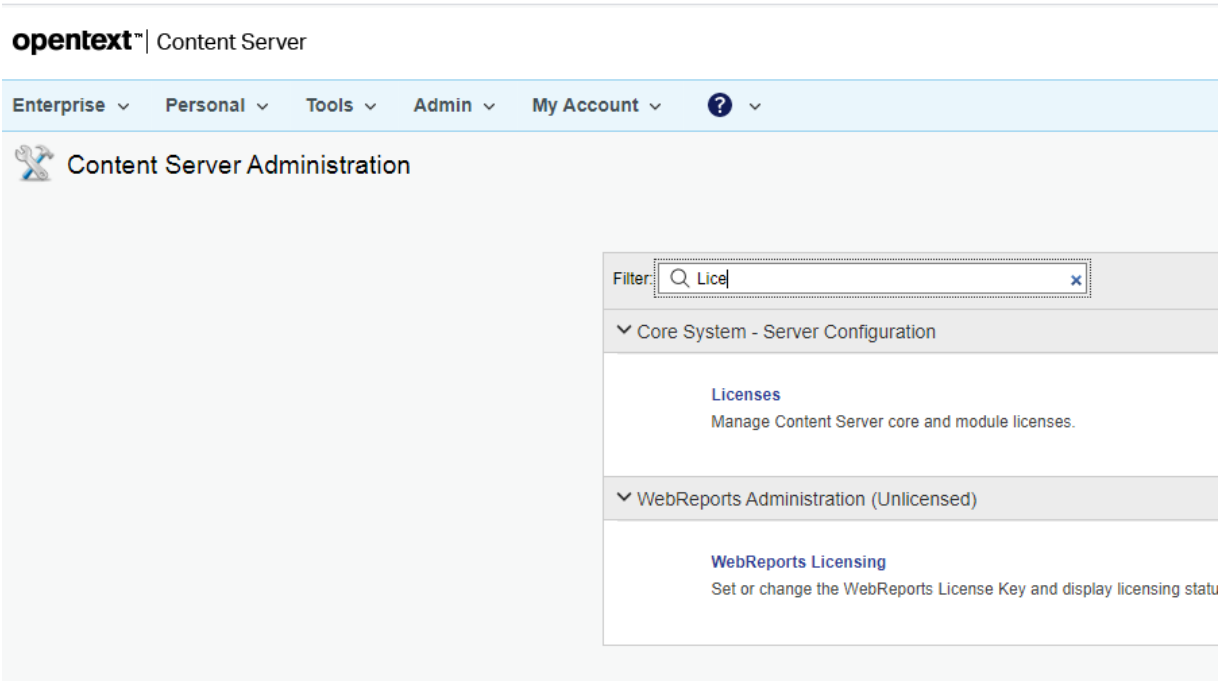
Activate and Configure¶

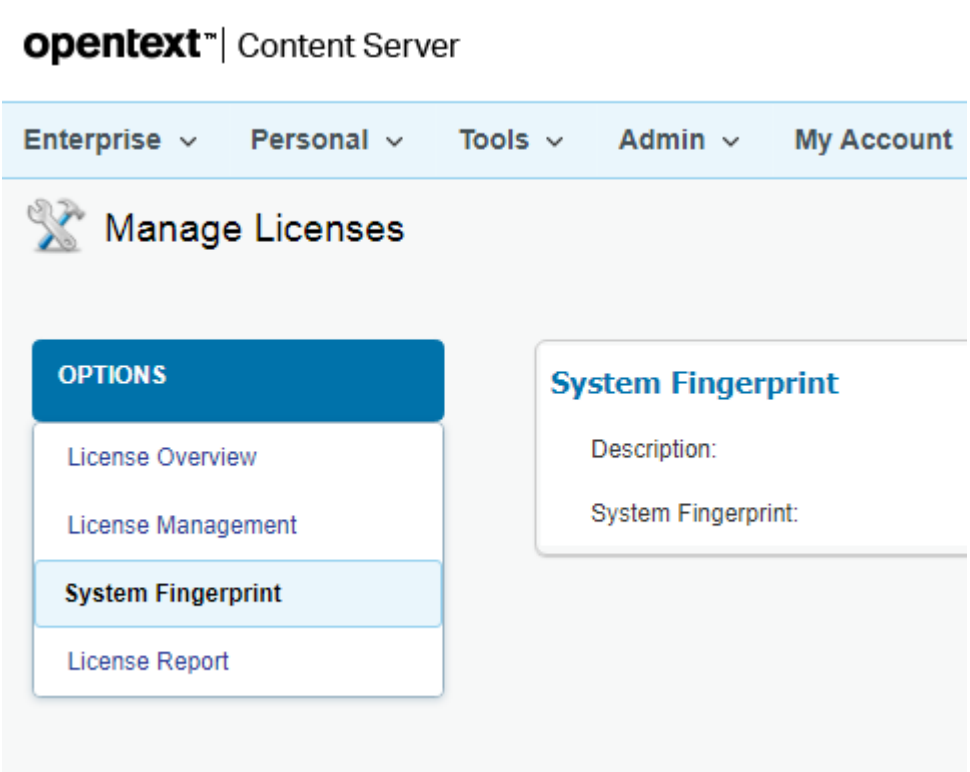
From the Administration Home, select **AnswerModules Administration > Base Configuration**, then enter the activation License in order to activate the product. License shall be entered in the **Module Suite - Activation Key** property

How do I get an activation key?

In order to activate Module Suite you need a valid activation key. Activation keys can be requested to AnswerModules Support *(https://support.answermodules.com)* by providing the OpenText Content Server System Fingerprint. You can read your's environment fingerprint from the OpenText Admin Pages as shown below

Copyright © 2013-2022 AnswerModules Sagl





Since version 3.1.0, if the installation is performed on a multi-server architecture is no longer necessary to repeat the activation process on all the node of the cluster since the License Key information is stored in the Content Server's database.

- [ ] Save the Base Configuration and restart Content Server
- [ ] Import the Module Suite components and widgets library

Installation complete

Copyright © 2013-2022 AnswerModules Sagl

The Module Suite's initial setup is complete

Installing the Suite on Unix

Installation procedure¶

We will refer to the Content Server installation directory as **%OTCS_HOME%**

Unix Consultant

This manual presumes that the user executing the intallation process has a good knowledge of a Unix System and its commands

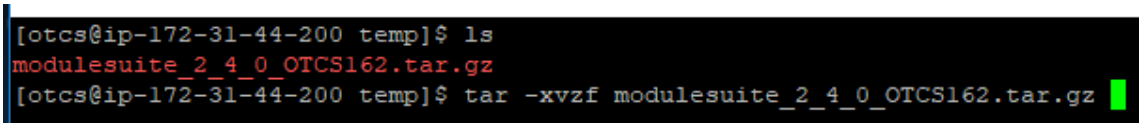
Deploy Content Server Modules¶

- Stop the Content Server
- Run the **Module Suite Master Installer** and unpack: Content Script module, Beautiful WebForms, Smart Pages modules and all the desired Module Suite Extension packages

Step-by-step procedure

The following screens will guide you through the deployment of Module Suite modules.

1. Staging folder: extract ModuleSuite compressed archive file into a temporary location



2. EULA : Acceptance of the end-user license agreement is mandatory for proceeding with the installation
   A copy of the agreement will be available, upon installation, in:
   %OTCS_HOME%/module/amcontentscript_X_Y_Z/license/EULA Accepting the End User Agreement is mandatory to proceed with the installation.
   Enter “Y” when ready.

Copyright © 2013-2022 AnswerModules Sagl

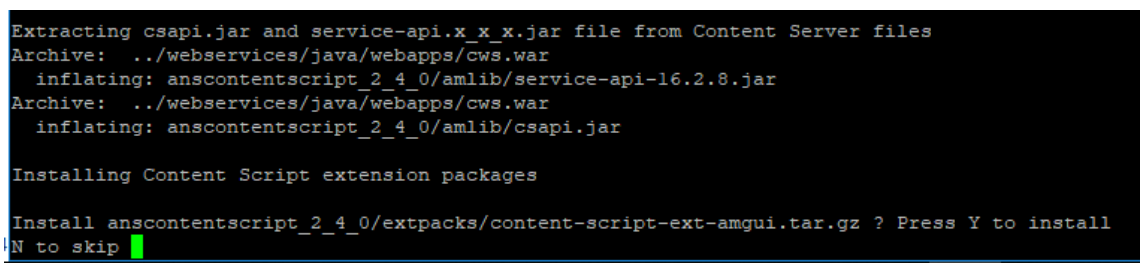
3. OTHOME selection: The installer will prompt you for the location where Content Server is installed. Either confirm (ENTER) the default location or enter the correct location to proceed with the installation.

4. Automatic import of Content Server dependencies: The installer will automatically attempt to load a few libraries from Content Server.
   In case of failure, a warning message could appear during this phase of the installation. In such case, the operation must be performed manually

5. Module Suite Extension Packages: Enter “Y” to install the extension when prompted.

CSEP SAP* - DO NOT INSTALL THEM UNLESS YOU WANT TO CONFIGURE THEM

This extension package requires the SAP™ JCo library *(https://support.sap.com/en/product/connectors/JCo.html)* to be available in the extension repository `<OTHOME>/module/anscontentscript_x_y_z/amlib/sap` and is certified for use with SAP™ JCo version (3.0.6) when used on OpenText Extended ECM and version (3.0.10) when used on CSP. SAP™ JCo library *(https://support.sap.com/en/product/connectors/JCo.html)* can be downloaded from SAP™ website. More on this extensions *(/installation/extpacks/#content-script-extension-for-sap)*.



Copyright © 2013-2022 AnswerModules Sagl

6. Installation completed: return to the installation checklist to finalize the module setup.

What to do if the installer raises the error: Unable to automatically extract...

Some Content Script extension packages require two Java libraries that are specific to the target Content Server environment and are not distributed with the module.
The required library files are:

- csapi.jar
- service-api-X.X.XX.jar

and can be found in the web app located in:

```
%OTCS\_HOME%\\webservices\\java\\webapps\\cws.war
```

- classificationsservice-api-X.X.XX.jar

which can be found in the web app located in:

```
%OTCS\_HOME%\\webservices\\java\\webapps\\cs-services-classifications.war
```

- physicalobjectsservice-api-X.X.XX.jar

which can be found in the web app located in:

```
%OTCS\_HOME%\\webservices\\java\\webapps\\cs-services-physicalobjects.war
```

- recordsmanagementservice-api-X.X.XX.jar

which can be found in the web app located in:

```
%OTCS\_HOME%\\webservices\\java\\webapps\\cs-services-recordsmanagement.war
```

To retrieve the files:

- copy the file named XXX.war to a temporary folder
- rename the file XXX.war in XXX.
- extract the zip archive contents locate the files in the WEB-INF/lib folder

Once the files have been located, copy them to the folder:

%OTCS_HOME\staging\anscontentscript_x_y_z\amlib

Install Content Server Modules¶

- Start Content Server
- Login as Administrator and access the Module administration panel
- Select **“Install Modules”**
- From the available modules, select **“AnswerModules Content Script 2.7.0”**
- Follow the installation steps and restart Content Server when prompted.
- From the Administration Home, access the Module administration panel

Copyright © 2013-2022 AnswerModules Sagl

- Select **“Install Modules”**
- From the available modules, select **“Answer Modules - Beautiful Web Forms 2.7.0”**
- Follow the installation steps and restart Content Server when prompted.
- From the Administration Home, access the Module administration panel
- Select **“Install Modules”**
- From the available modules, select **“Answer Modules - Smart Pages 2.7.0”**
- Follow the installation steps and restart Content Server when prompted.

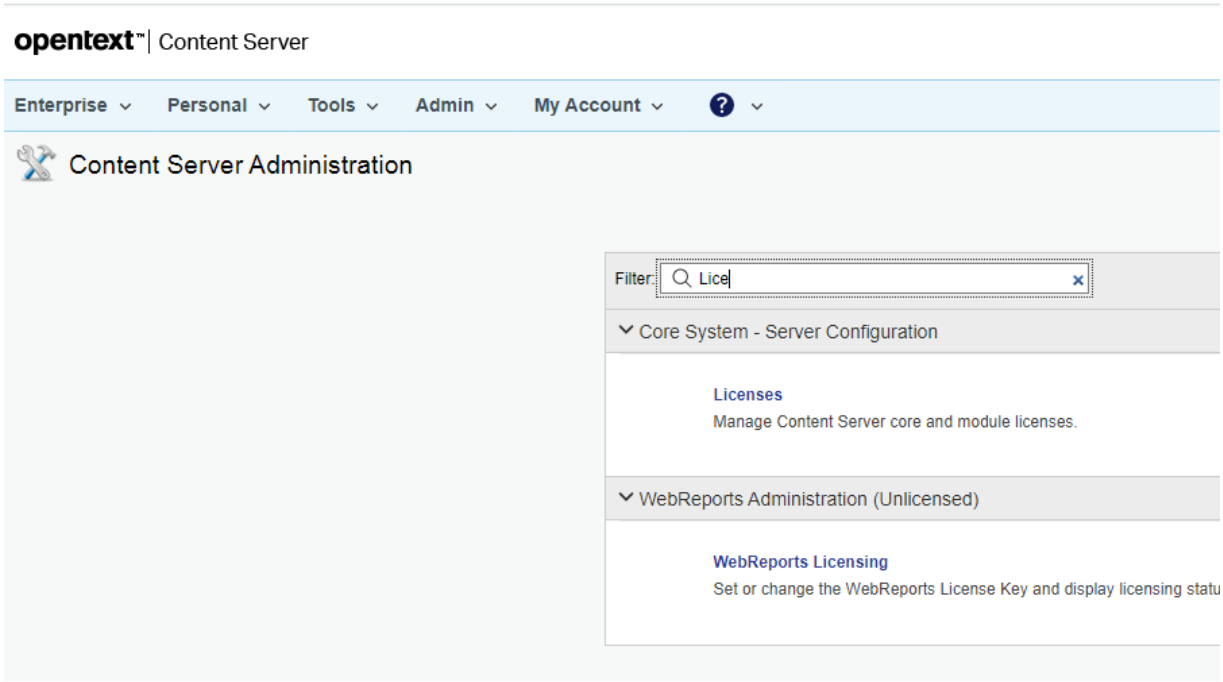
Activate and Configure¶

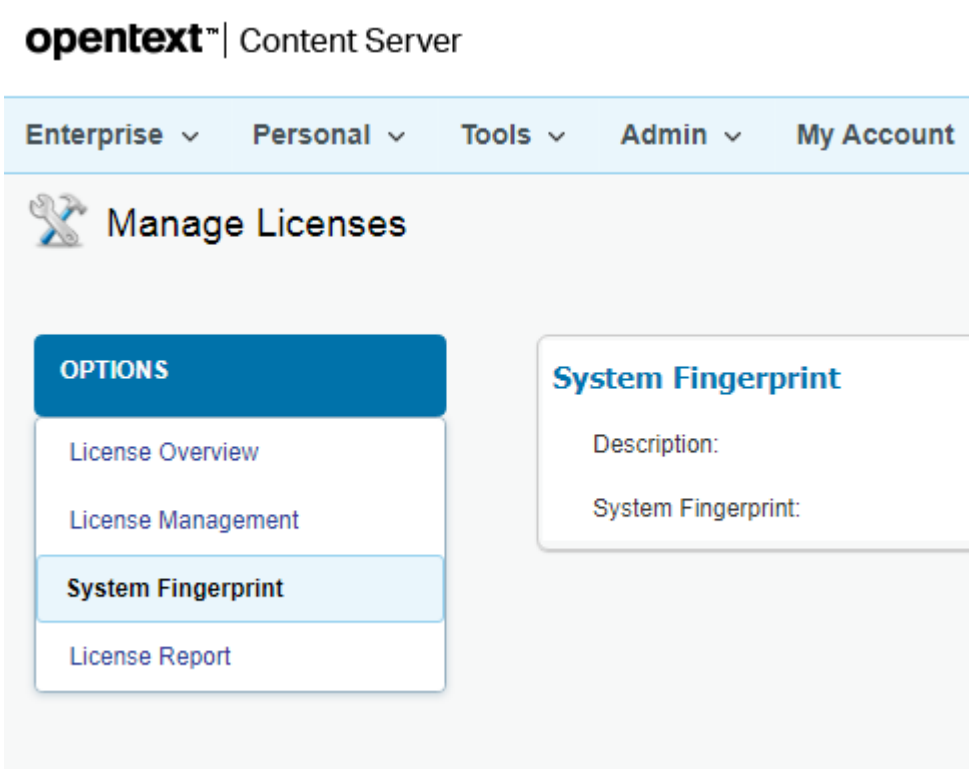
From the Administration Home, select **AnswerModules Administration > Base Configuration**, then enter the activation License in order to activate the product. License shall be entered in the **Module Suite - Activation Key** property

How do I get an activation key?

In order to activate Module Suite you need a valid activation key. Activation keys can be requested to AnswerModules Support *(https://support.answermodules.com)* by providing the OpenText Content Server System Fingerprint. You can read your's environment fingerprint from the OpenText Admin Pages as shown below

Copyright © 2013-2022 AnswerModules Sagl





Since version 3.1.0, if the installation is performed on a multi-server architecture is no longer necessary to repeat the activation process on all the node of the cluster since the License Key information is stored in the Content Server's database.

In order to do so, there are two alternative options:

- Edit the .ini file manually on each machine, adding the licensing information.
- From the Administrative pages of each machine, perform the “save Base Configuration” operation.

☐ Save the Base Configuration and restart Content Server

Copyright © 2013-2022 AnswerModules Sagl

- [ ] Import the Module Suite components and widgets library

Installation complete

The Module Suite's initial setup is complete

Installing Content Script

Installation procedure (Windows)¶

We will refer to the Content Server installation directory as **%OTCS_HOME%**

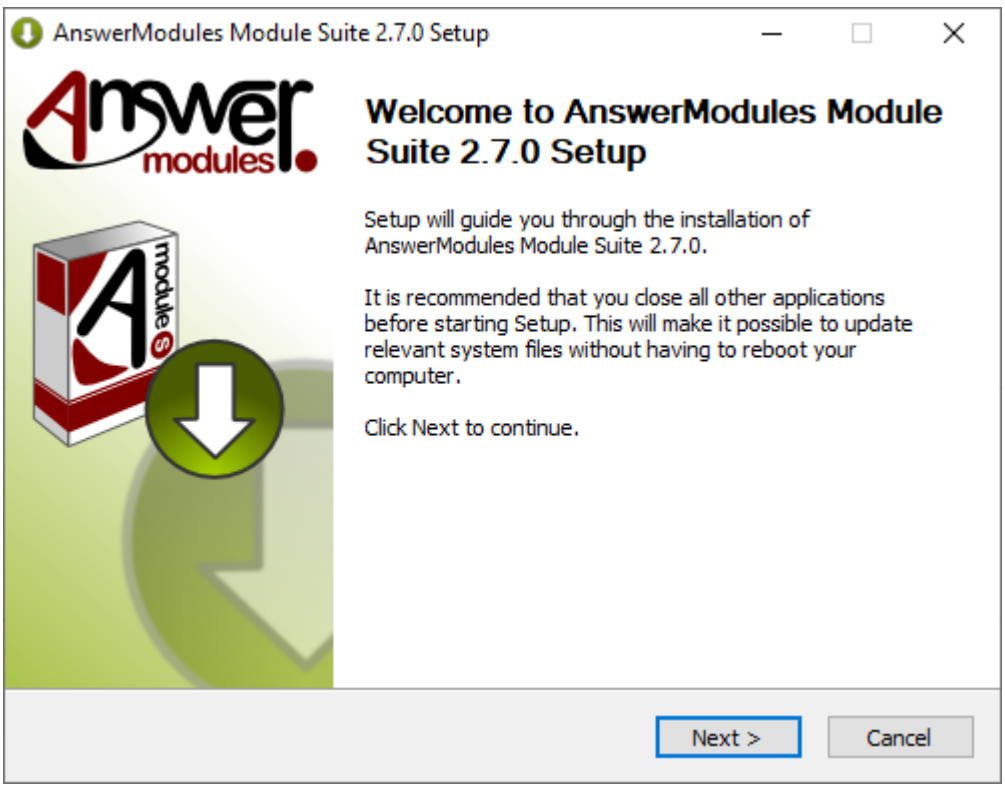
- [ ] Deploy Content Server Modules

  - Stop the Content Server
  - Run the **Module Suite Master Installer** and install the Content Script module and the desired Module Suite Extension packages

Step-by-step procedure

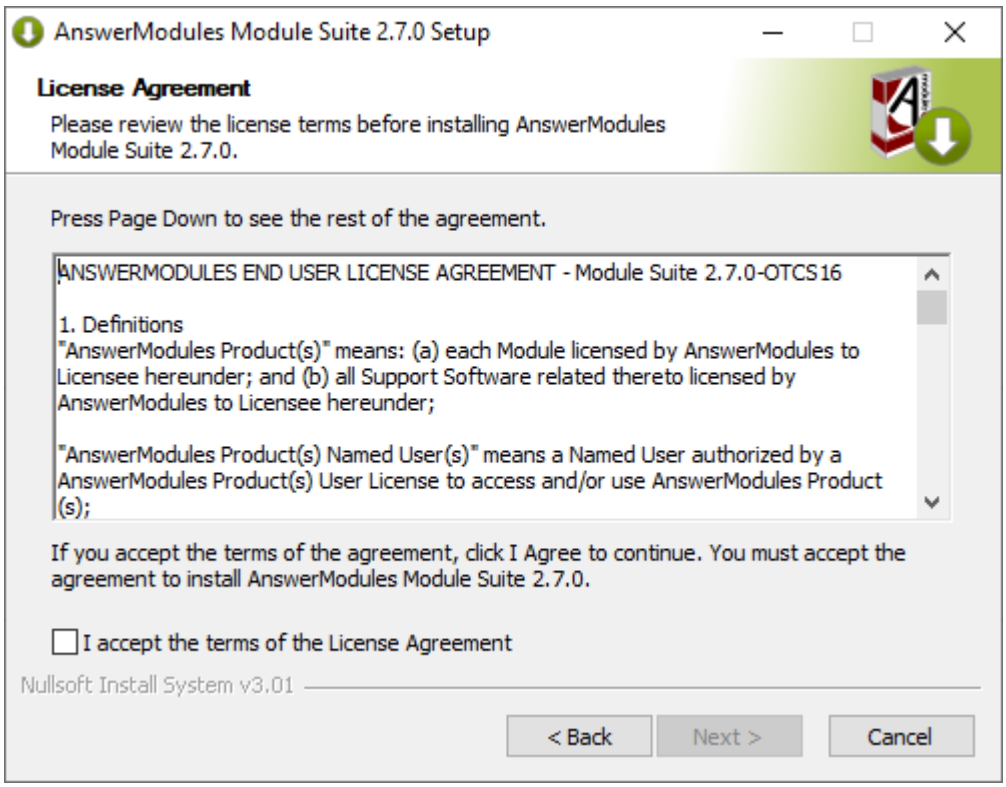
The following screens will guide you through the deployment of Module Suite modules.

1. Welcome Screen: Select “Next” when ready to start the installation.



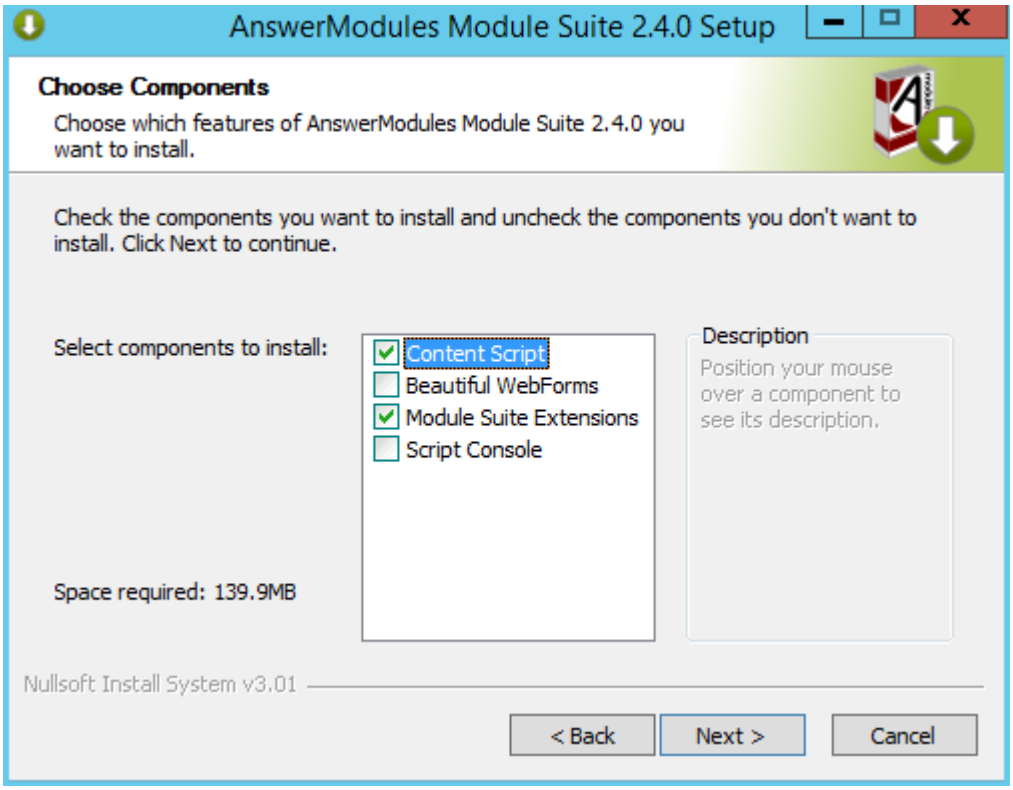
Copyright © 2013-2022 AnswerModules Sagl

2. EULA Screen: Acceptance of the end-user license agreement is mandatory for proceeding with the installation
   A copy of the agreement will be available, upon installation, in:
   %OTCS_HOME%/module/amcontentscript_X_Y_Z/license/EULA
   Select “Next” when ready.

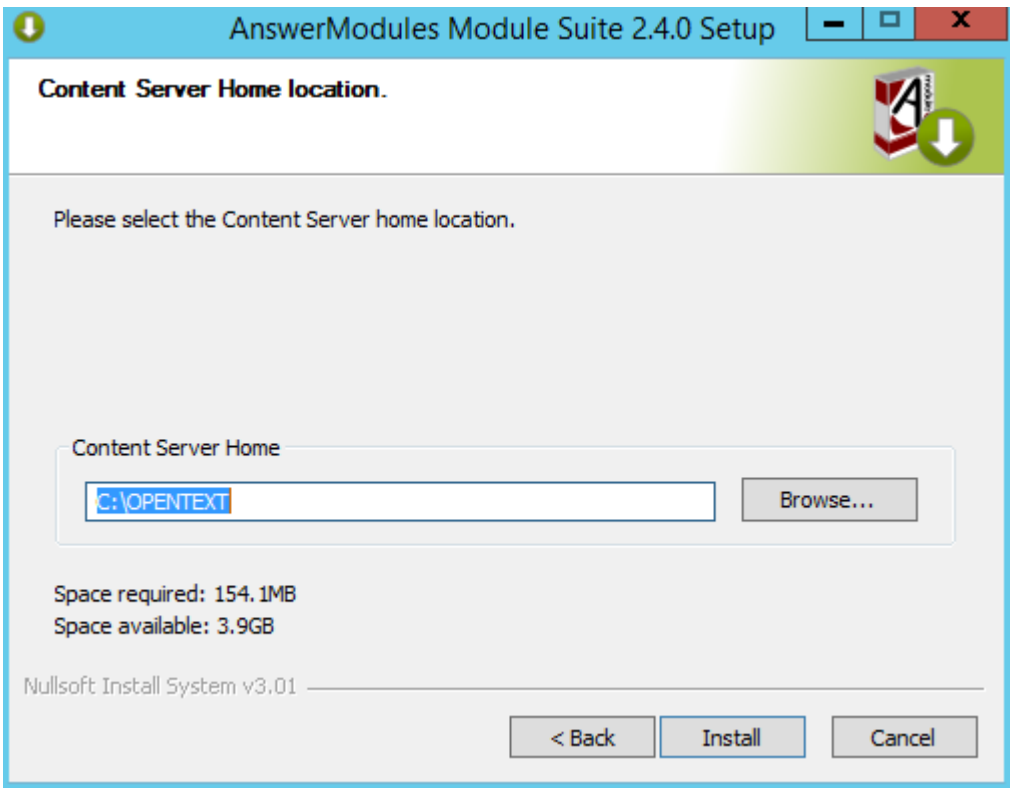


3. Components selection: Script console is unselected by default because it is not a Content Server module. A standard Module Suite installation does not require this component to be installed.
   Select “Next” when ready.

Copyright © 2013-2022 AnswerModules Sagl



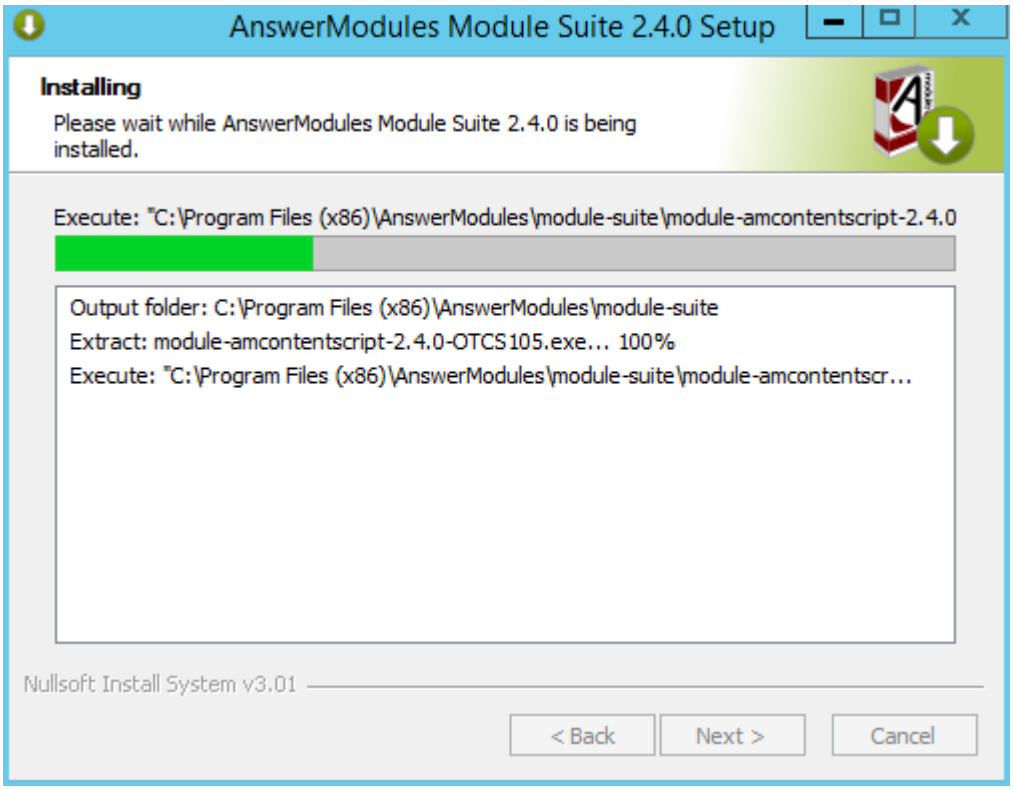
4. OTHOME selection: The installer will prompt you for the location where Content Server is installed. Browse to your **OTCS_HOME** and select "Next" when ready to start the installation.



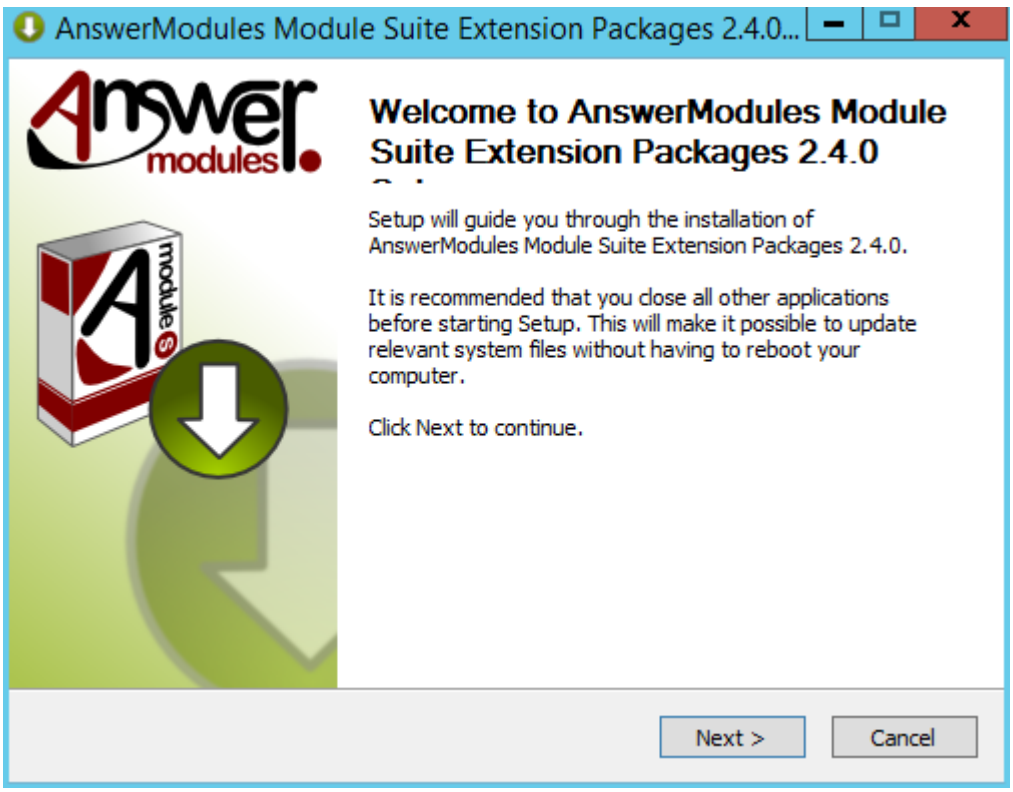
5. Automatic import of Content Server dependencies: The installer will automatically attempt to load a few libraries from Content Server.
   In case of failure, a warning message could appear during this phase of the installation. In such case, the

Copyright © 2013-2022 AnswerModules Sagl

operation must be performed manually



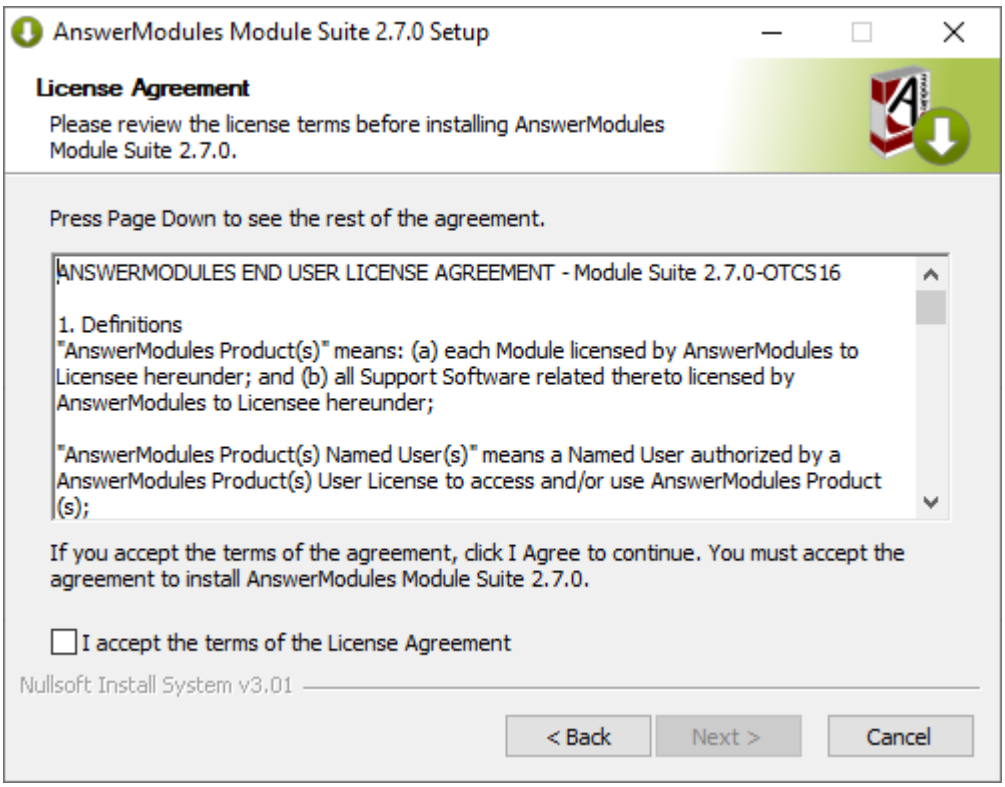
6. Module Suite Extension Packages: Select “Next” when ready.



Copyright © 2013-2022 AnswerModules Sagl

7. Welcome Screen: Select “Next” when ready to start the installation.

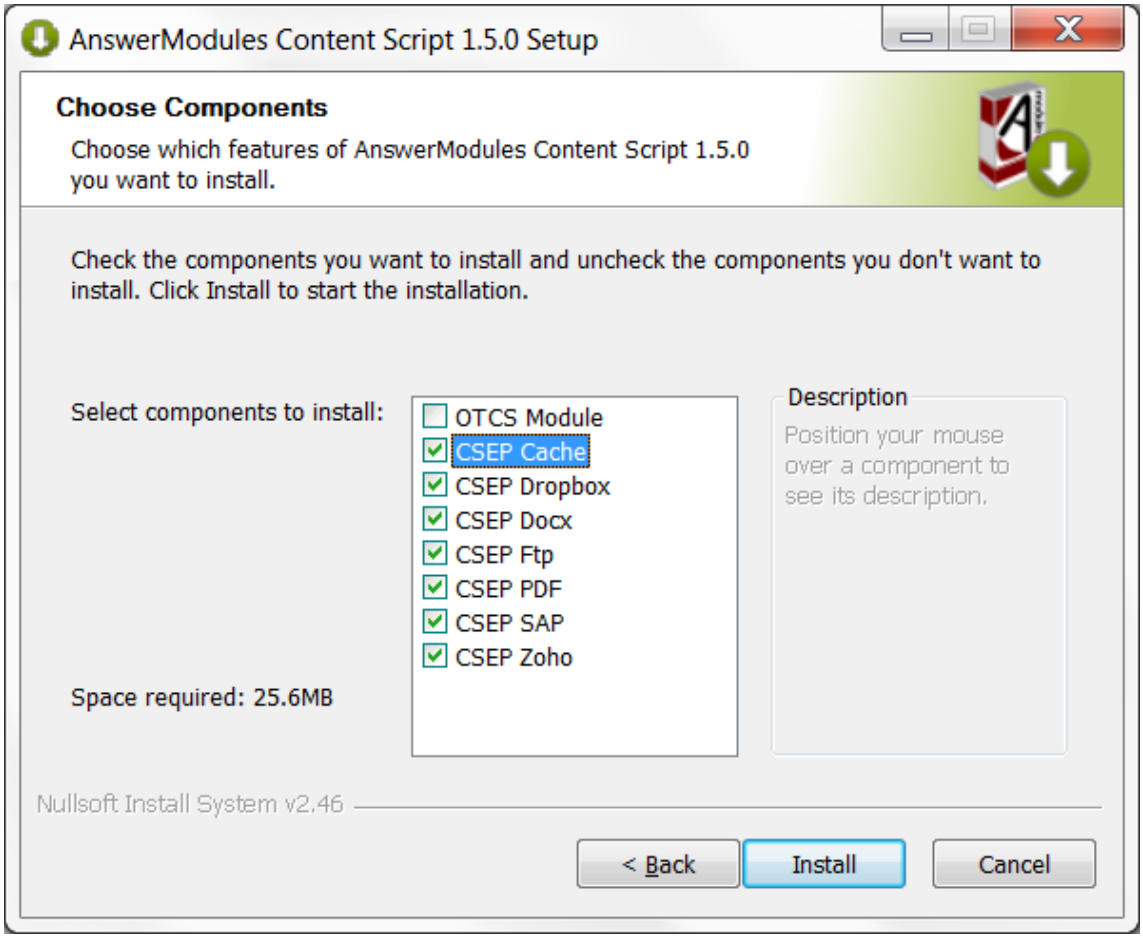
8. EULA Screen: Acceptance of the end-user license agreement is mandatory to proceed with the installation
   A copy of the agreement will be available, upon installation, in:
   %OTCS_HOME%/module/amcontentscript_X_Y_Z/license/EULA Accepting the End User Agreement is mandatory to proceed with the installation.
   Select “Next” when ready.



9. Components selection:Unselect the *OTCS Module* component. Select all of the extension components that are to be installed

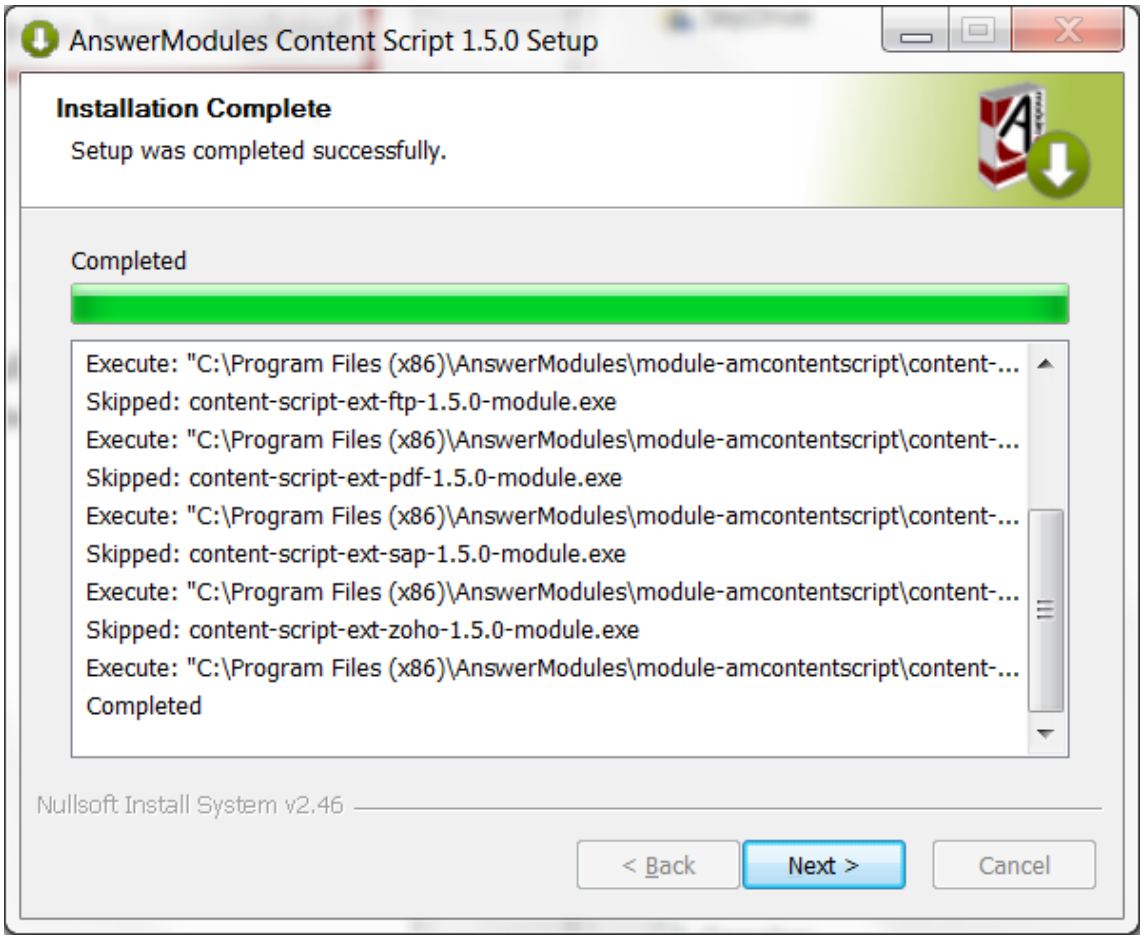
Copyright © 2013-2022 AnswerModules Sagl

Select “Install” when ready.



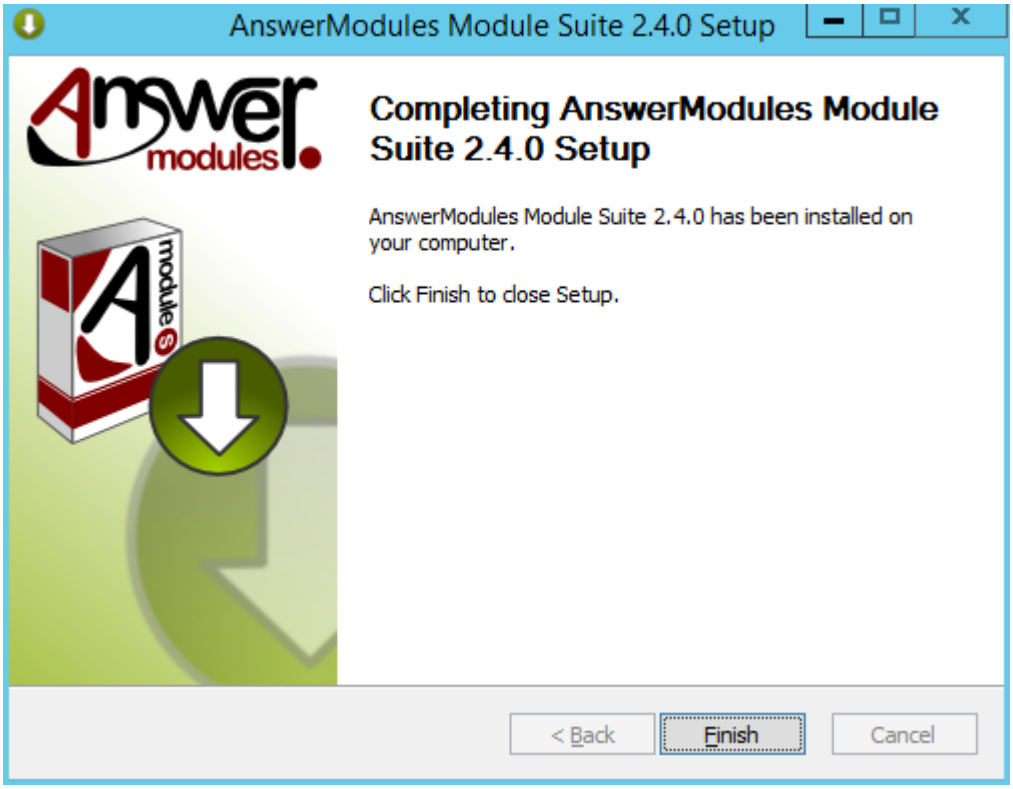
10. Installation: The extension packages are automatically installed.
    Select “Next” when the procedure is complete.

Copyright © 2013-2022 AnswerModules Sagl



Copyright © 2013-2022 AnswerModules Sagl

11. Installation completed: Select “Finish” and return to the installation checklist to finalize the module setup.



- [ ] Install Content Server Modules

  - Start Content Server
  - Login as Administrator and access the Module administration panel
  - Select **“Install Modules”**
  - From the available modules, select **“AnswerModules Content Script X.Y.Z”**
  - Follow the installation steps and stop Content Server when prompted.

If an earlier version of Content Script module was installed, and you are performing an upgrade, it is mandatory once the upgrade is completed to check for the presence of the previous module version in %OTCS_HOME%/module. In the eventuality, the previous version of the module has not been removed automatically, stop Content Server service and remove it manually. The Content Server will fail to start if two versions of the Content Script module are present at the same time.

What to do if the installer raises the error: Unable to automatically extract...

Some Content Script extension packages require two Java libraries that are specific to the target Content Server environment and are not distributed with the module.
The required library files are:

- csapi.jar
- service-api-X.X.XX.jar

Copyright © 2013-2022 AnswerModules Sagl

and can be found in the web app located in:

- %OTCS_HOME%\webservices\java\webapps\cws.war (on CS 16.X)
- %OTCS_HOME%\webservices\java\webapps\cws.war (on CS 10.5.X)

To retrieve the files:

- copy the file named XXX.war to a temporary folder
- rename the file XXX.war in XXX.
- extract the zip archive contents locate the files in the WEB-INF/lib folder

Once the files have been located, copy them to the folder:

%OTCS_HOME\staging\anscontentscript_x_y_z\amlib

- Post installation checks:

  Verify that the following optimization for the Java Server JVM has been added in **opentext.ini**

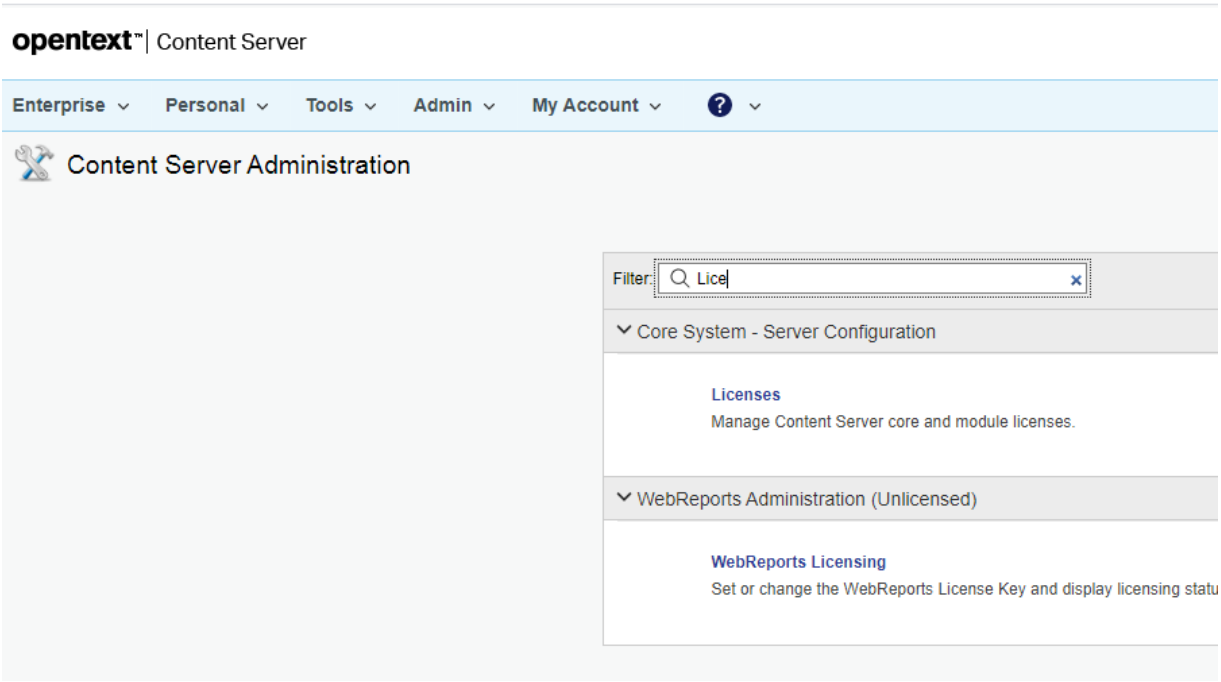
```
JavaVMOption\_4=-Xmx2048m
JavaVMOption\_5=-Xms128m
JavaVMOption\_6=-XX:+CMSClassUnloadingEnabled
JavaVMOption\_7=-XX:+UseConcMarkSweepGC
JavaVMOption\_8=-Dfile.encoding=UTF-8
JavaVMOption\_9=-Dlog4j.ignoreTCL=true
```

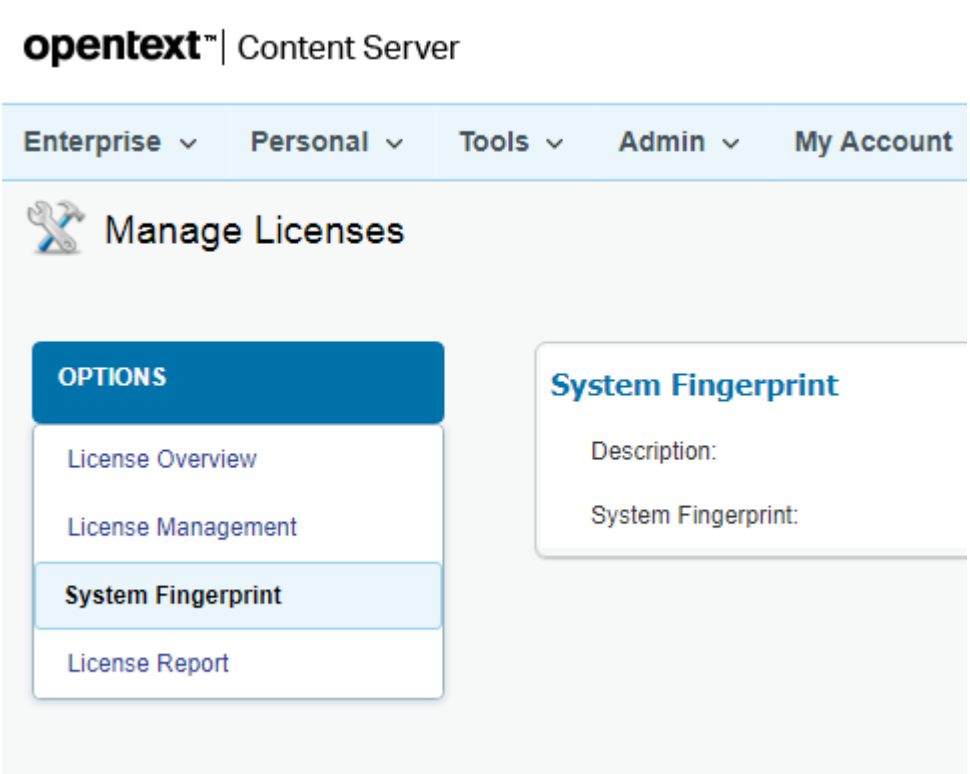
☐ From the Administration Home, select **AnswerModules Administration > Base Configuration**, then enter the activation License in order to activate the product. License shall be entered in the **Module Suite - Activation Key** property

How do I get an activation key?

In order to activate Module Suite you need a valid activation key. Activation keys can be requested to AnswerModules Support *(https://support.answermodules.com)* by providing the OpenText Content Server System Fingerprint. You can read your's environment fingerprint from the OpenText Admin Pages as shown below

Copyright © 2013-2022 AnswerModules Sagl





Since version 1.7.0, if the installation is performed on a multi-server architecture, the License Key must be made available on every single machine. License Key information is stored in the Content Script module **anscontentscript.ini** file.

In order to do so, there are two alternative options:

- Edit the .ini file manually on each machine, adding the licensing information.
- From the Administrative pages of each machine, perform the “save Base Configuration” operation.

☐ Save the Base Configuration and restart Content Server

Copyright © 2013-2022 AnswerModules Sagl

- [ ] Import the Module Suite components and widgets library

Installation complete

The Content Script initial setup is complete

Installing Beautiful WebForms

Installation procedure (Windows)¶

We will refer to the Content Server installation directory as **%OTCS_HOME%**

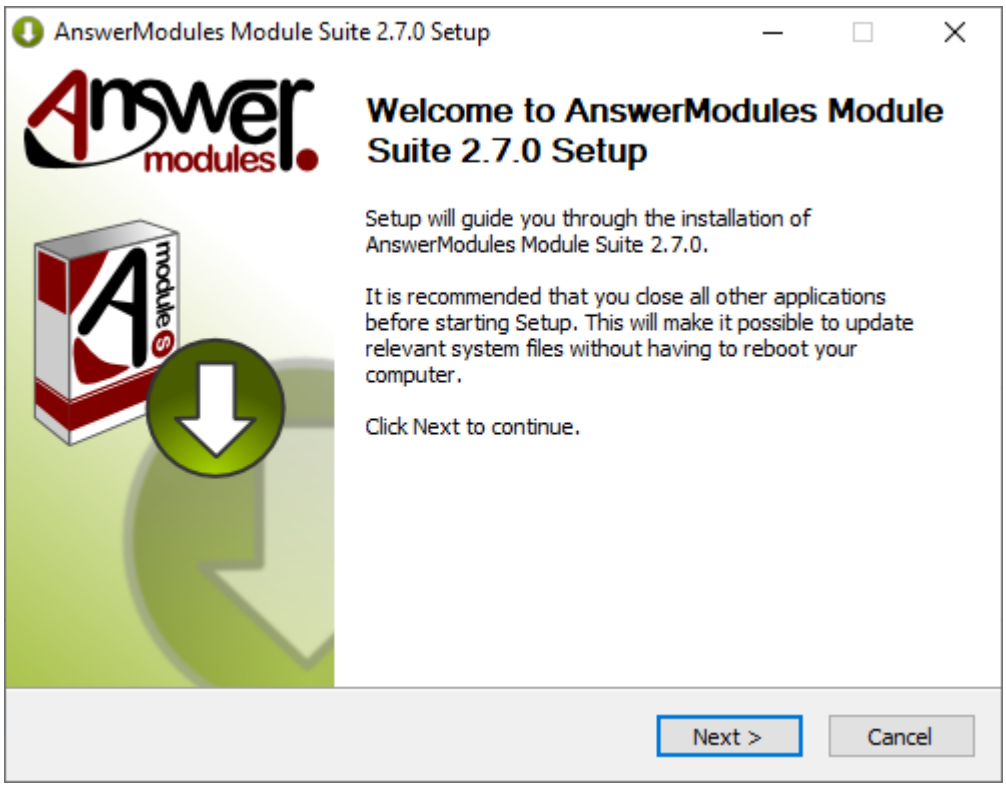
- [ ] Deploy Content Server Modules

  - Stop the Content Server
  - Run the **Module Suite Master Installer** and install the Beautiful WebForms module.

Step-by-step procedure

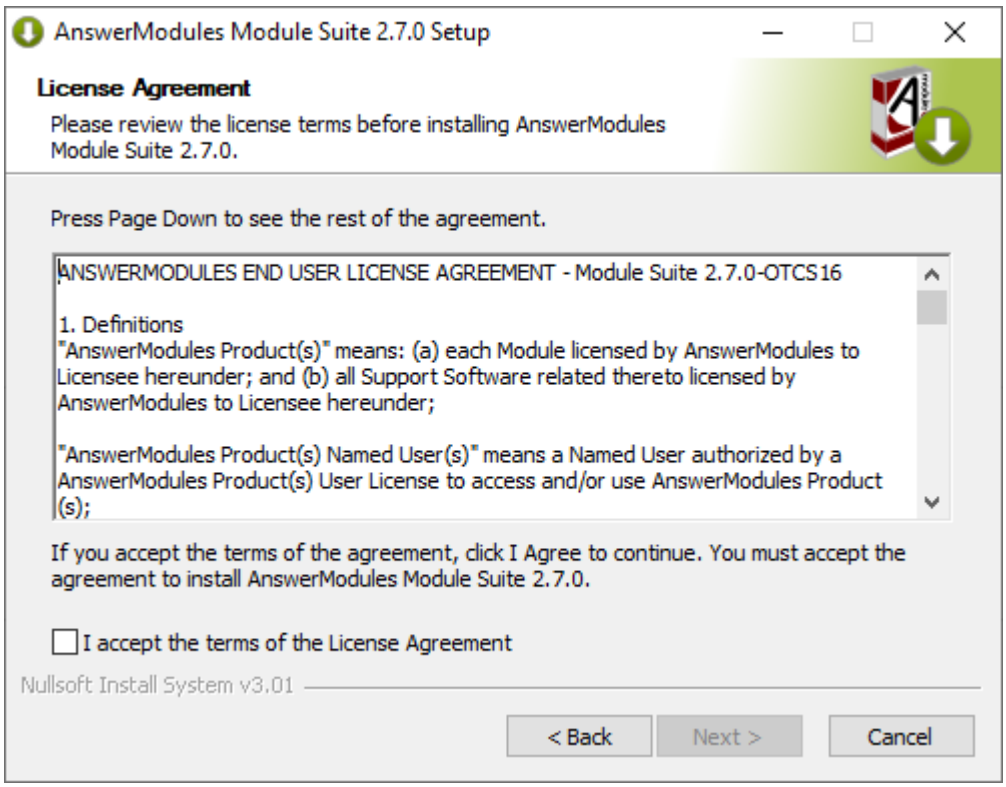
The following screens will guide you through the deployment of Module Suite modules.

1. Welcome Screen: Select “Next” when ready to start the installation.



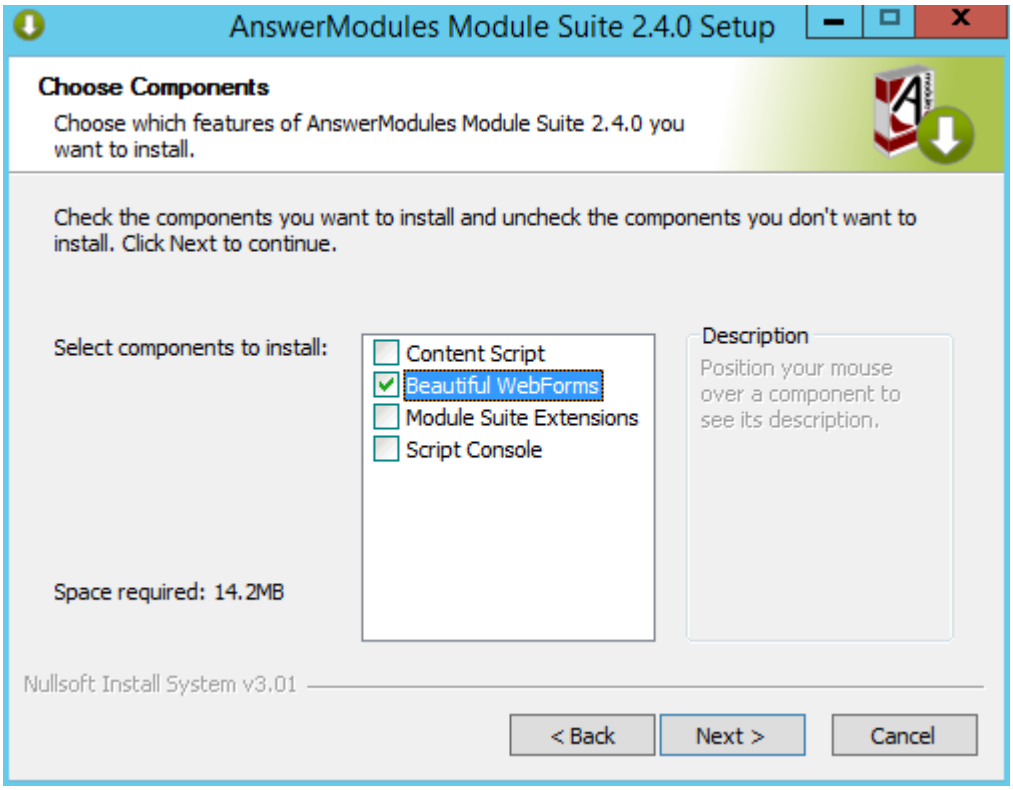
Copyright © 2013-2022 AnswerModules Sagl

2. EULA Screen: Acceptance of the end-user license agreement is mandatory for proceeding with the installation
   A copy of the agreement will be available, upon installation, in:
   %OTCS_HOME%/module/amcontentscript_X_Y_Z/license/EULA
   Select “Next” when ready.

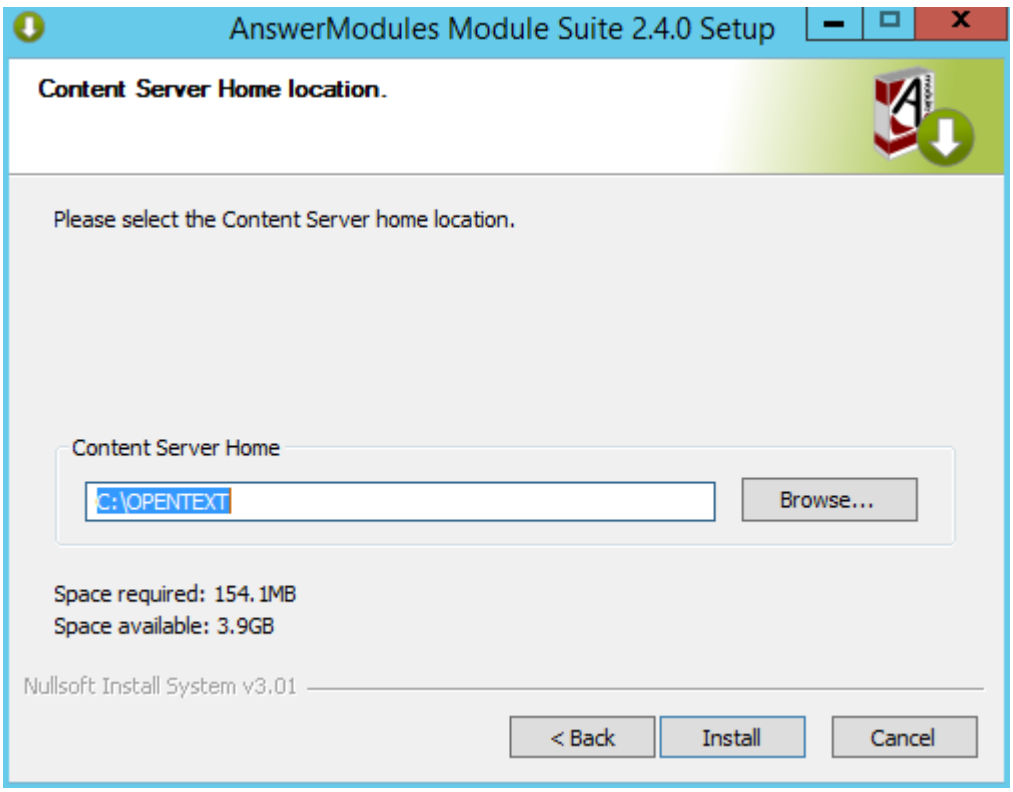


3. Components selection: Script console is unselected by default because it is not a Content Server module. A standard Module Suite installation does not require this component to be installed.
   Select “Next” when ready.

Copyright © 2013-2022 AnswerModules Sagl

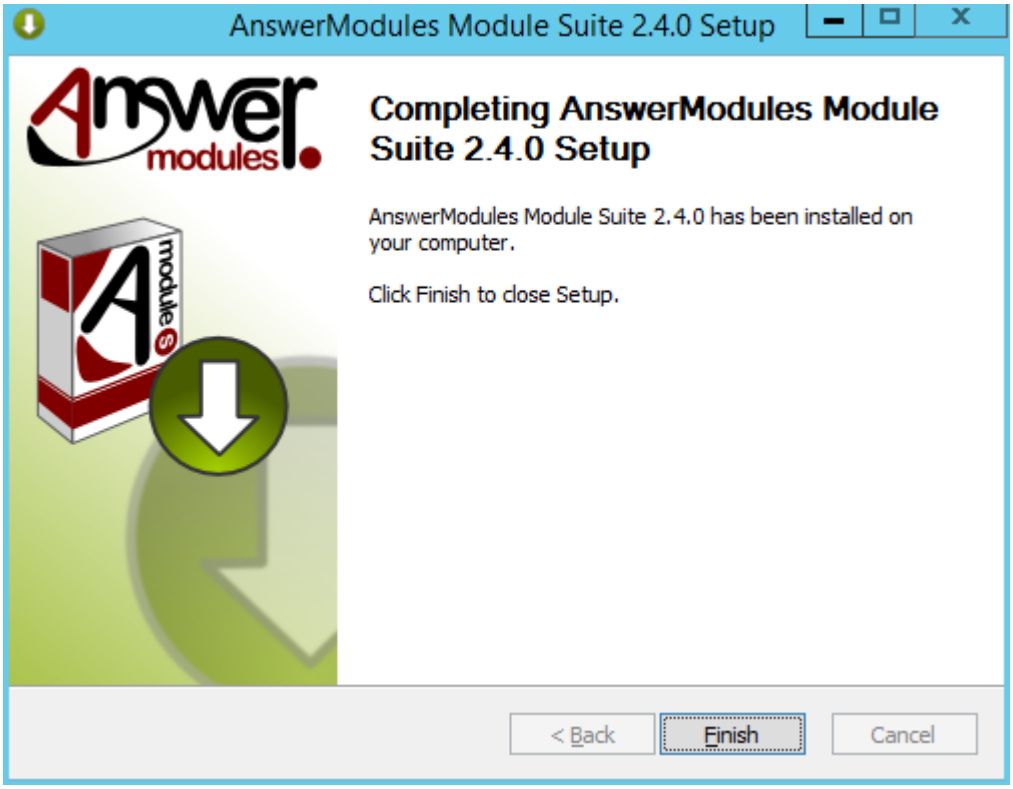


4. OTHOME selection: The installer will prompt you for the location where Content Server is installed. Browse to your **OTCS_HOME** and select "Next" when ready to start the installation.



Copyright © 2013-2022 AnswerModules Sagl

5. Installation completed: Select “Finish” and return to the installation checklist to finalize the module setup.



- [ ] Install Content Server Modules

  - Start the Content Server
  - Login as Administrator and access the Module administration panel
  - Select **“Install Modules”**
  - From the available modules, select **“Answer Modules - Beautiful Web Forms X.Y.Z”**
  - Follow the installation steps and stop Content Server when prompted.

Installation complete

The Beautiful WebForms initial setup is complete

Copyright © 2013-2022 AnswerModules Sagl

Installing Smart Pages (f.k.a. Module Suite Extension for SmartUI)

Prerequisites¶

This guides assumes the following components to be already installed and configured:

- AnswerModules' ModuleSuite

We will refer to the Content Server installation directory as **OTCS_HOME**

Installation procedure¶

The **AnswerModules Smart Pages** includes the following parts:

- *AnswerModules Smart Pages* Content Server module
- SmartUI Extension library objects to be imported in the Content Script Volume

Installing the Module Suite extension for SmartUI¶

Run the Module Suite SmartUI installer:

```
module-anscontentsmartui-X.Y.Z-OTCS16.exe
```

Follow the installation wizard steps:

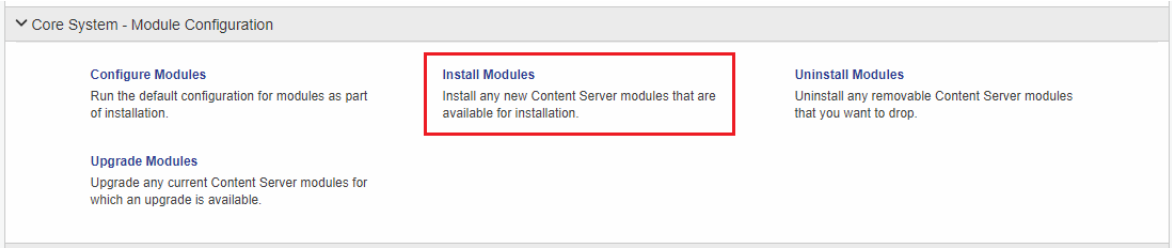
- [x] Select "Next" when ready to start the installation.
- [x] The installer will prompt you for the location where Content Server is installed. Browse to your **OTCS_HOME** and select "Next".
- [x] Review the installation steps for each component to be installed.
- [x] Click "Finish" to complete the unpacking of the module

Staging

At this point, the Module has been deployed in the Content Server Staging folder and is available for installing it through the Content Server administration pages.

- [x] Access the Content Server Admin pages > **Core System - Module Configuration** > **Install Modules**

Copyright © 2013-2022 AnswerModules Sagl



- ☑ Locate the **AnswerModules Smart Pages** module and proceed with installation
- ☑ Restart the OTCS services when prompted in order for the installation to be completed.

Deploying Beautiful WebForms static resources¶

The Smart Pages include a set of static resources (javascript libraries, css, etc.) required to enable embedding of Beautiful WebForms within SmartUI perspectives. In order to deploy these resources, perform the following steps:

- ☑ On the server's filesystem, navigate to the **anscontentsmartui** module support folder, and locate the folder named **ansbwebform**

```
OTCS_HOME\support\anscontentsmartui\ansbwebform
```

- ☑ Copy the contents of the **ansbwebform** folder identified above to the Beautiful WebForms module support folder:

```
OTCS_HOME\support\ansbwebform
```

Merging

The two paths contain a similar folder structure, which will be merged to add SmartUI specific files to the Beautiful WebForms support folder

Importing the SmartUI Extension library objects¶

In order to import the **SmartUI Extension library**, perform the following steps.

Only once

This procedure imports objects within the Content Script Volume. In case of clustered environment, it is only necessary to perform this operation once, as the volume is shared between all instances.

- ☑ On the server filesystem, navigate to the location of the **AnswerModules Smart Pages** module and open the **library** folder

  OTCS_HOME\module\anscontentsmartui_X_Y_Z\library

Copyright © 2013-2022 AnswerModules Sagl

- [x] Copy the **smartui extension.lib** file to the same location within the **Content Script** module

  OTCS_HOME\module\anscontentscript_X_Y_Z\library

- [x] In a browser, open the Content Server Admin pages > **AnswerModules Administration** > **Base Configuration**

Check the URL

If you are in a clustered environment, make sure you are accessing the right server (the one on which you performed the steps above).

- [x] Open the **Import** tool to import the content of the **SmartUI Extension** library. For details on how to use the **Import** tool, check the dedicated guide.

- [x] Copy the Smart View's Beautiful WebForms widgets from the **SmartUI Extension** library into their standard location in the **Content Script Volume**

```
Content Script Volume:SmartUI Extension:CSFormSnippets:V2:* -> Content Script Volume:CSFormSni
Content Script Volume:SmartUI Extension:CSFormSnippets:V3:* -> Content Script Volume:CSFormSni
```

- [x] Copy the Smart View's Content Script snippets from the **SmartUI Extension** library into their standard location in the **Content Script Volume**

```
Content Script Volume:SmartUI Extension:CSScriptSnippets:* -> Content Script Volume:CSScriptSr
```

Installing on a clustered environment

Installation approach¶

In a Content Server cluster environment, it is mandatory to install Module Suite modules on each node that makes up the cluster. The installation process in a cluster is more complex than installing on a single server, as a slightly different procedure must be performed on each remaining node in the cluster after installing the modules on the first one. The recommended approach is to install Module Suite on a primary node (the node on which the primary Opent Text Admin Content Server services are installed and configured) and then copy the installed modules to each node in the cluster. This approach ensures that all installed modules are identical and that the patch level on all nodes is the same.

We will refer to the Content Server installation directory as **%OTCS_HOME%**.

Copyright © 2013-2022 AnswerModules Sagl

Deployment on the primary node¶

Module Suite package installation on a primary node is identical to the installatoin process into the non-clustered environment.

Installation procedure¶

- [ ] Stop Content Server services on all the node of the cluster but the primary one
- [ ] Deploy Content Server Modules

Detailed description of this step can be found in `Deploy Content Server Modules` section.

- [ ] Install Content Server Modules

Detailed description of this step can be found in `Deploy Content Server Modules` section.

- [ ] Apply hotfixes on the Primary node

To apply hotfixes please follow the steps on a page `Applying HotFixes`

- [ ] Secondary node(s) installation

Once the Module Suite modules are installed on the primary node, the module packages can be deployed on the remaining cluster nodes.

- Primary node: Make a copy of the following resources and make them available in a working folder on the secondary node:

    1. %OTCS_HOME%/module/anscontentscript_x_x_x
    2. %OTCS_HOME%/module/ansbwebform_x_x_x
    3. %OTCS_HOME%/module/anscontentsmartui_x_x_x
    4. %OTCS_HOME%/support/anscontentscript
    5. %OTCS_HOME%/support/ansbwebform
    6. %OTCS_HOME%/support/anscontentsmartui

- Secondary node: Stop Content Server\u2019s services

- Secondary node: Run the **Module Suite Master Installer** and unpack: Content Script, Beautiful WebForms, Smart Pages modules and all the desired Module Suite Extension packages.

- Secondary node: Remove extracted files form the staging folder:

    1. %OTCS_HOME%/staging/anscontentscript_x_x_x
    2. %OTCS_HOME%/staging/ansbwebform_x_x_x
    3. %OTCS_HOME%/staging/anscontentsmartui_x_x_x

Copyright © 2013-2022 AnswerModules Sagl

- Secondary node: Copy the modules and support folders
- **Reconcile the opentext.ini** (with particular reference to the [Modules] section) file in **%OTCS_HOME%/config**
- Start the Content Server\u2019s services
- Any node: Continue with Modules Suite installation and configuration. Once modules are installed on all instances in the Content Server environment it is possible to contine with the configuration. Configuration is performed in a Browser on any server in the environment.

For configuration steps please refer to `Installing the Suite` page.

Installing Script Console

Installation procedure¶

Script Console can be configured to run in different modes. Common scenarios are:

1. standalone interactive console, connected to OTCS: mainly used for batch processing and administration tasks
2. standalone script interpreter, connected to OTCS: mainly used for scheduling administration tasks
3. standalone lightweight webserver (based on embedded application server), connected or not connected to OTCS
4. web application deployed on external application server, connected or not connected to OTCS

This guide covers the standard installation procedure of the Content Script Console (standalone based on embedded application server) which is compliant with the options 1, 2 and 3 of the above list.

For alternative deployment scenarios, including deployment on an external application server, please make reference to AnswerModules Support Team and guides available through Support Portal.

- [x] Run the **Script Console Installer** (WINDOWS), or extract the Script Console archive, and install the Script Console in your favourite location (this step should be executed by an user having local administrative privileges)

Environment variables

Copyright © 2013-2022 AnswerModules Sagl

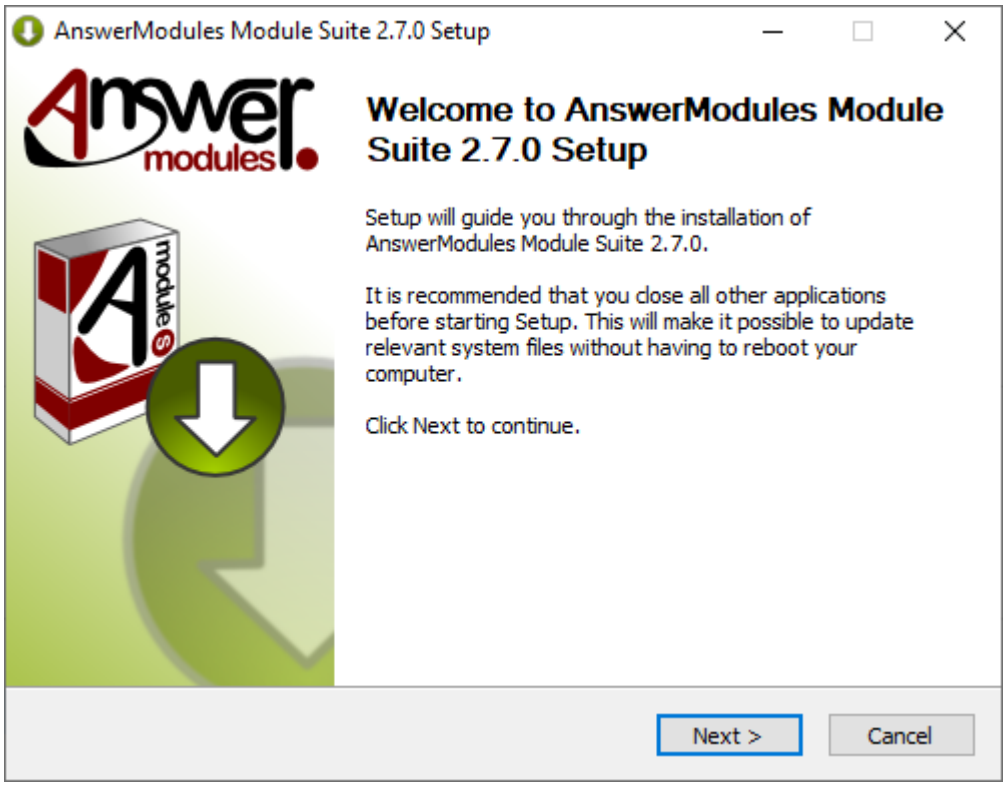
The Script Console requires an environment variable to be defined in order to work properly, for your convenience this variable is automatically defined on windows server by the Script Console installer:

- AM_CONSOLE_DATA: the Script Console's root folder

Step-by-Step procedure

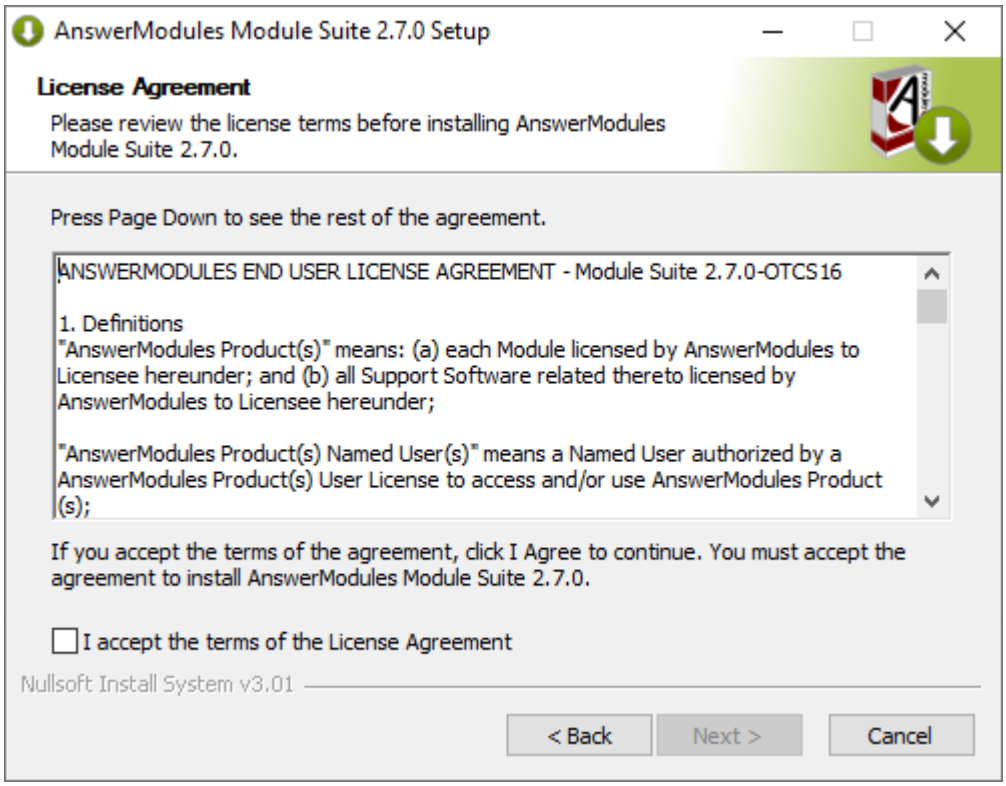
The following screens will guide you through the deployment of Script Console runtime.

1. Welcome Screen: Select “Next” when ready to start the installation.



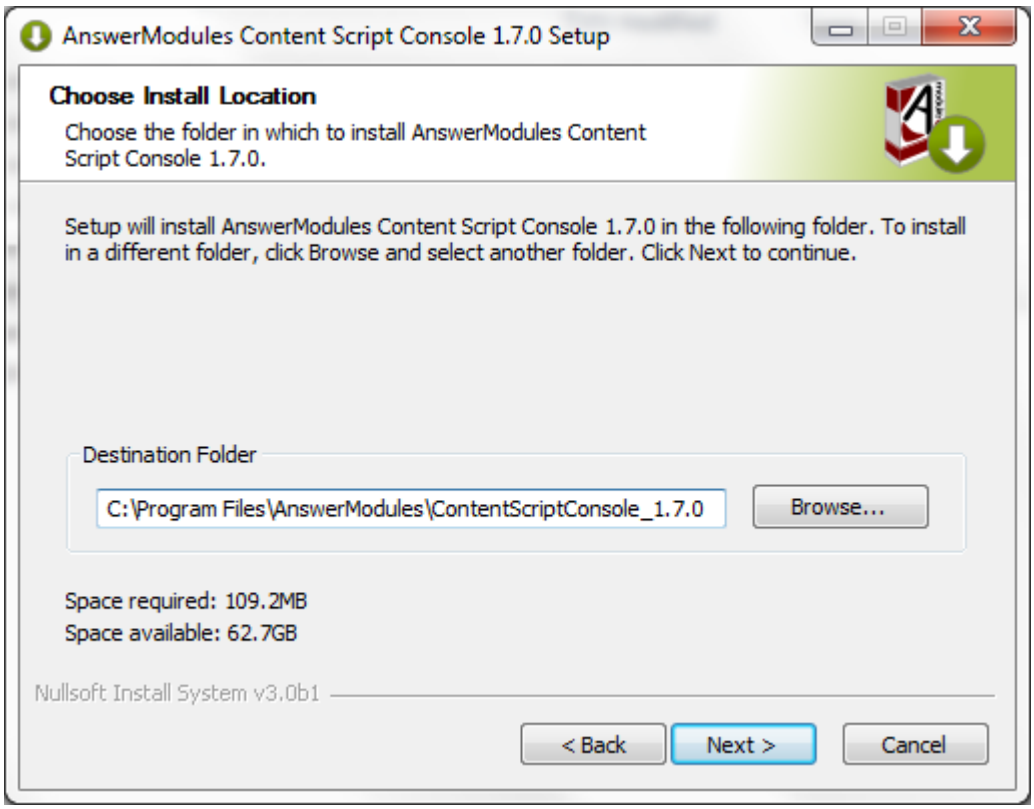
2. EULA Screen: Acceptance of the end-user license agreement is mandatory for proceeding with the installation
   A copy of the agreement will be available, upon installation, in:
   %AM_CONSOLE_DATA%/license/EULA Select “Next” when ready.

Copyright © 2013-2022 AnswerModules Sagl



3. AM_CONSOLE_DATA selection: Choose the location where the Script Console components will be installed. E.g.

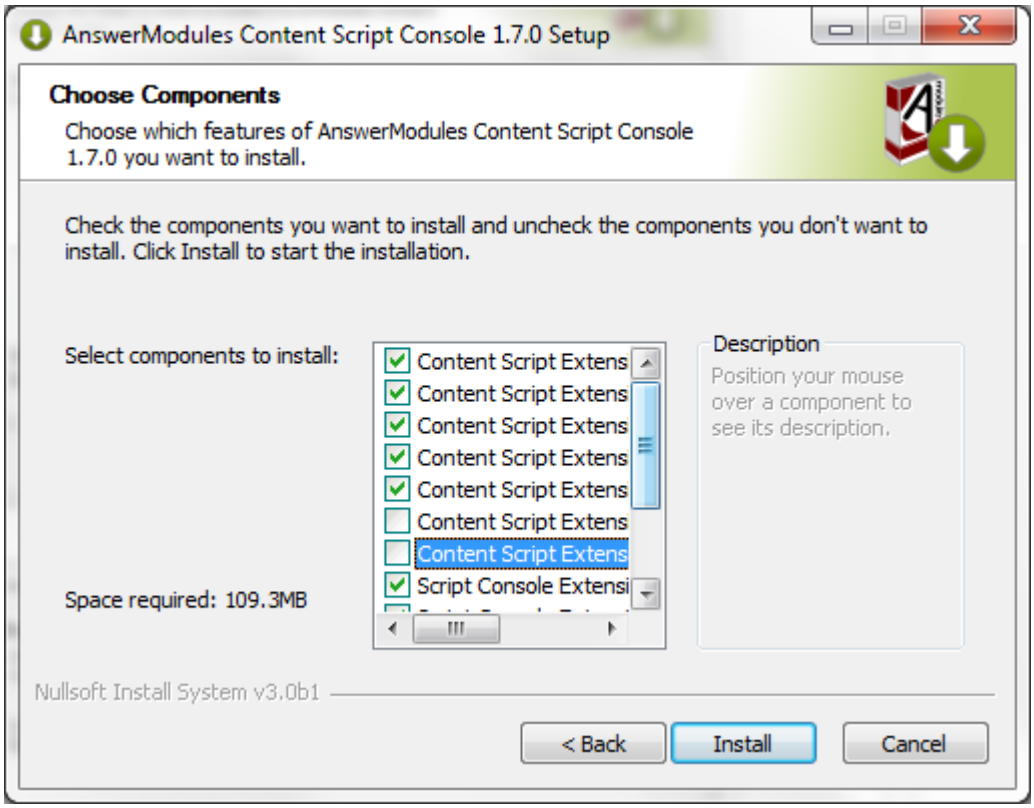
```
E:\AM\SC_2_7_0\
```



4. Script Console Application and Content Script Extension Packages: there are two different types of extensions that can be installed:

5. Content Script Extensions are extensions for the embedded Content Script Engine.

Copyright © 2013-2022 AnswerModules Sagl



11. Installation completed: Select “Finish” and return to the installation checklist to finalize the module setup.

- [x] Copy Content Server's libraries to the Script Console runtime

Copyright © 2013-2022 AnswerModules Sagl

Content Server libraries required

Some Content Script extension packages require additional Java libraries that are specific to the target Content Server environment, and are not distributed with the module. The required library files are:

- csapi.jar
- service-api-X.X.XX.jar

and can be found in the web app located in:

`%OTCS\_HOME%\\webservices\\java\\webapps\\cws.war`

- classificationsservice-api-X.X.XX.jar

which can be found in the web app located in:

`%OTCS\_HOME%\\webservices\\java\\webapps\\cs-services-classifications.war`

- physicalobjectsservice-api-X.X.XX.jar

which can be found in the web app located in:

`%OTCS\_HOME%\\webservices\\java\\webapps\\cs-services-physicalobjects.war`

- recordsmanagementservice-api-X.X.XX.jar

which can be found in the web app located in:

`%OTCS\_HOME%\\webservices\\java\\webapps\\cs-services-recordsmanagement.war`

- oml.jar

which can be found in: `%OTCS\_HOME%\\ojlib`

To retrieve the files:

- copy the file named XXX.war to a temporary folder
- rename the file XXX.war in XXX.zip
- extract the zip archive contents locate the files in the WEB-INF/lib folder

Once the files have been located, copy them to the folder: **%AM_CONSOLE_DATA%/runtime/amlib**

Copy libraries form Content Script

All the libraries mentioned above but ** oml.jar ** are usually also found in the installation folder of the Content Script module: `%OTCS_HOME/module/anscontentscript_X_Y_Z/amlib`

- [ ] Perform basic configuration of the Script Console. The main configuration file is located in: **%AM_CONSOLE_DATA%/config/cs-console-systemConfiguration.xml**
Default configuration will be similar to the following:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <config>
      <systems>
          <system id="TEST">
              <name>Content Server TEST Environment</name>
              <serverHost>localhost</serverHost>
              <serverPort>2099</serverPort>
              <local-repository-home>TEST</local-repository-home>
```

Copyright © 2013-2022 AnswerModules Sagl

```
            <local-repository-encoding>UTF-8</local-repository-encoding>
            <otcs-repository-encoding>UTF-8</otcs-repository-encoding>
            <systemVars>
                <systemVar name="img">/</systemVar>
                <systemVar name="url"></systemVar
                <systemVar name="csModulePath">
            </systemVars>
            <serviceVars>
                <serviceVar service="core" name="amcs.core.httpProxyHostname" ></serviceVar>
                <serviceVar service="core" name="amcs.core.httpProxyPort">80</serviceVar>
                <serviceVar service="core" name="amcs.core.httpProxyUsername"></serviceVar>
                <serviceVar service="core" name="amcs.core.httpProxyPassword"></serviceVar>
                <serviceVar service="core" name="amcs.core.httpMaxConnPerRoute">20</serviceV
                <serviceVar service="core" name="amcs.core.httpMaxConnTotal">50</serviceVar>
                <serviceVar service="core" name="amcs.core.httpOTCSSchema">http</serviceVar>
                <serviceVar service="core" name="amcs.core.tempFilePath">/tmp/</serviceVar>
            </serviceVars>
            <users></users>
        </system>
    </systems>
</config>
```

The base configuration allows to specify one or more “system” objects which represent OTCS instances to which the console will be able to connect.

How to setup your base configuration

The base configuration can be edited manually, or, alternatively, configuration parameters can be downloaded from a target Content Server instance. This feature comes particularly handy for installations that include multiple Content Script Extension Packages, each with its own configuration settings.

- [x] Apply any available hotfix(es)

Hot to install a hotfix

Before you install any hotfix, please backup all essential files. To install the hotfix, download the hotfix from the Support portal and save it to a temporary location. Make sure Script Console services (or executable) are completely stopped. From the temporary location, extract the contents of the hotfix to the <Script_Console_home> directory and then restart it.
The directory (directories) and file(s) contained in the hotfix(es) you install will be copied to <Script_Console_home>

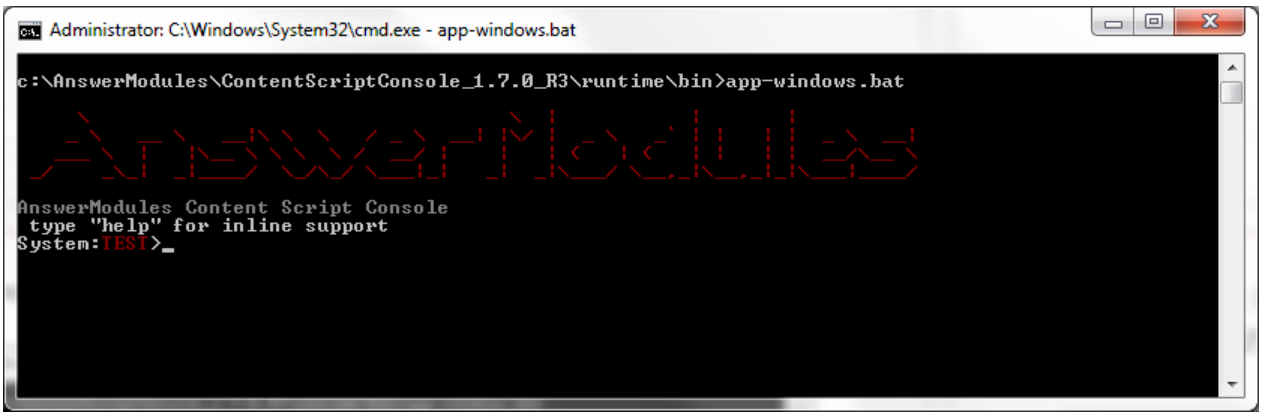
Please always make reference to the hotfix's description file:/hotfixes/hotFix_ANS_XXX_YYY_ZZZ.hfx for specific installation instructions or pre/post installation procedures

Configure Script Console¶

To perform configuration against an OTCS instance, run the Script Console in shell mode. To do so, open a Windows Commands Processor and move to the folder: **%AM_CONSOLE_DATA%/runtime/bin** which includes the Script Console's executables scripts

- Run the `app-windows.bat` or `app.sh` script
- The following prompt should appear:

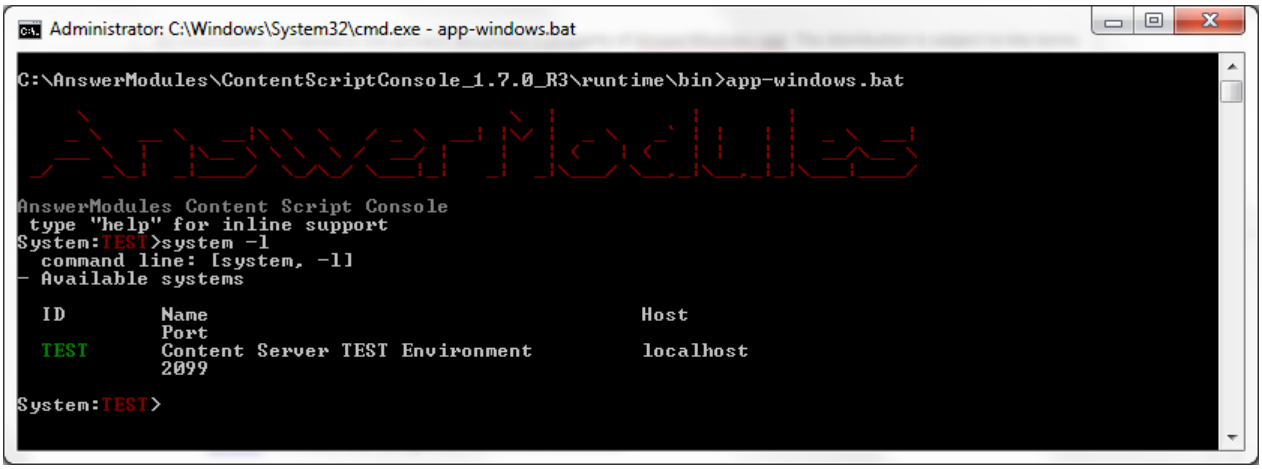
Copyright © 2013-2022 AnswerModules Sagl



Unix

```
centos:/opt/am/sc/runtime/bin$ export AM_CONSOLE_DATA=/opt/am/sc
centos:/opt/am/sc/runtime/bin$ ./app.sh
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

AnswerModules Content Script Console
 type "help" for inline support
System:TEST>
```

- The default TEST system is selected. To list all available systems, use the system command with the list flag (`-l`, `--list`). E.g. `system -l`:



Unix

Copyright © 2013-2022 AnswerModules Sagl

```
AnswerModules Content Script Console
 type "help" for inline support
System:TEST>system -l
- Available systems

  ID        Name                                Host                                Port
  TEST      Content Server TEST Environment     localhost                           2099

System:TEST>
```

- To create a new system (for example, *LOCAL*) use the system command with the add flag (`-a, --add`) followed by the ID of the new system. E.g. `system -a LOCAL`

  The shell will prompt for the required base values, such as `hostname` and `port` number.

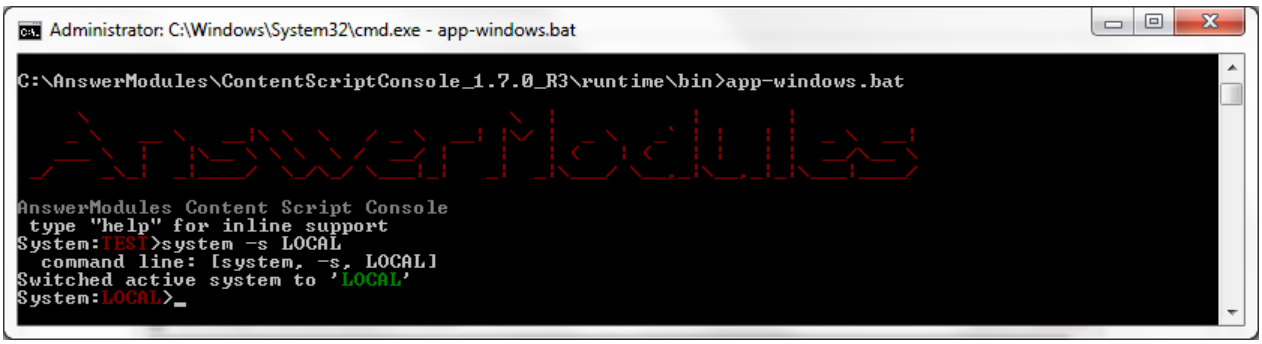
Unix

```
System:TEST>system -a LOCAL
  Host: localhost
  Port: 2099
  Username (opt): Admin
  Password (opt): ************
- New System details
  ID:   LOCAL
  NAME: Default
  HOST: localhost
  PORT: 2099

Shutting down..
centos:/opt/am/sc/runtime/bin$
```

Upon creating a new system, the Script Console will require a restart and will automatically shutdown.

- Switch the active system to LOCAL using the system command with the system flag (`-s`) followed by the ID of the target system. E.g. `system -s LOCAL`

Copyright © 2013-2022 AnswerModules Sagl



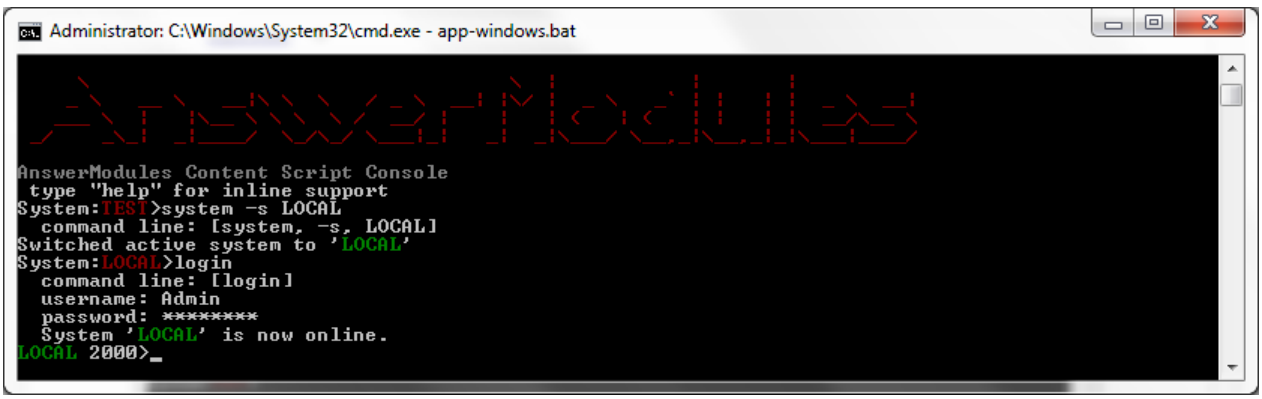
Unix

```
centos:/opt/am/sc/runtime/bin$ ./app.sh
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

```
AnswerModules Content Script Console
 type "help" for inline support
System:TEST>system -s LOCAL
Switched active system to 'LOCAL'
System:LOCAL>
```

The active system indicator in the command prompt should now indicate LOCAL.

- Login to the LOCAL system using the `login` command



Unix

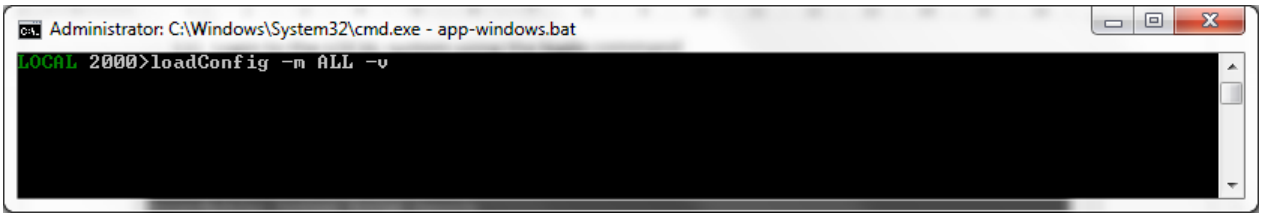
```
System:LOCAL>login
  System 'LOCAL' is now online.
LOCAL 2000>
```

Copyright © 2013-2022 AnswerModules Sagl

The active system indicator in the command prompt should now turn green to indicate that the system is ONLINE

- Synchronize ModuleSuite configuration parameters from the LOCAL system using the loadConfig command with the mode flag (`-m`, `--mode`) followed by the ALL value, and the verbose flag (`-v`, `--verbose`). E.g. `loadConfig -m ALL -v` The LOCAL system base configuration will be transfered and stored in the Script Console configuration file

- The configuration is complete. Try a simple ls command to test the connection



Unix

```
System:LOCAL>login
  System 'LOCAL' is now online.
LOCAL 2000>loadConfig -m ALL

LOCAL 2000>
```

Installing Extension Packages

Installation procedure¶

We will refer to the Content Server installation directory as **%OTCS_HOME%**

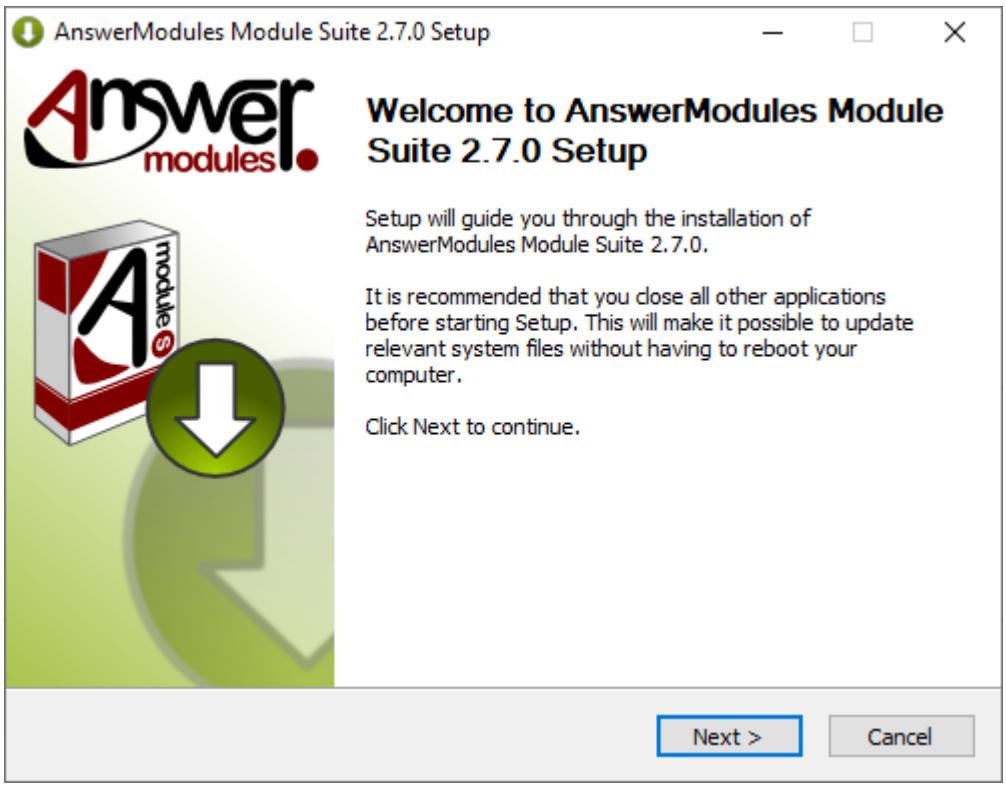
- Run the **Module Suite Content Script Master Installer** and install the desired extension packages.

Step-by-step procedure

Copyright © 2013-2022 AnswerModules Sagl

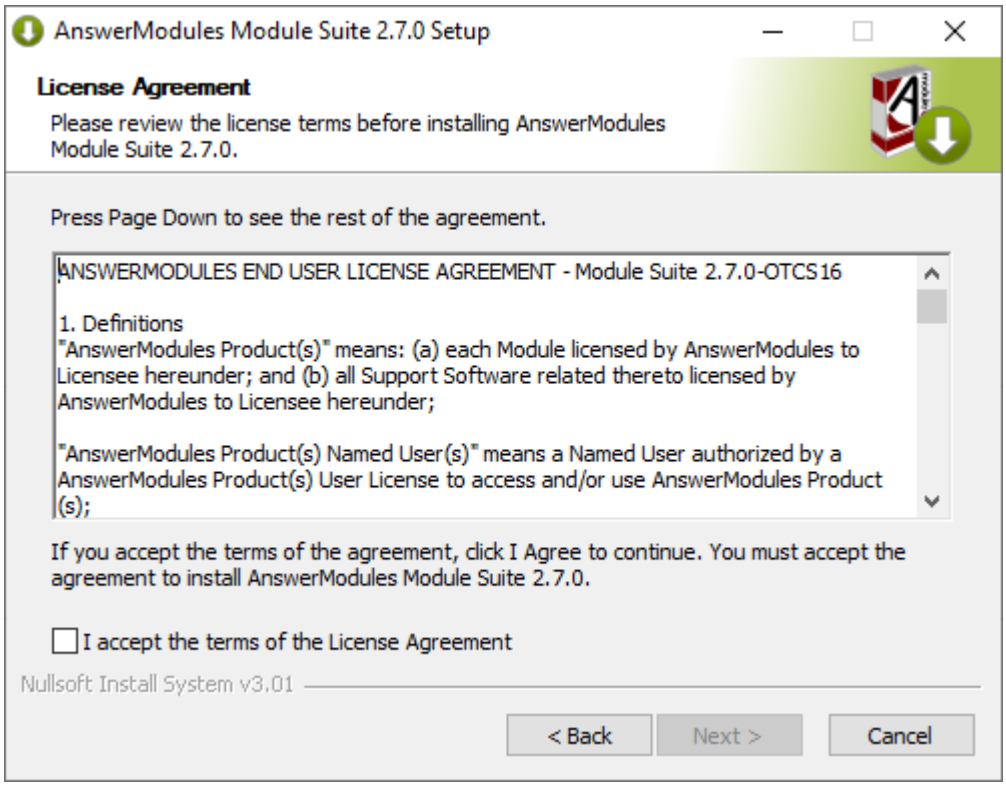
The following screens will guide you through the Content Script Module Master Installer steps required to install optional extension packages:

1. Welcome Screen: Select “Next” when ready to start the installation.

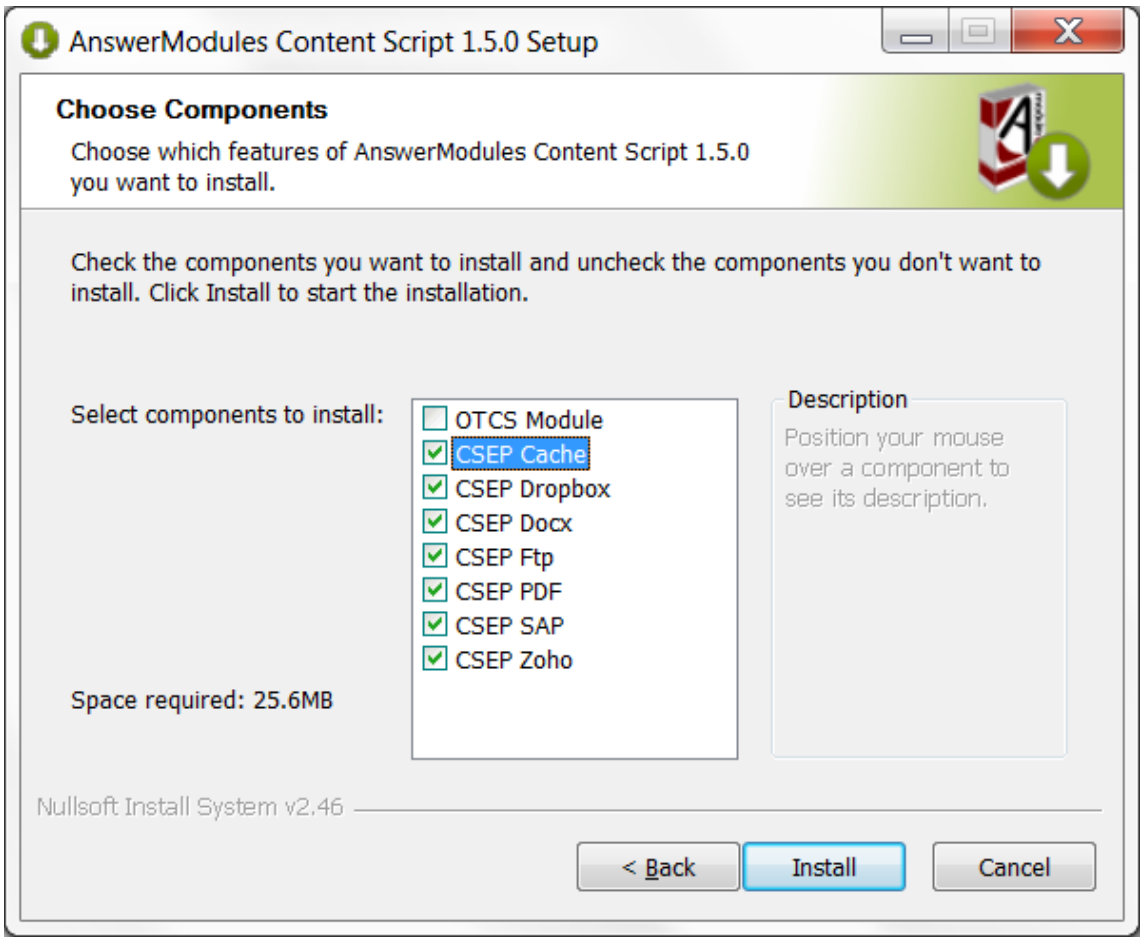


2. EULA Screen: Acceptance of the end-user license agreement is mandatory to proceed with the installation
   A copy of the agreement will be available, upon installation, in:
   %OTCS_HOME%/module/amcontentscript_X_Y_Z/license/EULA Accepting the End User Agreement is mandatory to proceed with the installation.
   Select “Next” when ready.

Copyright © 2013-2022 AnswerModules Sagl

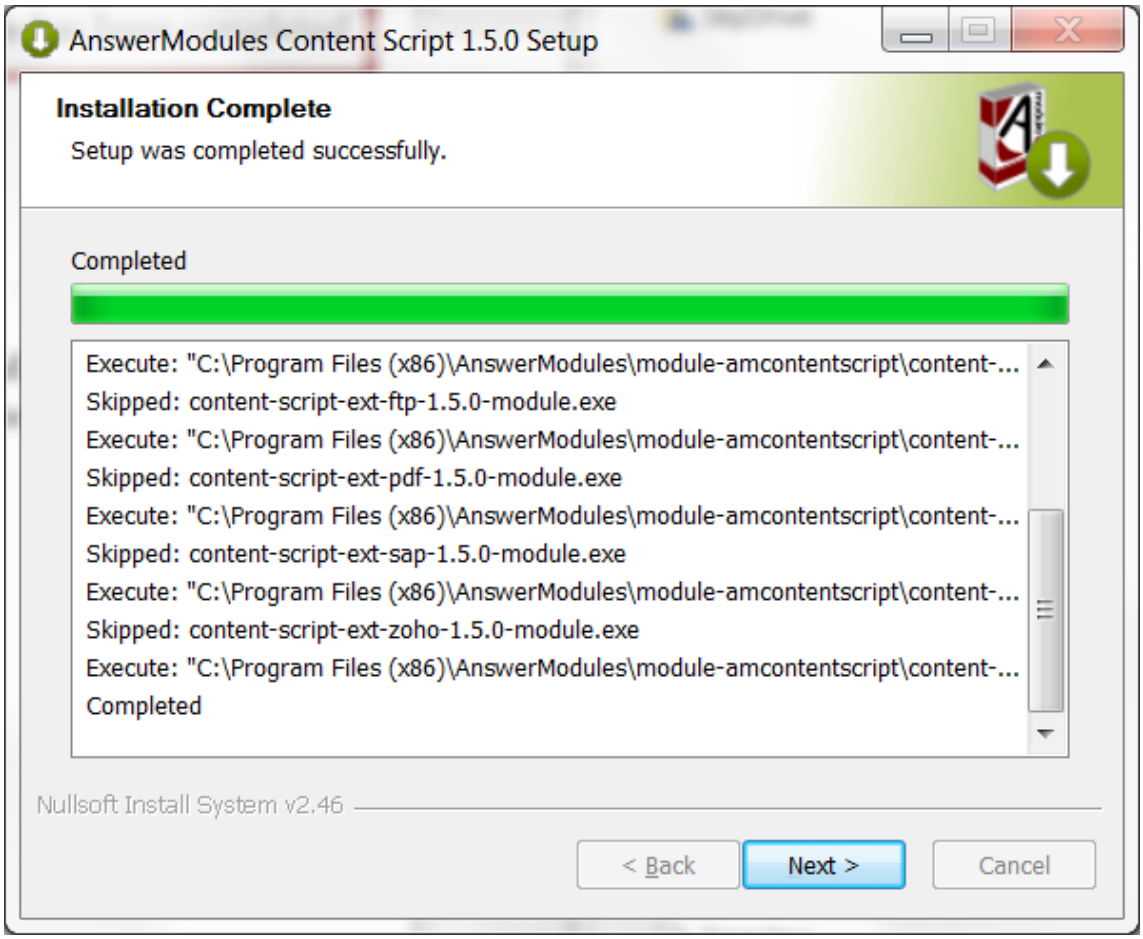


3. Components selection:Unselect the *OTCS Module* component. Select all of the extension components that are to be installed
   Select “Install” when ready.



Copyright © 2013-2022 AnswerModules Sagl

4. Installation: The extension packages are automatically installed.
   Select “Next” when the procedure is complete.



Configure the Extension Packages

If you are installing extension packages on an already installed and properly configured Module Suite instance you have to update the module's **Base Configuration** following the procedure below:

- Stop and Start Content Server service to let the system load the newly installed Extension Packages
- Login as Administrator and access the Module administration panel
- From the Administration Home, select **AnswerModules Administration > Base Configuration**
- If necessary, change the core configuration or the configuration of the extension modules.
- Save the Base Configuration (even in case no changes were applied) and restart Content Server

In order for any change in the base configuration to become effective, a restart of Content Server is required

Copyright © 2013-2022 AnswerModules Sagl

Rendition Extension Package¶

What is it?¶

The rendition extension package allows you to programmatically invoke a third party rendition engine to convert documents from one format to another, the most common use case is to convert HTML documents to PDF documents. Using the rendition extension package, you will be able to convert documents in real time and without interrupting the script execution flow.

The installation procedure for the rendition extension package isn't different from any other extension package, although it requires a couple of additional steps to be completed.

Install the third party rendition engine¶

The CS Rendition Extension package only provides the API to interface with a third party engine capable of converting documents.

This software is distributed separately by the third party and has to installed separately.

Although potentially compatible with different engines, the rendition extension package is pre-configured and tested to use on of the following options:

- an open source engine named wkhtmltopdf
- an open engine AnswerModules R&D Team derived from the open source project Puppeteer *(https://github.com/puppeteer/puppeteer)* named **rend**

The installation and configuration of the two above mentioned solutions is pretty similar.

wkhtmltopdf¶

Installation¶

- [x] Follow the software developers instructions to perform the installation on each server in the OTCS cluster on which the extension is needed.
- [x] Upon a successful installation, the main executable has to be made available to the Content Script Extension Package as a dropin.

  To do so:

  - locate the wkhtmltopdf installation path
  - locate the wkhtmltopdf.exe executable in the folder
  - copy the wkhtmltopdf.exe in the CS Rendition Extension package dropin folder, located in:

Copyright © 2013-2022 AnswerModules Sagl

```
<OTCS_HOME>/module/anscontentscript_x_x_x/amlib/rend/dropin
```

Configuration¶

- [x] Configure the Rendition extension package in order to use the ***wkhtmltopdf*** executable in the Module Suite Base Configuration *(/administration/modulesuite/#base-configuration)*

  Section ***rend***

Configuration Property	Configuration Property Value
amcs.rend.html2pdf.dropin	wkhtmltopdf
amcs.rend.html2pdf.cmdline	-B 10 -T 10 -L 5 -R 5 --viewport-size 1920x1080 ${source} --print-media-type --cookie ${cookie} --run-script "am_printFix()" ${destination}
amcs.rend.html2pdf.timeout	60000
Configuration Property	**Configuration Proerty Meaning**
amcs.rend.html2pdf.dropin	The relative path to the engine's executable. For security reasons, the root of this path is the extension package's dropin folder.
amcs.rend.html2pdf.cmdline	The template of the command line instruction to be used when performing rendition ***(**)***. A few replacement tags can be used in this command line template. (a) ${source} : represent the absolute path for the input resource you want to render. Its value is automatically injected by the rendition extension package. Since the rendition extension package works on Content Script Resources, you do not have to worry about file system housekeeping. (b) ${destination} :represent the absolute path for the output resource, the engine is going to generate. Its value is automatically injected by the rendition extension package. Since the rendition extension package works on Content Script Resources, you do not have to worry about file system housekeeping. c ${cookie} : represent a local authentication cookie
amcs.rend.html2pdf.timeout	the default maximum wait time, in milliseconds, after which a rendition attempt will be aborted.

(**)

Copyright © 2013-2022 AnswerModules Sagl

Please refer to the third-party rendition engine's guide for a detailed explanation of all the available command line parameters

rend¶

Installation (Windows)¶

External conversion engine package is provided as a compressed archive **rend-win.zip**. The Archive contains following items:

- ***chromium*** – folder containing an up to date version of Chromium *(https://www.chromium.org/Home)* engine.
- **rend** – pre-built NodeJS application leveraging Puppeteer *(https://github.com/puppeteer/puppeteer)*

To install it:

- ☑ Extract the conversion engine package in the following location:

```
<OTCS_HOME>/module/anscontentscript_x_x_x/amlib/rend/dropin
```

Installation (Unix)¶

External conversion engine package is provided as a compressed archive **rend.tar.gz**. The Archive contains following items:

- ***chromium*** – folder containing an up to date version of Chromium *(https://www.chromium.org/Home)* engine.
- **rend** – pre-built NodeJS application leveraging Puppeteer *(https://github.com/puppeteer/puppeteer)*
- **run_rend** – a script that will be called by the Content Suite and will launch the application

To install it:

- ☑ Extract the conversion engine package in the following location:

```
<OTCS_HOME>/module/anscontentscript_x_x_x/amlib/rend/dropin

e.g.
>tar -C <OTHOME>/module/anscontentscript_x_y_0/amlib/rend/dropin -xvf rend.tar.gz
```

Note: files inside dropin folder should belong to user that is used to run Content Server service. Thus you can either perform extraction under the OTCS service user or change ownership of the extracted files accordingly.

Copyright © 2013-2022 AnswerModules Sagl

Configuration¶

- [x] Configure the Rendition extension package in order to use the ***rend*** executable in the Module Suite Base Configuration *(/administration/modulesuite/#base-configuration)*

Section ***rend***

Windows

Configuration Property	Configuration Property Value
amcs.rend.html2pdf.dropin	rend-win
amcs.rend.html2pdf.cmdline	"${source}" --cookie "${cookie}" -p "${destination}" --format A4 --marginBottom 100px --marginTop 120px --marginLeft 30px --marginRight 30px --scale 0.8 --viewport 1240x1754
amcs.rend.html2pdf.timeout	60000

Unix

Configuration Property	Configuration Property Value
amcs.rend.html2pdf.dropin	run_rend
amcs.rend.html2pdf.cmdline	"${source}" --cookie "${cookie}" -p "${destination}" --format A4 --marginBottom 100px --marginTop 120px --marginLeft 30px --marginRight 30px --scale 0.8 --viewport 1240x1754
amcs.rend.html2pdf.timeout	60000

Configuration Property	Configuration Proerty Meaning
amcs.rend.html2pdf.dropin	The relative path to the engine's executable. For security reasons, the root of this path is the extension package's dropin folder.

Copyright © 2013-2022 AnswerModules Sagl

Configuration Property	Configuration Proerty Meaning
amcs.rend.html2pdf.cmdline	The template of the command line instruction to be used when performing rendition ***(**)***. A few replacement tags can be used in this command line template. (a) ${source} : represent the absolute path for the input resource you want to render. Its value is automatically injected by the rendition extension package. Since the rendition extension package works on Content Script Resources, you do not have to worry about file system housekeeping. (b) ${destination} :represent the absolute path for the output resource, the engine is going to generate. Its value is automatically injected by the rendition extension package. Since the rendition extension package works on Content Script Resources, you do not have to worry about file system housekeeping. c ${cookie} : represent a local authentication cookie
amcs.rend.html2pdf.timeout	the default maximum wait time, in milliseconds, after which a rendition attempt will be aborted.

Dropin options

-"${source}" – replacement tag that will be substituted by the URL to the generated HTML Form. This argument is mandatory and not editable.

-ck, --cookie [cookie] – value will be replaced by replacement tag that corresponds to the current user's session cookie. Should be in form "Name Value". This argument is mandatory and not editable.

-p, --path \<path> – identifies target PDF file location. Value will be substituted by the replacement tag. This argument is mandatory and not editable.

-f, --format [format] – PDF option. Paper format. If set, takes priority over width or height options. Defaults to 'Letter'. Available options: Letter, Legal, Tabloid, Ledger, A[0-6].

-d – Debug is on. If specified debugging information is written to the log file. Use only for debugging purposes. Log file located in \<OTHOME>\logs\cs_rend.log or when running application manually in \<appDir>\log\cs_rend.log

-mb, --marginBottom [margin] - Bottom margin, accepts values labeled with units.

-mt, --marginTop [margin] - Top margin, accepts values labeled with units.

-mr, --marginRight [margin] - Right margin, accepts values labeled with units.

-ml, --marginLeft [margin] - Left margin, accepts values labeled with units.

-vp, --viewport [cookie] - PDF option. Set the viewport. Width and height of the page in pixels

-prt, --printmediatype - Use print media type. Boolean. Default: true.

-s, --scale [scale] - Scale of the webpage rendering.

Copyright © 2013-2022 AnswerModules Sagl

-dhf, --displayHeaderFooter - Display header and footer. Boolean. Default: false.

-ht, --headerTemplate [template] - HTML template for the print header.

-ft, --footerTemplate [template] - HTML template for the print footer.

-pb, --printBackground - Print background graphics. Boolean. Default: true.

-pr, --pageRanges - Paper ranges to print, e.g., '1-5, 8, 11-13'. Defaults to the empty string, which means print all pages.

-w, --width [width] - Paper width, accepts values labeled with units.

-h, --height [height] - Paper height, accepts values labeled with units.

-wu, --waitUntil [choice] - WaitUntil accepts choices load, domcontentloaded, networkidle0, networkidle2. Defaults to 'networkidle2'.

For more detailed description of the option please refer to official Puppeteer documentation *(https://pptr.dev/#?product=Puppeteer&version=v3.0.1&show=api-class-page)*

Content Script Extension for SAP¶

What is it?¶

Content Script Extensions for SAP allows to integrate Content Script with the SAP™ ERP through RFCs (Remote Functions Calls).

The integration allows you to perform the following:

- connect to multiple SAP™ systems through JCo APIs;
- invoke standard and custom SAP™ functions for retrieving ERP's information;
- invoke standard and custom SAP functions for updating ERP's information;

SAP™ JCo Library Required

This extension package requires the SAP™ JCo library *(https://support.sap.com/en/product/connectors/JCo.html)* to be available in the extension repository `<OTHOME>/module/anscontentscript_x_y_z/amlib/sap` and is certified for use with SAP™ JCo version (3.0.6) when used on OpenText Extended ECM and version (3.0.10) when used on CSP. SAP™ JCo library *(https://support.sap.com/en/product/connectors/JCo.html)* can be downloaded from SAP™ website.

Extension setup¶

The Content ^Scripting extension for SAP is part of the Module Suite bundle but is not included in the main Module Suite installer. The extension package is provided on request by the AnswerModule support team, the reason why it is not included in the main intaller is because, if not configured correctly, it could cause problems with the installation of Module Suite.

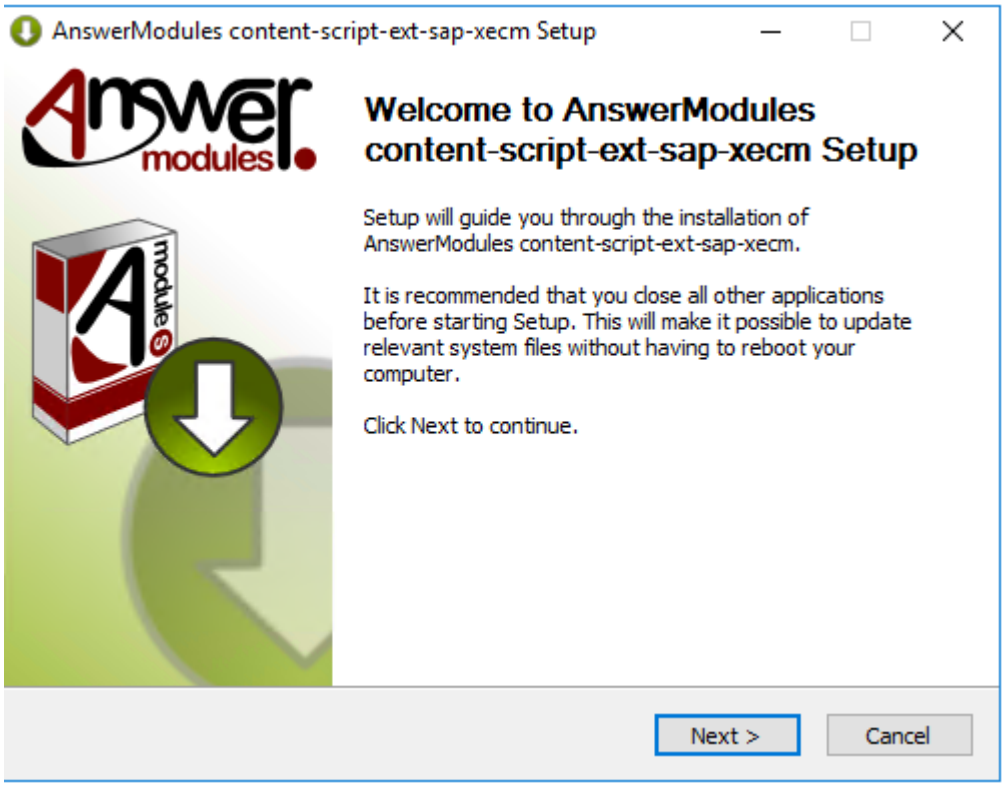
Copyright © 2013-2022 AnswerModules Sagl

Below is the step by step guide on how to install the Extensions for SAP. **Note**: For the general Module Suite and Module Suite Extensions Packages installation procedure please refer to "Installing the suite" *(/installation/installation/)* section

Installing the Content Script Extension for SAP¶

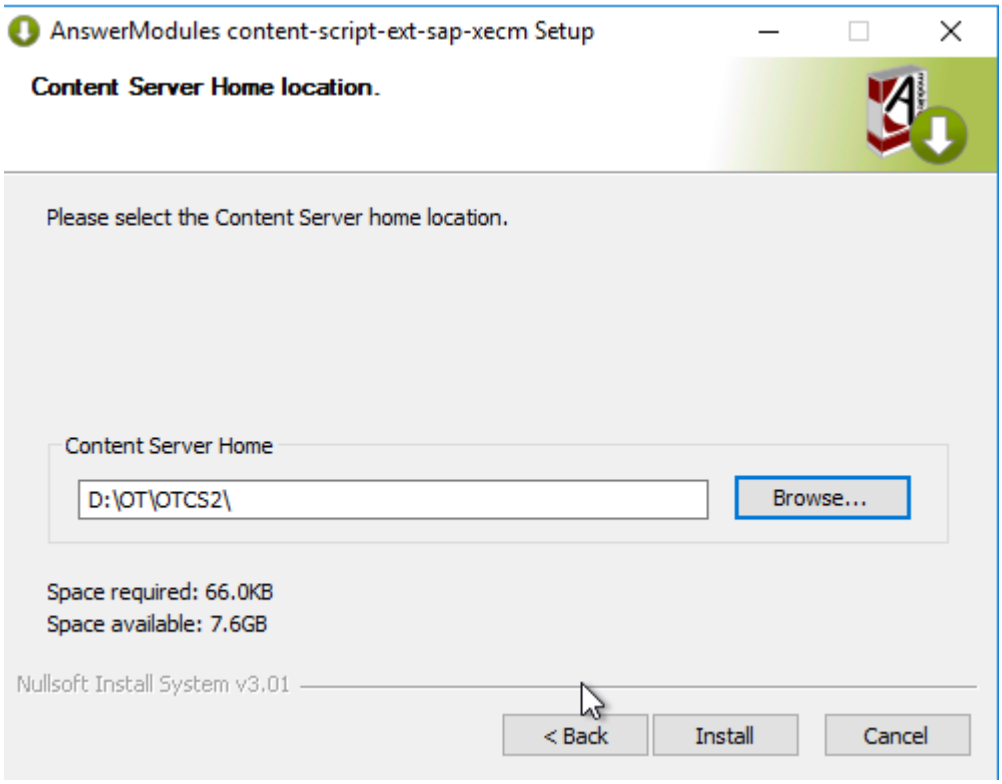
Run the Content Script SAP Extension installer and follow the installation wizard steps:

- Select "Next" when ready to start the installation.

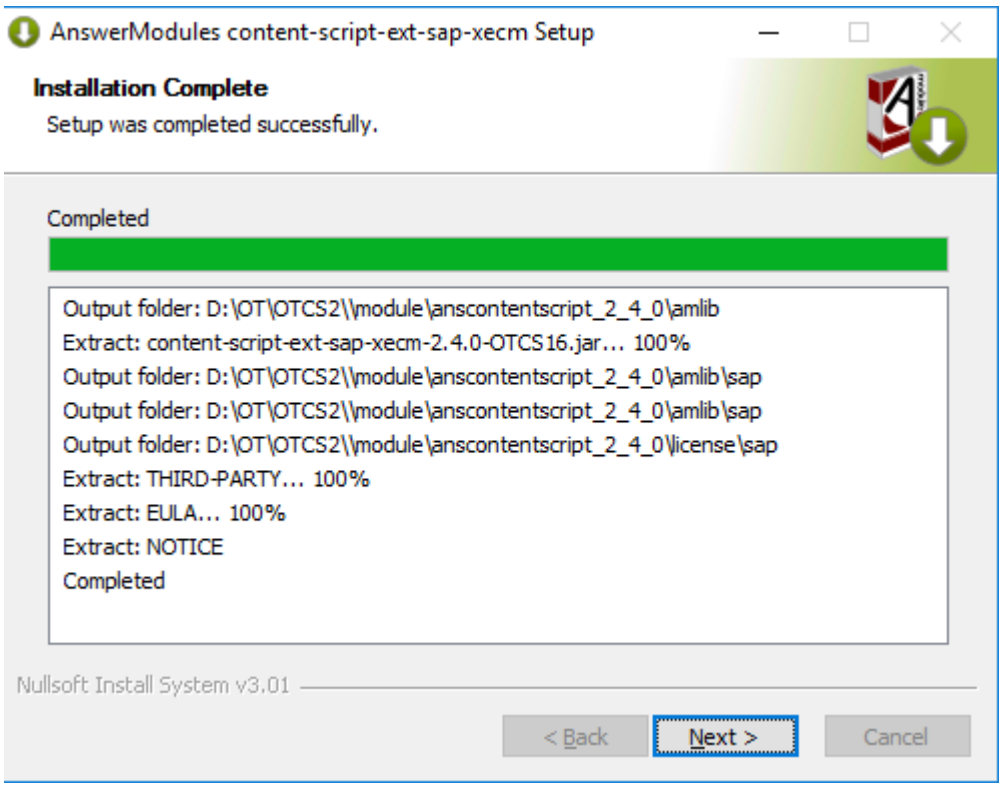


- Accept all the required license agreements
- The installer will prompt you for the location of the installed Content Server. Browse to your **OTCS_HOME** and select "Next".

Copyright © 2013-2022 AnswerModules Sagl



- Click "Install" to start the installation
- The installation of the required libraries will be performed



- Deploy SSAP™ JCo in the extension package repository: `<OTCS_HOME>/module/anscontentscript_2_x_0/amlib/sap`. The Content Script extension for SAP relies on SAP Java Connector (SAP JCo) to support outbound communication with the SAP Server. SAP JCo

Copyright © 2013-2022 AnswerModules Sagl

relies on a native bridge to implement the communication with the SAP server. This native bridge is implemented by the SAP JCo native library (sapJCo.dll). Both the SapJCo jar file and dll must be copied in the extension package repository.

To deploy SapJCo library follow this simple procedure:

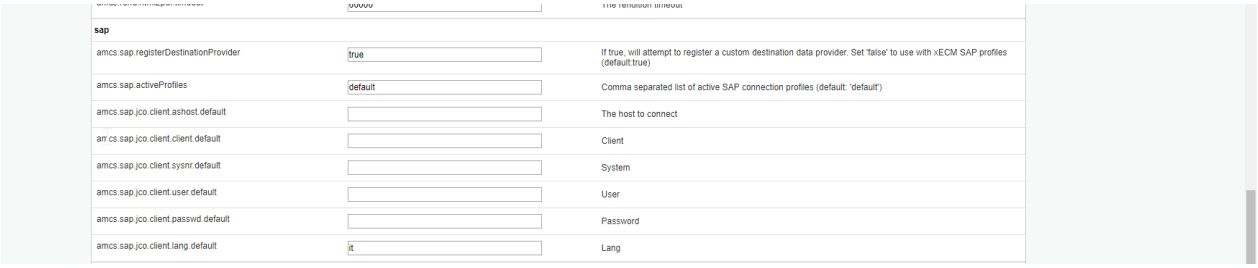
- Stop Content Server service
- Copy library files to the destination mentioned above
- Start Content Server service

Deploy on clustered environment

In case of a clustered Content Server installation the above steps should be performed on every cluster node.

Installation validation¶

If the Content Script Extension for SAP has been successfully installed, a new configuration section should appear in the Base Configuration *(/administration/modulesuite/#base-configuration)* page:



Configuration options¶

List of available parameters specified below:

Configuration Property	Configuration Property Meaning
amcs.sap.registerDestinationProvider	Determines whether the existing xECM connection or a custom connection should be used.When set to TRUE the custom destination data provider is used; when set to FALSE the existing configured SAP xECM connection is used.
amcs.sap.activeProfiles	List of the currently active and configured sap extension profiles. As many other extension packages Content Script Extension for SAP allows you to define multiple configuration profiles in order to manage multiple connections towards different systems.
amcs.sap.JCo.client.ashost.default	Target SAP System server hostname

Copyright © 2013-2022 AnswerModules Sagl

Configuration Property	Configuration Property Meaning
amcs.sap.JCo.client.client.default	Target SAP System Client number
amcs.sap.JCo.client.sysnr.default	Target SAP System ID
amcs.sap.JCo.client.user.default	Target SAP System username to logon with
amcs.sap.JCo.client.passwd.default	Target SAP System password for the specified username
amcs.sap.JCo.client.lang.default	Language to use for the connection

OpenText Activator

If you have not installed the "OpenText Activator for SAP Solutions" module on your system, you can only use the custom destinations. In this case it is necessary to install the SAP JCo version compatible with your environment.

Installing Extension for DocuSign

Prerequisites¶

This guides assumes the following components to be already installed and configured:

- AnswerModules ModuleSuite
- Script Console (*OPTIONAL - only for DocuSign webhook configuration*)

The following information will be required to complete the configuration procedure:

- DocuSign API key
- Docusign API credentials

Authentication Options

The Content Script extension supports two different authentication options when invoking DocuSign APIs:

- Username / Password
- Account GUID / RSA Certificate

Refer to the official DocuSign REST API guides *(https://developers.docusign.com/esign-rest-api/guides/building-integration/)* for details on how to generate your credentials.

We will refer to the Content Server installation directory as OTCS_HOME

We will refer to the Script Console installation directory as SCRIPT_CONSOLE_HOME

Copyright © 2013-2022 AnswerModules Sagl

Installation procedure¶

The **Module Suite DocuSign Extension** includes two components:

- Content Script Extension for DocuSign

  *This component enables the **docusign* service API in Content Script. The service is the entry point to integrating DocuSign functionality within your applications.**

- Script Console Extension for DocuSign (*Optional*)

  *This component enables a **DocuSign webhook endpoint** on Script Console. It is only required if you want to receive automatic update notification from DocuSign whenever an envelope status changes. For more details, refer to the official DocuSign REST API Guides (https://developers.docusign.com/esign-rest-api/code-examples/webhook-status) related to this topic.**

Installing the Content Script Extension for DocuSign¶

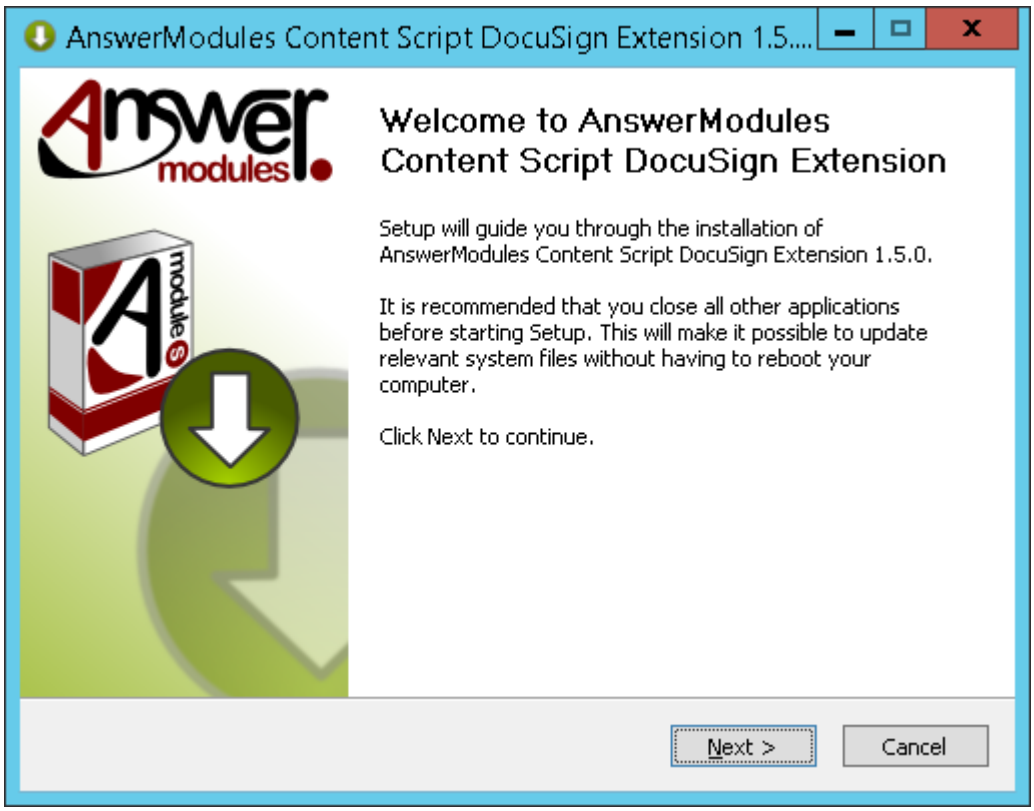
Run the Module Suite DocuSign installer:

```
module-ansmodulesuitedocusign-1.5.0-OTCSxxx.exe
```

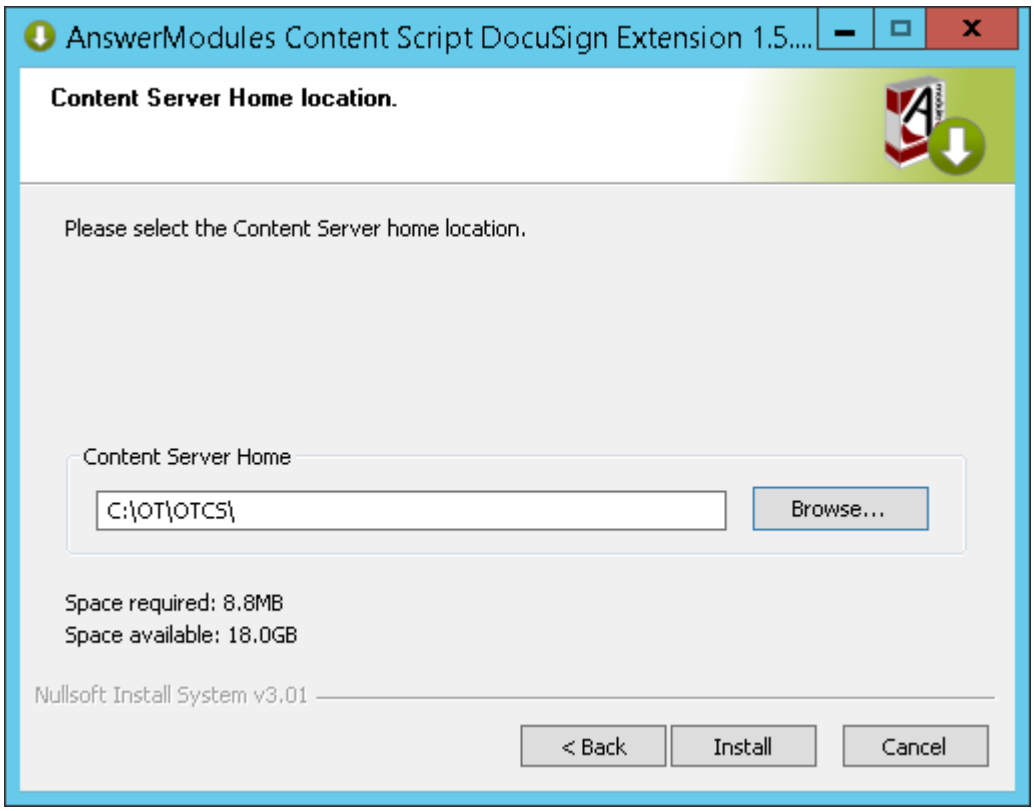
Follow the installation wizard steps:

- [ ] Select "Next" when ready to start the installation.

Copyright © 2013-2022 AnswerModules Sagl

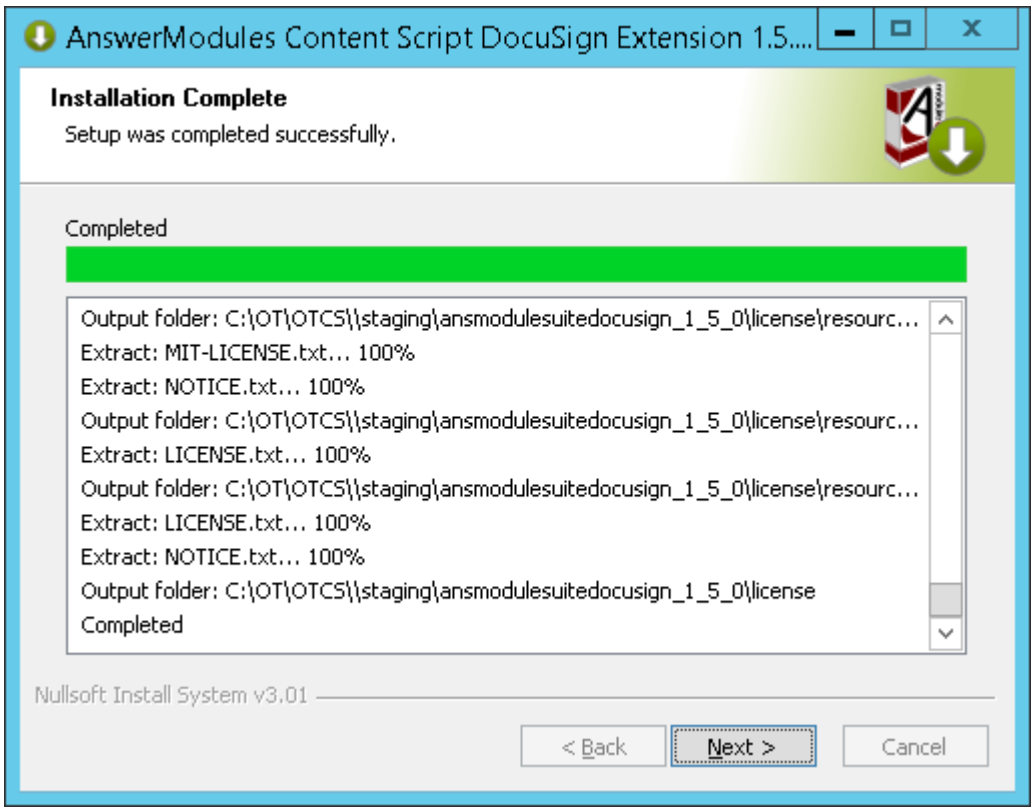


☐ The installer will prompt you for the location where Content Server is installed. Browse to your **OTCS_HOME** and select "Next".

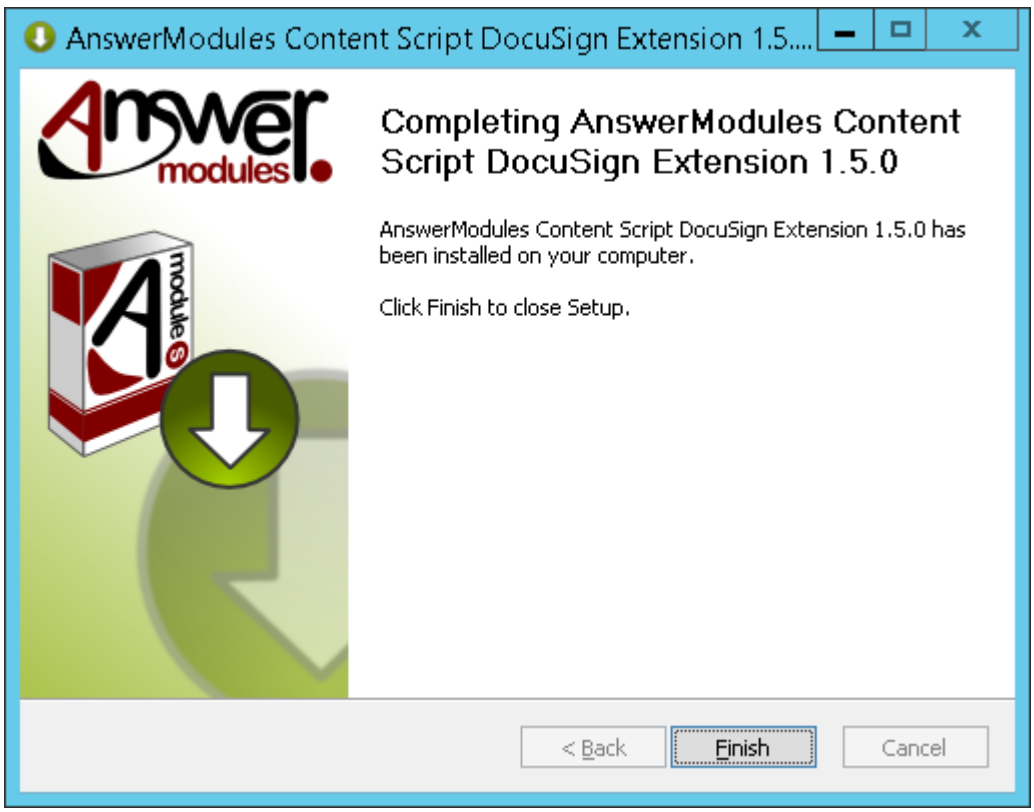


☐ Review the installation steps for each component to be installed.

Copyright © 2013-2022 AnswerModules Sagl



- [ ] Click "Finish" to complete the unpacking of the module

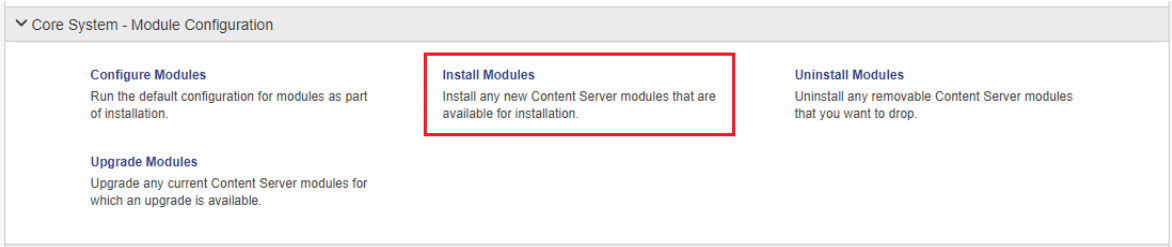


Copyright © 2013-2022 AnswerModules Sagl

Staging

At this point, the Module has been deployed in the Content Server Staging folder and is available for module install through the Content Server administration pages.

- [ ] Access the Content Server Admin pages > **Core System - Module Configuration** > **Install Modules**



- [ ] Locate the **AnswerModules Module Suite extension for Smart UI** module and proceed with installation

- [ ] Restart the OTCS services when prompted in order for the installation to be completed.

Installing the Script Console Extension for DocuSign (OPTIONAL)¶

Run the Script Console DocuSign Extension installer:

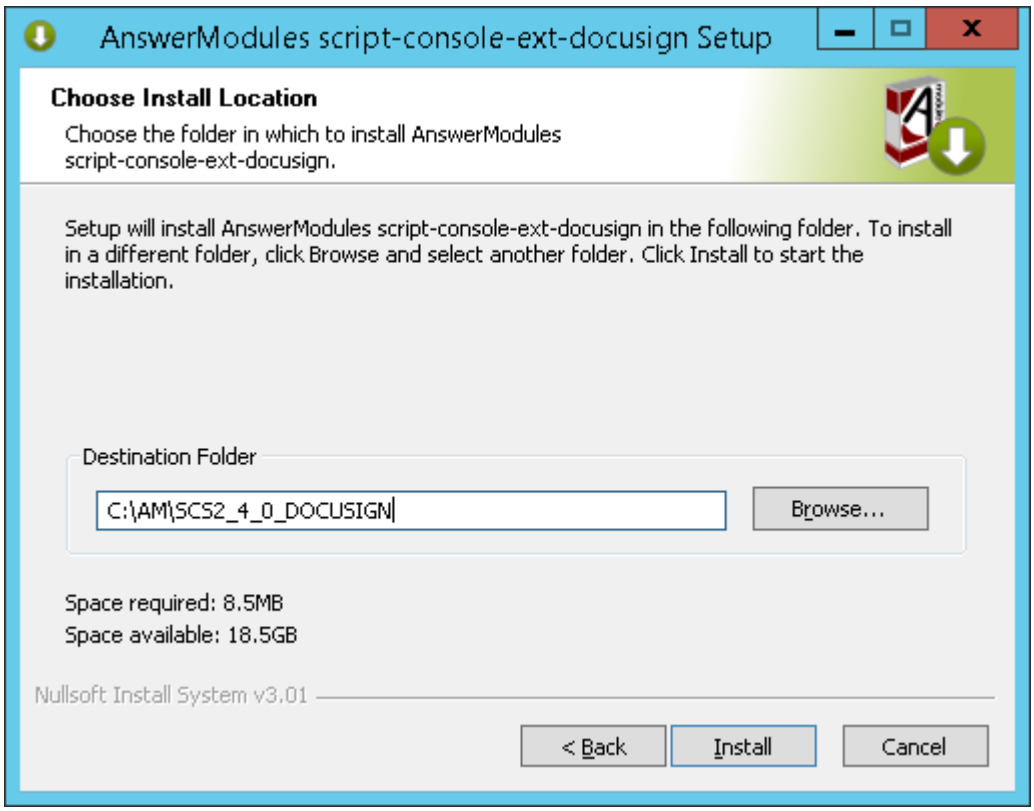
```
script-console-ext-docusign-2.4.0-OTCSxxx.exe
```

Follow the installation wizard steps

- [ ] Select "Next" when ready to start the installation.

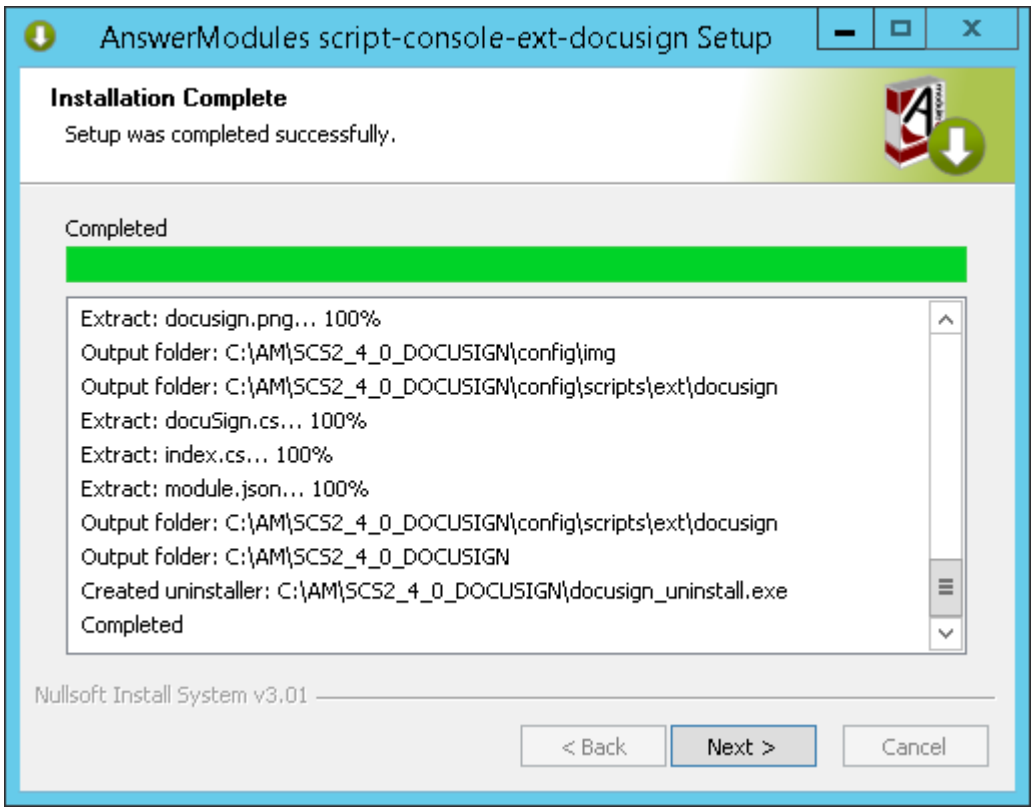
Copyright © 2013-2022 AnswerModules Sagl

☐ The installer will prompt you for the location where your target Script Console instance is installed. Browse to your SCRIPT_CONSOLE_HOME and select "Next".

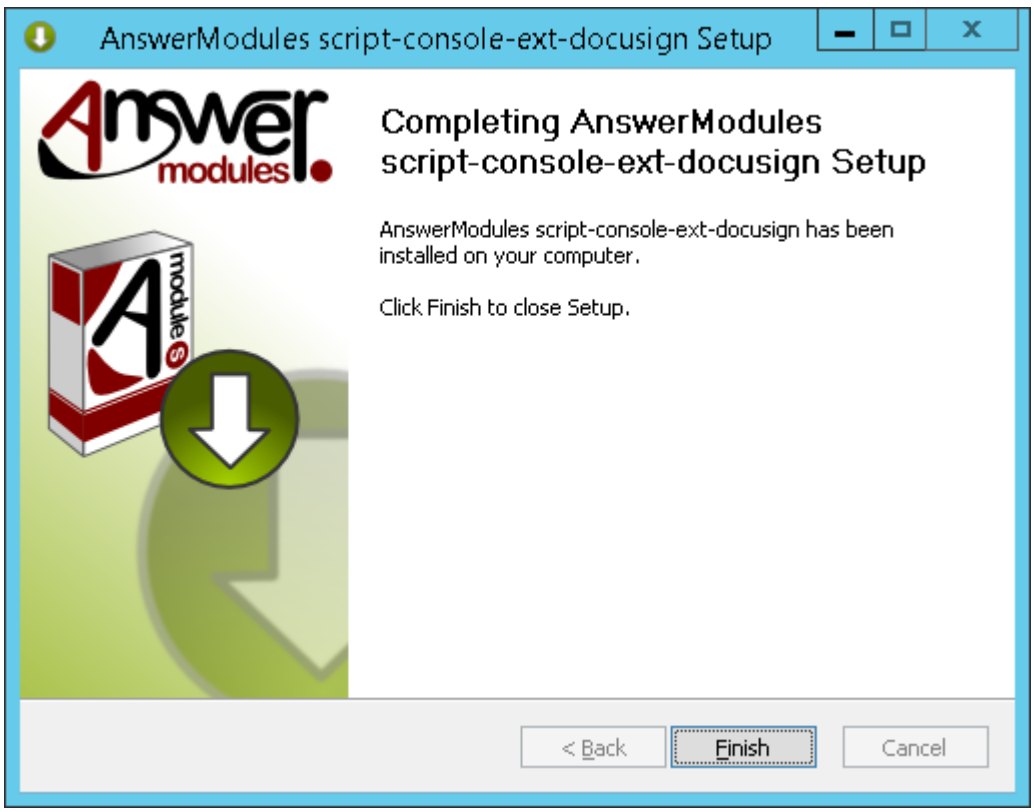


☐ Review the installation steps for each component to be installed.

Copyright © 2013-2022 AnswerModules Sagl



- [ ] Click "Finish" to complete the installation



Update the security configuration to allow access to the webhook endpoint. Edit the Script Console security config file:

Copyright © 2013-2022 AnswerModules Sagl

```
<SCRIPT_CONSOLE_HOME>\config\cs-console-security.xml
```

Add the following rule:

```
<s:http pattern="/ext/docusign/docuSign.cs" security="none"/>
```

Configuration¶

The DocuSign Connector requires a few configuration parameters in order to be able to communicate with DocuSign systems using the eSignature REST APIs.

In the OTCS Admin pages > AnswerModules Administration > Base Configuration section, complete the "docusign" API configuration.

The following parameters are available:

Key	Description
amcs.docusign.activeProfiles	Comma separated list of active DocuSign Accounts profiles (default: "*default*"). This is a local identifier and will not be sent over to DocuSign. It is only relevant when more than one set of configurations has to be specified.
amcs.docusign.appKey.default	DocuSign Integration Key: identifies your app for the DocuSign platform.
amcs.docusign.authUser.default	DocuSign Account GUID or Username
amcs.docusign.authServer.default	DocuSign authentication endpoint. This can be either **account-d.docusign.com** for sandbox testing or **account.docusign.com** for a production account.

Copyright © 2013-2022 AnswerModules Sagl

Key	Description
amcs.docusign.appSecret.default	DocuSign Account Password or RSA Certificate. If an Account GUID has been provided in the "*amcs.docusign.authUser.default*" field, than this MUST be an RSA Certificate private key. Otherwise, if a Username has been provided, this MUST be the account password.
amcs.docusign.appBasePath.default	DocuSign Integration Base Path. This can be either https://demo.docusign.net/restapi *(https://demo.docusign.net/restapi)* for sandbox testing or https://www.docusign.net/restapi *(https://www.docusign.net/restapi)* for a production account.
amcs.docusign.notifURI.default	DocuSign Notification WebHook URI. This is the **absolute**, **publicly accessible URL** that DocuSign will call for push notifications. It refers to the endpoint installed on your Script Console instance. This value is **OPTIONAL** and only required if using the push notifications.

RSA Certificate format

If using the RSA certificate authentication (combined with an account GUID), the following requirements must be met:

- RSA Certificate must be stored on a single line.
- Line breaks must be replaced with line feeds (*\n*).
- The *"-----BEGIN RSA PRIVATE KEY-----"* block and *"-----END RSA PRIVATE KEY-----"* must be included.

Example:

```
-----BEGIN RSA PRIVATE KEY-----\nxxx....xxx\nxxx....xxx=\n-----END RSA PRIVATE KEY-----\n
```

Save the Base Configuration and restart Content Server services when requested

Admin dashboard¶

The **Module Suite DocuSign Extension** supports the storage of a local copy of the signing envelope details within Content Server. The envelope status can either be periodically updated through a scheduled job, or automatically updated using push notifications by DocuSign (using a webhook pattern). An overview of the status of current and past envelopes can be visualized using the DocuSign Connector Admin dashboard.

The dashboard is a Content Script based tool that can be installed in the Content Script Volume using the Module Suite import/upgrade tool.

Copyright © 2013-2022 AnswerModules Sagl

Before running the import, you should make the lib file available to the tool with the following steps:

- [ ] On the server, navigate to the DocuSign Extension Module folder

```
<OTCS_HOME>\module\ansmodulesuitedocusign_1_5_0\library
```

and locate the file named **docusign integration.lib**.

- [ ] Copy the file to the **library** folder within the Content Script Module:

```
<OTCS_HOME>\module\anscontentscript_2_4_0\library
```

Now that the library is available, proceed to the import with the following steps:

- [ ] In a web browser, open the Module Suite Administration Base Configuration page. If working in a clustered environment, make sure you connect to the same server on which the library file has been copied.

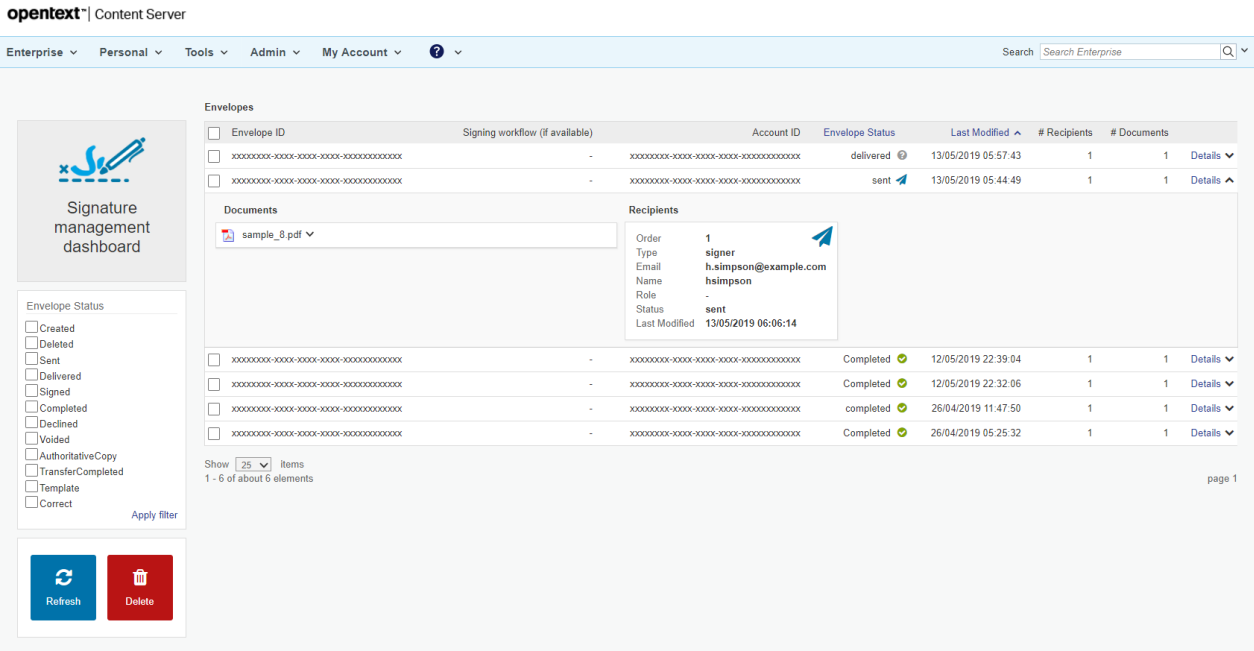
- [ ] Use the "*Import*" tool within the base configuration to import the **DocuSign Integration** library

Once the import is complete, you will be able to access the dashboard by navigating to the following Content Server location:

```
Content Script Volume > DocuSign Integration > CSTools
```

and running the **Dashboard** script.

Copyright © 2013-2022 AnswerModules Sagl



Applying HotFixes

Module Suite hotfixes are typically distributed in the form of compressed file archives (.zip files).

The content of the archive is a folder structure that mirrors the structure of the Content Server installation directory (e.g. "E:\Opentext" or "/opt/opentext/otcs").

Below an exemplar structure of an "hotfix" archive:

```
├ module
|    ├ anscontentscript\_X\_Y\_Z
|    ├ amlib
|    ├ ...
|    ├ ...
|    ├ hotfixes
|    |  └──── hotFix_ANS_XYZ_###.hfx
|    ├ ...
|    ├ ojlib
|    └ ...
|
└ support
     ├ anscontentscript
     ├ ...
     ├ amui
     |  └──── js
     |        └────...
     └ ...
```

Naming convention

Copyright © 2013-2022 AnswerModules Sagl

AnswerModules hotfixes follow a simple naming convention: they are all preceded by **hotFix_ANS_** followed by an optional string that identifies the AnswerModules product (e.g. **DS** for DocuSign Connector) (if absent the hotfix must be consider for Module Suite) followed by three digits identifying the version of the AnswerModules product followed by three digits identifying the hotfix followed by an optional string that identifies the OpenText Content Suite version the hotfix is compatible with.

e.g.
hotFix_ANS_240_001.zip
: Hotfix 001 for Module Suite version 2.4.0
hotFix_ANS_DS_150_002_CS16.zip
: Hotfix 002 for DocuSign Connector version 1.5.0 to be utilized on Content Server version 16.0.X
hotFix_ANS_SMUIEXT_150_001.zip
: Hotfix 001 for AnswerModules Smart View extension version 1.5.0
cumulative_hotFix_ANS_240_CS16X_009_024
: Cumulative hotfix (containing hotfixes from 009 to 024) for Module Suite version 2.4.0 to be utilized on Content Server 16.0.X

Hotfixes deployment¶

To install an hotfix the files provided in the hotfix archive must be deployed within the Content Server installation directory in order to overwrite existing files and/or to add new files to the AnswerModules product binaries.

The suggested procedure for installing an hotfix is the following:

- ☑ Extract the archive in a temporary folder;
- ☑ Read the patch installation notes carefully. The installation notes come in the form of a text file ending with **.hfx** located within the *module/anscontentscript_x_y_z/hotfixes* folder. The installation notes contains information about the issues addressed by the hotfix and any additional deployment instructions to follow;

cumulative hotfix

In case of a cumulative hotfix, carefully read all the hotfixes installation notes.

- ☑ Check the contents of the archive and backup all files in installation folder of the Content Server that will be overwritten by the hotfix;

Unless otherwise instructed by the hotfix installation notes:

- ☑ Stop the Content Server services;
- ☑ Copy the contents of the hotfix in the Content Server installation directory or follow hotfix's more specific instructions for deployment;
- ☑ Restart the Content Server services

Important notes

- Always read the hotfix notes before deploying the hotfix. Some hotfixes require additional operations to be performed before or after deploying the binaries;

Copyright © 2013-2022 AnswerModules Sagl

- Always perform a backup of the patched binaries;
- Make sure that the version of the hotfix matches exactly the version of the target AnswerModules product and OpenText Content Suite environment.
- Hotfixes are identified by a progressive numbering. It is imperative that hotfixes are deployed respecting the **correct sequential order**, as it is possible that the same resources are patched by different hotfixes (e.g. ***hotFix_ANS_260_002.zip*** (progressive number: 2) **must not** be installed after ***hotFix_ANS_260_003.zip*** (progressive number: 3). If, for any reason, an hotfix has been skipped and has to be later installed on a system, all subsequent hotfixes **must be reinstalled** in order to ensure that no newer change has been reverted
- When OpenText Content Suite is running on a clustered environment, hotfixes must be installed on all the servers on which Content Suite is deployed.

Upgrading Module Suite

Upgrading from a previous version¶

Whenever a new release of Module Suite is released, it is highly recommended for customers to update their installation. New releases not only contains fixes for the identified bugs, but most importantly new features that might open new usage scenarios for your Module Suite applications. Updating Module Suite is quite a straight forward procedure, that should take between 15 to 45 minutes (depending on how complex your Content Server architecture is). The system down time is limited to the two restarts required for each node.

We will refer to the Content Server installation directory as **%OTCS_HOME%**

Upgrading the primary node¶

This section only applies to your primary cluster node, or to non-clustered environments (single box). Please see section "Upgrading a secondary node" for the update procedure on secondary nodes.

- Deploy the Content Script Module and the Beautiful WebForms Module to the staging folder (Install Content Script, Install Beautiful WebForms).
- Install the desired Content Script extension packages for the newer version (Install Content Script extension packages)

Do not complete the installation procedure

For both the two procedures mentioned above stop at the step: **Select "Install Modules"**

- Perform the modules upgrade using the standard Content Server tools
    1. Login as Administrator and access the Module administration panel

Copyright © 2013-2022 AnswerModules Sagl

2. Select “Upgrade Modules”
3. From the available modules, select “Content Script X.Y.Z”
4. Follow the installation steps and re-start Content Server when prompted.
5. Return to the Module administration. In the “Update Modules” panel, select “Beautiful WebForms X.Y.Z”
6. Follow the installation steps but ***do not restart Content Server***

- Stop Content Server
- Apply relevant hot fixes
- Start Content Server
- From the Administration Home, select **AnswerModules Administration > Base Configuration**, then if necessary, change the core configuration or the configuration of the extension modules. ***Additionally, make sure you apply the new License Key in the relevant configuration entry***.

Saving the Base Configuration is required even if no changes have been applied to the configuration page, since some of the values (typically configuration options of newly installed extension packages) have to be saved at least once. Additionally, saving the Base Configuration is necessary to the licensing process.

- Using the import and upgrade tool perform the library upgrade

What do I need to upgrade ?

How the library upgrade works¶

The 'Upgrade' operation will rename the existing library folders in the Content Script volume, and import a new version of the same (the only exception is the 'CSFormTemplates' folder, which will be discussed later). As such, any modification that has been applied to one of the libraries will be relocated and no longer available.

Examples include:

- any custom Beautiful WebForms components added to the CSFormSnippets folder
- any custom Rest API endpoints added to the CSServices folders
- any callbacks configured in the CSEvents or CSSynchEvents folders
- any Classic UI modifications applied through the CSMenu, CSAddItems, CSBrowseView, CSBrowseViewColumns
- any other object created or modified within one of the upgraded folders

As part of the upgrade operation, you should identify such changes and make sure they are ported to the new libraries.

CSFormTemplates have a slightly different upgrade process. Since objects in this folder are referenced by object DataID (their unique identifier on OTCS) they can't be replaced with the updated version, since this would potentially cause issues in any existing form using the template. For this reason, the upgrade process for

Copyright © 2013-2022 AnswerModules Sagl

CSFormTemplates automatically updates each single template by adding a new version to the object, thus preserving the original DataID. For this reason, no "backup" folder will be found for CSFormTemplates..

- Cleanup. The folders named “Backup-_yyyyMMdd-AAAAAA” are backup folders containing the previously installed library scripts/snippets. They can be safely exported and removed

In case any of the standard components were customized, patched or otherwise modified, or new custom components were added within the standard library, make sure that you transfer any relevant changes to the new libraries before deleting the old version.

Script Console

Script Console can be upgraded performing a so-called "parallel" upgrade, which means installing on the same/different server the newer version of the console and configure it as the previous one. This typically requires to copy over the relevant configuration files from the previous Script Console together with any custom script you might have created/deployed on the console: %SCHOME%/config/cs-console-schedulerConfiguration.xml, %SCHOME%/config/cs-console-security.xml %SCHOME%/config/cs-console-systemConfiguration.xml

Upgrading a secondary node¶

This procedure only applies to cluster secondary nodes. The primary node should always be upgraded first: the steps below assume that the primary node has already been upgraded. To upgrade the primary node (or if your environment is a non-clustered environment) please refer to section “Upgrading the primary node”. Upgrading the secondary nodes in a cluster has the purpose of re-aligning the installed modules and support folders with the primary node in the cluster. Any cluster-wide configuration or update (for example, the library upgrade procedure) is not necessary on secondary nodes as it has already been executed on the primary node.

The following steps should be executed on each secondary node

- Make a copy of the following resources from the primary node, and make them available in a working folder on the secondary node:
  1. %OTCS_HOME%/module/anscontentscript_x_x_x
  2. %OTCS_HOME%/module/ansbwebform_x_x_x
  3. %OTCS_HOME%/support/anscontentscript
  4. %OTCS_HOME%/support/ansbwebform
- Stop Content Server’s services
- Move the old module and support folders to a backup location
- Copy the new folders in place of the old folders
- **Reconcile the opentext.ini** (with particular reference to the [Modules] section) file in **%OTCS_HOME%/config**

Copyright © 2013-2022 AnswerModules Sagl

Start the Content Server's services

Uninstalling Module Suite

Uninstallation procedure¶

We will refer to the Content Server installation directory as **%OTCS_HOME%**

Before proceeding with the uninstallation of Module Suite modules you need to complete some housekeeping routines. These routines are not strictly mandatory and should only be performed if you do not intend to reinstall the Module Suite on your system in the future.

- ☐ Shutdown CSEvents feature:

  This feature generates records in the Distributed Agent framework table, which are then managed by the **CallbacksManagerCS** handler. After uninstalling the Content Script module this type of handler will not be longer available, with the result that several errors will be generated in the DA framework's tables. To prevent these errors from occurring, it is safer to disable the feature completely and wait for all occurrences of this type of activity to be processed by the DA.

  - From the Administration Home, select **AnswerModules Administration > Base Configuration**, then enter **34** in the **amcs.core.debugEnabled** property and save the current configuration.

RESTART REQUIRED

A service rest of all the nodes that are part of your cluster is required.

amcs.core.debugEnabled is now 'Module Suite - Configuration Options'

In recent version of Module Suite the property **amcs.core.debugEnabled** has been associated with the label **Module Suite - Configuration Options** in the Base Configuration

  - Once all the nodes have been restarted wait until all the occurences of CallbacksManagerCS jobs have been processed and removed from the DA table. You can monitor this process by executing the query below:

```
select count(1) as "Total", 'WorkerQueue' as "Queue" from  WorkerQueue where Handle
union all
select count(1) as "Total", 'WorkerQueuePending' as "Queue" from  WorkerQueuePendin
union all
select count(1) as "Total", 'WorkerQueueCurrent' as "Queue" from  WorkerQueueCurren
```

Copyright © 2013-2022 AnswerModules Sagl

- [ ] Delete all Content Server's columns or facets having a Content Script script as their datasource.

- [ ] Stop and delete all instances of worklows using Module Suite modules. Upon Module Suite uninstallation all the currently active workflows, which make use of a feature related to one of the Module Suite modules, will not be able to continue correctly, to avoid errors you must wait for these workflows to end or stop and delete them.

> **Modify Workflow Map**
>
> Remove any Content Script Step, Content Script Workpackage, Content Script Event Script from all your Workflow Maps

- [ ] Stop any scheduled script

  - From the Administration Home, select **AnswerModules Administration > Manage Content Script Scheduling** unschedule any previously scheduled Content Script script.

  - Wait the completion of any previously scheduled script execution. You can monitor this process by executing the query below:

```
select count(1) as "Total", 'WorkerQueue' as "Queue" from  WorkerQueue where Handle
union all
select count(1) as "Total", 'WorkerQueuePending' as "Queue" from  WorkerQueuePendin
union all
select count(1) as "Total", 'WorkerQueueCurrent' as "Queue" from  WorkerQueueCurren
```

- [ ] (OPTIONAL) Collect and delete all the Content Script, Smart Pages, and Beautiful WebForm Views Object objects on your system.

  - Although not strictly necessary, this action will prevent you from having objects on your system that the application can no longer handle correctly. In order to easily find collect and delete the afore mentioned objects we suggest you to create and execute the script below, which it will create in the same container where the script was created a collection containing all the scripts pages and views in your system.

```
collection = docman.createCollection(self.parent, "Module Suite Objects", "Module S
/*
43100 BWF Views
43200 Content Script
43300 SmartPages
*/
nodes = docman.getNodesFastWith(sql.runSQLFast("""select distinct DataID "DataID" f
collection.addNodes(nodes)
```

> **Execute the script as Admin**

Copyright © 2013-2022 AnswerModules Sagl

Don't forget to create and run the above script as an "Admin" user to make sure you can collect all objects on your system regardless of the associated permissions.

- [ ] (OPTIONAL) Delete the Content Script Volume and its content.
    - Although not strictly necessary, this action will prevent you from having objects on your system that the application can no longer handle correctly. From the Administration Home, select **AnswerModules Administration > Open The Content Script Volume** once in the volume delete the volume's content.
- [ ] Delete Beautiful WebForm SmartEditor table.
    - From the Administration Home, select **AnswerModules Administration > Base Configuration** then click on the link **DELETE** under the **Manage Beautiful WebForms database** section. The action will require confirmation.
- [ ] Using standard Content Server features uninstall all the Module Suite modules

Uninstallation complete

The Module Suite is no longer on your system. We miss you already.

Copyright © 2013-2022 AnswerModules Sagl

Content Script

Content Server object

Content Script objects are document-class objects on Content Server.

Content Scripts are **restricted** objects: as such, users must be **enabled** to the creation of new objects through the Administration pages.

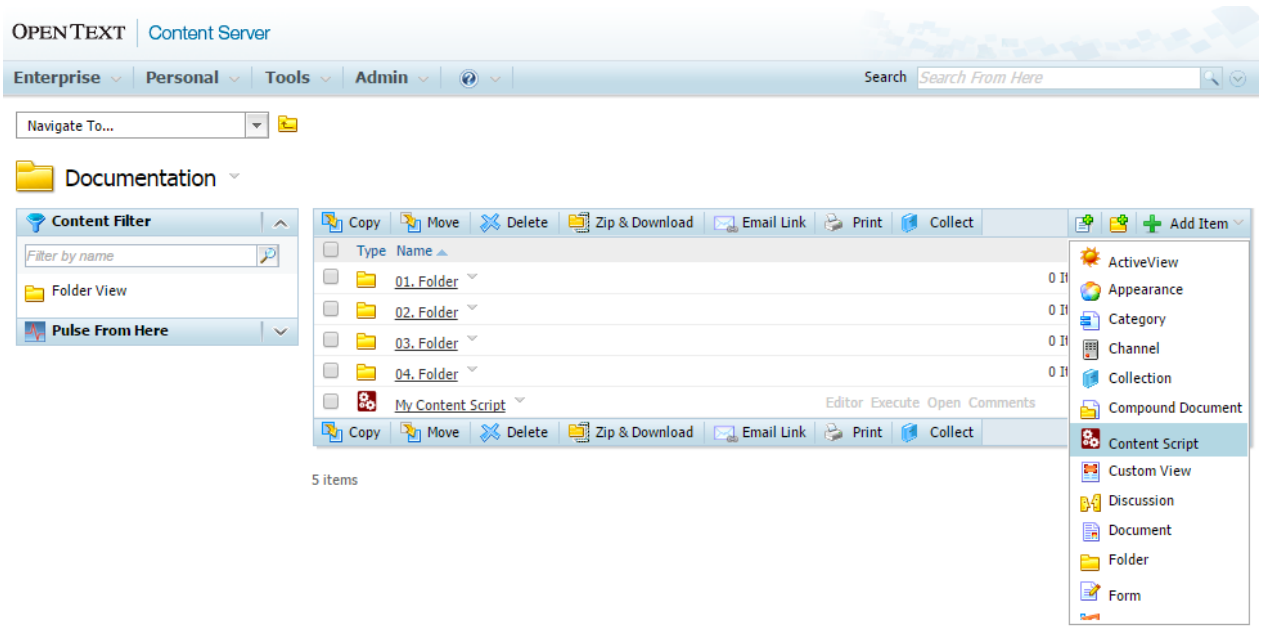
Content Scripts are executable objects, and the **execution** is the default action associated to the object.

Being standard objects, Content Scripts comply with Content Server **permissions** model. Make sure you assigned the proper permissions to your scripts.

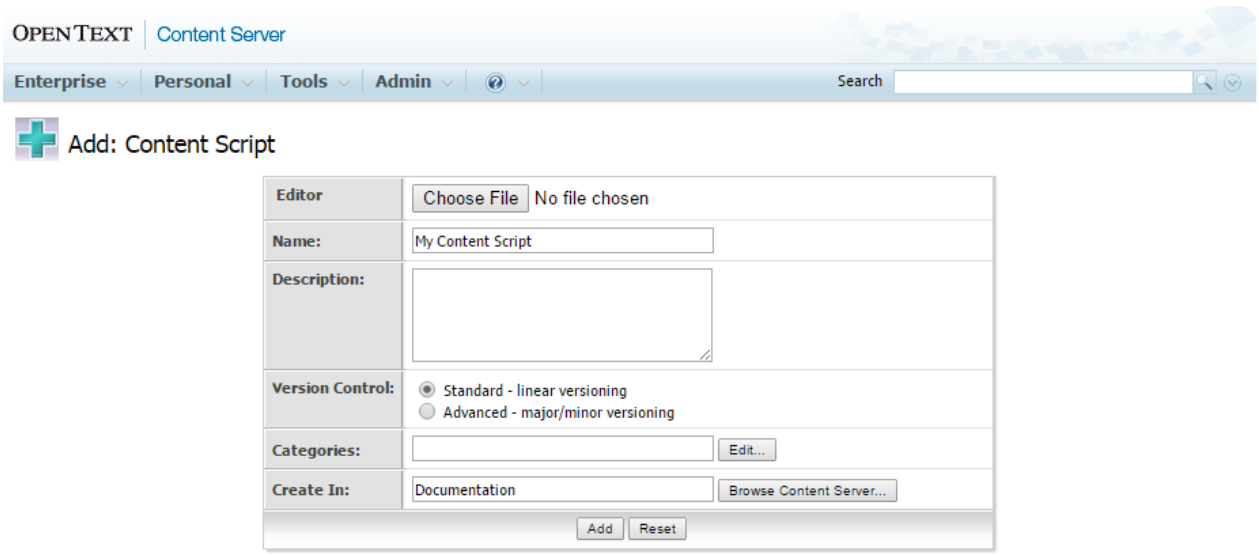
Upon creation, the object can be edited with the web-based IDE selecting the **'Editor'** function in the object function menu. The function is also available as a promoted function.

Creating a Content Script¶

To create a new Content Script object you can leverage the standard **add item menu**:



Copyright © 2013-2022 AnswerModules Sagl



Object's properties¶

This section covers the following topics:

- Content Script Static variables
- Scheduling a Content Script
- Running a Content Script as a different user
- Changing the default GUI icon for a Content Script object

Static variables¶

For every Content Script, it is possible to define a set of static, precompiled variables whose values will be available when the script is executed.

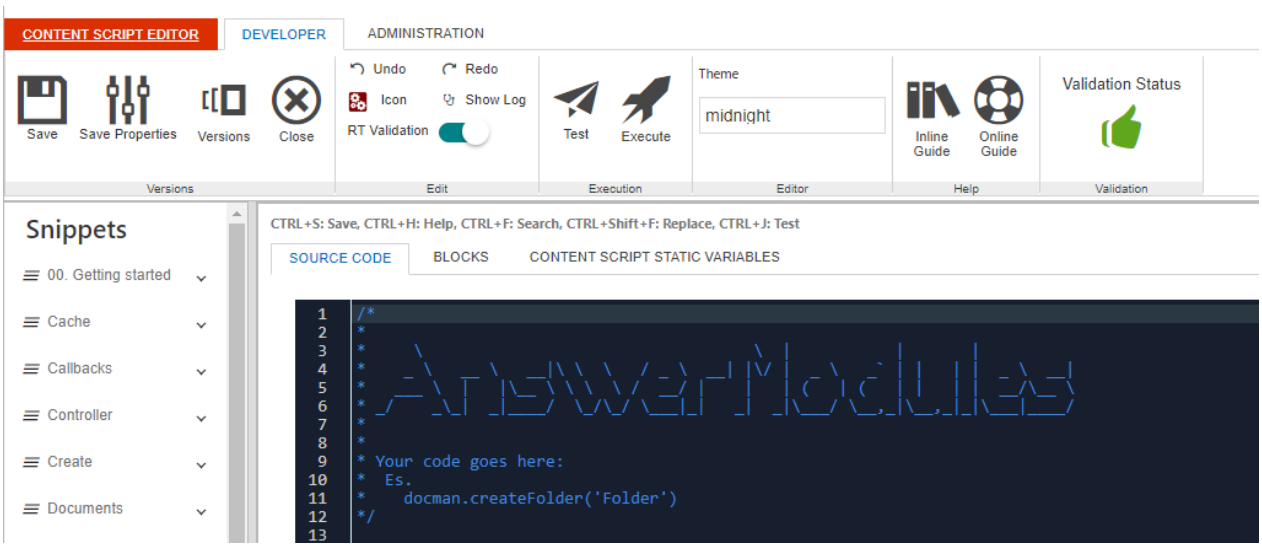
The framework supports the definition of these variables by means of a second script, whose outcome is the data map containing the values. For performance reasons, this second script is executed only when it changes (or when execution is explicitly forced by an editor), and the results are stored as part of the script object.

One of the reasons for having a script to define a static variable (instead of explicitly setting the value of the variable itself) is code portability: instead of defining the value of a variable, it is possible to define a rule to calculate that value. A typical example would be the Object ID of an object located in a specific position in Content Server: in case the code is moved to a different environment, the ID would be recalculated automatically.

Static variables are accessible within the Content Script through the **csvars** object.

> Each Content Script object has its own csvars constants. In complex applications, that include multiple Content Script objects, it is often useful to have all constants defined in one single file. This can be done by creating a Content Script dedicated to be the “constants” script, that will be run by the single scripts in the application to load the variable values in the context.

Copyright © 2013-2022 AnswerModules Sagl



Scheduling¶

Content Script supports the automatic execution of scripts through its internal **scheduler**.

The Content Script **scheduling utility** is available from within the **Specific > Advanced Settings** tab or from the Content Script Editor in the **Administration** tab (if visible). The utility allows to schedule the automatic execution of Content Scripts (that is, without the need for a user to trigger the execution explicitly).

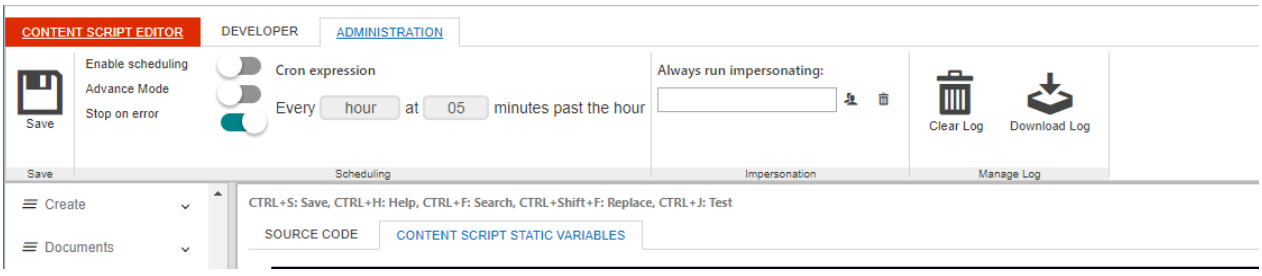
The scheduling is configured by means of a **cron expression**. A cron expression is a string comprising a set of fields separated by spaces, and identifies a set of times.

Cron expressions are powerful but can also be quite complex. For this reason, a simplified configurator with drop-down menus can be used to create the desired cron expression.

Skilled users can always flag the **“Advanced Mode”** checkbox to disable the configurator and compose their own expressions.

Once ready, the scheduler can be enabled by flagging the **“Enable Scheduling”** checkbox.

It is possible to stop a script from being rescheduled in case of execution errors. To do so, simply flag the **“Stop on Error”** checkbox.



Where is the log ?

Copyright © 2013-2022 AnswerModules Sagl

Content Script scheduling takes advantage of the Content Server's Distributed Agent framework. While normally executing a script will cause it to be run in the current front end server, a scheduled script could actually be executed on any server on which Distributed Agents are activated.

Impersonate¶

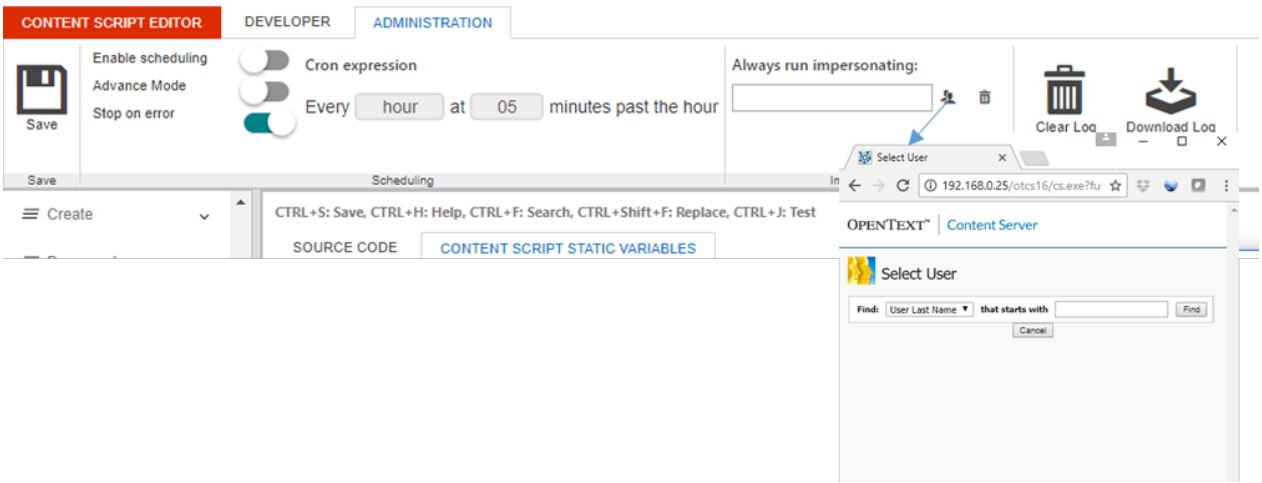
Content Script supports the execution of a script **impersonating** specific users.

This configuration applies for both:

- scripts explicitly executed by users
- scripts executed by the system (scheduled, workflow steps, callbacks, etc...)

The **"Run As"** configuration panel is accessible within the **Specific > Advanced Settings** tab or from the Content Script Editor in the **Administration** tab (if visible).

In order to be able to perform a "**Run As**" configuration, a user must have impersonation privileges.

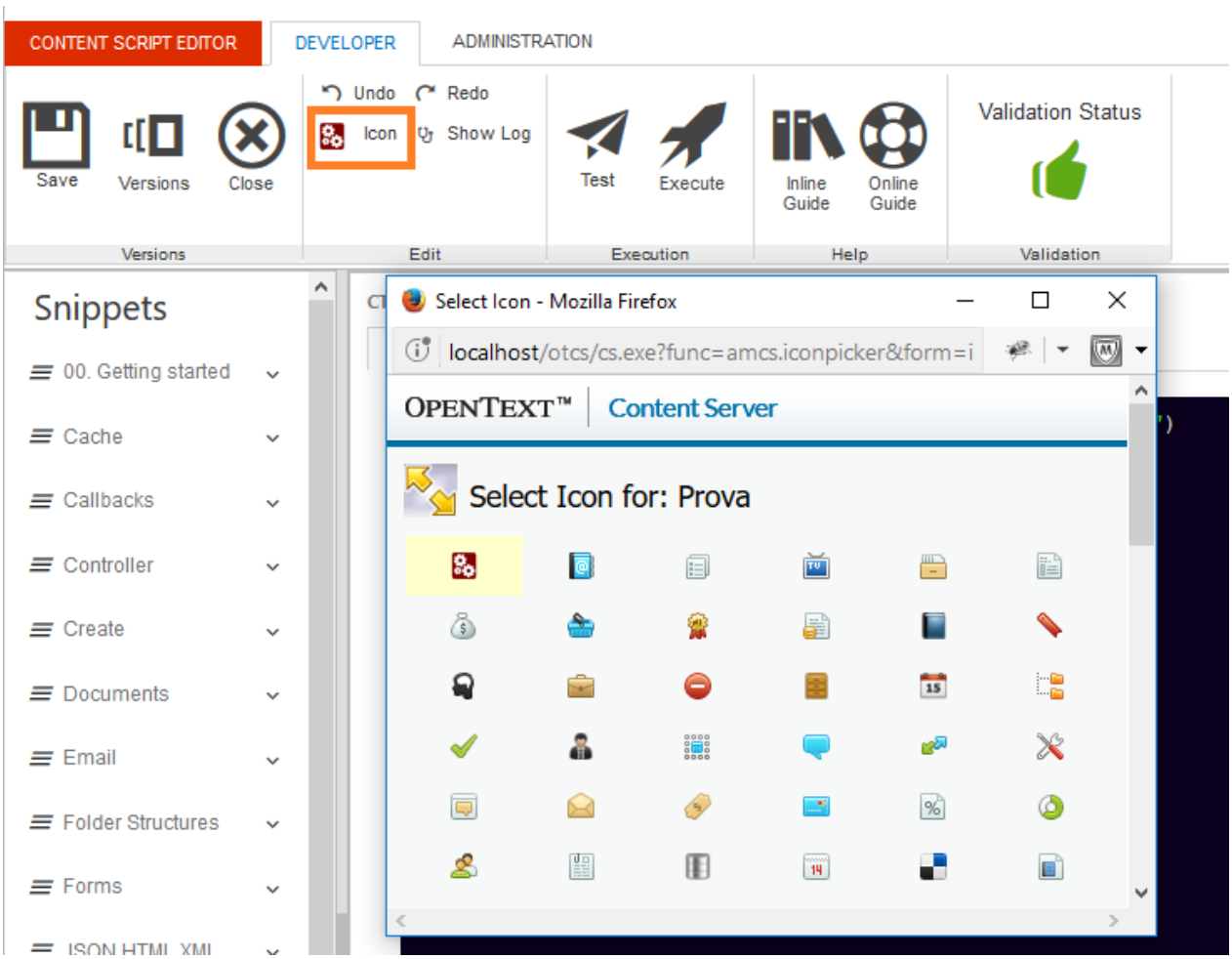


Icon Selection¶

Given the flexible nature of Content Script objects (both in terms of behaviour and execution outcome) it is often useful to be able to distinguish them at-a-glance. One way is to customize the default **icon** used by Content Server for the object.

The desired icon can be selected by clicking on the icon button on the Content Script's Developer tab and selecting a specific icon within a set of available icons.

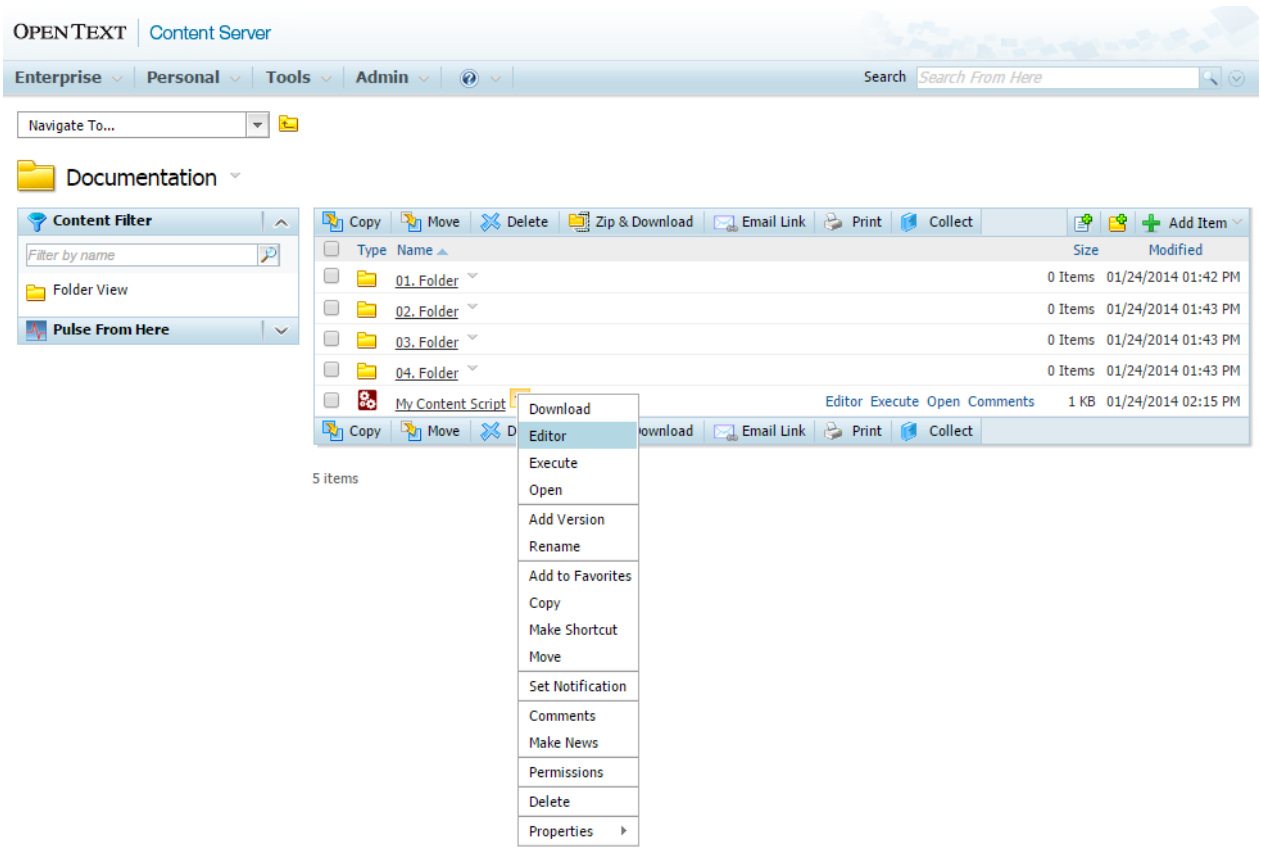
Copyright © 2013-2022 AnswerModules Sagl



Content Script editor

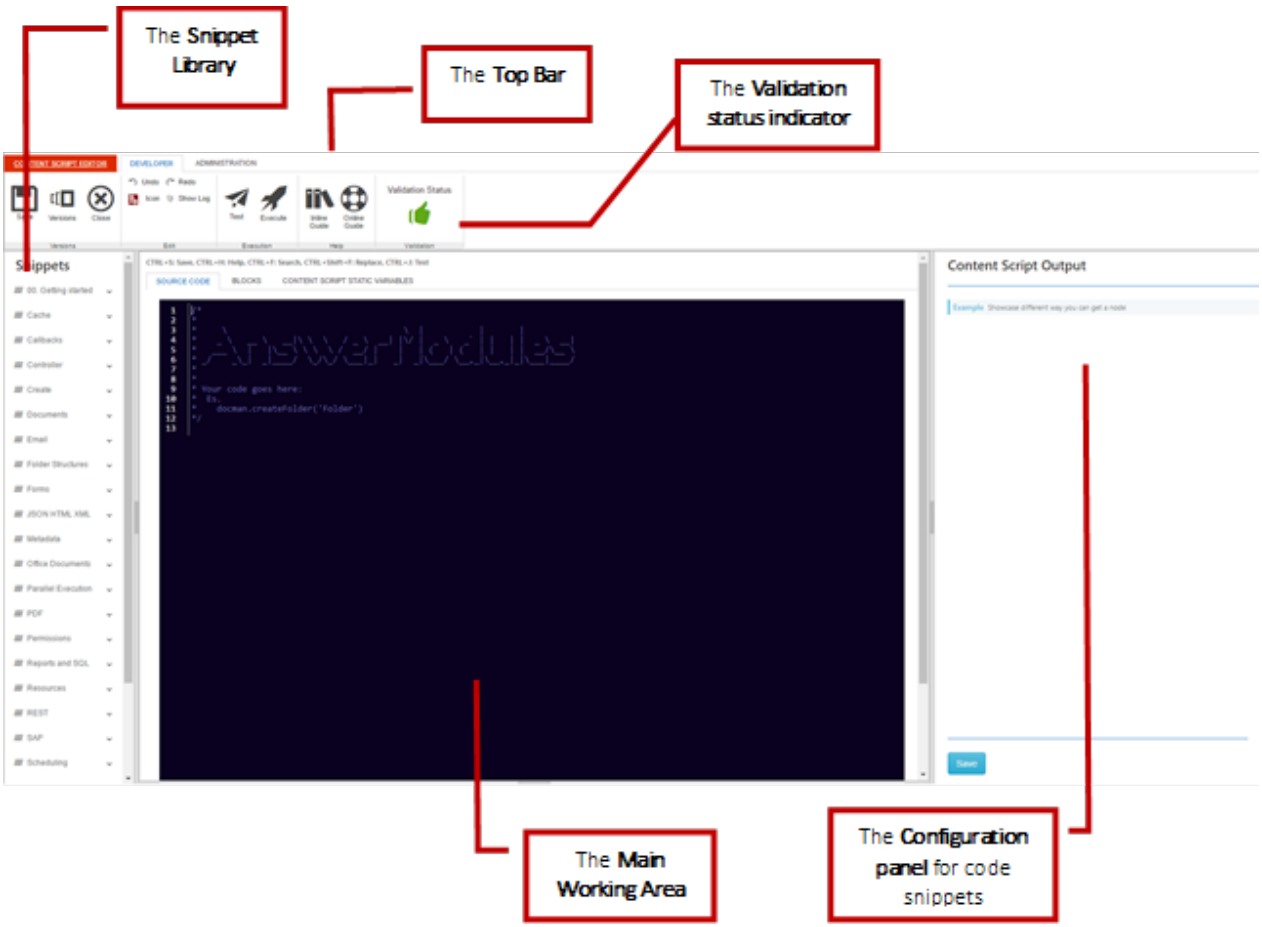
Content Script objects can be edited with the dedicated web-based IDE selecting the **'Editor'** function in the object function menu. The function is also available as a promoted function.

Copyright © 2013-2022 AnswerModules Sagl



The web-based IDE (Integrated Development Environment) for Content Script appears as follows:

Copyright © 2013-2022 AnswerModules Sagl



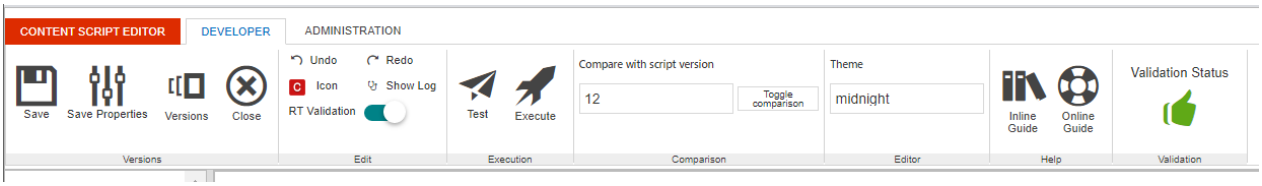
Shortcuts¶

The following keyboard shortcuts are available while using the editor:

Shortcut	Description
Ctrl + S	Save the current script (add a new version)
Ctrl + H	Toggle the online Help window
Ctrl + F	Open the 'Search' tools panel
Ctrl + Shift + F	Open the 'Search and Replace' tools panel
Ctrl + Space	Show the code autocompletion hints
Ctrl + J	Trigger the execution in the test frame
Ctrl + P	Inject the full path of the selected node in the Content Script editor

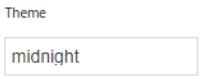
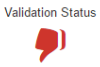
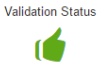
Copyright © 2013-2022 AnswerModules Sagl

Top Bar controls (DEVELOPER)¶

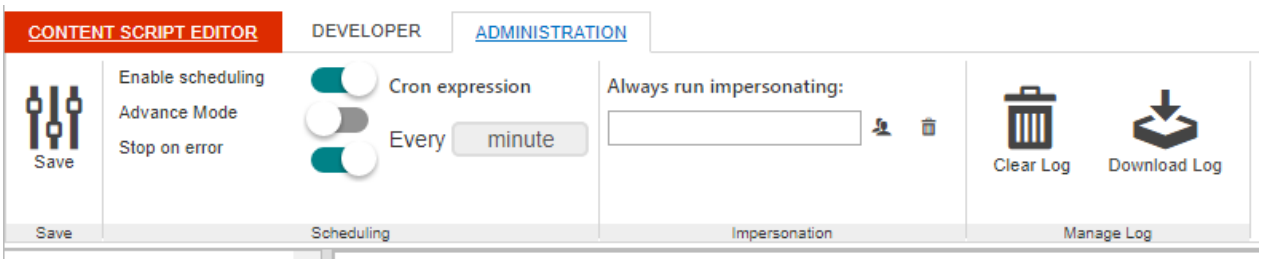


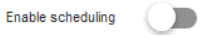
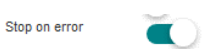
Command	**Description**
Versions	
Save	**Save** the script (adds a new version)
Save Properties	**Save** the content script properties (i.e. the icon) as well as the static variables (does not add a new version)
Versions	Open the object's **Versions** tab
Close	Close the Content Script Editor
Edit	
Undo	**Erases** the last change done
Redo	Opposite of Undo
Icon	Change the script's associated icon
Show Log	Display the last 200 lines of the ModuleSuite'smaster log file
RT Validation	Disable the script's real-time validation
Execution	
Test	**Run** the script in the current window (CTRL + J)
Execute	**Save** the script and **run** it, showing the result in the editor's bottom panel
Comparison	
Toggle comparison	**Toggle comparison**: toggles the comparison of the current version of the script with the selected.

Copyright © 2013-2022 AnswerModules Sagl

Command	**Description**
Editor	
	**Theme**:Changes the theme applied to **all** the RPA embedded editors
Help	
	Access the module's online **guide** and the **support portal**
Validation	
	**Red label**: The script failed the validation and most likely will **fail** to compile
	**Green label:** The script is **well-formed**

Top Bar controls (ADMINISTRATOR)¶



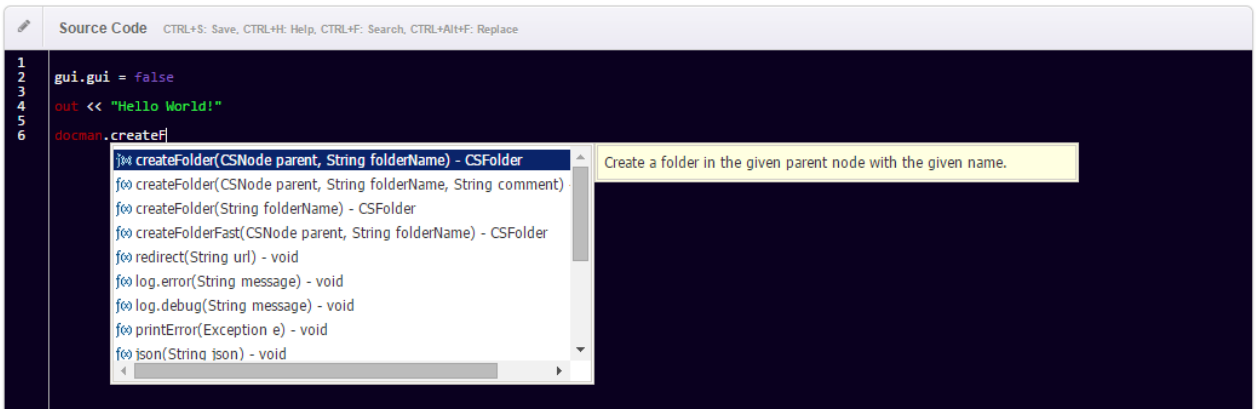
Command	**Description**
Versions	
	Save the content script properties (i.e. the icon) as well as the static variables (does not add a new version)
Scheduling	
	**Toggle** script scheduling
	Toggle script advance scheduling mode
	**Toggle** re-scheduling abortion on script's execution error
Impersonation	
	Select the user that will be always used to run the script
	Clear impersonation setting

Copyright © 2013-2022 AnswerModules Sagl

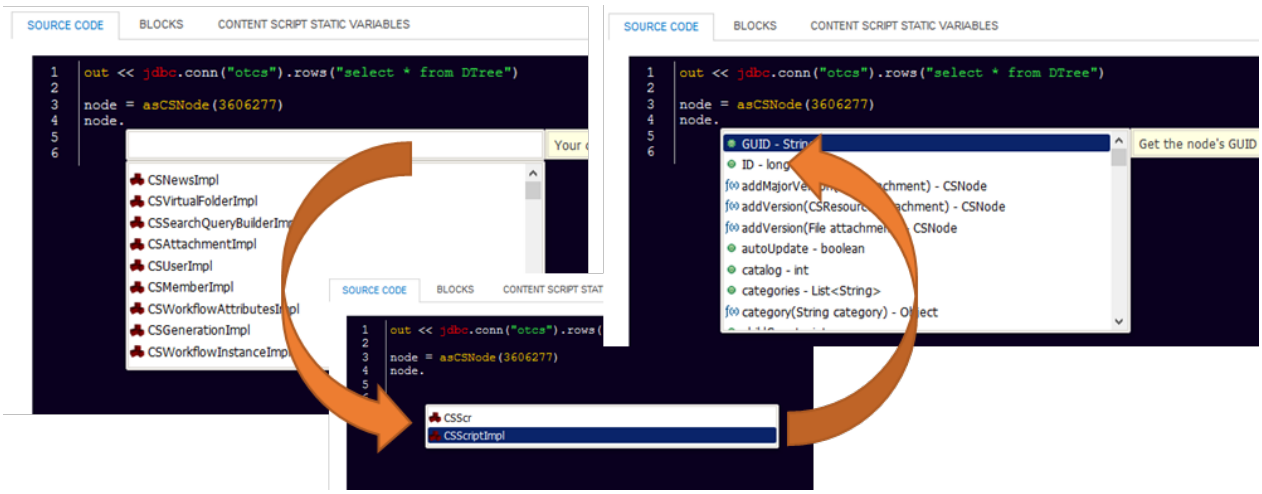
Command	**Description**
Manage Log	
Clear Log	**Trigger** ModuleSuite's master log rotation
Download Log	**Trigger** ModuleSuite's master log download

Auto-completion¶

The Content Script Editor features a code completion assistant functionality. While typing use the **ctrl + space** key combination to retrieve the suggested values.



In some cases the Content Script's inference engine might not be capable of determining the actual type of the expression you are trying to auto-complete. In these cases the auto-complete feature will prompt you to firstly specify the type against which the auto-completion should be performed and then will switch to the standard behaviour.



If the actual type (class) of your expression is not listed among the results you can still specify the fully qualified class name to autocomplete against that class: e.g. (java.lang.String)

Copyright © 2013-2022 AnswerModules Sagl

List of the most common API objects returned by Content Script APIs

Content Script API Objects			
ACSBrowseViewRowProvider	CSMemberImpl	CSRMRecordTemplate
AMBWFWidgetsLib	CSMemberPrivilegesImpl	CSRMRecordTraits
AdlibJobResult	CSMemberRightImpl	CSRMUserFunctions
CSANSTemplateFolderImpl	CSMenu	CSRMXReference
CSAssignmentImpl	CSMenuItem	CSReportImpl
CSAttachmentImpl	CSMilestoneImpl	CSReportResultImpl
CSBeautifulWebFormViewImpl	CSMilestoneInfoImpl	CSResourceImpl
CSBrowseViewAddItemButton	CSNewsBuilderImpl	CSScriptImpl
CSBrowseViewColumn	CSNewsImpl	CSSearchQueryBuilderImpl
CSBrowseViewMultiItemButton	CSNodeAuditDataPageImpl	CSSearchResultImpl
CSBrowseViewRow	CSNodeAuditRecordImpl	CSSetAttributeImpl
CSCategoryFolderImpl	CSNodeImpl	CSShortcutImpl
CSCategoryImpl	CSNodePageImpl	CSSpreadsheet
CSCategoryTemplateImpl	CSNodeResultImpl	CSSubMenu
CSChannelImpl	CSNodeRightImpl	CSTaskBuilderImpl
CSCollectionImpl	CSNodeRightsImpl	CSTaskGroupImpl
CSCompoundDocImpl	CSPDFFormField	CSTaskGroupInfoImpl
CSCompoundDocReleaseImpl	CSProjectImpl	CSTaskImpl
CSDiscussionImpl	CSProjectInfoImpl	CSTaskInfoImpl
CSDiscussionItemImpl	CSProjectPartecipantsImpl	CSTaskListImpl
CSDocumentImpl	CSProjectRoleUpdateInfoImpl	CSTaskListInfoImpl
CSEmailImpl	CSRMClassification	CSUnreadInfoImpl
CSEmailMessage	CSRMClassificationTypes	CSUrlImpl
CSExportOptionsImpl	CSRMField	CSUserImpl
CSFTPFile	CSRMFieldsInfo	CSVersionImpl
CSFolderImpl	CSRMHold	CSVirtualFolderImpl
CSFormImpl	CSRMHoldDistribution	CSWebReportImpl
CSFormTemplateDefinitionImpl	CSRMHoldDoc	CSWordDoc
CSFormTemplateImpl	CSRMHoldPage	CSWorkPackageImpl
CSGenerationImpl	CSRMProvenance	CSWorkflowAssignedTaskImpl
CSGroupImpl	CSRMRSIRetention	CSWorkflowAttachmentsImpl
CSImportOptionsImpl	CSRMRecord	CSWorkflowAttributesImpl
CSWorkflowAuditRecordImpl	CSWorkflowCommentsImpl	CSWorkflowFormDataImpl
CSWorkflowInstanceImpl	CSWorkflowMapImpl	CSWorkflowQueryBuilderImpl
CSWorkflowSearchHandleImpl	CSWorkflowStartDataImpl	CSWorkflowFormsImpl
CSWorkflowTaskActionsImpl	CSWorkflowTaskCommentImpl	CSWorkflowTaskDetailsImpl
CSWorkflowTaskImpl	CSWorksheet	FTPConfigProfile
FieldInfo	Form	GCSAdlibJob
GCSCategory	GCSTableOfContents	GCSWatermark
LDAPConnection	NodeListRowProvider	PDFOverlayText
PDFWaterMark	SQLQueryRowProvider	SampleContextAwareObject
SampleObject	SearchResultRowProvider	SinglePageRowProvider

Language basics

Content Script is a **Domain-Specific Programming Language (DSL)** for OpenText Content Server.

The language is based on **Oscript** and exposes a **Groovy** interface to developers. **Groovy** (http://www.groovy-lang.org/ *(http://www.groovy-lang.org/)*) is a widespread dynamic language for the Java Virtual Machine, particularly indicated for the creation of DSLs.

Copyright © 2013-2022 AnswerModules Sagl

Content Script language syntax is fully compatible with Groovy.

Under the hood, a mix of **Oscript** and **Java** features allow for a deep integration with Content Server functionalities, as well as for an extreme ease of **integration with external systems**.

The following sections are meant to be an introduction to the language.

Statements¶

The definition of variables can be either generic or restricted to a specific type. Assigning a value to a variable that does not match its type will force the engine to attempt to cast its value to the given type. In case no conversions can be done, it will result in an error.

```
// Defining a local variable can be done either by
// 1) declaring explicitly its type
// 2) using the "def"
int anInt = 1

String aString = "text"

def anObject = "anything"
anObject = 123
```

With String variables, a few useful tricks are available:

```
// Strings can be defined both with quotes ('') or double quotes ("")
String aString = "text"
String anotherString = 'text'

// Selecting the alternative "" or '' can be useful if quotes are present in the string content
String aQuote = "this is a quote: 'My words...' "
String anotherQuote = 'this is a quote: "My words..." '

// using triple """ allows to span across multiple lines for string
// definition. Useful for readable SQL queries, for example..
String multilineString = """SELECT *
                            FROM DTREE
                            WHERE DATAID = 2000"""

Lists and Maps can be defined very easily
def aList = ["firstElement", "secondElement"]

def aMap = [firstKey:"firstValue", secondKey:"secondValue"]

// statements can span across multiple lines
def multilineDefinition = ["firstElement",
                            "secondElement"]

// collections can contain different kinds of elements
def aMapWithStringsAndInts = [first:"one", second:2
```

Copyright © 2013-2022 AnswerModules Sagl

Basic Control Structures¶

Below are the basic structures for flow control and iteration

- **if – else** statement
- **if – else if – else** statement
- **inline if** statement
- **switch** statement
- **while** loop
- **for** loop

Flow control: if – else¶

```
if(a == b){
    //do something
} else {
    //do something else
}
```

Flow control: if - else if - else¶

```
if(a == b){
    // do the first thing
} else if(c == d){
    // do a second thing
} else {
    // do something else
}
```

Flow control: inline if - else¶

```
a = (b == c) ? "c is equal to b" : "c is different from b"
```

Flow control: switch¶

```
switch ( a ) {
    case "a":
        result = "string value"
        break

    case [1, 2, 3, 'b', 'c']:
        result = "a mixed list of elements"
        break

    case 1..10:
```

Copyright © 2013-2022 AnswerModules Sagl

```
        result = "a range"
        break

    case Integer:
        result = "is an Integer"
        break

    case Number:
        result = "is a Number"
        break

    default:
        result = "default"
}
```

Looping: while¶

```
def a = 0

while (a++ < 10){
    // do something ten times
}

def b = 10

while ( b-- > 0 ) {
   // do something ten times
}
```

Looping: for¶

```
// Standard Java loop
for (int i = 0; i < 5; i++) {

}

// range loop
for ( index in 0..100 ) {
    // do something
}

// list or array loop
for ( index in [0, 10, 20, 40, 100] ) {
    // do something 5 times
}


// map looping
def aMap = ['first':1, 'second':2, 'third':3]

for ( entry in aMap ) {
    // do something for each entry (the values can be accessed and used)
    entry.value
}
```

Operators¶

All Groovy operators can be used in Content Scripts:

Copyright © 2013-2022 AnswerModules Sagl

Operator Name	Symbol	Description
Spaceship	<=>	Useful in comparisons, returns -1 if left is smaller 0 if == to right or 1 if greater than the right
Regex find	=~	Find with a regular expression
Regex match	==~	Get a match via a regex
Java Field Override	.@	Can be used to override generated properties to provide access to a field
Spread	*.	Used to invoke an action on all items of an aggregate object
Spread Java Field	*.@	Combination of the above two
Method Reference	.&	Get a reference to a method, can be useful for creating closures from methods
asType Operator	as	Used for groovy casting, coercing one type to another.
Membership Operator	in	Can be used as replacement for collection.contains()
Identity Operator	is	Identity check. Since == is overridden in Groovy with the meaning of equality we need some fallback to check for object identity.
Safe Navigation	?.	returns nulls instead of throwing NullPointerExceptions
Elvis Operator	?:	Shorter ternary operator

Methods and Service Parameters¶

Methods on objects can be called using the dot "." followed by the method signature and parameter clause.

```
out << template.evaluateTemplate("""
#@csform(false, "Submit")
    <label for="myFile">File to be uploaded</label>
    <input type="file" name="myFile"  />
#end
""")

if(params.myFile && params.myFile.filelength){
    def parentNode = docman.createFolder("MyFolder")
    def file = new File(params.myFile)
    if(file && file.canRead()){
        docman.createDocument( parentNode, params.myFile_filename, file, "", false, parentNode)
        //Redirect after submit
        redirect "${url}/open/${self.ID}"
    }
}
```

Methods can be called omitting the parenthesis in the parameter clause, given that (a) there is no ambiguity and (b) the method signature has at least one parameter.

Copyright © 2013-2022 AnswerModules Sagl

```
// In certain cases, parenthesis can be omitted
docman.createFolder "MyFolder"
```

Properties and Fields¶

Properties and public fields of objects can be accessed using the dot "." followed by the property or field name.

```
def folder = docman.createFolder("myFolder")

// Accessing an object property
def me = folder.createdBy
```

A safe syntax to navigate through fields is available in Groovy by adding a "?" before the dot. In this case, the chain will be interrupted if one of the intermediate values is undefined, avoiding an exception to be raised.

```
// Safe field access (no exception raised if folder is NULL)
def me = folder?.createdBy
```

Comments¶

```
// Comments are available as single line // and multiline /* */

def a = 1 // A comment can close a line

/* Or span
over multiple
lines */
```

Closures¶

Content Script inherits from Groovy the concept of Closures. A closure is an open, anonymous, block of code that can take arguments, return a value and that can be assigned to a variable.

```
// Define a closure and assign it to a variable
def addNumbers =  {  def num1 , def num2 -> //Arguments
    return (num1 as int)+(num2 as int)
}

out << "Calling the addNumbers closure:${addNumbers(4, "5")} <br/>"

addNumbers = { String... arguments -> // Variable number of arguments (MUST be the last parameter)
    def total =0
    arguments.each{total+=(it as int)}
    return total
}

out << "Calling the addNumbers closure:${addNumbers("1", "2","3")} <br/>"
```

Copyright © 2013-2022 AnswerModules Sagl

```
def createNewFolder = {  String name,  def parentNode = docman.getEnterpriseWS() ->
    docman.createFolder(parentNode, "name" )
}

def node = createNewFolder( new Date().format("yyyyMMddHHss"))
out << "Calling the createNewFolder with One arguments:${node.ID} <br/>"

def newNode = createNewFolder( new Date().format("yyyyMMddHHss"), node)
out << "Calling the createNewFolder with Two arguments:${newNode.ID} <br/>"
```

Content Script programming valuable resources¶

A number of resources can be extremely useful to the Content Script developer at different times. A few of the most important ones are:

Online help

The Content Script Module features an online guide that covers the basic language syntax and functionalities. It also contains quick references to context variables and methods.

Code Snippet Library

When using the Content Script Editor, a library of ready-to-use code snippets is available to bootstrap new scripts without having to start from scratch. The library includes usage examples and code templates for many common use cases, and can be easily extended by the developer.

**Groovy reference guide **

The Apache Groovy language is supported by a wide community of adopters worldwide. Groovy is supported by the Apache Software Foundation: a significant amount of documentation and examples are available online.

http://www.groovy-lang.org/ *(http://www.groovy-lang.org/)*

Velocity reference guide

The Apache Velocity engine powers the templating features in Content Script. Velocity is supported by the Apache Software Foundation: lots of documentation and examples can be found throughout the web and on the project's website.

http://velocity.apache.org/ *(http://velocity.apache.org/)*

Writing and executing scripts

Content Script scripts are "document" class objects stored on Content Server. The primary usage for a script is its execution. When you "execute" a script, you are basically programmatically invoking a series of APIs that perform actions over Content Server's or other systems' data. In

Copyright © 2013-2022 AnswerModules Sagl

the following paragraphs, we are going to analyze all the Content Script architecture's elements and components that play a role in turning a textual file into an actionable object.

As said, scripts are persisted as "documents" on Content Server. Whenever you execute a script a component named Script Manager retrieves the script's last version and, either compiles it (and caches the compiled version) or loads a pre-compiled version of it for execution. Scripts' execution is managed by another component named Content Script Engine. The Content Script Engine executes the script's code against the provided execution context (the execution context is the "container" through which the script's code can access the Content Script's services, environment variables, support variables, database, etc..). The internals of both the Script Manager and the Script Engine are not relevant for the purpose of this manual and won't be discussed.

API Services¶

Content Script API Service¶

Content Script APIs are organized in classes denominated **services**. Each Content Script API service acts as a container for a set of *homogeneous* APIs (API releated to the same kind of objects or features). Content Script APIs can be extended creating and registering new services *(/working/contentscript/sdk/#create-a-custom-service)*.

Content Script APIs are, in their most essential form, the methods exposed by the service classes. In order to be recognize as a Content Script API a service class method must be decoretad with the `@ContentScriptAPIMethod` annotation.

> **Content Script API Services Interfaces**
>
> When working with Content Script APIs developers program against interfaces. As a matter of fact all Content Script API services and objects implement one or more interfaces. Implementation classes can be easily distinguished from their interfaces because their name ends with the "Impl" suffix.

Content Script API Objects¶

Content Script APIs return or accept, as parameters, objects representing OTCS objects or features. In Content Script, these objects are referred to as **Content Script API objects**. Content Script API objects are *active* information containers. We define them *active* because they expose APIs designed to manipulate the information stored in themselves.

In order to be recognize as a Content Script API Object a class must be decoretad with the `@ContentScriptAPIObject` annotation.

When the script Execution Context is initialized by the Content Script engine, all registered API services are injected into it. These **services** allow a Content Script to perform operations on

Copyright © 2013-2022 AnswerModules Sagl

Content Server, to use internal utilities (such as PDF manipulation utilities or the templating service), to access external systems and services, etc.

Here after are some of the main services that are currently available as part of Content Script APIs.

API Service Name	Description
Base API	Base API is constituted by methods and properties that are exposed directly by each script. Some of the most important API are: logging, redirection (used to redirect users navigation through a server side redirection i.e. http code 302, outputting HTML, XML, JSON and Files)
`docman`	The docman service is the main access point to the Content Server Document Management functionalities. With docman service it is possible, among other things, to: create and manipulate documents and containers, access and modify meta-data, access and modify object permissions, access volumes, perform database queries, manipulate renditions and custom views, run reports, consume OScript request handlers, programmatically import/export content through Content Server native XML import/export feature
`users`	The users service is the main collector for all APIs related to Content Server users and groups. With users service is it possible, among other things, to: create/modify/delete users and groups, impersonate different users, access and modify user privileges, perform member searches
`workflow`	The workflow service allows to programmatically manipulate workflows. With the workflow service is it possible, among other things, to: start, stop, suspend, resume, delete workflows, access and manipulate workflow and task data, accept, complete, reassign workflow tasks, perform searches within workflows and tasks, change workflows' and steps' title
`search`	The search service allows to programmatically search over Content Server's repository. With the search service is it possible, among other things, to: easily build/execute complex search queries programmatically, easily build/execute query based on categories attributes, retrieve search result with or without pagination
`collab`	The collab service is the main access point to the Content Server collaborative functionalities. With collab service is it possible, among other things, to: create and manage projects, tasks and milestones, create and manage discussions, list and manage users’ assignments

Copyright © 2013-2022 AnswerModules Sagl

API Service Name	Description
`mail`	The mail service allows to programmatically create/send and receive emails from scripts. With the mail service is it possible, among other things, to: create and send email message through multiple mailboxes, scan mailboxes and retrieve incoming messages and attachments, create email messages (both html and text messages are supported) with custom templates, send email to internal users and groups, attach files and Content Server documents to emails, configure multiple email service profiles to use different IMAP/SMTP configuration at the same time
`template`	The template service can come in handy anytime you have to dynamically create documents. With the template service is it possible, among other things, to: evaluate documents and plain text strings as templates, replace place holders and interpret template-expressions
`admin`	The admin service allows to programmatically perform administrative tasks. With the admin service is it possible, among other things, to: perform XML import/export operations, programmatically schedule/unscheduled Content Script executions
`classification`	The classification service is the main access point to the Content Server classification features. With classification service is it possible, among other things, to: access, apply, remove classifications from objects
`pdf`	The pdf service allows to programmatically manipulate PDF documents. With pdf service is it possible, among other things, to: create and manipulate PDF documents, write in overlay on PDFs, extract PDF pages as images, merge PDFs, add watermarks to PDF documents, add barcodes (mono and bi-dimensional) on PDF pages, remove print/modify permissions from PDF, add PDFs in overlay to existing PDFs, extract images from pages or portion of pages, read bar-codes form PDF's pages, remove/insert pages
`ftp`	The ftp service allows to interact with FTP services. With ftp service it is possible, among other things, to: access, read, write files and folders on multiple FTP servers
`docx xlsx`	The docx/xlsx services allow to programmatically manipulate Microsoft Office documents. With docx/xlsx services is it possible, among other things, to: create and manipulate Word, PowerPoint and Excel documents, read and write documents' properties
`forms`	The forms service is the main access to the Content Server web-forms features. With forms service it is possible, among other things, to: create and modify `form` and `form template` objects, read/modify/delete submitted form records, submit new form records, export/import form records

Copyright © 2013-2022 AnswerModules Sagl

API Service Name	Description
`adlib`	The adlib service allows to programmatically drive the AdLib rendition engine. With adlib service it is possible, among other things, to: create jobs for AdLib PDF Express Engine and fetch renditions results
`rend`	The rend service allows to programmatically invoke external rendition engines. With rend service it is possible, among other things, to: transform on the fly HTML pages to PDF documents, rend WebForms as PDFs, invoke external services through an "all-purpose" generic rendition api
`sap`	The sap service allows to integrate Content Script with the well known SAP ERP through RFCs. With sap service it is possible, among other things, to: connect to multiple SAP systems through JCO APIs, invoke standard and custom SAP functions to retrieve/update ERP information

APIs evolution

New service APIs are constantly added or updated with every subsequent release of Content Script. Optional APIs are usually available through Content Script Extension Packages, and can be installed separately using the master installer or the extension packages' own installers.

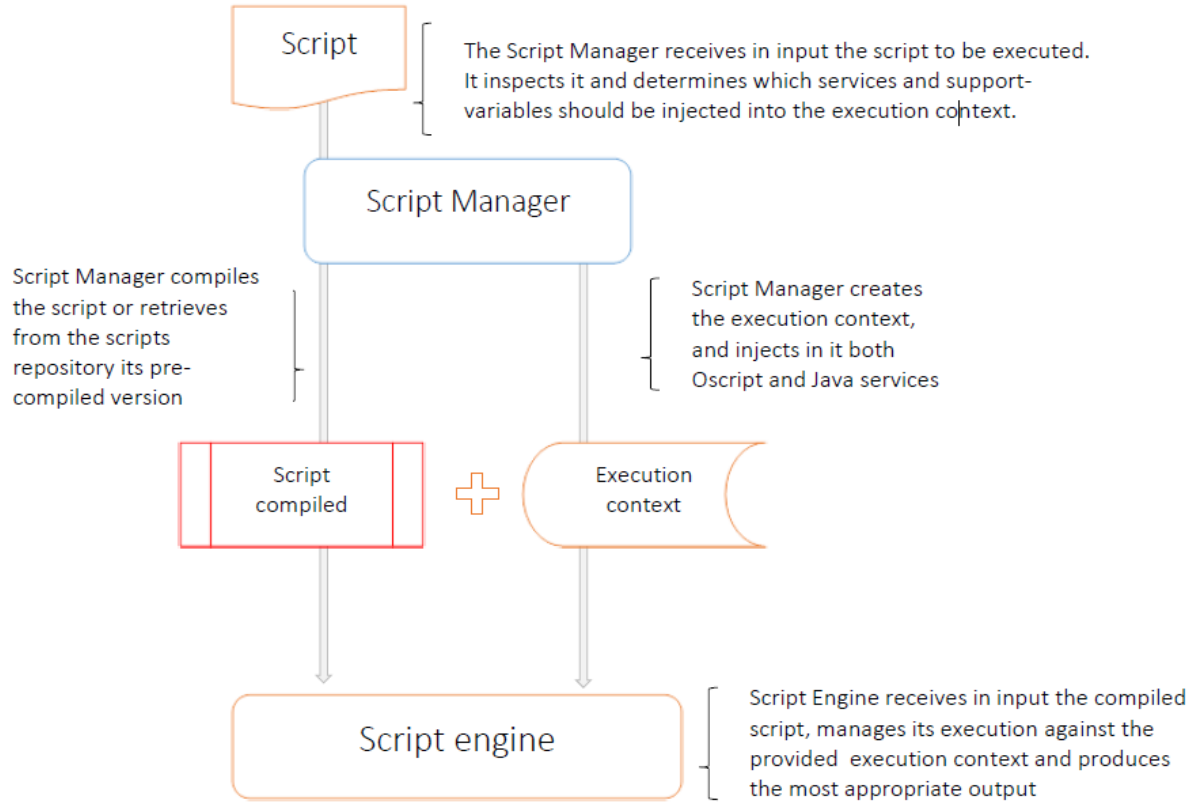
Execution context¶

Upon execution, every Content Script is associated to a Groovy binding. The binding can be seen as a container for Objects that are part of the context in which the script is executed. We make reference to this context as Content Script Execution Context or as Script Binding.

The Script Manager creates the most appropriate execution context on the basis of:

- the script's code;
- the system's current configuration;
- the user context (user's permission, user's roles, etc..)
- the cause that triggered the script's execution (direct invocation, scheduler, callback, etc..)

Copyright © 2013-2022 AnswerModules Sagl



The Script Manager initializes the Script Binding before execution, injecting a set of objects, which include:

- API Services
- Request variables
- Support Objects
- Support Variables

Additionally, a set of script utility methods are available in the Content Script (Base API). The methods grant access to short-cuts for commonly used features or can pilot the execution result.

Request variables¶

Request variables are variables injected into the execution context by the Script Manager whenever a script is directly invoked as a result of a user's browser request.

Copyright © 2013-2022 AnswerModules Sagl

Variable	Description
params	A container for the Script's request parameters. It's a non-case sensitive map that provide access to all the parameters passed to the script when executed. In the params map are injected by default also the following variables (where available): • myurl:The URL string used to execute the Content Script • node: the id of the Content Script object • useragent: the user's browser useragent • cookies: the user's browser cookies (as strings) • method: the HTTP verb used to request the script • lang: the user's locale • port: the HTTP port used to request the script • server: the HTTP host used to request the script • pathinfo: the request's URL path information
request	A synonym for the previous variable (for backward compatibility)

Support variables¶

The number and the nature of the variables that are injected by the Content Script Engine depends primarily from the mode through which the script has been executed. Content Script scripts used for example to implement Node Callbacks or columns' Data Sources will have injected in their Execution Context, respectively: the information regarding the Node that triggers the event or the Node for which the column's value is requested. Please refer to the Content Script module online documentation for the name and type of the variables made available in the Execution Context in the different scenarios. The following variables are always injected.

Variable	Description
img	Content Server static resource context path (es. /img/).
webdav	WebDav path
supportpath	Content server support path
url	Content Server CGI Context
SCRIPT_NAME	A synonym for the previous variable (for backward compatibility)
csvars	A map containing the script's static variables *(/working/contentscript/otcsobj/#static-variables)*
originalUserId	The ID of the user that triggered the execution of the Script (not considering impersonation)

Copyright © 2013-2022 AnswerModules Sagl

Variable	Description
`originalUsername`	The username of the user that triggered the execution of the Script (not considering impersonation)

IMG

Please note that most of the time the img context variable ends with a trailing slash. To correctly use it as a replacement variable in Content Script strings or velocity templates we suggest you to use the ${img} notation. E.g.:

```
"""<img src="${img}anscontentscript/img/am_logo.png" />"""
```

Support objects¶

Support objects are instances of Content Script classes that the Script Manager creates, configures and injects into every execution context in order to provide a simple mean for accessing very basic or commonly required functionalities.

Variable	Description
`self`	An object representing the Content Script node being currently executed.
`response`	An instance of the ScriptResponse class that can be used to pilot the Content Script output.
`gui`	A map of standard Content Server UI Components that can be enabled/disabled at the time of rendering the page. E.g. • gui.search = false • gui.sideBar = false **Disable standard UI** To completely disable the standard Content Server UI use: `gui.gui = false`

Copyright © 2013-2022 AnswerModules Sagl

Variable	Description
`log`	Each Content Script is associated with an instance logger that can be used to keep track of the script execution. From within a script you can access the logger either using the Script's method getLog() or the shortcut **log**. The Content Script logging system is based on a framework similar to the one used internally by OTCS. The logger supports five different levels: trace, debug, info, warn, error. The default log level for any script is: **error** this means that log messages at level for example **debug** won't be outputted in the ModuleSuite's master log file (cs.log). Logging level can be overridden per script basis through a dedicated administrative console.
`out`	A container for the script textual output

Base API¶

The Content Script "Base API" or "Script API" is constituted by methods and properties that are exposed directly by each Content Script script.

API	Description
`asCSNode(Map)`	An alternative to loading a node explicitly using one method out of: `docman.getNode`, `docman.getNodeByPath`, `docman.getNodeByNickname`
`asCSNode(Long)`	An alternative to loading a node explicitly using the `docman.getNode` method
`redirect(String)`	A shortcut for sending a redirect using the response object
`json(String)`	A shortcut for sending json using the response object
`json(Map)`	A shortcut for sending json using the response object
`json(List)`	A shortcut for sending json using the response object
`sendFile(File[,String])`	A shortcut for sending a file using the response object
`success(String)`	A shortcut for setting the result of the script execution to "success"
`runCS(Long)`	A utility method to run a second Content Script (identified by ID) within the same context
`runCS(String)`	A utility method to run a second Content Script (identified by nickname) within the same context

Copyright © 2013-2022 AnswerModules Sagl

API	Description
`runCS(String, Object[])`	A utility method to run a second Content Script (identified by nickname) using a cleaned execution context (the new execution context shares with the caller's context only the Content Script services and the following variables: out, gui, response). In the sub-script code the parameters that have been used to call the sub-script can be accessed through the context variable "args". Using this variant it's possible to intercept the result of the sub-script execution.
`printError(Ex)`	A utility method to print out any exception raised by script's execution

Examples

Usage example for runCS(String, Object[]) API

```
//Parent Script
node = asCSNode(123456)
map  = runCS("mySubScript", node, users.current)
out << map.user

//SubScript "mySubScript"
def retVal = [:]
retVal.name = args[0].name
retVal.user = args[1].with{
 [
    name:it.displayName,
    id:it.ID
 ]
}
return retVal
```

Usage example for asCSNode(...)API

```
// Load a CSNode
asCSNode(2000)

// A node can be loaded also by path or nickname
asCSNode(nickname:"MyNode")
asCSNode(path:"path:to:myNode")
asCSNode(id:2000) //=== asCSNode(2000)
```

Usage example for printError(...)API

```
try{
    out << asCSNode(12345).name
}catch(e){
    log.error("Error ",e) //Prints the full stack trace in the log file
    printError(e) //Outputs the error

}
```

Copyright © 2013-2022 AnswerModules Sagl

Script's execution¶

As shown in previous sections, the execution of Content Scripts can be triggered in different ways. Here after are a few examples:

- **Direct execution by a user**. This can happen, for example:
  - Using the **Execute** action in the object function menu or promoted object functions
  - While using the Content Script Editor, using the **Execute** or **Execute in Modal** buttons (useful for debug and testing purposes, shown in the figure below)
  - A **URL** associated to the execution of a Content Script is invoked
  - A **Content Script backed SmartUI widget** is displayed
- **Direct execution by an external system**
  - A URL associated to a **Content Script REST API** is invoked
- **Automatic execution by the system**. This happens when:
  - The script is **scheduled**, at the configured execution time
  - A **callback** is configured, and the associated event is triggered
  - A Content Script **Workflow step** is configured as part of a workflow, and the step is activated
  - A Content Script is configured as a **Data Source for a WebReport**, and the WebReport is executed
  - A Content Script serves as a **Data Source** for a custom column

Copyright © 2013-2022 AnswerModules Sagl

Script's output¶

As you can easily imagine by analysing the examples in the previous paragraph, the expected result from the execution of a Content Script varies significantly from case to case.

When a user executes a Content Script directly from the Content Server user interface, he/she would probably expect, in most of the cases, the result to be either a **web page**, a **file to download**, or a **browser redirection** to a different Content Server resource.

When a remote system invokes a REST service API backed by a Content Script, it will most probably expect structured data in return (probably **XML** or **JSON data**).

When a Content Script is executed as part of a workflow and the next step is to be chosen depending on the execution outcome, the script will probably be expected to return a single variable of some kind (**a number or a string**) or an indication that the execution was either **successful** or encountered **errors**.

Content Script is flexible enough to cover all of these scenarios. The next section will include examples of how to provide the different output necessary in each situation.

HTML (default)¶

The default behaviour in case of a successful script execution is to return the content of the "out" container

```
def contentToPrint = "This content will be printed in output"
out << contentToPrint
```

```
def contentToPrint = "This content will be printed in output"

//If the object returned by the script is a String, it will be printed in output
return contentToPrint
```

JSON¶

JSON content can be easily returned

```
def builder = new JsonBuilder()
builder.companies {
    company "AnswerModules"
    country "Switzerland"
    }

// Stream JSON content, useful for restful services
response.json(builder)
```

```

```

Copyright © 2013-2022 AnswerModules Sagl

```
String jsonString = '{"key":"value"}'

// A string containing JSON data can be used
response.json(jsonString)
```

```
// or with the shorthand method
json(jsonString)
// or
json([[key:”value1”], [key:”value2”]])
```

XML¶

XML content can be easily returned

```
gui.gui = false
gui.contentType = "application/xml"

def builder = new StreamingMarkupBuilder()
def parent  = asCSNode(2000)
def nodes   = parent.childrenFast //nodes are lazy loaded

def xml = builder.bind {
    node(id:parent.ID, name:parent.name,  isContainer:parent.isContainer){
        children {
            nodes.collect {
                node(id:it.ID, name:it.name, isContainer:it.isContainer)
            }
        }
    }
}
out << XmlUtil.serialize(xml)
```

Output of the above script:

```
<node id="2000" name="Enterprise" isContainer="true">
    <children>
            <node id="90064"    name="Import"   isContainer="true"/>
            <node id="3270165"  name="Training" isContainer="true"/>
    </children>
</node>
```

Using gui support object for tuning script's output

Note the usage of `gui.contentType` in order to change the response's “Content-Type” header.

Files¶

It is also possible to stream a file directly:

```
// Stream a file as result of the execution
def res  = docman.getTempResource("tempRes", "txt")
res.content.text = "Just a test"
```

Copyright © 2013-2022 AnswerModules Sagl

```
def file = res.content
response.file(file)
```

```
// Stream a file as result of the execution
def res  = docman.getTempResource("tempRes", "txt")
res.content.text = "Just a test"
def file = res.content
// Stream a file, specifying if it is a temporary file (will prevent deletion)
response.file(file, true)
```

```
// Stream a file as result of the execution
def res  = docman.getTempResource("tempRes", "txt")
res.content.text = "Just a test"
def file = res.content
// or with the shortcut method
sendFile(file)
```

```
// Stream a file as result of the execution
def res  = docman.getTempResource("tempRes", "txt")
res.content.text = "Just a test"

// or returing the CSResource directly
res.name = "My textFile.txt"
return res
```

Managed resources¶

In the context of developing against OTCS you will end up dealing with many different kind of contents most of which are (or are strictly related with) files. In order to reduce the amount of code needed to properly manage the disposition of temporary files, Content Script introduces the concept of "managed resource" or CSResource. A CSResource is basically a wrapper around the File class. CSResources are managed by the Content Script engine (no disposition required) and are returned any time you want to access the content of a CSDocument or you fetch a version from it (in these cases the CSResource will keep a reference, towards the source CSDocument, through its "owner" property.

CSResources are first class citizens in Content Script. A CSResrouce can be for example returned directly by a Content Script, triggering the download of the same.

Returning CSResource to trigger document download

Returning a CSResource from a script is the simplest way to stream out a file in this case is important to keep in mind that the name of the downloaded file will be determined using the following rule:

```
if the property onwer of the CSResource is != null
then
    use the name of the CSNode referenced by the CSResource's owner property
else
    use the CSResource's name property.
end
```

Copyright © 2013-2022 AnswerModules Sagl

Redirection¶

In alternative, the response could contain a redirection to an arbitrary URL:

```
String url = "http://www.answermodules.com"

// Send a redirect using the response
response.redirect(url)


// or with the shortcut method
redirect(url)
// or
redirect "${url}/open/2000"
// or
redirect asCSNode(2000).menu.open.url
```

HTTP Code¶

In certain cases (e.g. when Content Script is used to extend OTCS' REST APIs), it could be necessary to explicitly control the "error" or "success" status of the script execution:

```
// Force the script execution result to be "success" using the response
response.success("This is a success message")
response.success("This is a success message",200)

// or with the shortcut method
success("This is a success message")
success("This is a success message",200)

// Force the script execution result to be "success"
response.error("This is an error message", 403)

// or with the shortcut method
error("This is an error message", 403)
```

Advanced programming¶

Templating¶

Content Script features a flexible yet powerful templating engine based on Apache Velocity. Evaluating a template is just a matter of invoking one of the evaluate methods available through the template service.

Content Script velocity macros¶

Content Scripts defines a collection of macros that simplify the creation of OTCS UI embeddable interfaces. A developer can create his own macros simply defining them in a z_custom.vm file to be stored under the Content Script "Temp" folder (as defined in the Base Configuration page: amcs.core.tempFilePath).

Copyright © 2013-2022 AnswerModules Sagl

Name and description	Param	Type and description	Usage example
csmenu(dataid[,nextUrl]) Creates the standard OTCS context menu for the given node (identified by its dataid)	dataid	Integer node's dataid	`#csmenu(2000)`
	nextUrl	String	
csresource(retList) Loads static libraries from the module support directory	resList	List A list of resources to load. To be chosen from: query, jquery-ui, jquery-ui-css, bootstrap, bootstrap-css	`#csresource(['bootstrap'])`

csform(script[,submit])
Creates the HTML form needed to submit a request against the executed Content Script
	script	Integer
		The objId of the Content Script you'd like to execute
			`#@csform()`
			`//Custom form inputs go here`
			`#end`

	submit	String The value for the label of the submit button. If null the submit button will not be created	

cstable(columns,sortColumn, columnsClasses[,checkBoxes])
Creates an HTML table that fits nicely with the standard OTCS UI
	columns	List
		The list of column labels
			`#@cstable(['First Name'], {},{}, true)`
			`//Your rows here`
			`#end`

	sortColumns	Map A map of "Column Label", "Property" couples. The Property is used to build sort links for columns	

Copyright © 2013-2022 AnswerModules Sagl

Name and description	Param	Type and description	Usage example
	columnsClasses	Map A map of “Column Label”, “CSS Classes” couples. The “CSS Classes” are assigned to the THs tags.	
	checkBoxes	Boolean If TRUE the first column of the table will have width 1%. To be used to insert a checkboxes column	
cspager(skip,pageSize, pagerSize,elementsCount) Creates a pagination widget to be used	skip	Integer The index of the element to skip before to start rendering rows	`#cspager(0 25 3 $parent.childCount)`
	pageSize	Integer The page size (e.g. 25)	
	pagerSize	Integer The number of pages to show in the pager widget	
	elementsCount	Integer The total number of elements	

Copyright © 2013-2022 AnswerModules Sagl

OScript serialized data structures¶

Content Script Java layer is tightly bound with Content Script Oscript layer, thus quite frequently you will face the need of managing Oscript's serialized data structures obtained for example querying the OTCS' database or from nodes' properties.

Oscript serializes its data in the form of Strings, for this reason Content Script enhances the String class in order to provide a quick method for retrieving the corresponding Content Script's objects out of the OScript serialized representation.

Methods available on the String class are:

- getDateFromOscript
- getListFromOscript
- getMapFromOscript

In the exact same way Content Script enhances its most common types (List, Map, Date, Long, CSReportResult) in order to simplify the creation of the corresponding OScript serialized representation.

The below table shows an usage example of the mentioned features:

Statement	Result
"D/2011/7/19:18:10:51".getDateFromOscript()	Tue Jul 19 18:10:51 CEST 2011
"{1,2,3}".getListFromOscript()	[1, 2, 3]
"A<1,?,'key1'=1000,'key2'={1,2,A<1,?,'key3'=2002,'key4'=D/2017/7/19:18:10:51>}>".getMapFromOscript()	[key2:[1, 2, [key4:Wed Jul 19 18:10:51 CEST 2017, key3:2002]], key1:1000]
sql.runSQLFast("select ExtendedData EXT from DTree where DataId = %1",false,false, -1, 520305).rows[0].EXT.getMapFromOscript()	[DisplayAsLink:false, alignment:right, dataSource:attr_93202_3, NewWindow:false, sortable:false, DisplayValue:%value%, inheritedPermID:93202, columnDisplayWidth:20, locations:[90372], TitleText:, indexName:null, longText:0, columnEMWidth:9, columnName:WND_attr_93202_3, URL:? func=ll&objId=%objid%&objAction=attrvaluesedit&version=-1&nexturl=%nexturl%]
sql.runSQLFast("select DATAID, NAME from DTree where DataId = %1",false,false, -1, 520305).getOscriptSerialization()	V{<'DATAID','NAME'><520305,'Amount'>}
"January 1, 2013".asDate().getOscriptSerialization()	D/2013/1/1:12:0:0
[1,2,3].getOscriptSerialization()	{1,2,3}
['key1':1000, 'key2':[1,2, ['key3':2002, 'key4':new Date()]]].getOscriptSerialization()	A<1,?,'key1'=1000,'key2'={1,2,A<1,?,'key3'=2002,'key4'=D/2017/7/20:9:52:43>}>

Optimizing your scripts¶

Behaviors¶

You can use behaviors to decorate your scripts and let them implement a specific set of new functionalities. Behaviors are to be considered similar to inheritance. A behavior is defined as a collection (MAP) of closures and usually implemented in the form of a static class featuring a **getBehaviors** method.

When you add a behavior to your script, all the closures that have been defined in the behavior become part of your script thus becoming part of your script context.

Copyright © 2013-2022 AnswerModules Sagl

Behaviors are resolved at compilation time, this means that they should be considered as a static import.

Said otherwise, any changes applied directly on the script that implements your behaviors, won't effect the scripts that have imported such behaviors. In order to update the imported behaviors you have to trigger the re-compilation of the script that is importing them (target script).

BehaviorHelper¶

In order to add behaviors to a script you shall use the BehaviourHelper utility class.

The BehaviourHelper utility class, features three methods:

```
@ContentScriptAPIMethod (params = [ "script" , "behaviours" ], description = "Add behaviours to a Co
public static void addBeahaviours(ContentScript script, Map<String, Closure> closures)

@ContentScriptAPIMethod (params = [ "script" ,  "behaviours" ], description= "Remove behaviours from
public static void  removeBehaviours(ContentScript script, String... closures=null)

@ContentScriptAPIMethod (params = [ "script" ,  " behaviour " ], description= " Determine if the scr
public static void  hasBehaviour(ContentScript script, String name)
```

Through BehaviourHelper you can add, remove or check for the presence of an associated behavior.

Behaviors are of great help when it comes to structure your code base, optimize executions and reduce boilerplate code.

Module Suite comes with few predefined behaviors, you can easily implement yours by defining a map of closures to be passed to the above BehaviourHelper utility class.

Default Behaviours¶

The AMController behavior has been designed to simplify the creation of form-based application on Content Server.

It features the following closures:

1. **start**: this closure takes no parameters, and it is used to dispatch incoming requests. It creates (if not already provided) an app object to be made available in the execution context. It analyzes the request's pathinfo, to extract the information required to route towards a registered closure. Rebuilds any Beautiful WebForm object found in the request.

   This closure should be the last instruction of your script.

   ```
   app =  [:]
   app.product ="Module Suite"
   ```

Copyright © 2013-2022 AnswerModules Sagl

```
if(!BehaviourHelper.hasBehaviour(this, "start") ) {
    BehaviourHelper.addBeahaviours(this, AMController.getBehaviours())
}

home = {
    out << "Hello world from ${app.product}"
}

details = { String id = null->
    out << "This script ID ${id?asCSNode(id as int).ID:self.ID}"
}
start()
```

When directly executed (`http://my.server/otcs/cs.exe?func=ll&objId=12345&objAction=Execute&nexturl=`.. or `http://my.server/otcs/cs.exe/open/12345`) the script above will output:

```
Hello world from Module Suite
```

when executed using: `http://my.server/otcs/cs.exe/open/12345/details` it will output

```
The script ID 12345
```

when executed as: `http://my.server/otcs/cs.exe/open/12345/details/2000` it wll output:

```
The script ID 2000
```

In other words the requested path will always been interpreted using the follow schema: `http://my.server/otcs/cs.exe/open/12345/closurename/param1/param2/param3` where **closurename** will be defaulted to "home" if not found in the path.

2. **loadForm(def formID, def amSeq=0):** loads a Form data object, setting form.viewParams.contentScript = params.node (so that if the form data object will be used with a BeautifulWebForm view the form will submit on this very same content script) and form.viewParams.amapp_Action = params.pathinfo.
3. **submitForm(def form):** validates the form data object and performs the submit (executing pre-submit and on-submit scripts if defined)
4. **renderForm(def form, def context=null):** renders the form either in the script context or in the specified context

Copyright © 2013-2022 AnswerModules Sagl

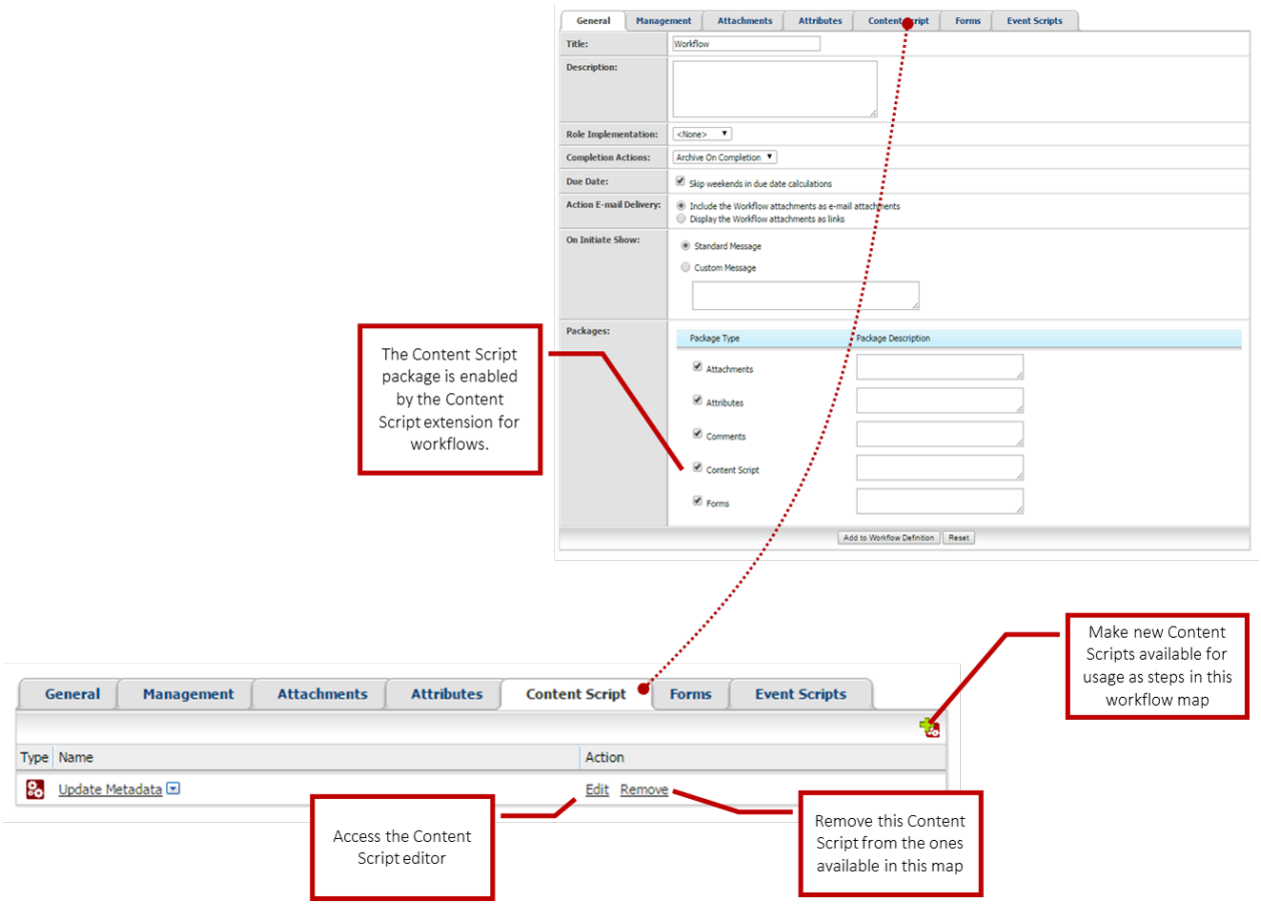
Working with workflows

Content Script Workflow Steps¶

The Content Script Extension for Workflows is automatically available upon installation of the Content Script module. The extension enables a new workflow package in Workflow maps (Content Script package) and custom type of workflow step (Content Script step).

Content Script Package¶

The Content Script package must be enabled in order to use Content Script steps within a workflow map.

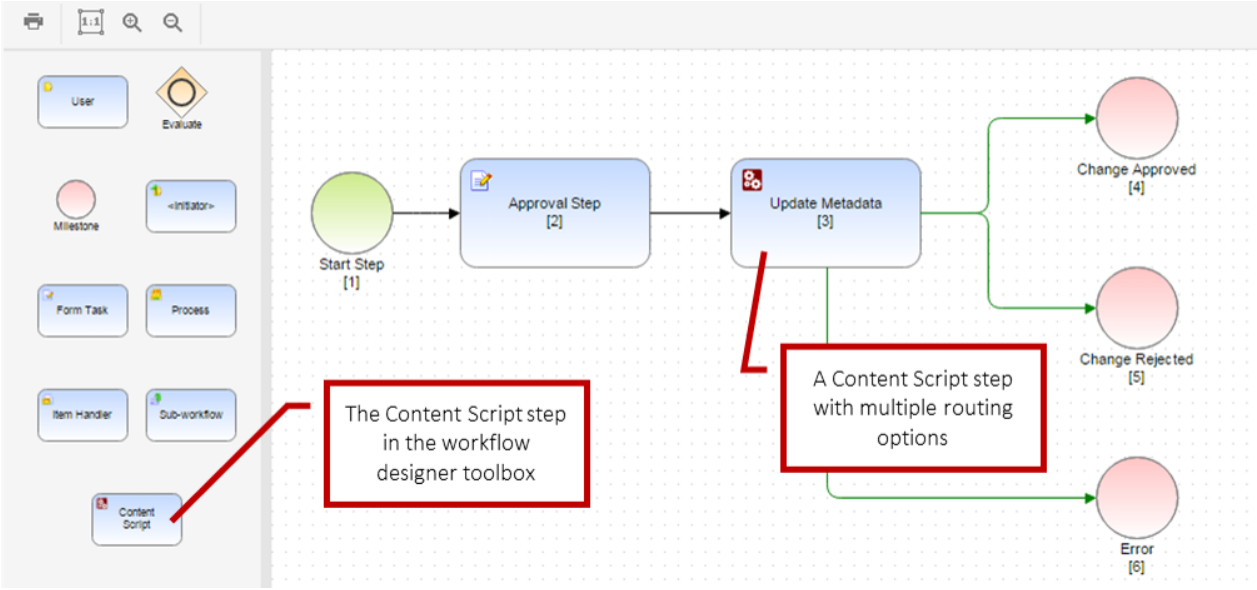


Once enabled, it will be possible to define the set of Content Script objects that will be available for inclusion in the current workflow map.

Content Script Workflow Step¶

Content Scripts enabled in the workflow package can be used in the workflow map as Content Script steps.

Copyright © 2013-2022 AnswerModules Sagl



Here below is an example of a Content Script step performing some basic operations on the current workflow task.

```
// Fetch the menu in its original format
def workflowStatus  = workflow.getWorkflowStatus(workID, subWorkID)
def workflowTask  = workflow.getWorkFlowTask(workID, subWorkID, taskID)
def allTasks   = workflowStatus.tasks

// Edit Workflow Attribute values
def workflowAttributes = workflowStatus.getAttributes()
workflowAttributes.setAttributeValues("Customer", "ACME inc.")
workflowAttributes.setAttributeValues("Country", "Switzerland")
workflow.updateWorkflowData(workID,  subWorkID, [workflowAttributes]) //Updates attributes

// Edit  Workflow Attribute values - different flavour
try{

    def atts =workflowStatus.getAttributes()

    // This API is not just for reading values...
    // Set the value
    atts.data.Customer = "ACME inc."
    atts.data.Country = "Switzerland"

    workflowStatus.updateData() // COMMIT CHANGES

}catch(e){
    log.error("Unable to access workflow's attributes ",e)
}

// Access a workflow form
def form = forms.getWorkFlowForm(workflowTask, "Form")
form.myattribute.value = "A new value"
forms.updateWorkFlowForm(workflowTask, "Form", form, false)

// Update Task's title
workflow.updateTaskTitle(
                          workID,
                          subWorkID,
                          taskID,
                          "Title with form field: ${form.myattribute.value}"
                        )

// Access a workflow form and workflow attributes - different flavour
```

Copyright © 2013-2022 AnswerModules Sagl

```
//Mapping
node = asCSNode(path:"Some Path:On Content Server:Node")

workflowStatus.attributes."Account Folder" = node.ID
workflowStatus.forms.Form.data."Lead Owner" = node.Account."Account Manager"
workflowStatus.forms.Form.data."Company" = node.Account."Company name"
workflowStatus.forms.Form.data."First Name" = node.Account."Contacts"."First Name"
workflowStatus.forms.Form.data."Last Name" = node.Account."Contacts"."Last Name"
workflowStatus.forms.Form.data."Email" = node.Account."Contacts"."Email"
workflowStatus.forms.Form.data."Addresses"."Street" = node.Account."Addresses"."Street"
workflowStatus.forms.Form.data."Addresses"."City" = node.Account."Addresses"."City"
workflowStatus.forms.Form.data."Addresses"."Zip Code" = node.Account."Addresses"."ZipCode"
workflowStatus.forms.Form.data."Addresses"."Country" = node.Account."Addresses"."Country"
workflowStatus.updateData()   // COMMIT CHANGES

// Updating Workflow title
workflow.updateWorkFlowTitle(
                              workID,
                              subWorkID,
                              "Company: ${workflowStatus.forms.Form.data."Company" as String}"
                             )

// Add documents to the attachments folder (an empty spreadsheet in this case)
def workflowAttachments = workflowStatus.getAttachmentsFolder()
workflowAttachments.createDocument("Spreadsheet", xlsx.createSpreadsheet().save())
```

In the above example, the script is:

- fetching information related to the current workflows status and tasks
- performing changes on some workflow attributes
- fetching and updating a workflow form
- adding attachments to the workflow attachments folder

Note that the above script makes use of some context variable available in the execution context that are peculiar only to workflow steps. The variables are:

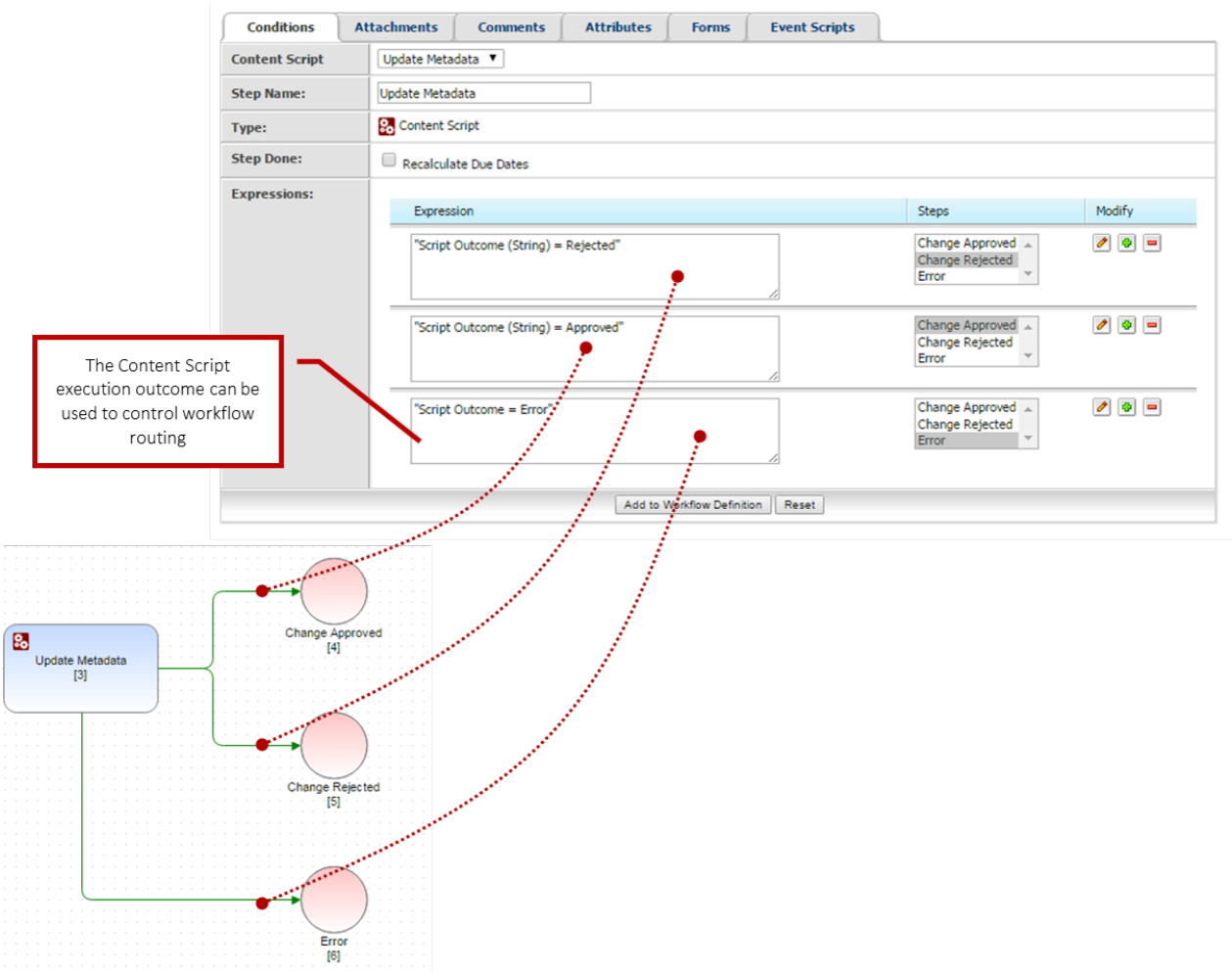
Expression type	Type	Description
workID	Integer	The workflow ID
subWorkID	Integer	The subworkflow ID
taskID	Integer	The current task ID

The above variables can be used in combination with the **workflow** service API to access all the information related to the current workflow. See the complete API documentation for a complete list of operations available on workflow instances.

Workflow routing¶

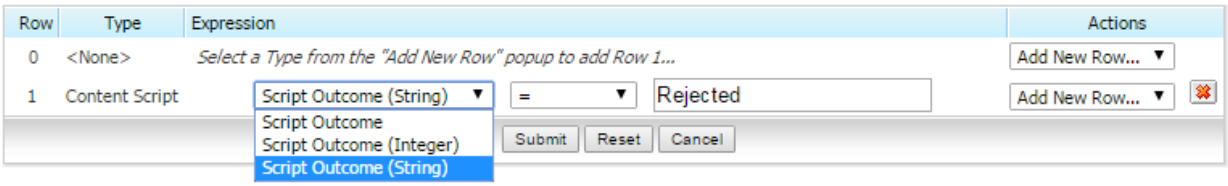
Content Script execution outcome, which MUST always be a String, can be interpreted in different ways, and used to route the next steps of the workflow.

Copyright © 2013-2022 AnswerModules Sagl



The following routing expression types are currently supported:

Expression type	Values	Description
Content Script Outcome	Success or Error	Error in case the script returns an exception
Content Script Outcome (Integer)	Any Integer value	Supports evaluation based on numeric comparison
Content Script Outcome (String)	Any String value	Evaluation based on string comparison



Copyright © 2013-2022 AnswerModules Sagl

Managing events (callbacks)

Synchronous and Asynchronous callbacks¶

Since version 1.5, Content Script supports the definition of **Event Callbacks**: in response to specific actions performed on Content Server, it is possible to execute one or more Content Scripts.

The callbacks can be:

- **synchronous**: the script is executed within the same transaction as the triggering action. Synchronous callbacks are configured through the **CSSynchEvents** container.
- **asynchronous**: the triggering action completes normally. The callback script is executed later on. Asynchronous callbacks are configured in the **CSEvents** container.

Since synchronous callbacks are performed in the same transaction as the event, any errors that occur during script execution will cause the transaction to roll back.

Performance

Since synchronous callbacks are executed in the same transaction as the event, make sure that any action performed by the script requires a reasonable time span for execution. Otherwise, the user experience could be affected negatively.

The definition of Content Script callbacks is based on a convention over configuration approach. In order to register a new callback, a script should be placed somewhere in a nested container structure in the **CSSynchEvents** or **CSEvents** container, following a specific naming convention.

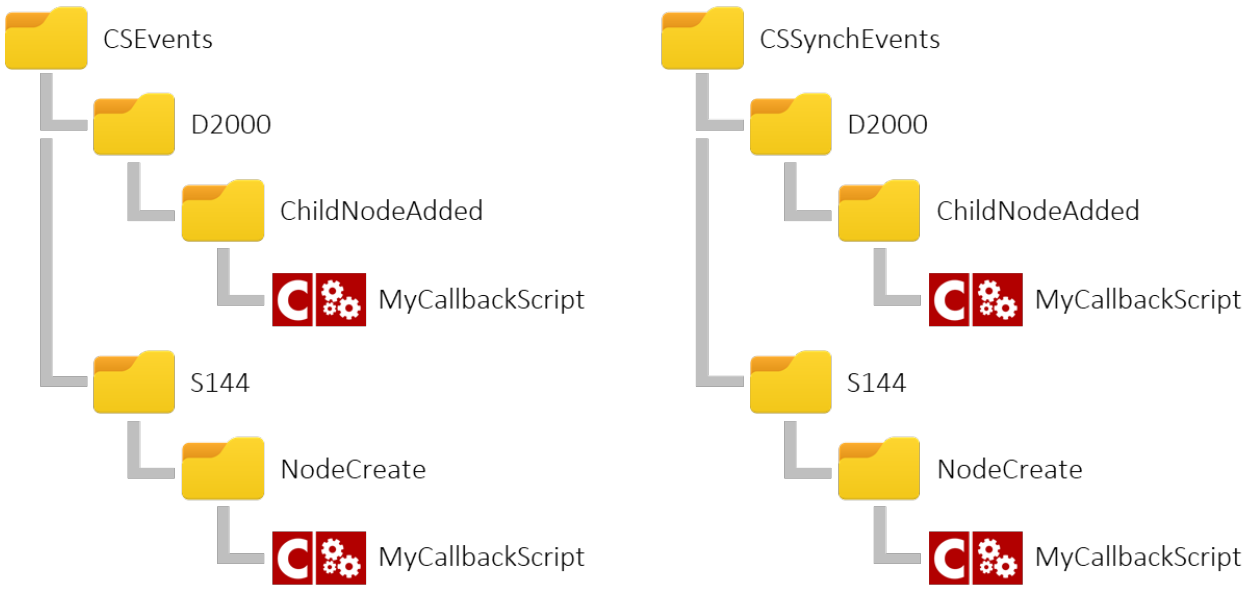
The first level under the container indicates the object or object subtype to which the callbacks are bound.

The naming convention is one of the following:

- `D<nodeID>`
- `S<subtype>`

where **nodeID** identifies the node unequivocally and **subtype** identifies a specific object subtype on Content Server.

Copyright © 2013-2022 AnswerModules Sagl



Examples:

D2000 will intercept events on the Enterprise Workspace

S144 will intercept event on Document type objects (subtype: 144)

The second level should be once again a container and specifies the **event type**. The name of this container should be one of:

- ChildNodeAdded
- ChildNodeCreate
- NodeAddVersion
- NodeAddVersionPre
- NodeCopy
- NodeCreate
- NodeCreatePre
- NodeMove
- NodeRename
- NodeUpdate
- NodeUpdateCategories

Inside the Event Type container it is possible to place one or more Content Scripts that will be invoked when the callback is triggered.

Copyright © 2013-2022 AnswerModules Sagl

> The Module Suite Administration pages feature a Manage Callbacks *(/administration/modulesuite/#manage-callbacks)* tool that can be used to verify, at any time, all the callbacks that are bound to a specific object or subtype.

In the following tables we present a summary of the supported Events and the information regarding the variables that are injected in the Execution Context, automatically by the framework, for each event. These variables can be useful to implement the required business logic within the Script.

```
| Event Name            | Execution Context Param | Type               | Description
|-----------------------|-------------------------|--------------------|-----------------------------
| All                   | callbackID              | String             | The CSEvent Name (NodeAddVer
|                       | eventSourceID           | Integer            | The dataid of the node that
| NodeAddVersion        | nodeID                  | Integer            | The document that has receiv
| NodeAddVersionPre     | nodeID                  | Integer            | The document that has receiv
| NodeUpdateCategories  | nodeID                  | Integer            | The updated node's id
|                       | addedCategories         | List&lt;String&gt;       | The list of added cate
|                       | deletedCategories       | List&lt;String&gt;       | The list of removed ca
|                       | changes                 | ChangeAssoc        | The list of applied attribut
| NodeCopy              | nodeID                  | Integer            | The id of the node that has
|                       | newNodeID               | Integer            | The newly created node's id
| ChildNodeAdded        | nodeID                  | Integer            | The id of the node where a n
|                       | newNodeID               | Integer            | The newly created node's id
| NodeCreate            | newNodeID               | Integer            | The newly created node's id
| NodeCreatePre         | newNodeID               | Integer            | The newly created node's id
| ChildNodeCreate       | nodeID                  | Integer            | The id of the node where a n
|                       | newNodeID               | Integer            | The newly created node's id
| NodeMove              | nodeID                  | Integer            | The moved node's id
| NodeRename            | nodeID                  | Integer            | The renamed node's id
|                       | oldName*Can be null*    | String             | The previous node's name
|                       | newName                 | String             | The current node's name
| NodeUpdate            | nodeID                  | Integer            | The updated node's id


The following table is related to the structure of the **ChangeAssoc**
object, necessary to manage **NodeUpdateCategories** type events.

| Property name | Type     | Description
|---------------|----------|-----------------------------------------------------------------------------
| attributePath | List     | The path of the modified attribute inside the category.
                             {"Name",0}: represents the path to the first value of the attribute "Nam
                             {"Name",1}: represents the path to the second value of the attribute "Na
                             {"Addresses", 2, "ZipCode", 0}: represents the path to the first value o
                             occurrence of the Set attribute Addresses
| oldValue      | Dynamic  | The previous attribute's value
| newValue      | Dynamic  | The present attribute's value
| categoryName  | String   | The category name
```

InterruptCallbackException - transaction roll-backed¶

There are cases in which you might want your synchronous callback to cause the roll-back of the original event transaction (to prevent its completion), e.g. you implemented a synch-callback triggered by the NodeCreate event and you want to use it to ensure that the node that is going to be created respects some specific business rule, for example, it's a PDF document. In this cases, you can just raise an un-catched InterruptCallback exception from within your callback script.

Copyright © 2013-2022 AnswerModules Sagl

E.g.

```
log.error("Running ${self.parent.parent.name}:${self.parent.name}:${self.name} for $nodeID")
out << "This is the mother of all failures..."
throw new InterruptCallbackException("New Callback Exception...")
```

Returning meaningful messages to your users

To return a message to your users you have just to add an output statement to your script.

Extending REST APIs

Extending REST APIs:CSServices¶

The **CSServices** container is dedicated to Content Scripts that should be made available as REST services.

The name of scripts placed in this container can be used to invoke the script directly through two dedicated HTTP endpoint (**amcsapi**, **amcsapi/v1**)

The **amcsapi** can be used to consume the REST service from within the Content Server GUI (it will in fact use the standard Content Server authentication mechanism to authenticate the user).

On the other hand the **amcsapi/v1** can be used to consume the REST service using the Content Server REST Apis authentication token *(https://developer.opentext.com/webaccess/#url=%2Fawd%2Fresources%2Farticles%2F6102%2Fcontent%2Bserver%2Brest%2Bapi%2B%2Bquick%2Bstart%2E*

When invoked, unless otherwise specified (for example, in the script's "Run As" configuration), each script is executed as the currently logged in user.

Basic REST service¶

As a very simple example, the script **getuserbyname** can be invoked by using an URL built as follows:

```
http://localhost/otcs/cs.exe/amcsapi/getuserbyname
```

```
http://localhost/otcs/cs.exe/amcsapi/v1/getuserbyname
```

Additional parameters can be passed to the service, and will be available in the Content Script (via the **params** object). For example, invoking the previous script as:

Copyright © 2013-2022 AnswerModules Sagl

```
http://localhost/otcs/cs.exe/amcsapi/getuserbyname?term=admin
```

the REST service framework will run the backing **getuserbyname** script adding the value of the GET parameter **term** in the **params** container variable. In the script, the value will be accessible by simply using the expression:

```
params.term
```

Behaviour based REST services¶

Since version 1.7.0, Content Script supports a "behaviour" based approach for the creation of REST services. This allows for an easier set-up of new services, enhance maintainability and better compliance with REST service commonly used conventions and de-facto standards.

A skeleton for a behaviour-based REST service is shown below.

A REST service can specify multiple operations, identified with behaviours. Each behaviour is implemented as a *closure*. By convention, the **home** behaviour is bound to the root of the API.

Service example¶

```
log.debug("Content Script REST Service {} - START", self?.name)

section = { String elemID=null, String method=null, String param=null ->
    try {
        if(elemID){
            switch(params.method){
                case "GET": //Read
                json(
                    [
                        operation:"section",
                        elemID:elemID,
                        method: method,
                        param: param
                    ]
                )
                return;
                default :
                response.error("Unsupported operation",500)
                return
                    }
        }else{
            json(
                [
                    operation:"section",
                    elemID:elemID,
                    method: method,
                    param: param
                ]
            )
            return
                }
    } catch(e){
        log.error("An error has occurred while managing the request", e)
        json([error:"Unable to complete your request $e?.message"])
    }
}
```

Copyright © 2013-2022 AnswerModules Sagl

```
//Default service method
home = { String elemID ->
    try {
        //Single element
        if(elemID){
            switch(params.method){  //request verb
                // CRUD operations
                case "POST": //Create
                //Your code here...
                break;
                case "GET": //Read
                json(["elemID":elemID])
                return;
                case "PUT": //Update
                //Your code here...
                break;
                case "DELETE": //Delete
                //Your code here...
                break;
            }
        }else{
            switch(params.method){  //request verb
                // CRUD operations
                case "POST": //Create
                //Your code here...
                break;
                case "GET": //Read
                //Your code here...
                break;
                case "PUT": //Update
                //Your code here...
                break;
                case "DELETE": //Delete
                //Your code here...
                break;
            }
        }
        // Default return
        json([ok:true])
    } catch(e){
        log.error("An error has occurred while managing the request", e)
        json([error:"Unable to complete your request $e?.message"])
    }
}

if(!BehaviourHelper.hasBehaviour(this, "start")) {
    BehaviourHelper.addBeahaviours(this, AMRestController.getBehaviours())
}

return start()

log.debug("Content Script REST Service {} - END", self?.name)
```

Sample invocation path	Operation	Parameters passed to the closure
/training	home	elemID = null
/training/2000	home	elemID = “2000”
/training/2000/section	section	elemID = “2000” method = null param =null

Copyright © 2013-2022 AnswerModules Sagl

Sample invocation path	Operation	Parameters passed to the closure
/training/2000/section/100	section	elemID = “2000” method = “100” param =null
/training/2000/section/100/list	section	elemID = “2000” method = “100” param =“list”
/training/section/2000	section	elemID = “2000” method = null param =null
/training/section/2000/100	section	elemID = “2000” method = “100” param =null
/training/section/2000/100/list	section	elemID = “2000” method = “100” param =“list”

Extending Content Script

Create a Custom Service¶

One of the most important feature of ModuleSuite is its extensibility. ModuleSuite has been in fact designed in order to let you extend it, creating new services, new components, widgets, code snippets etc..

Creating a new service it's particularly helpful when it comes to integrate other services and/or systems, or to leverage existing libraries to extend the Content Server capabilities. Creating your extension in the form of a new Content Script service you will automatically benefit from all the existing ModuleSuite features such as, for example, the full support of the Content Script Editor.

New services can be easily created by using the Content Script SDK. The **Content Script SDK** is a toolkit that can be used by developers to **create custom Content Script services**. Services created with the SDK can be seamlessly deployed in the target Content Server instance, and be accessible within Content Script code.

The suggested way to setup and use the Content Script SDK is by using the well-known **Eclipse IDE**.

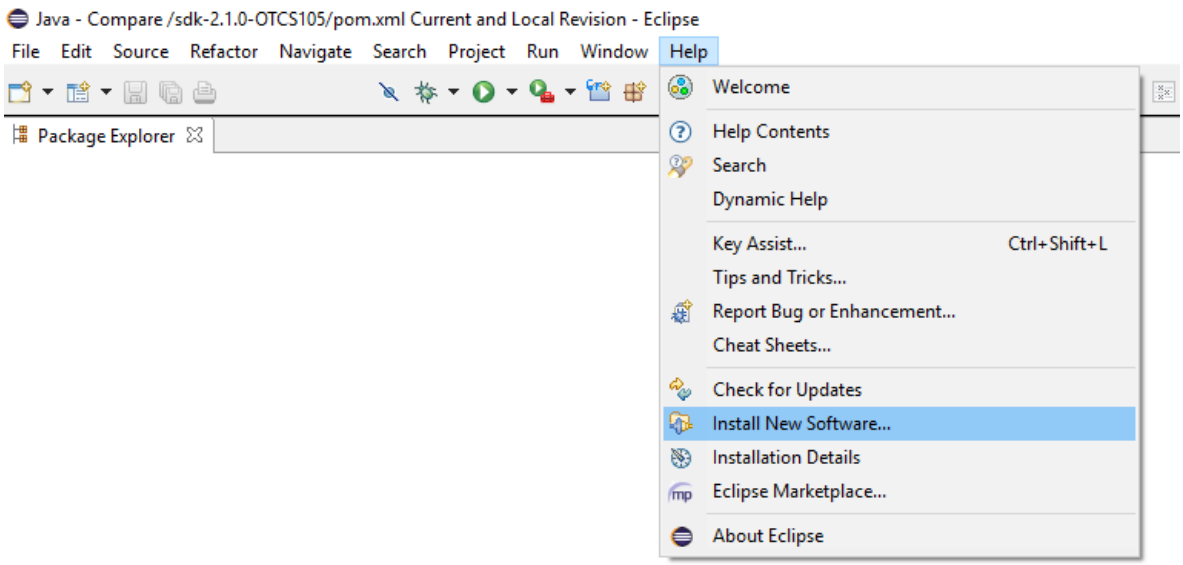
Copyright © 2013-2022 AnswerModules Sagl

The SDK is shipped in the form of an Eclipse Maven project. The project includes all the interfaces required for integration within Content Script, and can be used as a template to create a custom service.

Content Script SDK setup¶

1. Download **Eclipse Luna SR2** (https://eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr2c *(https://eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr2c)*) *

2. Run Eclipse. Use the **Help > Install new software** option to install some required additional components

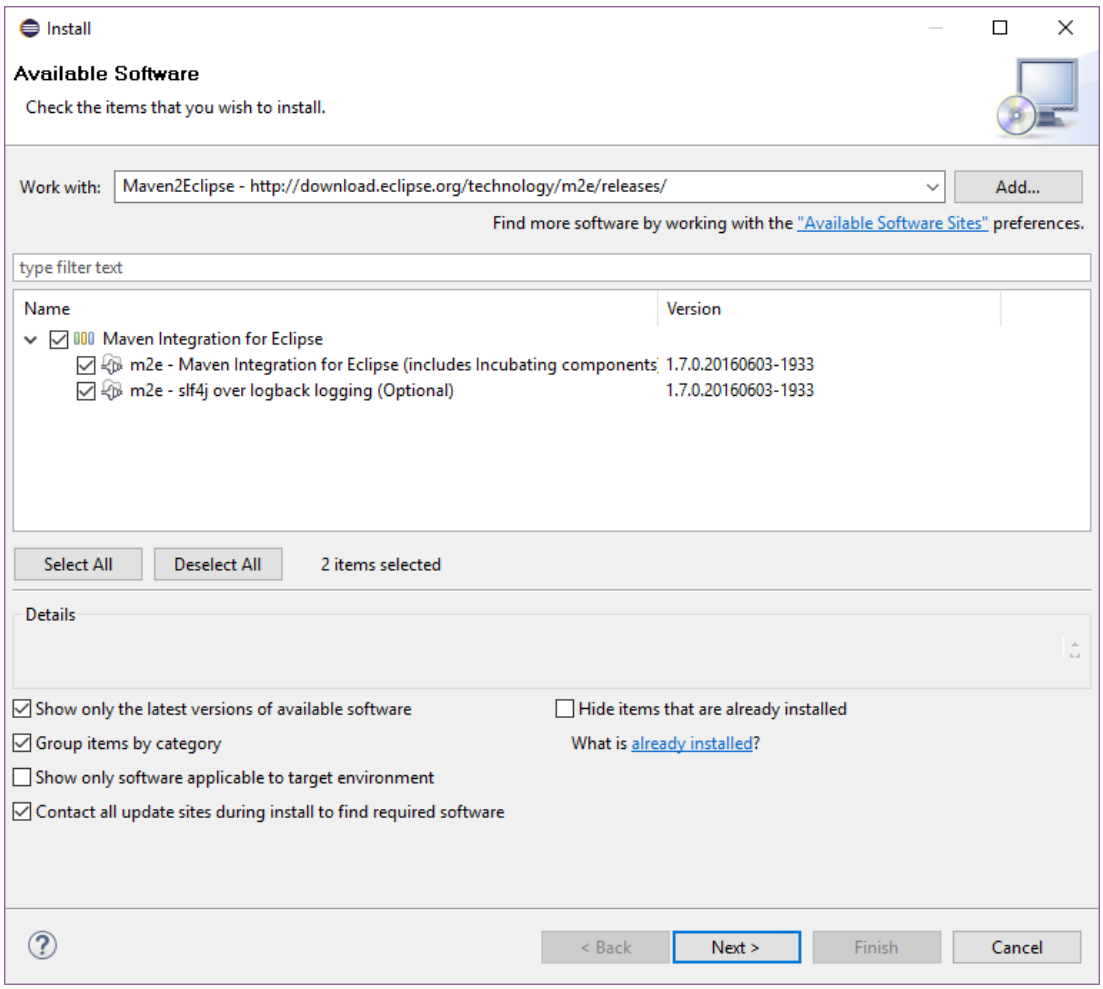


3. Install **Maven2Eclipse** components

    1. add the update site (http://download.eclipse.org/technology/m2e/releases/ *(http://download.eclipse.org/technology/m2e/releases/)*)

    2. install the components: **m2e - Maven integration for Eclipse**, **m2e - slfj over logback logging (Optional)**

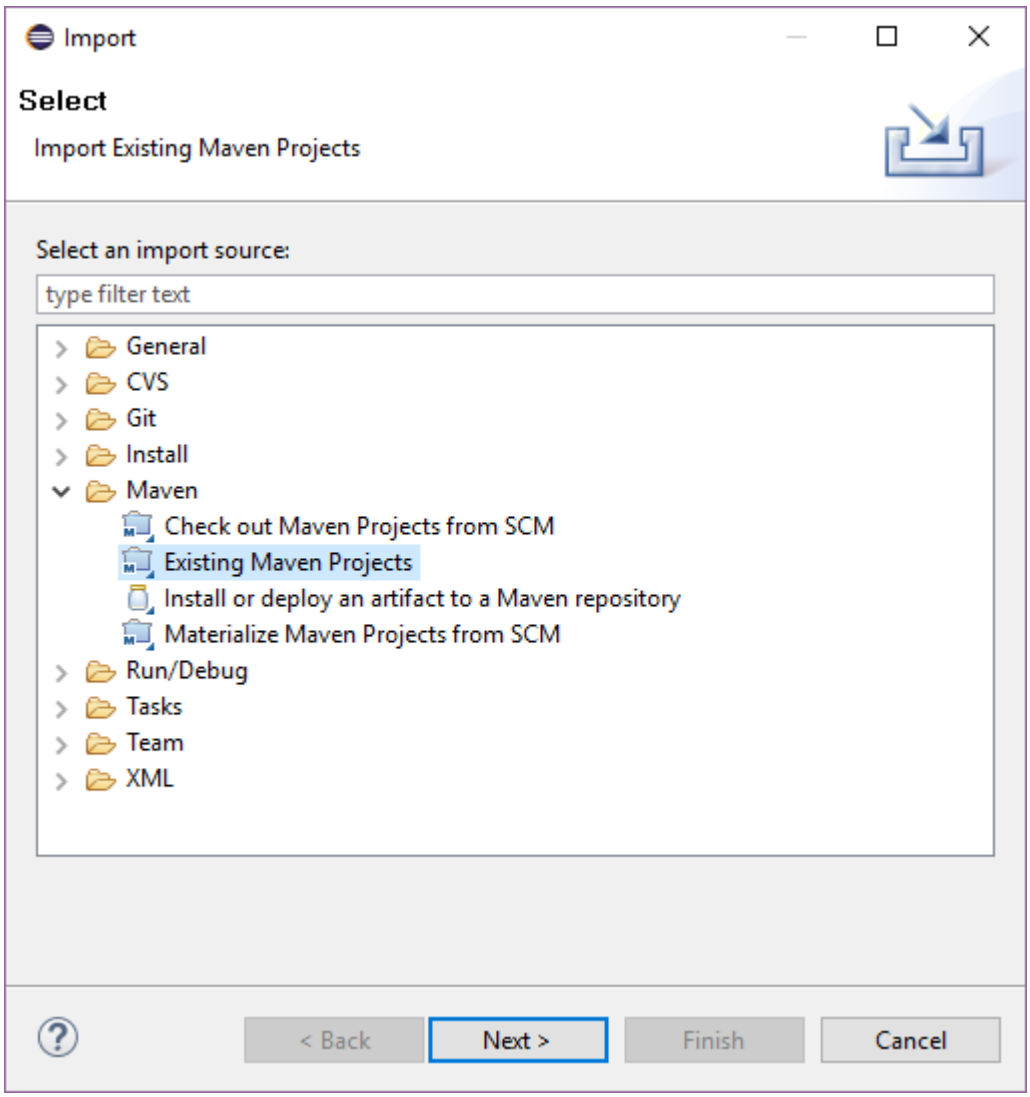
Copyright © 2013-2022 AnswerModules Sagl



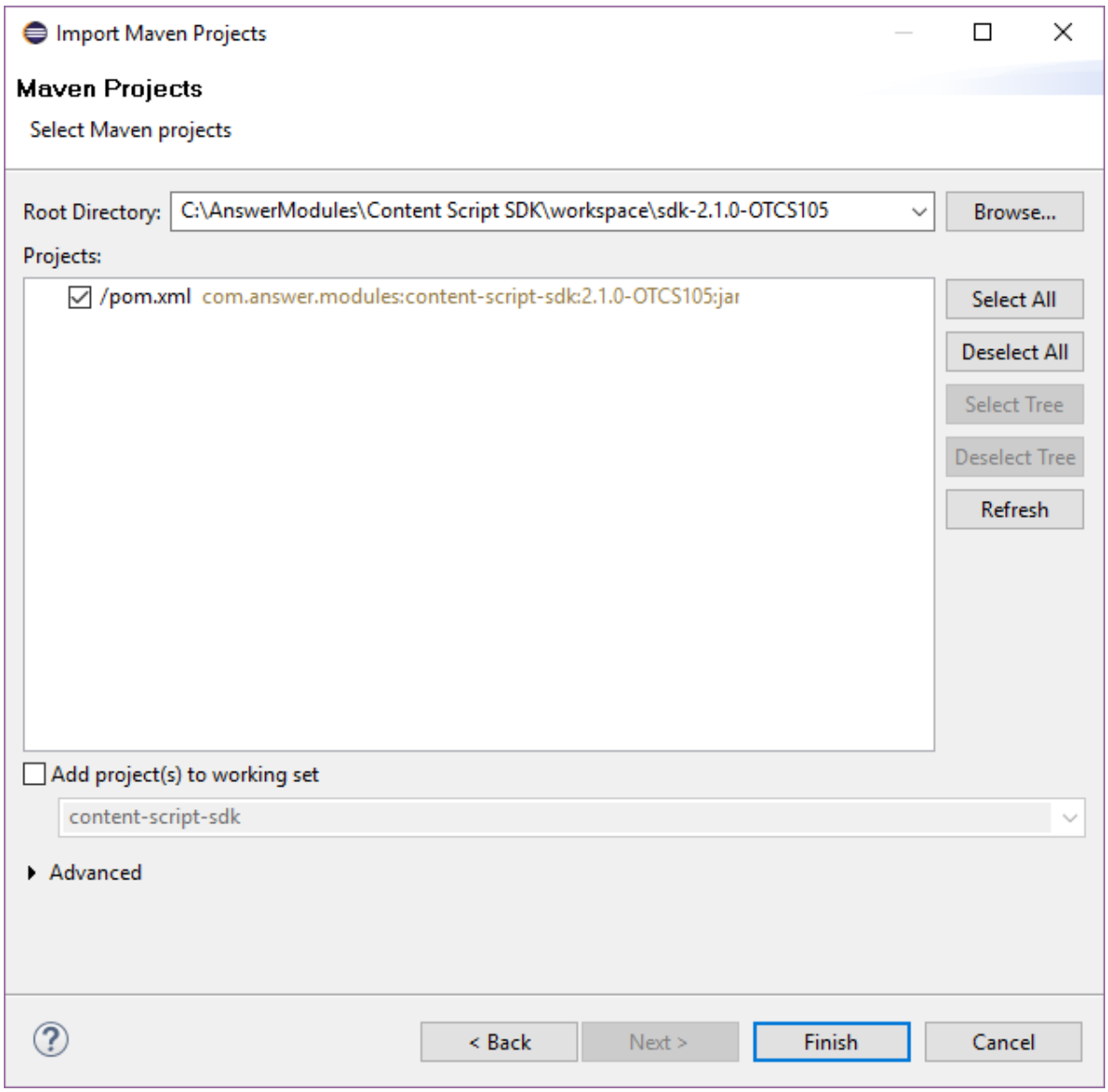
4. In your workspace folder, unpack the contents of the **Content Script SDK** archive

5. Import the unpacked project within your new Eclipse environment.

Copyright © 2013-2022 AnswerModules Sagl



Copyright © 2013-2022 AnswerModules Sagl

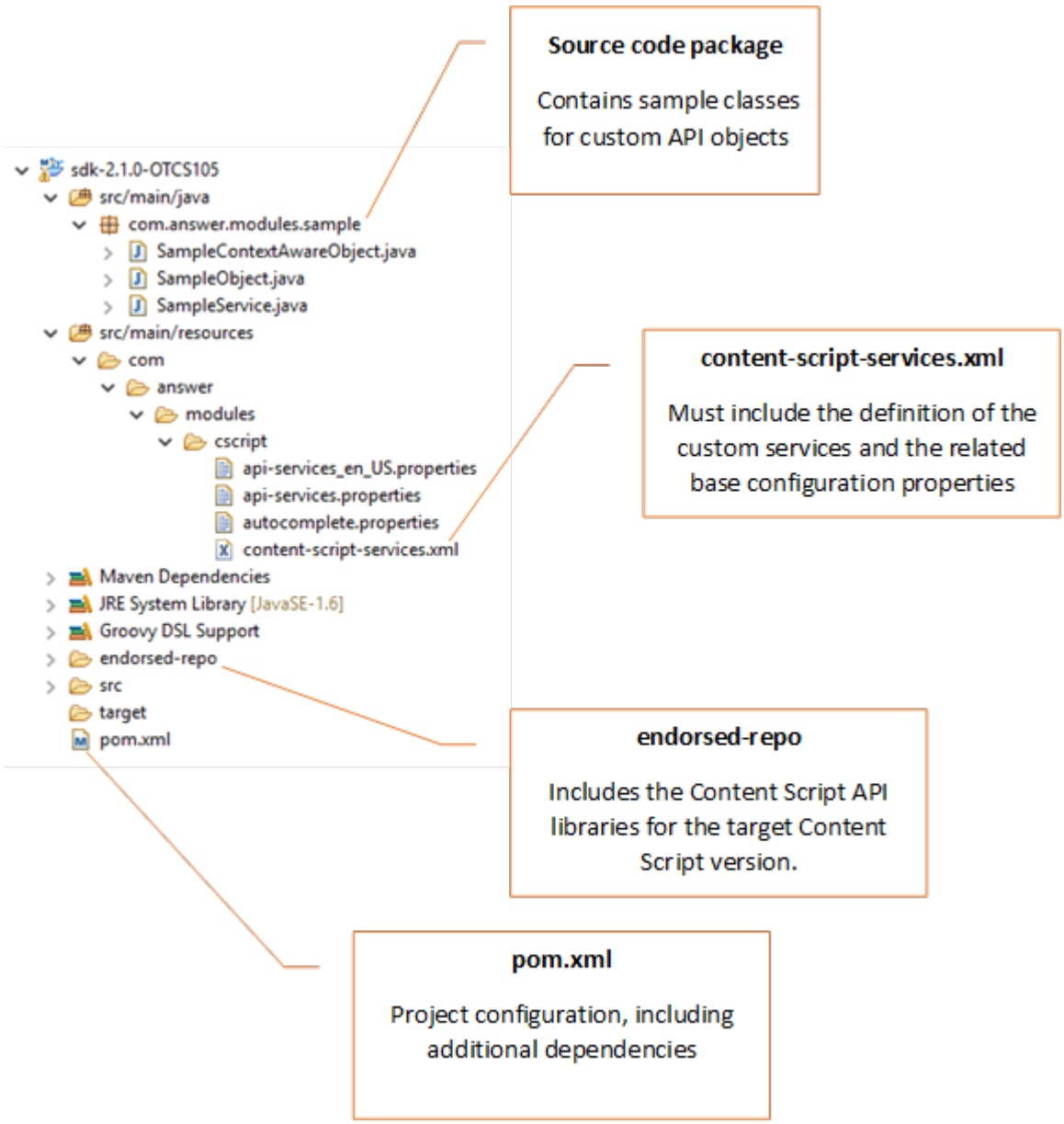


Navigate to the workspace folder and select the project directory, the project is identified by its pom.xml (Project Object Model) file. The Content Script SDK pom should appear in the listing.

Once selected, proceed with import.

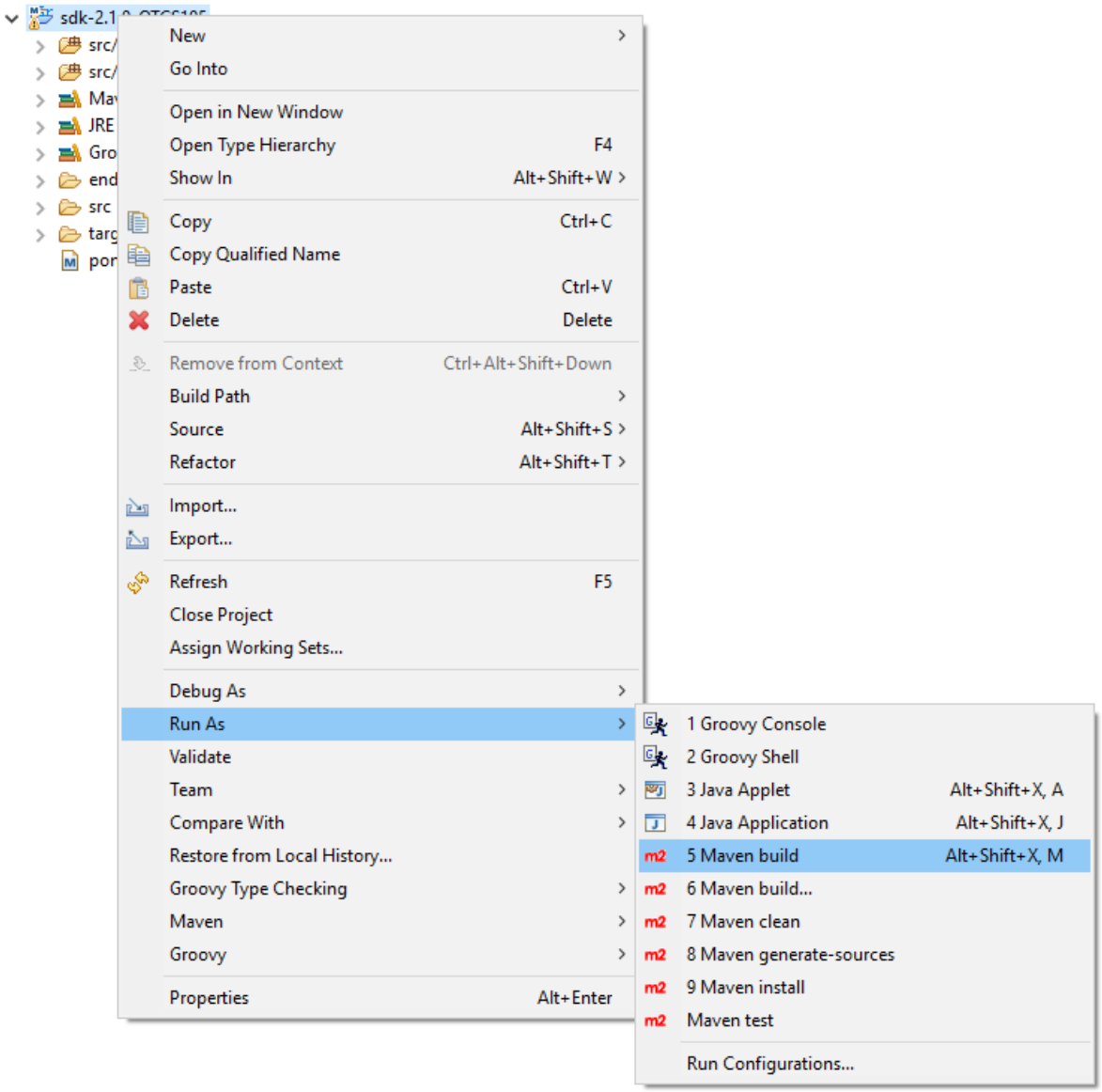
6. Review the imported SDK project layout

Copyright © 2013-2022 AnswerModules Sagl



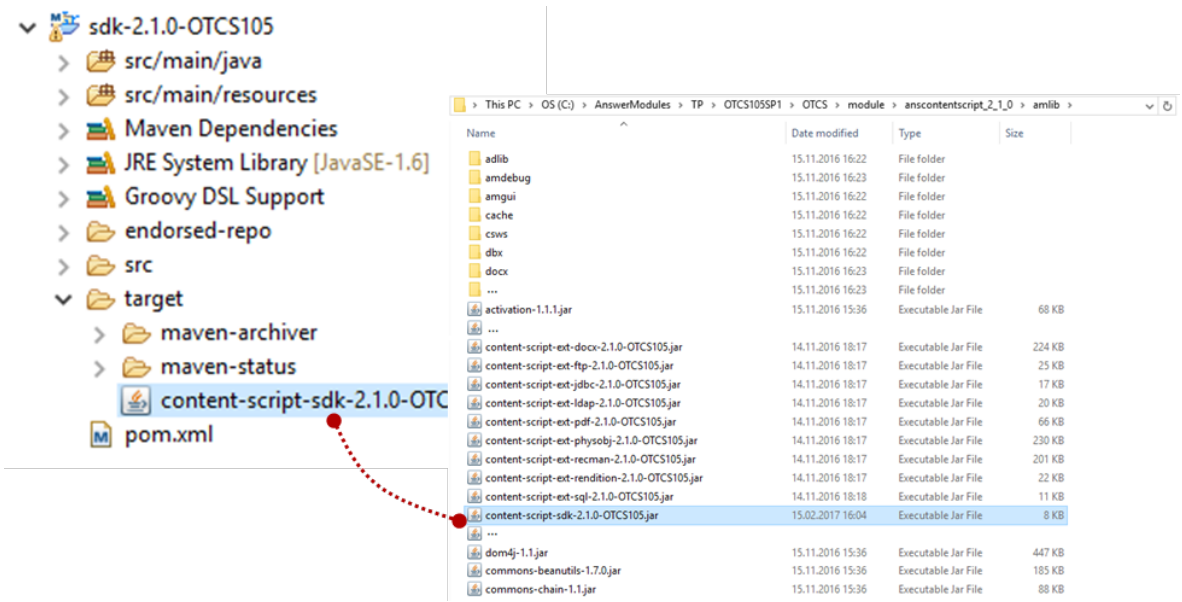
Copyright © 2013-2022 AnswerModules Sagl

7. Build the project using the Maven menu options.



8. Deploy the newly created service on your Content Server instance. The main artifact produced by a project build is a jar file containing the service classes. In order to install the custom services to the target OTCS instance, copy the jar file to: **<OTCS_Home>/module/anscontentscript_X_Y_Z/amlib**

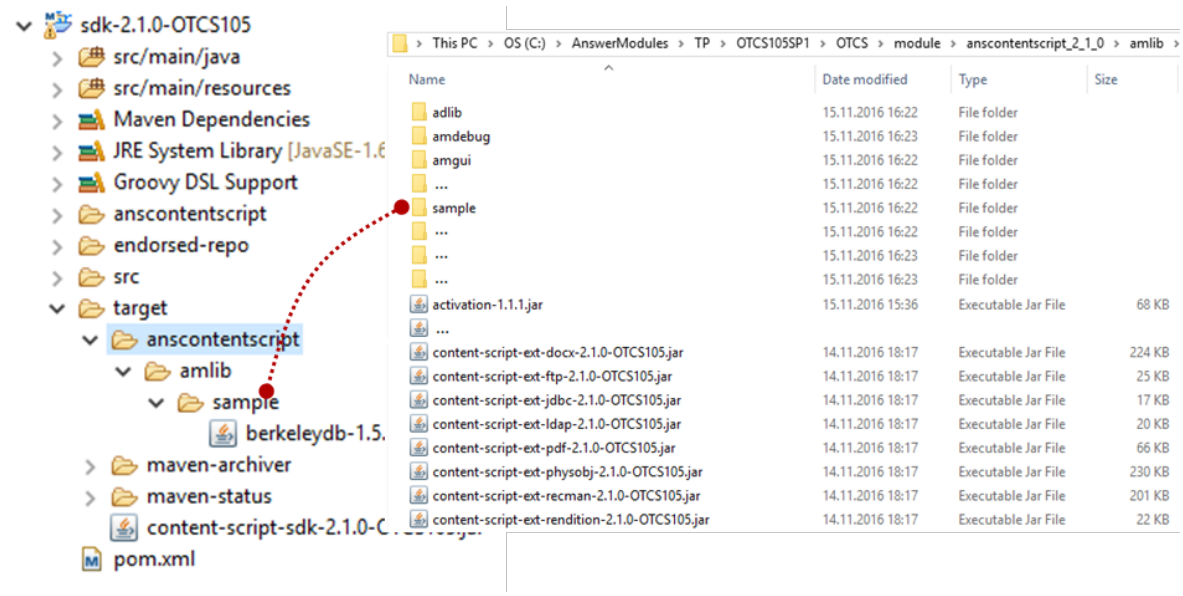
Copyright © 2013-2022 AnswerModules Sagl



Each service might load as many dependencies as it needs, service's dependencies are loaded with an high isolation level, thus several services might load the same dependency (same library) or even load different version of the same dependency (different version of the same library). Service's dependencies are loaded by default from a folder stored under /module/anscontentscript_X_Y_Z/amlib having the same name as the service identifier. Service's dependencies can be specified using the POM file you can find in the SDK project. E.g.

```
<dependency>
    <groupId>berkeleydb</groupId>
    <artifactId>berkeleydb</artifactId>
    <version>1.5.1</version>
</dependency>
```

Upon build, an additional target folder will include all direct and indirect dependencies needed at runtime:



Copyright © 2013-2022 AnswerModules Sagl

content-script-services.xml – Service description file¶

In order to let ModuleSuite be aware of your new service you have to properly describe it using the content-script-service.xml file. This xml files allows you not just to describe your service but also to provide some basic configuration for it.

The base structure of the file is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<services>
 <service id="sample" extRepoId="sample" class="com.answer.modules.sample.SampleService">
  <properties>
   <property name="sample.aProperty"
             description="A property with a default value (default: 'default')">default</property>
   <property name="sample.aSecret" type="hidden"
             description="A property with a hidden value"></property>
   <property name="sample.aNumber"
             description="A property with a numeric value (default: 1)">1</property>
  </properties>
 </service>
 <service id="anotherSample" extRepoId="sample" class="com.answer.modules.sample.ASampleService">
  <properties>
   <property name="sample.aProperty"
             description="A property with a default value (default: 'default')">default</property>
   <property name="sample.aSecret" type="hidden"
             description="A property with a hidden value"></property>
   <property name="sample.aNumber"
             description="A property with a numeric value (default: 1)">1</property>
  </properties>
 </service>
</services>
```

Using a single Content Scrip SDK project you can define as many services as you want. Each service should have its own service element descriptor in the description file. The mandatory attributes for the service element are: the service unique identifier (id) and the service implementation class (class). The extRepoId attribute is used if multiple services are defined in the same description file in order to inform ModuleSuite from where services' dependencies shall be loaded (in the above example both the services are loading their own dependencies from the same repository).

Content Script Extension for SAP¶

Using the extension¶

This section describes how to use the SAP API to retrieve data from the SAP system. The main Script API Object you are going to use is the SAPFunction object, which can be obtained from the sap service by calling sap.getFunction Script API Method. The SAPFunction object works the same for either an existing xECM connection or for a custom connection.

Copyright © 2013-2022 AnswerModules Sagl

```
def sapfunc = sap.getFunction("BAPI_TIMEQUOTA_GETDETAILEDLIST", "PRD")
```

Function's input parameters can be specified using the **setImpParam** method:

```
def sapfunc = sap.getFunction("BAPI_TIMEQUOTA_GETDETAILEDLIST", "PRD")
sapfunc.setImpParam("EMPLOYEENUMBER", cid)
sapfunc.setImpParam("DEDUCTBEGIN", now)
sapfunc.setImpParam("DEDUCTEND", now)
```

To invoke a function in the target system and retrieve the function's result just call the **execute** method of the SAPFunction object:

```
def sapfunc = sap.getFunction("BAPI_TIMEQUOTA_GETDETAILEDLIST", "PRD")
sapfunc.setImpParam("EMPLOYEENUMBER", cid)
sapfunc.setImpParam("DEDUCTBEGIN", now)
sapfunc.setImpParam("DEDUCTEND", now)
sapfunc.execute()
```

Function execution results¶

The extension package features several options that help you in properly manage a function's execution result:

1. Function export parameter is in Table form Get content of table parameter of function execution result, i.e. as SapTable Script API Object. See sample code below

```
//result as SAPTable class
def sapTblQuote = sapfunct.table("ABSENCEQUOTARETURNTABLE",
                            "QUOTATYPE",
                            "QUOTATEXT",
                            "DEDUCTBEGIN",
                            "DEDUCTEND",
                            "ENTITLE",
                            "DEDUCT",
                            "ORDERED",
                            "REST",
                            "REST_FREE",
                            "TIMEUNIT_TEXT" )

def quote = sapTblQuote.rows.collect{
            [
                "quotaType":it.QUOTATYPE,
                "quotaText":it.QUOTATEXT,
                "begin":it.DEDUCTBEGIN,
                "end":it.DEDUCTEND,
                "entitle":it.ENTITLE,
                "deduct":it.DEDUCT+it.ORDERED,
                "rest":it.REST_FREE
            ]}
```

Copyright © 2013-2022 AnswerModules Sagl

Please refer to SAPTable Script API Object for more detailed description of available methods and options.

1. Function export parameter is in Structure form Get content of a structure export parameter as a SapStructure Script API Object. See sample code below

```
def cumulateSAPStctr = sapfunct.table("CUMULATEDVALUES",
                                      "QUOTATYPE",
                                      "QUOTATEXT",
                                      "ENTITLE",
                                      "DEDUCT",
                                      "ORDERED",
                                      "REST",
                                      "REST_FREE",
                                      "TIMEUNIT_TEXT" )
//optionally you can call cumulateSAPStctr.getRows("QUOTATYPE","QUOTATEXT",...).collect()
def cumulate = cumulateSAPStctr.rows.collect{
                    [
                        "quotaType":it.QUOTATYPE,
                        "quotaText":it.QUOTATEXT,
                        "entitle":it.ENTITLE,
                        "deduct":it.DEDUCT+it.ORDERED,
                        "rest":it.REST_FREE
                    ]}
```

Please refer to SapStructure class API for more detailed description of available methods and options.

1. Get generic value of export parameter To get value of function export parameter you can use gertExportParam() method. Please see sample code below:

```
def empldet = sap.getFunction("Z_HR_MSD_RFC01_AD_EMPL_SINGLE", "PRD")
                    .setImpParam("I_PERNR", cid).execute()
                    .getExportParam("E_AD_EMPL")
```

All necessary conversions between Java and ABAP data types are done automatically.

Sample code listing below contains sample usage scenarios of SAP integration extension:

```
        // BAPI Function
getSAPHRData = {
    cid ->
    def now = new Date()
    def sapfunct = sap.getFunction("BAPI_TIMEQUOTA_GETDETAILEDLIST", "PRD")
                        .setImpParam("EMPLOYEENUMBER", cid)
                        .setImpParam("DEDUCTBEGIN", now)
                        .setImpParam("DEDUCTEND", now)
                        .execute()
    def quote = sapfunct.table("ABSENCEQUOTARETURNTABLE",
                            "QUOTATYPE",
                            "QUOTATEXT",
                            "DEDUCTBEGIN",
                            "DEDUCTEND",
                            "ENTITLE",
                            "DEDUCT",
                            "ORDERED",
                            "REST",
```

Copyright © 2013-2022 AnswerModules Sagl

```
                                "REST_FREE",
                                "TIMEUNIT_TEXT" ).rows.collect{
            ["quotaType":it.QUOTATYPE, "quotaText":it.QUOTATEXT, "begin":it.DEDUCTBEGIN, "end":it.DEDUCTI
        }
        def cumulate = sapfunct.table("CUMULATEDVALUES",
                                      "QUOTATYPE",
                                      "QUOTATEXT",
                                      "ENTITLE",
                                      "DEDUCT",
                                      "ORDERED",
                                      "REST",
                                      "REST_FREE",
                                      "TIMEUNIT_TEXT" ).rows.collect{
            ["quotaType":it.QUOTATYPE, "quotaText":it.QUOTATEXT, "entitle":it.ENTITLE, "deduct":it.DEDUCT
        }
        return ["quote":quote, "cumulate":cumulate]
    }


    quotaMap = getSAPHRData(cid)


    out << template.evaluateTemplate("""

    <div>
        #@cstable(['Quote', 'Begin', 'End', 'Entitle','Deduction', 'Rest'] {  '':'' } {  '':'' })
            #foreach(\$row in \$quotaMap.quote)
                <tr>
                    <td>\$row.quotaText</td>
                    <td>\$date.format('dd.MM.yyyy', \$row.begin)</td>
                    <td>\$date.format('dd.MM.yyyy', \$row.end)</td>
                    <td>\$row.entitle</td>
                    <td>\$row.deduct</td>
                    <td>\$row.rest</td>
                </tr>
            #end
        #end
    </div>

    """
    )
```

SAP service APIs¶

Method Summary	
SapFunction	**getFunction**(String functionName, String destinationName) Get a SAP function for the specified destination
SapFunction	**getFunction**(String functionName) Get a SAP function for the default destination ('default')

API Objects¶

SapField¶

Copyright © 2013-2022 AnswerModules Sagl

Method Summary	
SapField	**setValue**(Object value) Set the field value

Field Summary	
Object	**value** Get the field value

Extension: Classic UI

Customize an object's functions menu: CSMenu¶

Content Script can be used to perform changes to the standard object function menus, by adding new options or removing existing ones. This feature is enabled by defining a Content Script that “filters” the object menu and performs the desired modifications. The “**amgui**” service provides a user-friendly interface to perform modifications to the menu object.

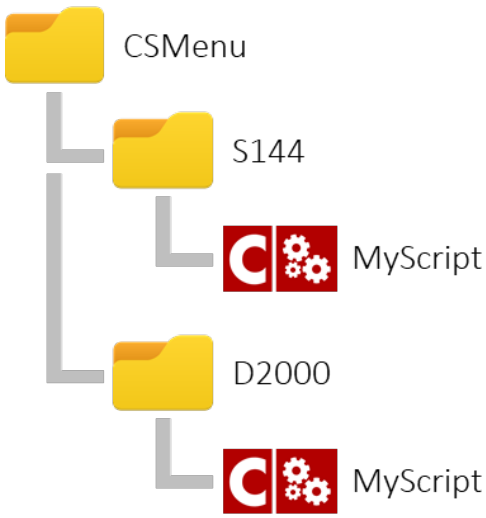
As for most other features configured through the Content Script Volume, a convention-over-configuration approach has been adopted.

The target container in which to place the Content Scripts is **CSMenu**. The first level under this container identifies the objects to which the customizations are applied. The naming convention is one of the following:

- `D<nodeID>`
- `S<subtype>`

where **nodeID** identifies the node unequivocally and **subtype** identifies a specific object subtype on Content Server.

Copyright © 2013-2022 AnswerModules Sagl



Examples:

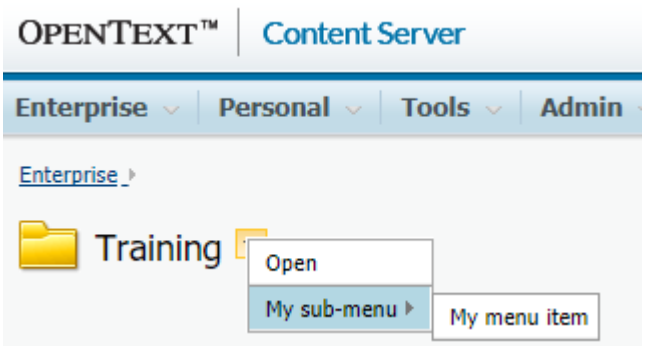
D2000 will change the function menu of the Enterprise Workspace

S144 will change the function menu of Document type objects (subtype: 144)

The following example shows a menu customization script that includes:

- fetching the original menu
- filtering the original menu entries (removing entries that match a specific expression)
- adding a divider row to split menu entries
- adding a submenu
- adding a custom menu entry to the new submenu
- returning the modified menu

E.g.




```
def csMenu = amgui.getCSMenu() //retrive the current object's menu
try{
    def node = docman.getNodeFast(nodeID)
    /**
```

Copyright © 2013-2022 AnswerModules Sagl

```
    A filter is a closure that returns true if the menu item shall be kept, false otherwise.
    In the filter function scope the object "it" represent the menu item.
    A menu item has the following properties:
    - name (string)
    - url (string)
    - openInNewTab (boolean ) *available only on 10.5
    - order (decimal)
    **/
    csMenu.filter {it.name == "Open"}
    csMenu.appendDivider() //use appendDivider(position) to specify a position

    def submenu = csMenu.appendSubMenu("My sub-menu") //use appendSubMenu(name, position) to specify
        submenu.appendItem("My menu item", "${url}?func=ll&objAction=properties&objId=${nodeID}&next
}catch(e){
    log.debug("Unable to apply changes to add items menu",e)
}

return amgui.returnCSMenu(csMenu)
```

Property	Type	Description
name	String	Label of the menu entry (only for menu items and submenus)
openInNewTab	Boolean	If true opens a new target browser window (only for menu items)
position	String	The order of the entry in the menu (available for menu items, submenus and dividers)
url	String	The target URL (only for menu items)

Notice that all operations are performed either through the **amgui** service or the **CSMenu** and **CSSubMenu** objects.

Return the proper value

The last operation performed in a CSMenu script should always be a call to the “returnCSMenu(...)” API of the amgui service

Customize a space's add-items menu: CSAddItems¶

Content Script can be used to perform changes to a container’s Add Item menu, by adding new options or removing existing ones. This feature is enabled by defining a Content Script that “filters” the menu and performs the desired modifications. The “**amgui**” service provides a user-friendly interface to perform modifications to the menu object.

As for most other features configured through the Content Script Volume, a convention-over-configuration approach has been adopted.

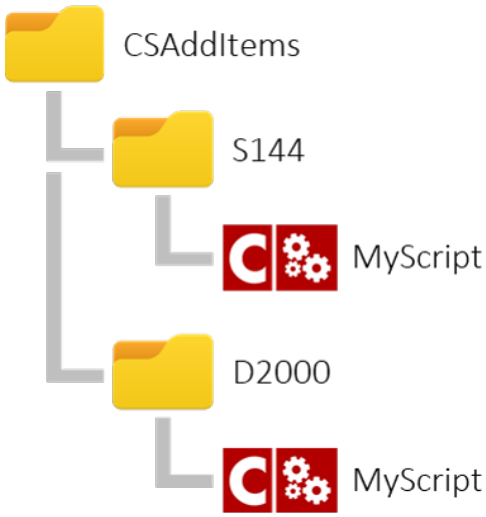
The target container in which to place the Content Scripts is **CSAddItems**. The first level under this container identifies the objects to which the customizations are applied. The naming convention is one of the following:

- `D<nodeID>`

Copyright © 2013-2022 AnswerModules Sagl

- S<subtype>

where **nodeID** identifies the node unequivocally and **subtype** identifies a specific object subtype on Content Server.



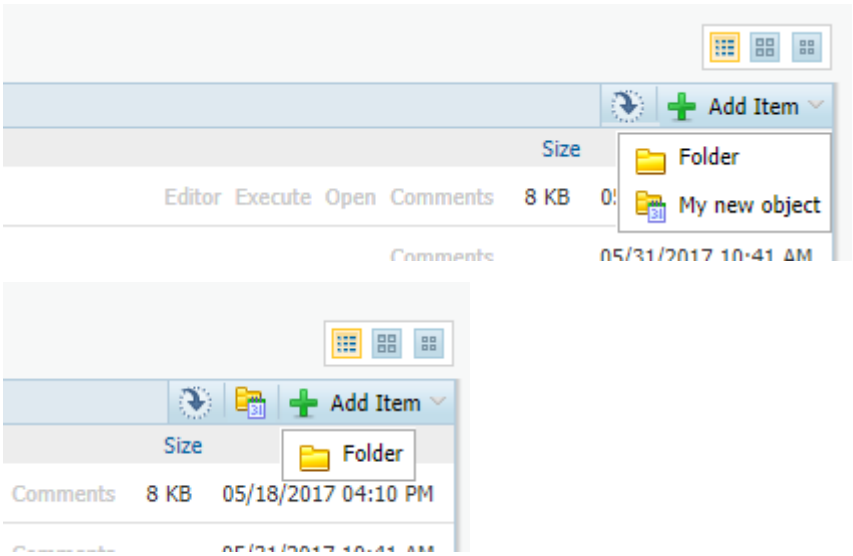
Examples:

D2000 will change the add items menu of the Enterprise Workspace

The following example shows a menu customization script that includes:

- filtering the original menu entries (removing entries that match a specific expression)
- adding a custom menu entry
- returning the modified menu

E.g.



Copyright © 2013-2022 AnswerModules Sagl

```
try{
    //The current space
    def node = docman.getNodeFast(nodeID)
    /**
        Other possibile filter examples:
        it.name == "Folder"
        it.subtype == 0
    **/
    amgui.filterAddItems {
        it.name == "Folder"
    }
    /**
        Other possibile filter examples:
        it.name  == "Folder"
        it.subtype == 0
    **/
    amgui.filterAddItems ({false}, true)
    amgui.addBrowseViewAddItem(
        amgui.newBrowseViewAddItemsMenu().builderUrl().setImg("${img}folder_icons/folder5.gif")
                                                      .setName("My new object")
                                                      .setPromoted(true)
                                                      .setUrl("${url}?func=ll&objAction=create&objTy
    )
}catch(e){
    log.debug("Unable to apply changes to add items menu",e)
}

return amgui.returnAddItemsMenus()
```

Invoke a Content Script

The url of the menu entry could be used to pass parameters to a custom Content Script that will perform the desired operations.

Return the proper value

The last operation performed in a CSAddItems script should always be a call to the “returnAddItemsMenu(...)” API of the amgui service

Customize a space's buttons bar: CSMultiButtons¶

Multi-action buttons can be added, removed or modified by using an approach similar to the CSMenu customization. In this case, customization scripts should be added in the **CSMultiButtons** container. The container structure is the same as the one described for the CSMenu.

As for most other features configured through the Content Script Volume, a convention-over-configuration approach has been adopted.

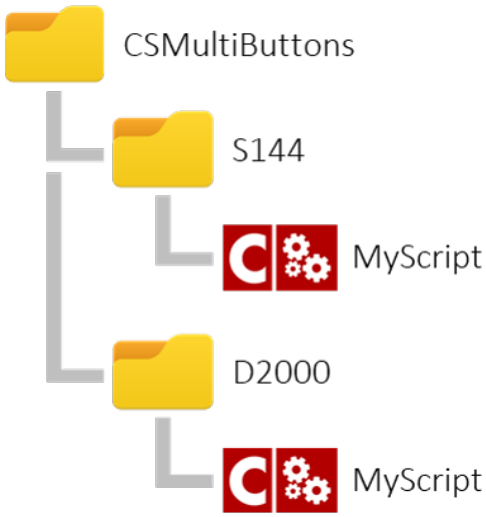
The target container in which to place the Content Scripts is **CSMultiButtons**. The first level under this container identifies the objects to which the customizations are applied. The naming convention is one of the following:

- `D<nodeID>`

Copyright © 2013-2022 AnswerModules Sagl

- S<subtype>

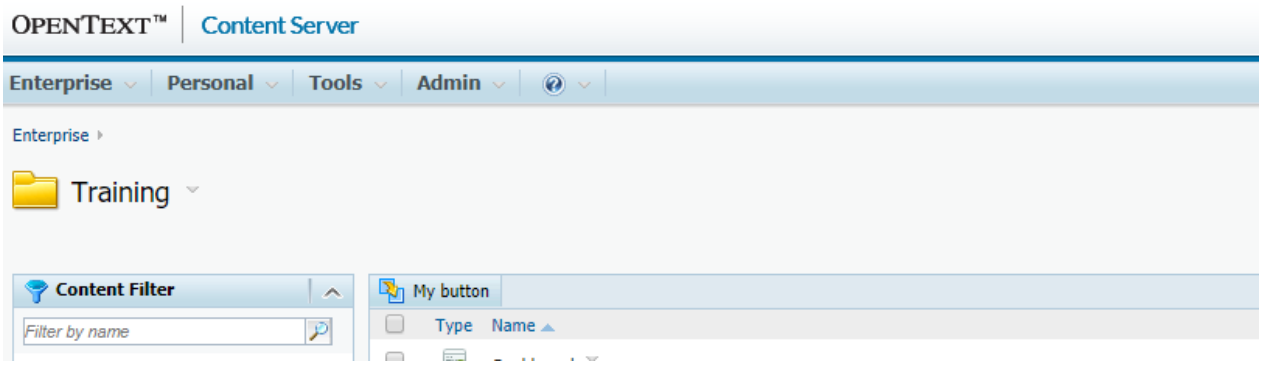
where **nodeID** identifies the node unequivocally and **subtype** identifies a specific object subtype on Content Server.



Examples:

D2000 will change the buttons bar menu of the Enterprise Workspace

E.g.




```
try{
    amgui.addBrowseViewMultiItemButton(
        amgui.newBrowseViewMultiItemButton()
            .builder()
             .setOrder(1100)
             .setJavascriptFunctionName('runContentScript')
             .setJavascriptFile("anscontentscript/js/contentScriptMultifileBar.js")
             .setImageMap("anscontentscript/contentscriptmultifilebar.png")
             .setImageXPos(0)
             .setImageYPos(0)
             .setImageXPosAlternative(-268)
             .setImageYPosAlternative(0)
             .setDisplayName('My button')
             .create()
    )
    /**
        Properties that can be used to filter the buttons bar:
        - action (the request handler to be executed e.g. ll.ProcessMultiCopy)
```

Copyright © 2013-2022 AnswerModules Sagl

```
            - Order
            - Name
            - DisplayName
            - ExecutesOnClient

        **/
        amgui.filterBrowseViewMultiItemButton {it.name == "mybutton"}

    }catch(e){
        log.debug("Unable to apply changes to add multi items buttons bar",e)
    }
    return amgui.returnBrowseViewMultiItemButtons()
```

where the following fields are mostly relevant:

Property	Type	Description
ImageMap	String	The path of the image map file (in the Support folder) containing the button icon
ImageXPos, ImageXPos2, ImageYPos, ImageYPos2	Integer	The coordinates of the portion of the image map to use for the button (normal and on mouse over)
Order	String	The order of the button in the menu bar
Type	String	The button type (should be “Content Script”)
ExecutesOnClient	boolean	If the button logic is on the client side (should be “true”)
DisplayName	String	The button label
Name	String	The name of the button
JavascriptFile	String	The javascript resource in which the function controlling the button behavior is defined
JavascriptFunctionName	String	The javascript function defined in the JavascriptFile that controls the button behavior

Invoke a Content Script

A sample Javascript file (contentScriptMultifileBar.js) is located in the Content Script Module support folder. Create a customized version of this file when adding new actions.

Customize a space's displayed columns: CSBrowseViewColumns¶

Content Scripts located in the **CSBrowseViewColumns** container can be used to perform modifications to how columns are presented in the standard Content Server Browse View.

Copyright © 2013-2022 AnswerModules Sagl

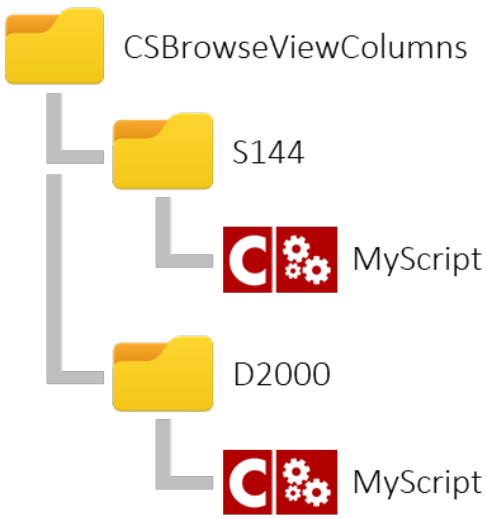
The modifications can be limited to specific portions of Content Server. This feature is enabled by defining a Content Script that “filters” the browse view columns configuration and performs the desired modifications. The “**amgui**” service provides a user-friendly interface to perform the modifications.

As for most other features configured through the Content Script Volume, a convention-over-configuration approach has been adopted.

The target container in which to place the Content Scripts is **CSBrowseViewColumns**. The first level under this container identifies the objects to which the customizations are applied. The naming convention is one of the following:

- `D<nodeID>`
- `S<subtype>`

where **nodeID** identifies the node unequivocally and **subtype** identifies a specific object subtype on Content Server.



Examples:

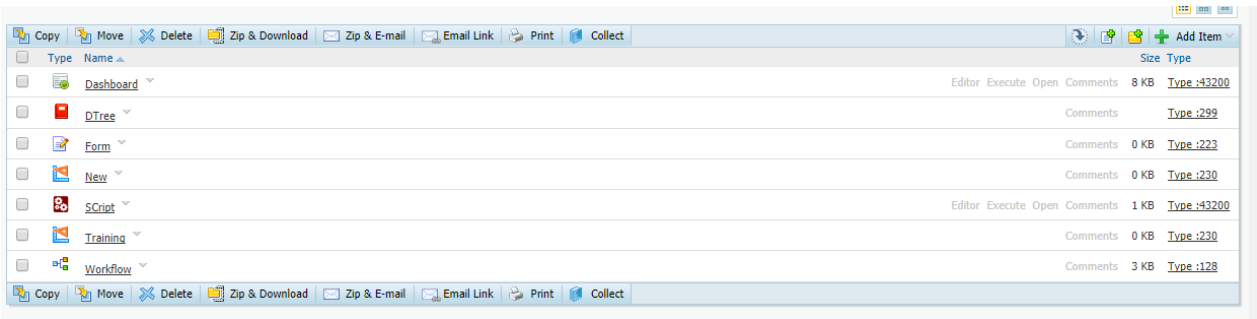
D2000 will change the columns visible in the Enterprise Workspace

The following example shows a browse view columns customization script that includes:

- create a new column using the builder
- filtering the original columns list (removing entries that match a specific expression)
- adding the column to the view
- returning the modified columns list

E.g.

Copyright © 2013-2022 AnswerModules Sagl




```
try{
    /**
    A browse view column is quite a complex object. The amgui service provides you with a builder in
    **/
    def columnBuilder = amgui.newBrowseViewColumn().builder()

        .setColumnName("type")   // Column name corresponds to the property
                                 //from the browse view row that will be used
                                 //to populate the column.
        .setDisplayName("Type") // Column display name is the label used for the column.
        .setAlignment("left")
        .setSortable(true)       // If sortable the Javascript sorting
                                 //function will look for a property named:
                                 // columnName+'SortStr' or columnID+'SortStr'
                                 // to perform sorting
        .setColumnEMWidth(1.0)
        .setDisplayAsLink(true)
        .setNewWindow(true)
        .setUrl("${url}?param=%value%")       // The url to be opened.
                                              // The following placeholder
                                              // can be used in the expression:
                                              // %value%, %objid%, %rawvalue%, %nexturl%"

        .setFormatValueMask("Type :%value%") // The format mask to be used to
                                              // present the column value.
                                              // The following placeholder
                                              // can be used in the expression:
                                              // %value%, %objid%, %rawvalue%, %nexturl%"
    /**
    A filter is a closure that returns true if the column shall be kept, false otherwise.
    In the filter function scope the object "it" represent the column object.
    For default columns the only attribute available is columnID (string) which might have one out tl
    dataidColumn, dateColumn, arbitraryColumn, columnWithURL, userColumnWithURL)

    All the other columns have the following properties:
    DisplayAsLink (boolean), DisplayValue (string), NewWindow (boolean), NewWindowTitle (string), URI

    **/
    amgui.filterBrowseViewColumn {
        it.columnID != "dateColumn"
    }
    amgui.addBrowseViewColumn(columnBuilder.create())

}catch(e){
    log.debug("Unable to apply changes to add items menu",e)
}

return amgui.returnBrowseViewColumns()
```

The following properties are available for each column object (they are managed through a builder *(https://en.wikipedia.org/wiki/Builder_pattern)* the `CSBrowseViewColumnBuilder` obtained :

Copyright © 2013-2022 AnswerModules Sagl

Property	Type	Description
isDefault	boolean	True if the column has a Javascript definition
sortable	boolean	True if the column has a Javascript definition
DisplayAsLink	boolean	The value of the column will be wrapped into an HTML link
DisplayValue	String	The column's value
NewWindow	boolean	If DisplayAsLink = true, opens the link in a new window
NewWindowTitle	String	If DisplayAsLink = true, the title of the window in which link will be opened
Url	String	If DisplayAsLink = true, the URL to be used for building the link
alignment	String	Column alignement. One out: 'left', 'right', 'center'
columnID	String	Column unique identifier
columnName	String	Column name
displayName	String	Column name as it will be displayed in the page
displayName	String	Column name as it will be displayed in the page

Filtering columns - lines from 39 to 41

A filter is a closure that returns true if the column shall be kept, false otherwise.

Default Columns¶

Default columns are columns for which a Javascript column definition exists. Default columns Javascript definitions can be found in webnode/browse.js file. The following **default** columns definition should exist in your environment:

Value	Description
checkBoxColumn	Used for selecting multiple nodes
typeColumn	Represents the node's type in the form of a web-icon
nameWthPrmtdCmdsColumn	Name with promoted commands column
sizeColumn	Size of the document or number of items in the space
dataidColumn	Node's unique system identifier
dateColumn	Node's last modification date
arbitraryColumn	Template for other columns (ABSTRACT)
columnWithURL	Template for other columns (ABSTRACT)
userColumnWithURL	Node's owner

Copyright © 2013-2022 AnswerModules Sagl

The amgui service features a method that can help you in creating your own custom column Javascript definition on the basis of a template that is stored in the Content Script Volume (`CSVolume:CSGui:BrowseViewColumnDefinition`). The custom Javascript column's definition can be rendered, for example, as part of a customview, an appearance or a Content Script

```
amgui.getBrowseViewColumnDefinition(
                                    String columnID, //The id of the column
                                    Map templateContext, // A map to be used as model for
                                                         // the column's definition template
                                    [,CSDocument param ] // An optional template document.
                                                         // If none is provided the default
                                                         // CSVolume:CSGui:BrowseViewColumnDefinitio
                                                         // will be used
                                    )
```

Here below a real-world usage example. The Script is used to create a custom view within the space in which is stored.

```
jsAddCell = """
        var cell;

        try
        {

            cell = rowStruct.insertCell( cellCount++ );
            cell.className = this.cellClassName;
            if ( true === this.nowrap )
            {
                cell.style.whiteSpace = 'nowrap';
            }
            cell.innerHTML = this.getCellValue( dataRow, rowNo );


        }
        catch(e)
        {
            exceptionAlert( e, "Issue occured in browse.js/htmlColumn.AddCell." );
        }
        return cellCount;
"""

jsGetCellValue ="""
    var val = dataRow[ 'pstatus' ];
    if ( val == undefined )
    {
        val = "";
    }
    return val;
"""

def customView = docman.getTempResource("customView", ".html")

customView.content.withWriter{
    it << amgui.getBrowseViewColumnDefinition("pstatus",
                                              ["jsAddCell":jsAddCell, "name":"Status", "jsGetCellVal
}
def cv = docman.createCustomView(self.parent, "customView", customView.content)
cv.setIsHidden()
cv.update()
```

Copyright © 2013-2022 AnswerModules Sagl

For **default** columns (listed in the table above) the only attribute available is **columnID** (string).

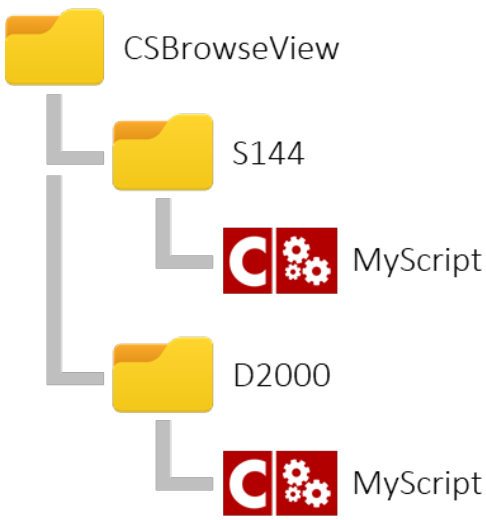
Customize a space content view: CSBrowseView¶

Content Scripts located in the **CSBrowseView** container can be used to perform modifications on the content of a browse view.

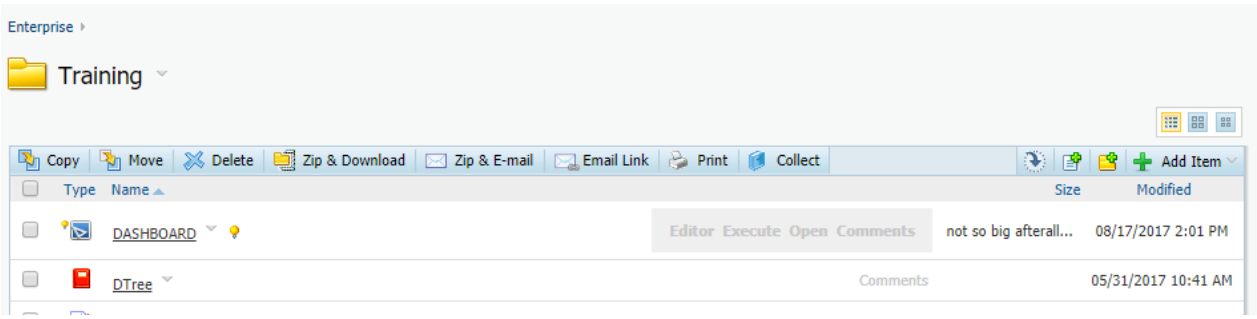
The target container in which to place the Content Scripts is **CSBrowseView**. The first level under this container identifies the objects to which the customizations are applied. The naming convention is one of the following:

- `D<nodeID>`
- `S<subtype>`

where **nodeID** identifies the node unequivocally and **subtype** identifies a specific object subtype on Content Server.



The following example shows a browse view customization script that will iterate on each row in the browse view and perform modifications for objects of subtype 43200 (Content Scripts)




```
try{

    /**
        Properties that can be used to filter the browse view rows:
```

Copyright © 2013-2022 AnswerModules Sagl

```
        - dataId (Numeric)
        - name (String/Html will be rendered inside an 'a' tag)
        - link (String)
        - size (String/Html e.g. '1 KB')
        - date (String e.g.
        - imgStr (String)
        - imgLargeStr (String)
        - imgThumbnailStr
        - promotedCmds (Html)
        - modifiedImgs (Html)
        - imgStatus (String)
        - statusName (String)


    **/
    amgui.filterBrowseView { row ->

        // Just for Content Scripts
        if(row.type == "43200"){
            row.checked = true
            row.name = "${row.name.toUpperCase()}"
            row.promotedCmds = """ <div style="font-weight:bold;background-color:#EFEFEF;padding:10p
            row.modifiedImgs =  "<img src='${img}webnode/new.gif' />"

            row.imgStatus  = "${img}webnode/new.gif"
            row.statusName = "Ready to be executed"
            row.link ="http://www.answermodules.com/products/content-script"
            row.size = "not so big afterall..."

            row.date = amgui.formatDateForBrowseView(new Date())  //This is a shortcut to format dat

            row.imgStr = "${img}anscontentscript/lib/img/icons/product-design.png"
            row.imgLargeStr = "${img}anscontentscript/lib/img/icons/product-design_large.png"
            row.imgThumbnailStr = "http://www.answermodules.com/img/content-script/content-script-ba
        }

        // This to be sure that the rows will be rendered
        return true
    }
}catch(e){
    log.debug("Unable to filter browse view rows for node {}", nodeID, e)
}
return amgui.returnBrowseViewRows()
```

Filtering rows - lines from 20 to 42

The filtering closure passed as parameter to the amgui.filterBrowseView(...) method should return a boolean value of “true”. If “false”, the row will not be rendered.

Add a new row

It is possible to add new rows from scratch by using the amgui.addBrowseViewRow(...) method. A blank row template can be obtained through the amgui.newBrowseViewRow() method

The following properties are available for filtering or modification on each **row** object that is being iterated:

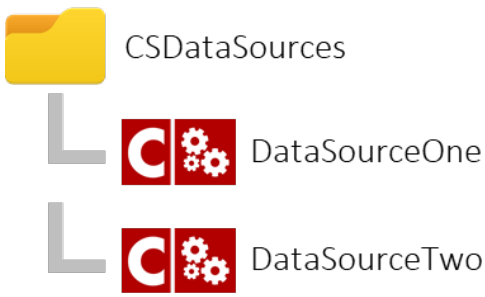
Property	Type	Description
dataId	Numeric	The node's unique identifier

Copyright © 2013-2022 AnswerModules Sagl

Property	Type	Description
name	String/ HTML	The node's name Html will be rendered inside an 'a' tag
link	String	The link to be associated to the node's name
size	String/ HTML	The node's side e.g. '1 KB'
date	String	The node's last modification date
imgStr	String	The url for the node's icon
imgLargeStr	String	The url for the node's icon when the node is featured
imgThumbnailStr	String	The url for the node's thumbnail
promotedCmds	HTML	The HTML code containing links to the node's promoted functions (can be any HTML)
modifiedImgs	HTML	The HTML code to be used to notify users that the node's has been modified
imgStatus	String	The url for the node's status icon
statusName	String	The node's status name

Create a custom column backed by Content Script: CSDataSources¶

Since version 1.5 Content Scripts can be used as Column Data sources. Content Scripts placed in the **CSDataSources** Template Folder will automatically be available as Column Data Sources.



The CSDataSource scripts will automatically be invoked by Content Server for each node of the system, and the resulting value will be used as a column value.

Return the proper value

A CSDataSource Content Script **MUST** always return a String object.

Copyright © 2013-2022 AnswerModules Sagl

In the Content Script code, the execution context *(/working/contentscript/scripts/#execution-context)* will be enriched by the framework with the following information related to the current node:

- volumeID
- parentID
- dataID
- createDate
- modifyDate

As per standard column data sources the developer is in charge of defining and implementing a reliable updating strategy. Most of the time the task can be accomplished implementing either a synchronous or an asynchronous (see Managing events *(/administration/csvolume/#cssynchevents-and-csevents)*) event script.

As a matter of fact, Content Script features two different APIs that can be used to update columns' datasources values.

```
docman.updateColumnValue( CSNode node, //The node for which you want to update the column's value
                          String dataSourceId, // The standard identifier for the column's datasource
                          String columnValue // The new value for the column
                        )

docman.updateContentScriptColumnValue( CSNode node, //The node for which you want to update the colu
                          String scriptName, // The name of the Content Script script that serves
                                             // datasource
                          String columnValue // The new value for the column
                        )
```

The first one is supposed to be used with standard columns' datasources, the latter with Content Script backed columns' datasources.

The **updateContentScriptColumnValue** takes as secondo parameter the name of the Script used to implement the column's datadasouce.

The **updateColumnValue** method takes as second parameter a **dataSourceIdentifier**, which can be easily determined inspecting the ExtendedData column's value of the corresponding Column object on the DTree table (property "**dataSource**").

E.g.

```
def exData = sql.runSQL( """ select ExtendedData EXT
                             from   DTree
                             where  DataId = %1 """,
                         false,
                         false,
                         -1,
                         2109 //The column object DataId
```

Copyright © 2013-2022 AnswerModules Sagl

```
                        ).rows[0].EXT
out << extData.getMapFromOscript().dataSource //Returns sys_CreateDate
                                              //(on most of the systems)
```

Copyright © 2013-2022 AnswerModules Sagl

Beautiful WebForms

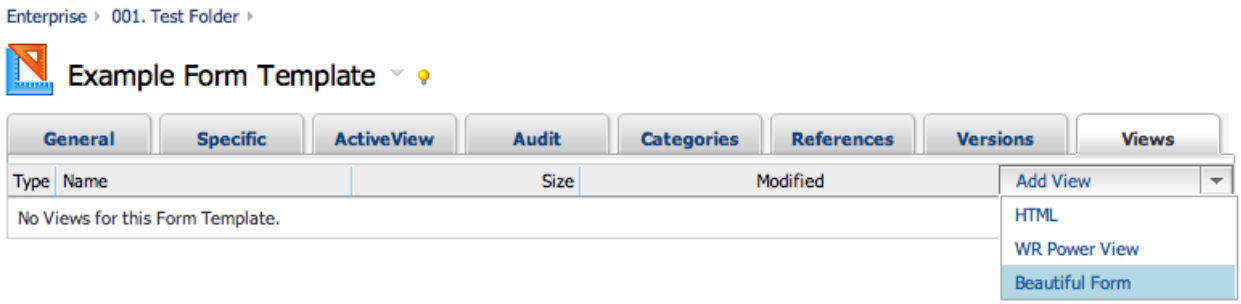
Content Server object

Beautiful WebForms views are document-class objects on Content Server.

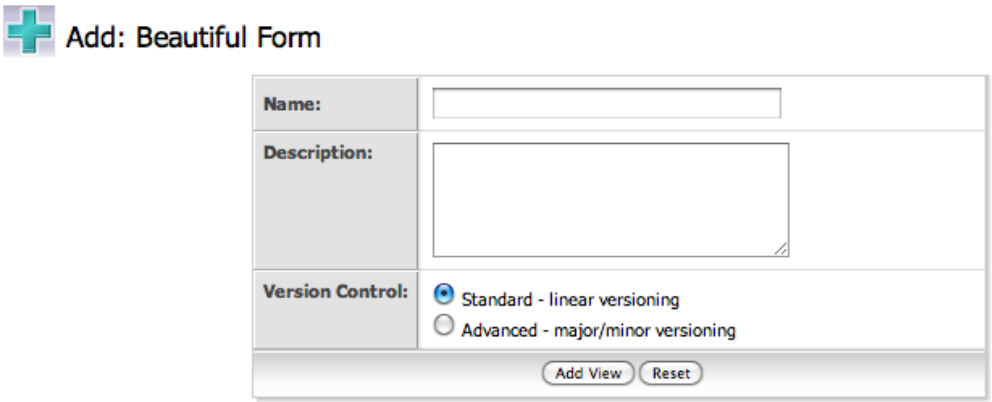
Being standard objects, Beautiful WebForms views comply with Content Server **permissions** model. Upon creation, the object can be edited with the web-based IDE selecting the **'Form Builder'** function in the object function menu.

Creating a Beautiful WebForms View¶

Beautiful WebForms views can be created in the same way as standard html views. In the 'views' tab of the 'form template', an additional 'Beautiful Form' entry will be available in the 'add view' dropdown menu.

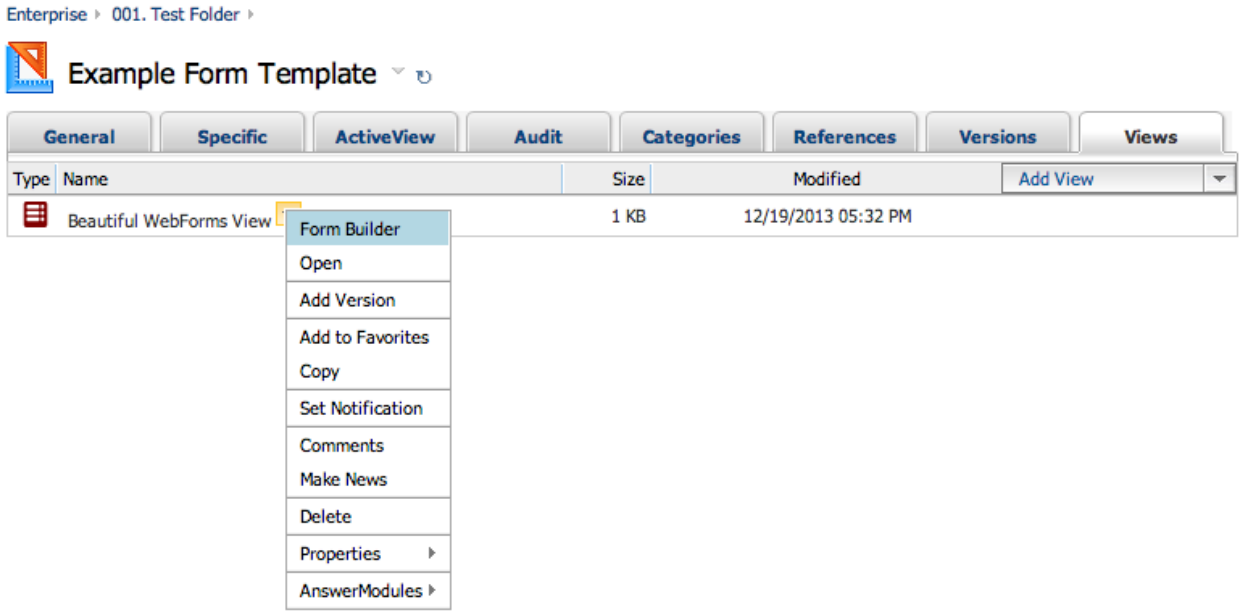


As per standard views, the creation requires a view name be specified. Standard versioning options apply to form views.

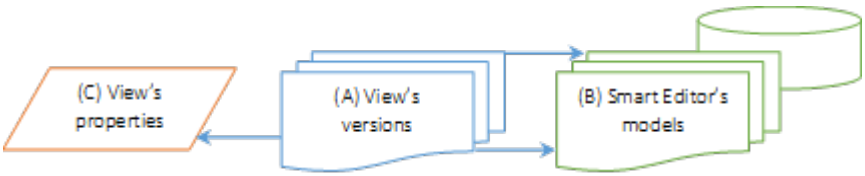


Copyright © 2013-2022 AnswerModules Sagl

Upon creation, the view can be edited with the web-based IDE selecting the **'Form Builder'** option in the object options menu.



Understanding the view object¶



Beautiful WebForms views are much more than simple html-views. They are **active** objects that can be used to create very complex applications. In order to implement all their additional functionalities, Beautiful WebForms views are decorated with a set of information used by the Beautiful WebForms framework for determining how to render, and how to display form's data within them.

In the image above a simplified representation of the information that constitutes a Beautiful WebForms view is highlighted:

- **(A)** View's versions: Beautiful WebForms views are standard FormTemplate's views thus versioned document-class objects. Each version is, in the very end, nothing but a Velocity *(http://velocity.apache.org/)* template document (HTML code + template expressions).
- **(B)** For each version created with the FormBuilder's smart-editor the BWF framework archives the smart-editor view's "model" into an internal database table. The smart-editor view's model is constituted by the list of the configurations used for each widget that build the view.

Copyright © 2013-2022 AnswerModules Sagl

- (C) View's properties: Beautiful WebForms views are associated with a set of predefined properties persisted as the object's extended data. These properties are related just to the last view's version.
  The view's predefined properties are:

  1. Form Builder mode used for creating the current view's version (either "source code" or "smart editor",
  2. The list of static "css" view's dependecies dynamically determined on the basis of the widgets used to build the view
  3. The list of static "javascript" view's dependecies dynamically determined on the basis of the widgets used to build the view
  4. The number of view's columns
  5. The identifier of the library of widgets used to build the view
  6. The ID of the view template (if any) associated to the view

Form builder

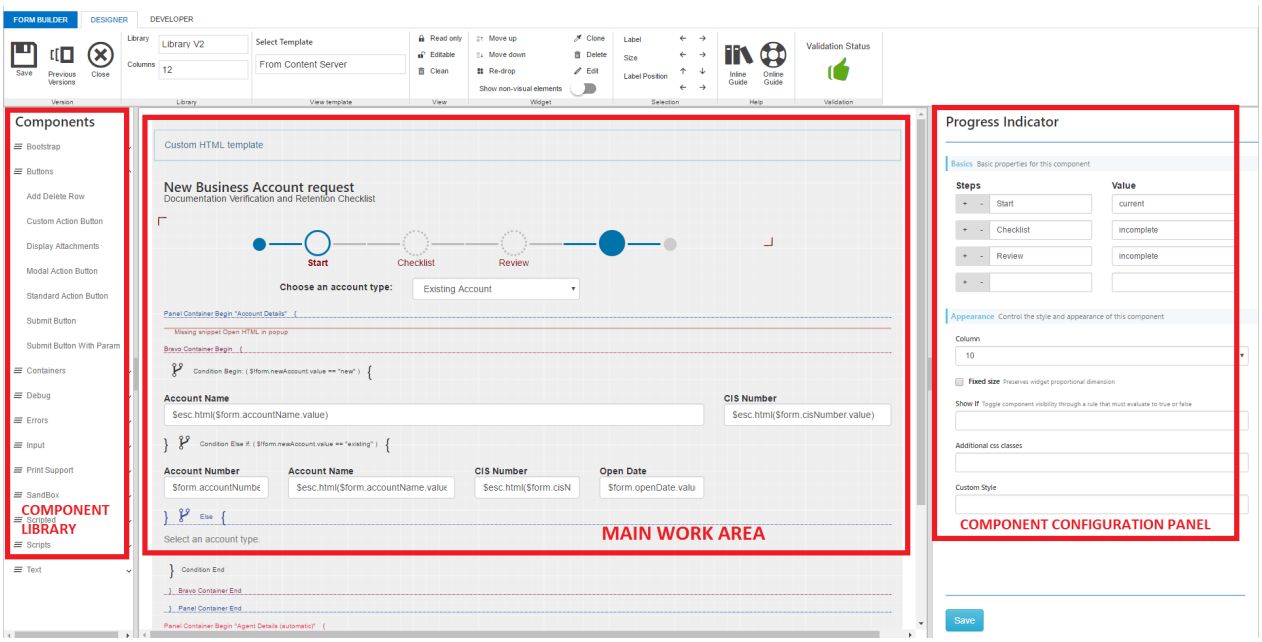
The Form Builder is the privileged IDE for Beautiful WebForms. On the **first load** of an empty view, the Form Builder will initialize it with a default input widget for every field in the form template. The view will then be available for further editing.

Layout¶

The IDE is composed of a set of areas and controls, with different purposes.

- The **Main Working Area** shows a preview of the current form view, with the available input fields
- The **Widget Library** (on the left) features a set of predefined widgets, which can be easily dragged and dropped in the working area
- The **Widget Configurator panel** (on the right) is linked to the widget currently selected in the main working area

Copyright © 2013-2022 AnswerModules Sagl



The Main Working area contains the two editor windows.

- **Smart Editor**: allows for drag & drop, editing and configuration based on visual tools
- **Advanced Editor**: allows for full control on the form view code.

By default, the Advanced Editor is locked and the Smart Editor is active. The two modes are mutually exclusive, and can't be active at the same time.

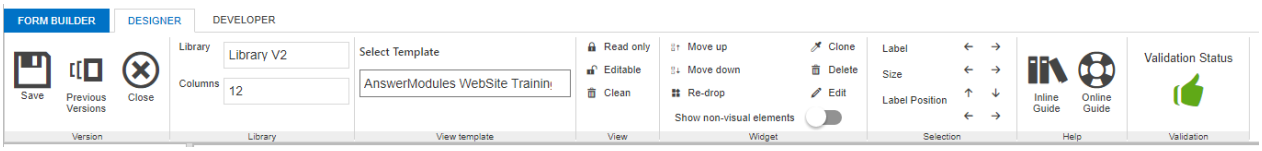
Shortcuts¶

The following keyboard shortcuts are available while using the editor:

Shortcut	Description
Ctrl + S	Save the current view (add a new version)
Ctrl + Canc	Delete the selected widget(s)
Ctrl + B	Clone the selected widget(s)
Shift + Left Ar.	Reduce the label's dimension for the selected widget(s)
Shift + Right Ar.	Increment the label's dimension for the selected widget(s)
Ctrl + Left Ar.	Reduce the dimension for the selected widget(s)
Ctrl + Right Ar.	Increment the dimension for the selected widget(s)
Ctrl + Mouse sel.	Select multiple widgets
Ctrl + Space	In sourcecode editor - show the code autocompletion hints
Ctrl + H	In sourcecode editor - Toggle the online Help window
F11	In widget's configuration panel – Maximize editor (full-screen mode)

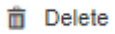
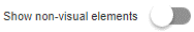
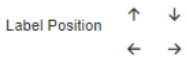
Copyright © 2013-2022 AnswerModules Sagl

Top Bar controls (DESIGNER)¶



Command	**Description**
Versions	
	Save the view (adds a new version)
	Open the object's **Versions** tab
	Close the FormBuilder
Libary	
Library	**Selects** the widgets' library to use for creating the view
Columns	Configures the number of columns in the view layout. In order to take effect, requires to save the view & reload the editor window
View template	
Select Template	The View's template associated with the form can be selected with the dropdown menu, or, as an alternative, selecting a suitable document from Content Server.
View	
Read only	Switch the whole view between Read Only and Editable mode (affects the way input widgets are rendered)
Editable	Switch the whole view between Read Only and Editable mode (affects the way input widgets are rendered)
Clean	Clear the entire working area
Widget	
Move up Move down	Reposition the widget, moving it one step up/down in the form
Re-drop	Pick Up the widget (to drop it elsewhere in view)
Clone	Duplicate the selected widget

Copyright © 2013-2022 AnswerModules Sagl

Command	**Description**
	Remove the widget from the form
	Open the widget's Configuration Panel
	Toggle the visibility of widgets that are not rendered in the final view (e.g. scripts)
Selection	
	Increase/decrease the size of the widget's label (if available). This option affects the number of columns spanned horizontally by the label.
	Increase/decrease the size of the widget. This option affects the number of columns spanned horizontally by the whole widget (including the label, if present).
	Change of the widget's position. This option affects the number of columns spanned horizontally by the whole widget (including the label, if present).
Help	
	Access the module's online **guide** and the **support portal**
Validation	
	Red label: The view failed the validation and most likely will fail to compile
	Green label: The view is **well-formed**

Widget Scope

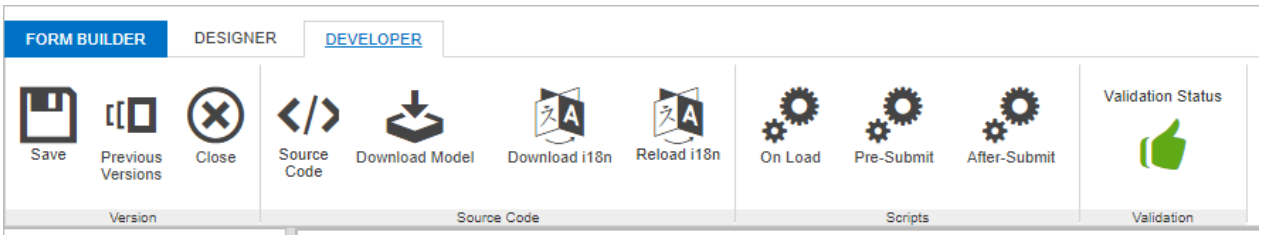
To enable the Widget Scope options in the menu, simply right click on the target widget in the working area.

Columns

When switching the number of columns, save & reload the page editor to force reload of all widgets in the working area

Copyright © 2013-2022 AnswerModules Sagl

Top Bar controls (DEVELOPER)¶



Command	**Description**
Versions	
	Save the view (adds a new version)
	Open the object's **Versions** tab
	Close the FormBuilder
Source code	
	Opens the view's source code editor
	Downloads the view's current model to be used for creating a new widget
	Downloads the view's localization file
	Reloads all the available localization files
Scripts	
	Opens the On-load CLEH Content Script Editor
	Opens the Pre-submit CLEH Content Script Editor
	Opens the On-submit CLEH Content Script Editor
Validation	

Copyright © 2013-2022 AnswerModules Sagl

Command	**Description**
Validation Status	**Red label**: The view failed the validation and most likely will fail to compile
Validation Status	**Green label:** The view is **well-formed**

Editing source code

View's versions created editing directly the source-code editor can't be further modified with the FormBuilder's smart-editor. If you switch from source-code editor to smart-editor any changes applied modifying the source code will be lost.

Building views

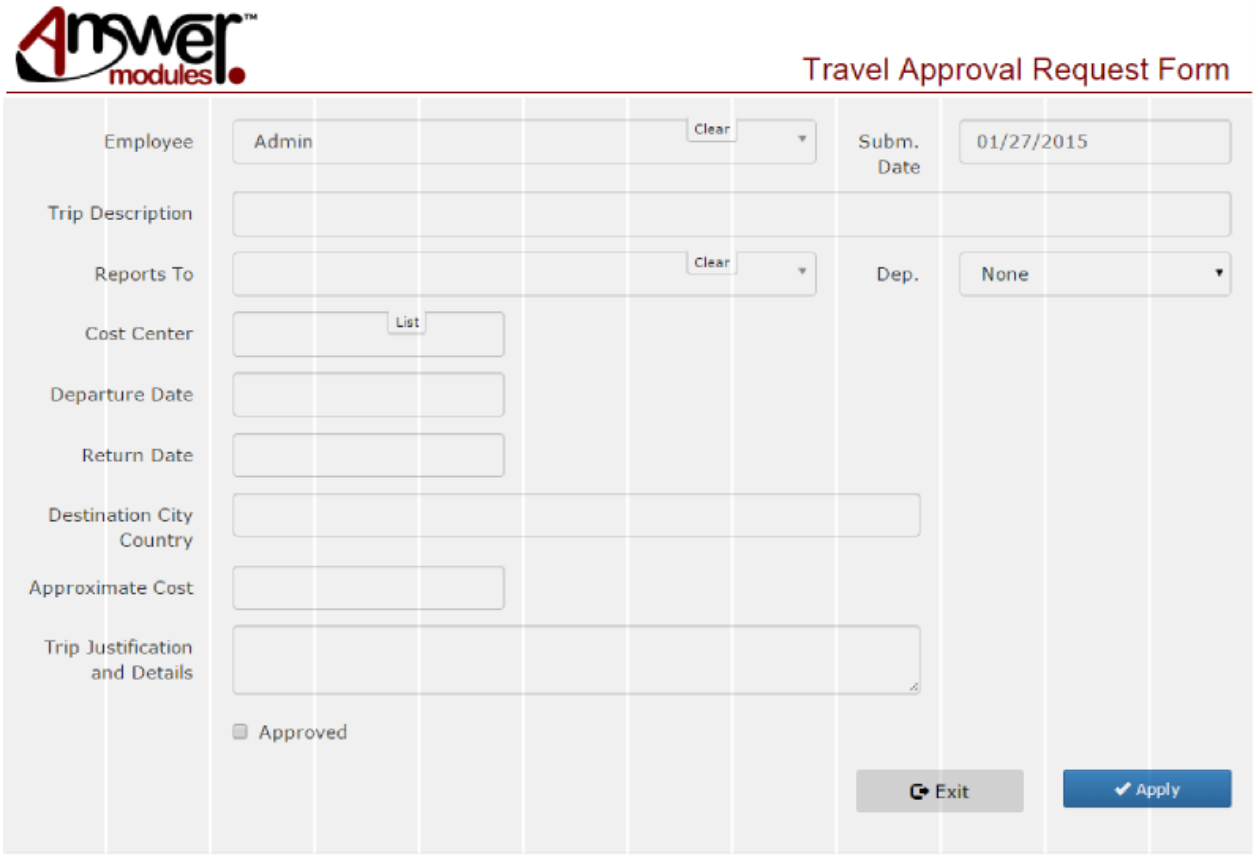
Understanding the grid system¶

In order to understand some of the features presented in the next sections, it is necessary to introduce the concept of **Grid System**, which has been adopted in the Beautiful WebForms Form Builder and views.

When creating or modifying a Form view, all of the widgets in the view appear neatly aligned to each other. The widgets can be modified in size only in discrete steps: that is, each widget can be assigned a size from a set of predefined options. When the view is presented to the user, the actual size of the widget will be proportional to the selected value.

To understand the logic behind this behaviour, you can imagine the Form fieldset area as if it was divided in a fixed number of columns (12 by default, 24 optional). By forcing each widget to span over a whole number of columns, we keep the overall layout of the form clean and tidy, eliminating the effort that is usually required to fine-tune the alignments and spacings. To better understand this concept, please take a look at the following image.

Copyright © 2013-2022 AnswerModules Sagl

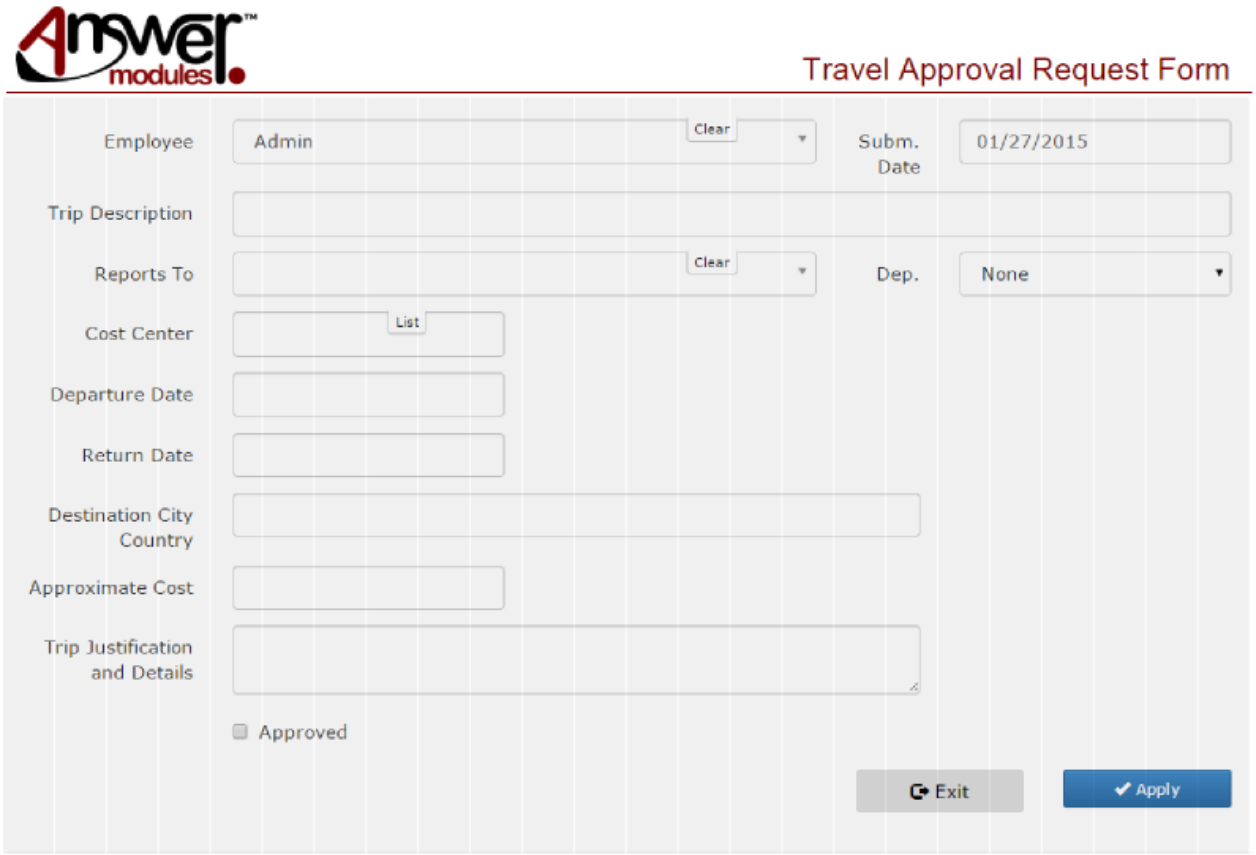


Additionally, the technology used for the grid layout is **responsive**. The form will automatically adjust to the size of the screen in which it is viewed, degrading gracefully in case of screen of small size.

In addition to the default 12-column grid, some of the built-in templates provided with Beautiful WebForms support a 24-column grid: this allows for a greater precision and more possibilities when creating the form layout. It is possible to configure the view to use one system or the other by toggling a menu in the Form Builder, as shown in the following sections.

Here after is an example presenting the same view as the previous example, but this time built on a 24-column grid.

Copyright © 2013-2022 AnswerModules Sagl



Understanding the Beautiful WebForms request life-cycle¶

Beautiful WebForms implement a slightly different lifecycle if compared to standard forms, thanks to their custom submission mechanism.

How incoming requests are processed¶

Beautiful WebForms are managed through a dedicated endpoint. Upon submission, the underlying engine performs server side validation. Only after successful validation, the form data is eventually submitted to Content Server.

The Beautiful WebForms life-cycle management of incoming requests can be schematized in the following steps:

1. Form rendering request: a user requests the form
2. ON LOAD - Custom logic execution hook
3. Form view rendering: the form page is rendered
4. User data input: the user interacts with the form and populates the input fields

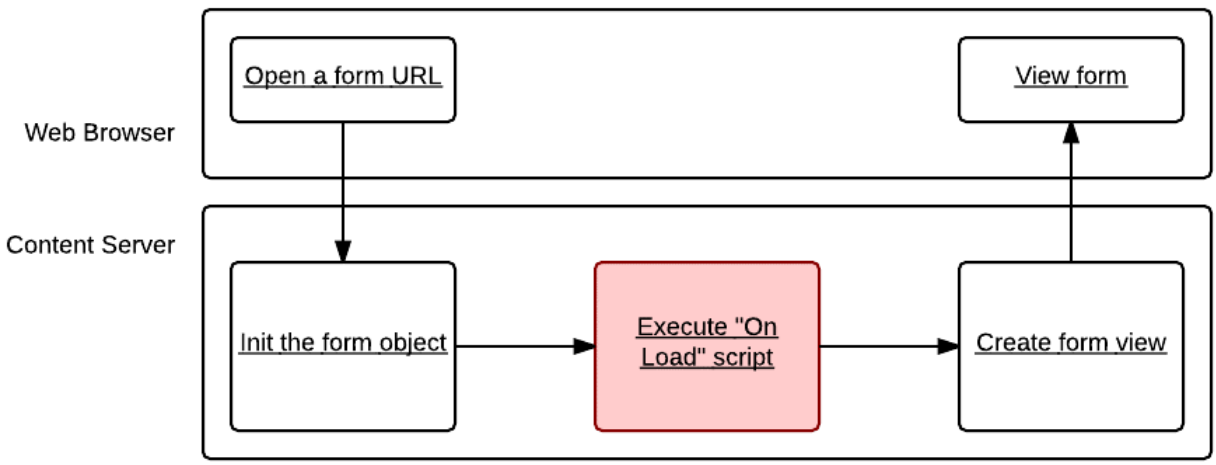
Copyright © 2013-2022 AnswerModules Sagl

5. Form submit action: the user attempts to submit the form data
6. Client side validation: the client side library validates the input fields
7. Actual data submission to Beautiful WebForms endpoint: in case of successful validation, data is submitted to the server
8. Server side validation: the Beautiful WebForms engine performs server side validation on the submitted data
9. PRE SUBMIT - Custom logic execution hook
10. Actual data submission to Content Server: form data is submitted to Content Server
11. POST SUBMIT - Custom logic execution hook
12. A validation error in any of the validation steps would interrupt the flow and return to step 1. Error information would be added to the form view, and used to populate inline error messages.

In case of validation errors, the data input by the user is preserved for the following view rendering.

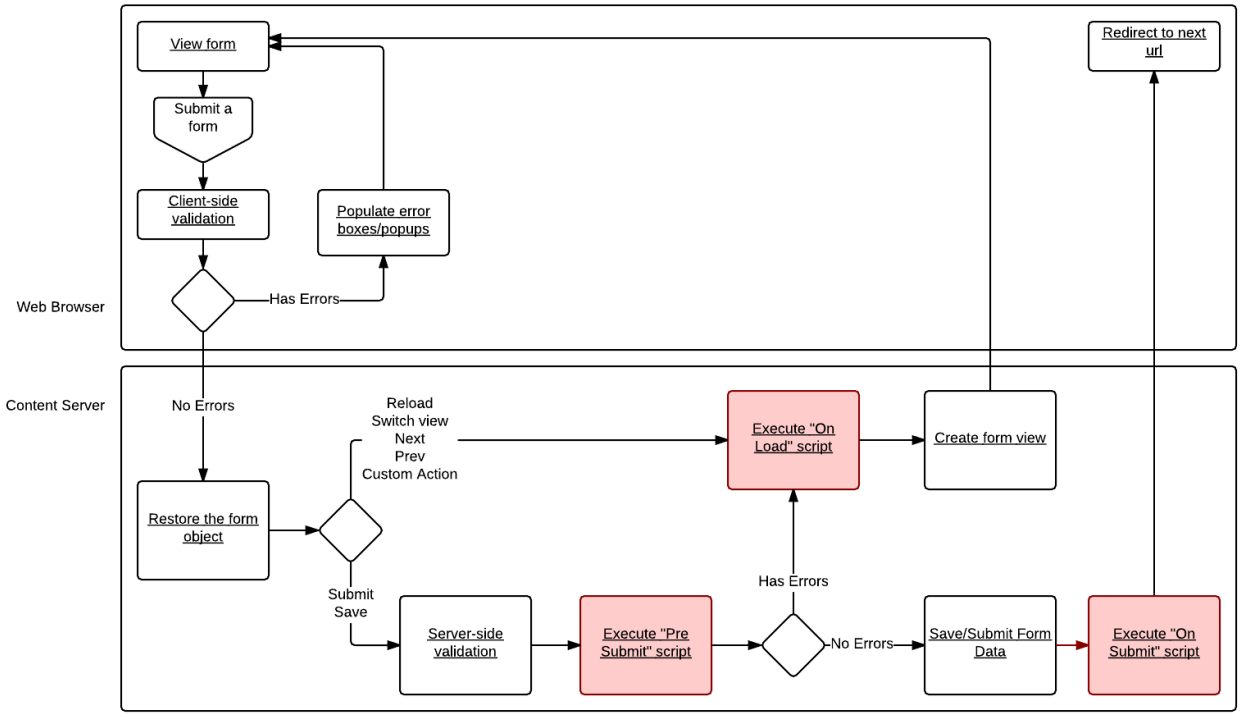
Lifecycle schema¶

The following schema considers a scenario in which a new form is requested by a user:



The following schema is related to a scenario in which the user attempts to submit the form (or otherwise performs an action that triggers a round trip to the server):

Copyright © 2013-2022 AnswerModules Sagl



Custom Logic Execution Hooks (CLEH)¶

In the two schemas above, there are several highlighted boxes that represent Custom Logic execution hooks. That is, steps in which it is possible to add customized business logic, in the form of Content Script code.

The scripts are:

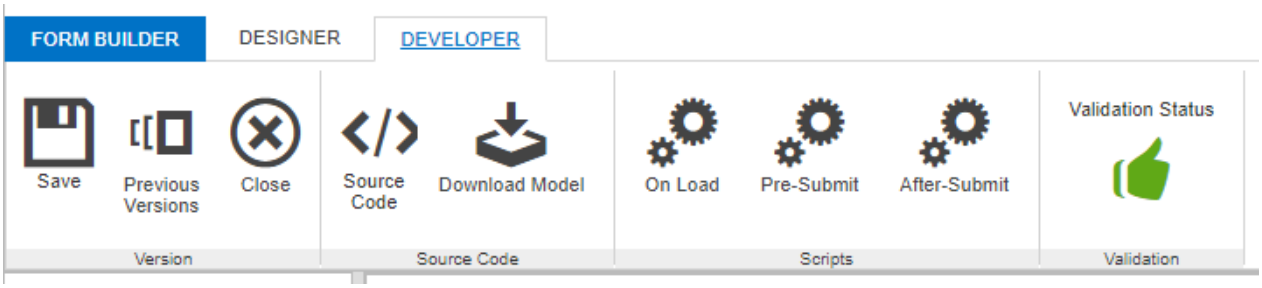
- **ON LOAD** view Content Script: this is the typical hook for prepopulating the form and manipulating the form view
- **PRE SUBMIT** view Content Script: this is the typical hook for extended validation and actions that must be performed before that the data is actually saved
- **POST SUBMIT** view Content Script: this is the typical hook for post submit actions (user notifications, document manipulation on content server, etc.)

> Starting with version 1.7.0, Beautiful WebForms Views have been transformed in container objects. Content Scripts associated to Beautiful WebForms views are standard Content Script nodes in the view container. The nodes are associated to the lifecycle steps *by name*

Throughout the whole process and in all of these scripts, a form object is available in the execution context. This object allows to fetch and manipulate the form data, as well as programmatically add or remove validation errors.

The Content Script objects associated to each execution hook can be accessed and edited through the **Specific Properties** tab of the Beautiful WebForm view object.

Copyright © 2013-2022 AnswerModules Sagl



The Content Scripts associated with CLEHs are regular Content Script objects. In the Script Context the Beautiful WebForms framework will inject additional items, such as the **form** object, which represents the form that is currently associated to the view.

The form object grant access to the form fields structure and the current values of each field, the form submitted data, the validation rules associated to the form, and provides utilities to manipulate this information.

E.g.

A commonly used function in the "ON LOAD view script" is

```
form.isFirstLoad()
```

The function allows to define actions which are executed only once per form view (the actions are not repeated in case of submission failure - for example, in case of validation errors). Typically, field prepopulation happens here.

The following sections provide information on common tasks that can be performed on the form programmatically in the various Content Scripts.

Managing form fields values¶

The state of the forms can be programmatically accessed and modified through the Content Script Custom Logic Execution Hooks.

In scripts, form field values can be accessed using the following notation:

```
form.*normalizedname*.value
```

where 'normalizedname' is the name of the field after normalization performed by the Beautiful WebForms framework.

Auto completion

Use the CTRL+Space keyboard shortcut to access autocomplete options on the form object. Options include all the fields in the form.

Copyright © 2013-2022 AnswerModules Sagl

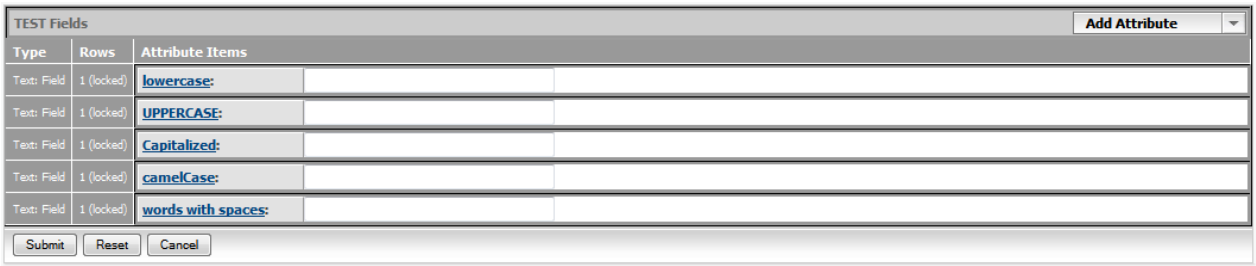
The rules applied when normalizing field names are:

- the only admitted characters are alphanumeric characters and whitespaces (using different characters can lead to unexpected behavior)
- all characters are transformed in lowercase
- all characters immediately after a whitespace are transformed in uppercase

As a rule of thumb, it is advised to adopt a naming convention for field names that would be compatible with SQL table column names.

To better understand the concept, consider the following Form Template, containing a few fields (using different possible naming conventions):

- a field named 'lowercase'
- a field named 'UPPERCASE'
- a field named 'Capitalized'
- a field named 'camelCase'
- a field named 'words with spaces'



The fields can be accessed in a script as follows:

- 'lowercase': form.lowercase.value
- 'UPPERCASE': form.uppercase.value
- 'Capitalized': form.capitalized.value
- 'camelCase': form.camelcase.value
- 'words with spaces': form.wordsWithSpaces.value

```
form.lowercase.value        = "TEST VALUE A" //Form template field name: lowercase
form.uppercase.value        = "TEST VALUE B" //Form template field name: UPPERCASE
form.capitalized.value      = "TEST VALUE C" //Form template field name: Capitalized
form.camelcase.value        = "TEST VALUE D" //Form template field name: camelCase
form.wordsWithSpaces.value  = "TEST VALUE E" ////Form template field name: words with spaces
```

Copyright © 2013-2022 AnswerModules Sagl

// Initalize form field values: some examples

form.lowercase.value = “TEST VALUE A” // Form template field name: lowercase

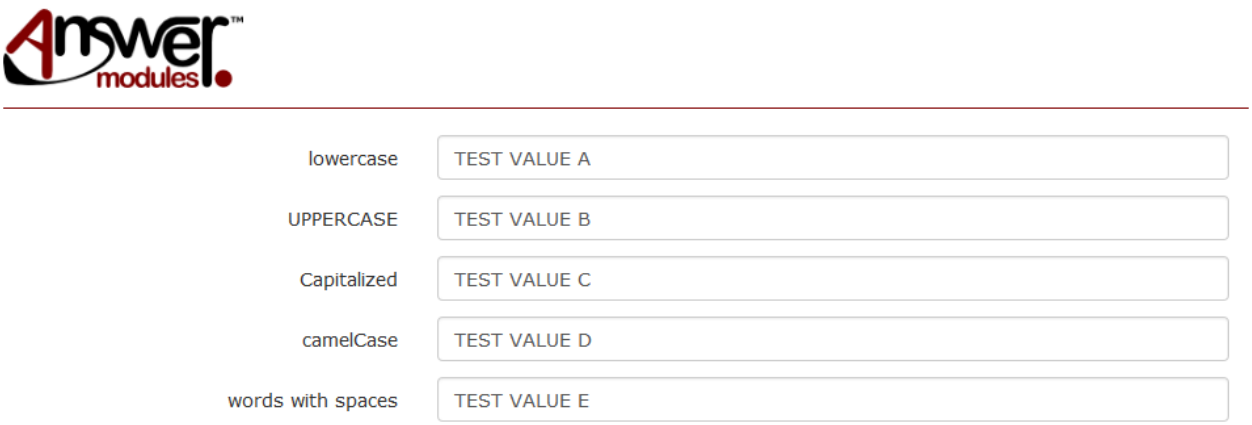
form.uppercase.value = “TEST VALUE B” // Form template field name: UPPERCASE

form.capitalized.value = “TEST VALUE C” // Form template field name: Capitalized

form.camelcase.value = “TEST VALUE D” // Form template field name: camelCase

form.wordsWithSpaces.value = “TEST VALUE E” // Form template field name: words with spaces

The resulting form (after initialization):



Adding and removing values from multivalue fields¶

In case of multi-value fields, it is possible to programmatically add new values (up to the max-values limit)

For each field, multiple values can be accessed directly by index (0-based).

By default, if a field value is accessed without specifying an index, the referenced value is the one with index 0.

form.textvalue.value = "My value" is equivalent to form.textvalue[0].value = "My value"

NOTE: The value at index 0 does not require initialization.

To access values at index > 0:

form.textvalue.addField(1)

form.textvalue[1].value = "My value"

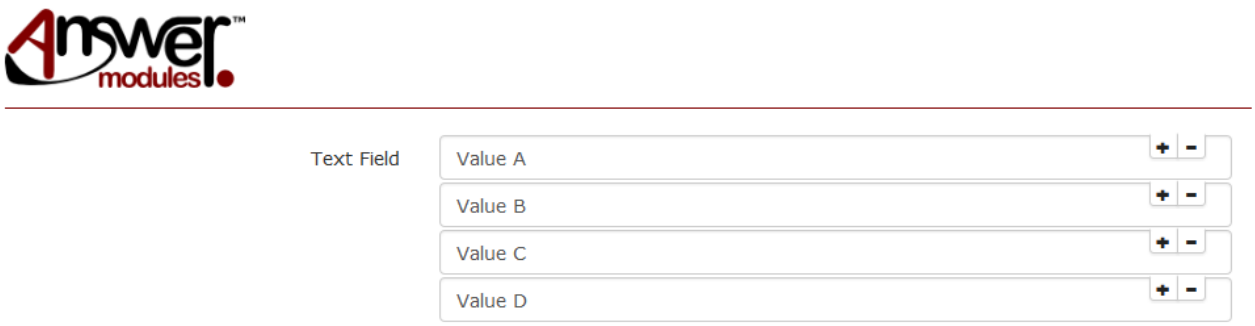
Example. Field initialization:

```
form.textField.value = "Value A"  // The first field (index:0) is always available. no need to add t
```

Copyright © 2013-2022 AnswerModules Sagl

```
form.addField("textField", 1) // Additional field's values can be added either through the form obje
form.textField.addField(2)    // or directly on the field

form.textField[1].value = "Value B"
form.textField[2].value = "Value C"

form.textField.addField(3)
form.textField[3].value = "Value D"
```

The resulting form (after initialization):



Form actions¶

An *action* is a piece of server side scripting code that is execute in response of a particular type of request. The action to be performed is identified by the request parameter (`am_Action`) submitted with the form. Another optional parameter (`am_ActionParams`) is sometimes included when specific information is required by the action.

Standard form actions¶

The framework is capable of handling a set of predetermined actions as part of the Beautiful WebForms lifecycle.

The following are the standard action managed by the framework:

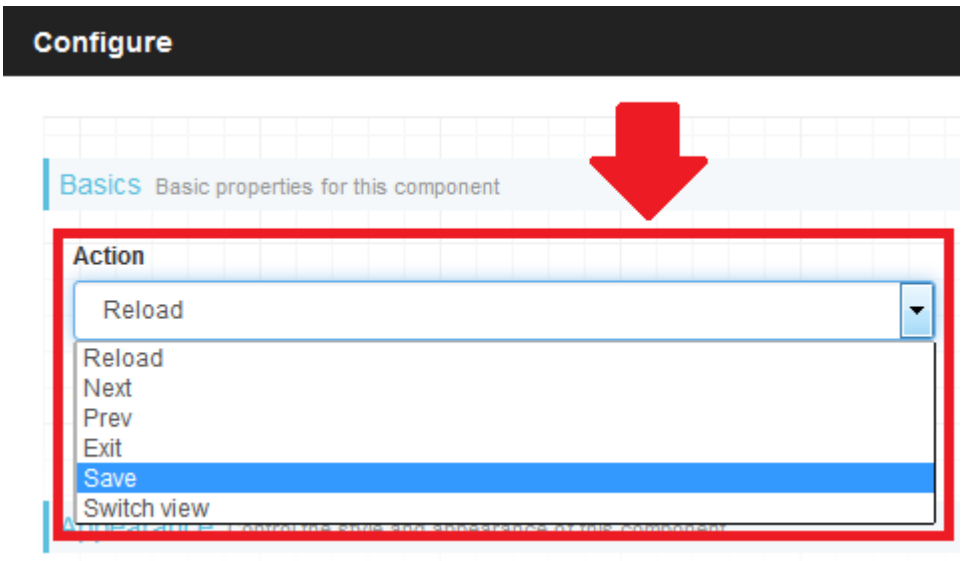
Action	Description	Action ID (am_Action)	Action parameter (am_ActionParams) usage
Reload	Performs a round trip to the server and re-renders the form view.	`am_reload`	*not required*
Save	Saves the current state of the form, without submitting. *Available in Workflow forms only*	`am_save`	*not required*
Exit	Exits without saving modifications to the form data	`am_exit`	*not required*

Copyright © 2013-2022 AnswerModules Sagl

Action	Description	Action ID (am_Action)	Action parameter (am_ActionParams) usage
Switch View	Switches the view and re-renders the form	`am_switchView`	The ID of the target view
Next	To be used together with "prev" to create a wizard-like experience, enabling the switching forwards through a sequence of different views	`am_wizardNext`	The ID of the next view. Alternatively, the target view can be configured on server side by setting the value of: form.viewParams.am_wizardNextView
Prev	To be used together with "next" to create a wizard-like experience, enabling the switching backwards through a sequence of different views	`am_wizardBack`	The ID of the previous view. If not and a "Next" action was invoked beforehand, the framework will attempt to switch back to that view. Alternatively, the target view can be configured on server side by setting the value of: `form.viewParams.am_wizardPrevView`

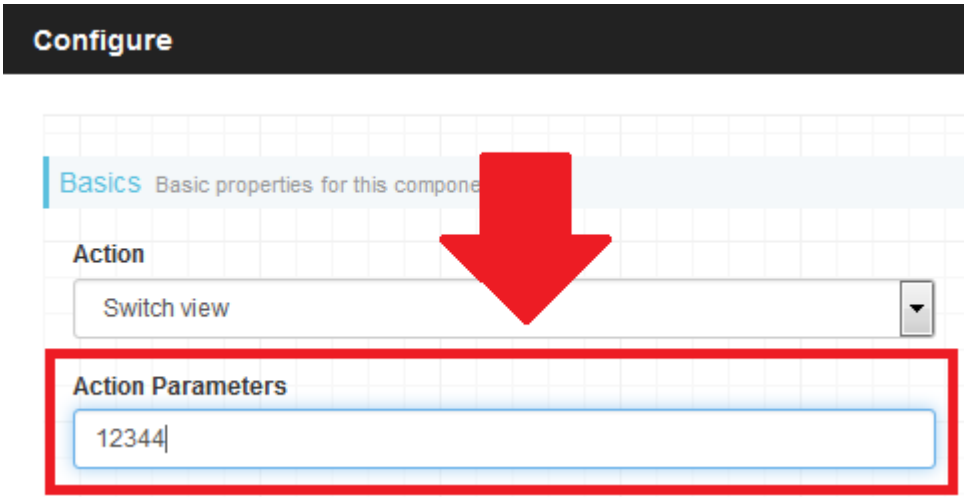
Standard form actions can be selected by using the **Standard Action Button** component.

The **Standard Action Button** component can be configured through the configuration panel to select the appropriate action



Copyright © 2013-2022 AnswerModules Sagl

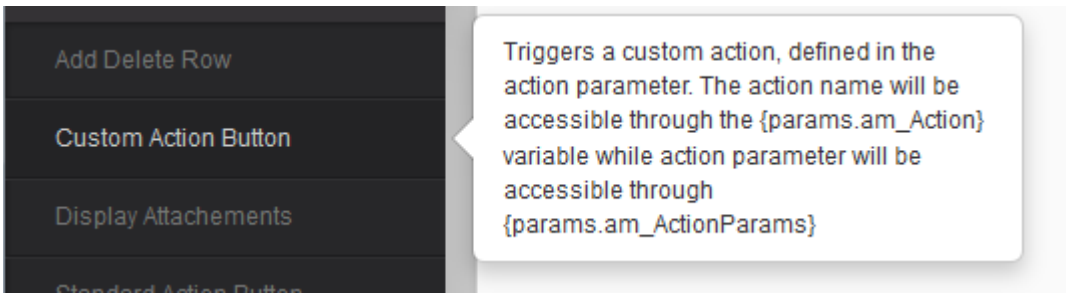
Whenever a parameter is required by the selected action (see above table) the appropriate value can be configured as follows:



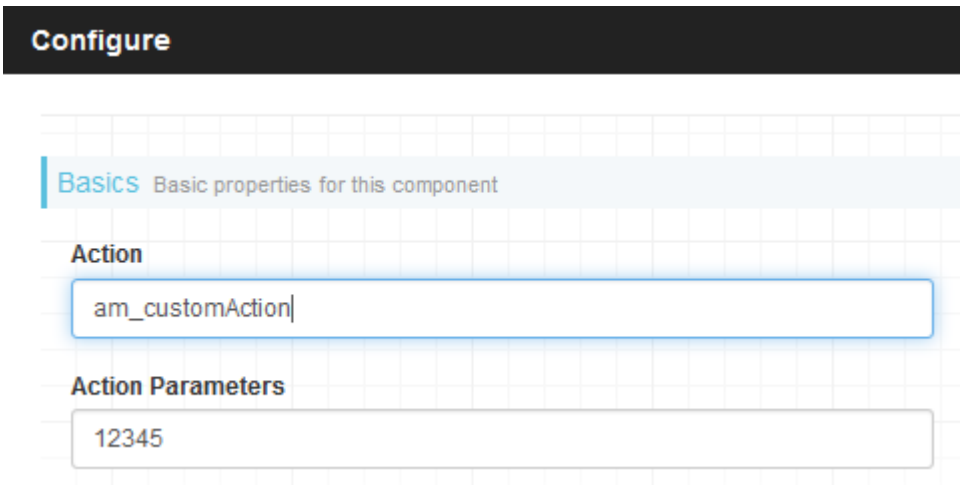
Custom form actions¶

It is also possible to define custom actions when submitting a form. In this case, the custom actions should be handled in the Content Script **Custom Logic Execution Hooks**.

Custom form actions can be selected by using the **Custom Action Button** component.



In this case, the configuration panel allows to specify a value for the name of the **action** and the value of the (optional) **actionParams**

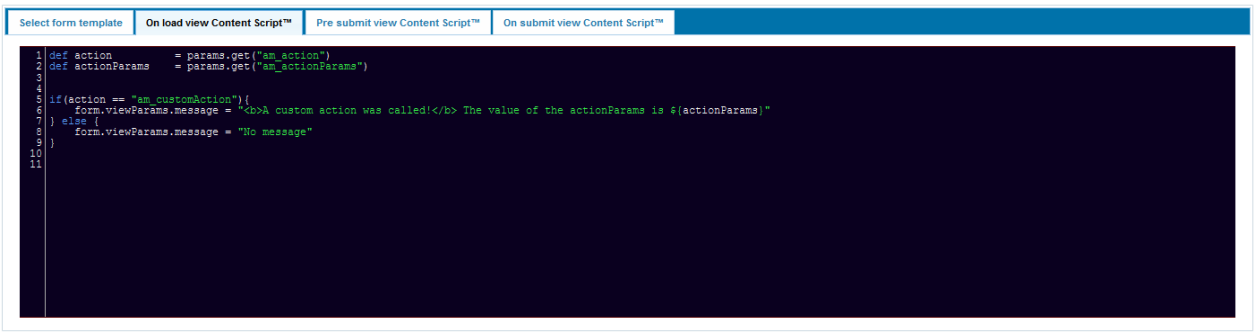


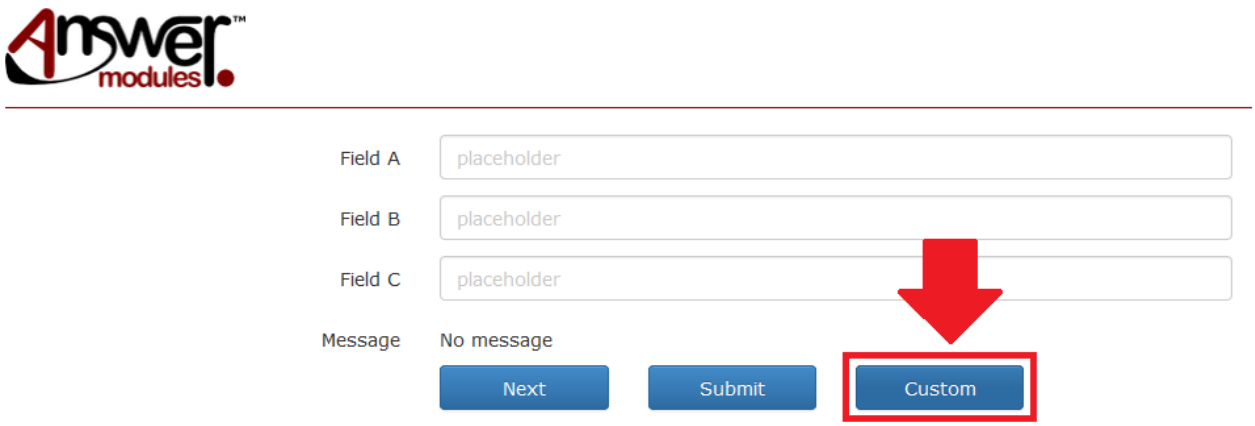
Copyright © 2013-2022 AnswerModules Sagl

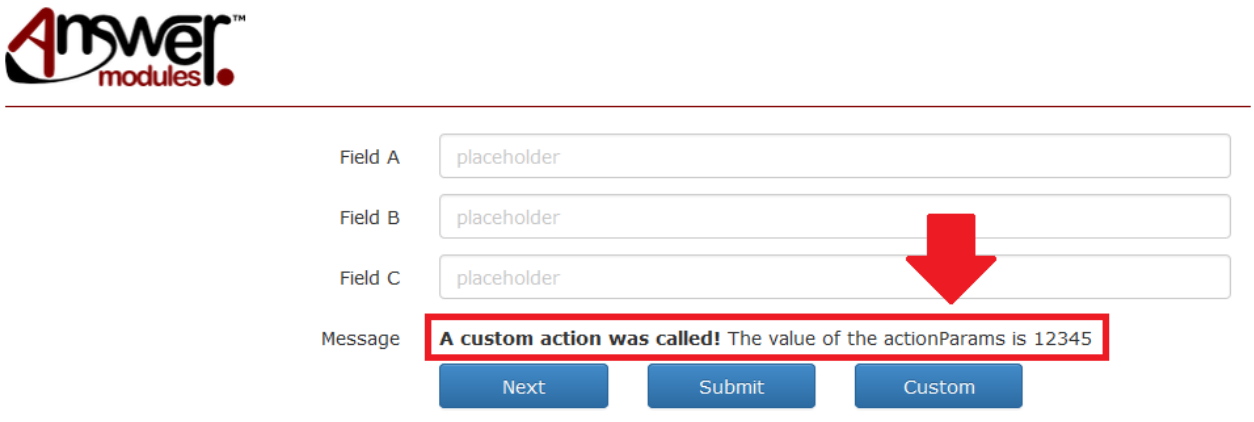
Whenever the button is used, the information related to **action** and **actionParams** will be available in the request params. It can be easily accessed as follows:

```
def action = params.get("am_action")
def actionParams = params.get("am_actionParams")
```

Below is a simple example showing how to use and manage a **Custom Action**:







Invoking an action

It is possible to manually trigger the execution of Actions in cases where the provided Form Components are not sufficient to meet specific needs.

Copyright © 2013-2022 AnswerModules Sagl

In such cases, the `am_setAction(form, action, actionParams)` javascript function can be used, where:

- **form** is the id of the html form (eg. `form_258191`)
- **action** is the action id (eg. `am_customAction`)
- **actionParams** is the optional value of additional parameters required by the action (eg. '12345')

The following is an example using an HTML button:

```
<button
onclick="am_setAction('form_258191','am_customAction','12345')"
type="submit"> Custom Action Button </button>
```

Attaching Custom information and data to a Beautiful WebForms view ¶

ViewParams¶

It is sometimes necessary to bind to the form object additional parameters and values that are not supposed to be stored in form fields. It is the case for parameters that are only needed to control the form page layout: an example is when the HTML template containing the form can be dynamically configured in some of its parts (for example, a title or logo).

To address this need, the **'form'** object is bound to a data map (named **'viewParams'**) which is meant to contain additional parameters that are not supposed to be persisted with the form data.

Entries in the **'viewParams'** map can be set and accessed programmatically as in the following examples.

Example 1. Within a Content Script, set the value of the parameter 'title':

```
form.viewParams.title = "My Form"
```

Example 2. Within a Content Script, read the value of the parameter 'title' and store the value in a variable 'myVar':

```
def myVar = form.viewParams.title
```

Example 3. When accessing the 'viewParams' in an HTML Form Template, the syntax is slightly different, as the templating engine syntax must be used. For example:

```
<h1>$form.viewParams.title</h1>
```

Copyright © 2013-2022 AnswerModules Sagl

You can include a '!' in your expression in order to avoid printing the output in the rendered HTML in case the value of the variable is not set:

```
<h1>$!form.viewParams.title</h1>
```

Serializable

any object programmatically added to the 'viewParams' map MUST be a serializable object.

ViewParams variables¶

Prior of each view rendering, the Beautiful Form Frameworks injects in the **viewParams** field of the Form object a set of variables. The number and type of these variables depend on the current execution scope. All the variables at the moment of the injection are serialized as String. The table here below summarizes all the possible variables that can be found in the **viewParams** field, indicating for each of them, the original type and name.

Warning

the actual case of the variable names could depend on the underlying database.

List of the variable automatically injected into the ViewParams map

Variable Name	Scope	Original Type
LL_CgiPath	Form, Workflow	String
LL_NextURL	Form, Workflow	String
LL_SupportPath	Form, Workflow	String
LL_UserContact	Form, Workflow	String
LL_UserFirstName	Form, Workflow	String
LL_UserFullName	Form, Workflow	String
LL_UserGroupName	Form, Workflow	String
LL_UserID	Form, Workflow	Integer
LL_UserLastName	Form, Workflow	String
LL_UserLogin	Form, Workflow	String
LL_UserMailAddress	Form, Workflow	String
LL_UserMiddleName	Form, Workflow	String
LL_UserTitle	Form, Workflow	String
MapTask_CustomData	Workflow	Assoc
MapTask_Description	Workflow	String
MapTask_Form	Workflow	Assoc
MapTask_Instructions	Workflow	String
MapTask_Priority	Workflow	Integer
MapTask_StartDate	Workflow	Date
MapTask_SubMapID	Workflow	Integer
MapTask_SubType	Workflow	Integer
MapTask_Type	Workflow	Integer
Map_Description	Workflow	String
Map_Instructions	Workflow	String
Map_SubType	Workflow	Integer
Map_Type	Workflow	Integer
SubWorkTask_DateDone	Workflow	Date
SubWorkTask_DateDue_Max	Workflow	Date
SubWorkTask_DateDue_Min	Workflow	Date

Copyright © 2013-2022 AnswerModules Sagl

```
| SubWorkTask\_DateMilestone | Workflow | Date    |
| SubWorkTask\_DateReady     | Workflow | Date    |
| SubWorkTask\_Flags         | Workflow | Integer |
| SubWorkTask\_IterNum       | Workflow | Integer |
| SubWorkTask\_PerformerID   | Workflow | Integer |
| SubWorkTask\_Status        | Workflow | Integer |
| SubWorkTask\_SubWorkID     | Workflow | Integer |
| SubWorkTask\_TaskID        | Workflow | Integer |
| SubWorkTask\_Title         | Workflow | String  |
| SubWorkTask\_Type          | Workflow | Integer |
| SubWorkTask\_WaitCount     | Workflow | Integer |
| SubWorkTask\_WorkID        | Workflow | Integer |
| SubWork\_DateCompleted     | Workflow | Date    |
| SubWork\_DateDue\_Max      | Workflow | Date    |
| SubWork\_DateDue\_Min      | Workflow | Date    |
| SubWork\_DateInitiated     | Workflow | Date    |
| SubWork\_Flags             | Workflow | Integer |
| SubWork\_MapID             | Workflow | Integer |
| SubWork\_Project           | Workflow | Dynamic |
| SubWork\_ReturnSubWorkID   | Workflow | Integer |
| SubWork\_ReturnTaskID      | Workflow | Integer |
| SubWork\_Status            | Workflow | Integer |
| SubWork\_SubWorkID         | Workflow | Integer |
| SubWork\_Title             | Workflow | String  |
| SubWork\_WorkID            | Workflow | Integer |
| Work\_DateCompleted        | Workflow | Date    |
| Work\_DateDue\_Max         | Workflow | Date    |
| Work\_DateDue\_Min         | Workflow | Date    |
| Work\_DateInitiated        | Workflow | Date    |
| Work\_Flags                | Workflow | Integer |
| Work\_ManagerID            | Workflow | Integer |
| Work\_OwnerID              | Workflow | Integer |
| Work\_Status               | Workflow | Integer |
| Work\_WorkID               | Workflow | Integer |
```

Form Components that make use of 'viewParams' values.¶

Various components available in the Form Builder are configurable and require one or more parameters to be programmatically set: these parameters can be made available to the component as values in the **'viewParams'** container variable.

The widgets library¶

The **Widgets library** is an extensible set of form widgets that can be used through the drag & drop visual editor. To simplify the navigation, the widgets are arranged in families of objects with similar functionalities.

> The mapping between form template fields and their default input widget used to initialize Beautiful WebForms Views can be customized by configuring the desired CSFormSnippet in the Content Script Volume.

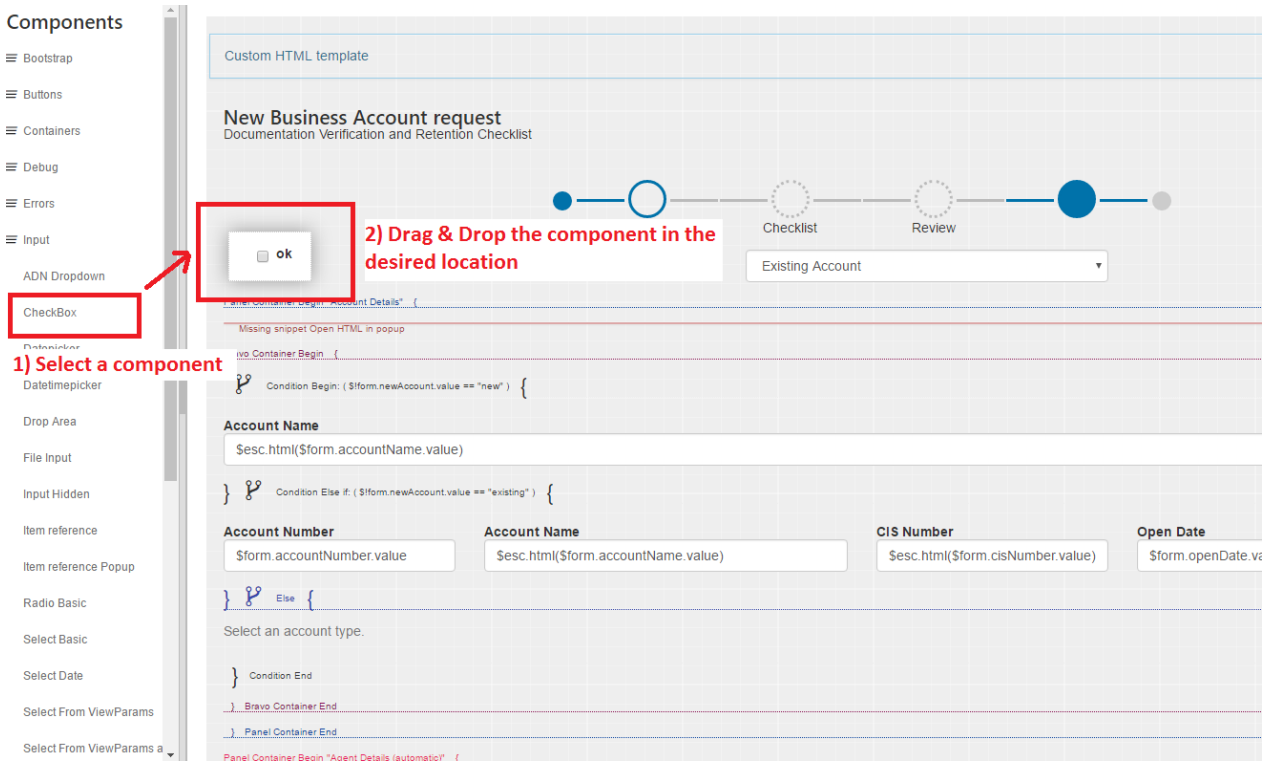
To add a new widget:

1. Open the widget library group that contains the widget
2. Click on the widget, holding the mouse button down

Copyright © 2013-2022 AnswerModules Sagl

3. Drag the widget to the desired position in the working area (a highlighted box will appear)

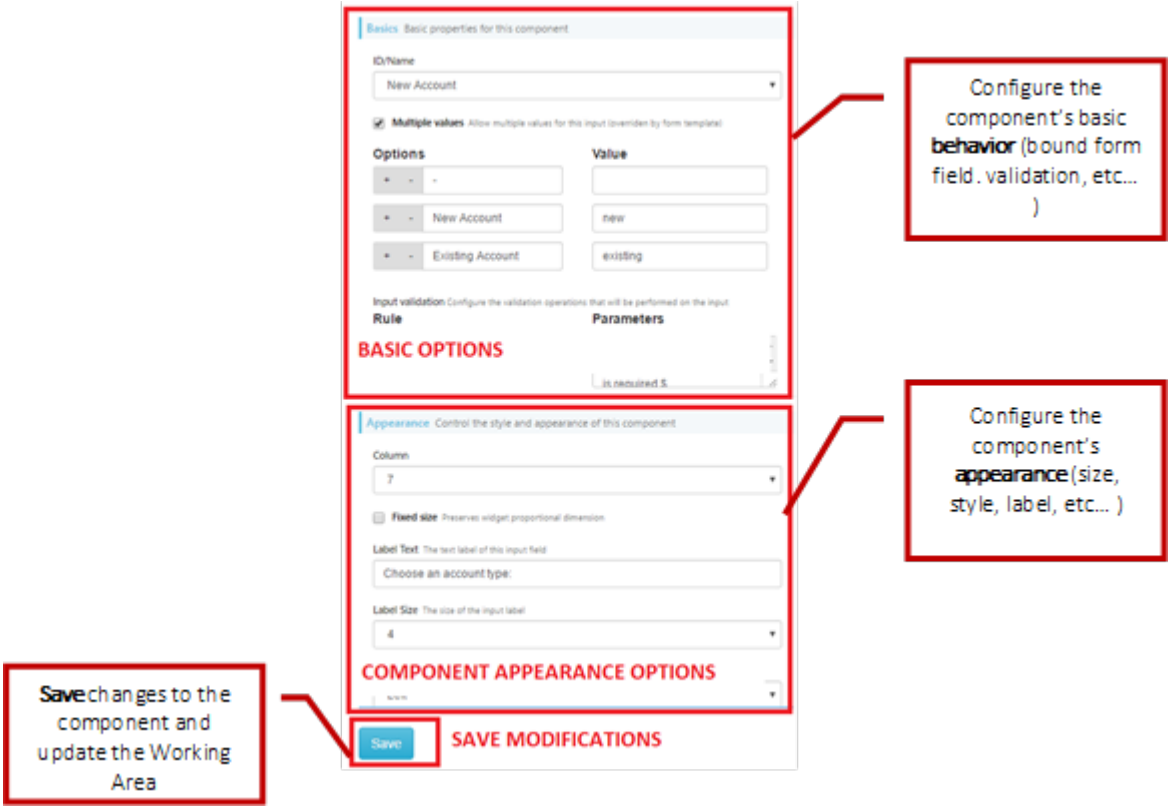
4. Drop the widget in the working area



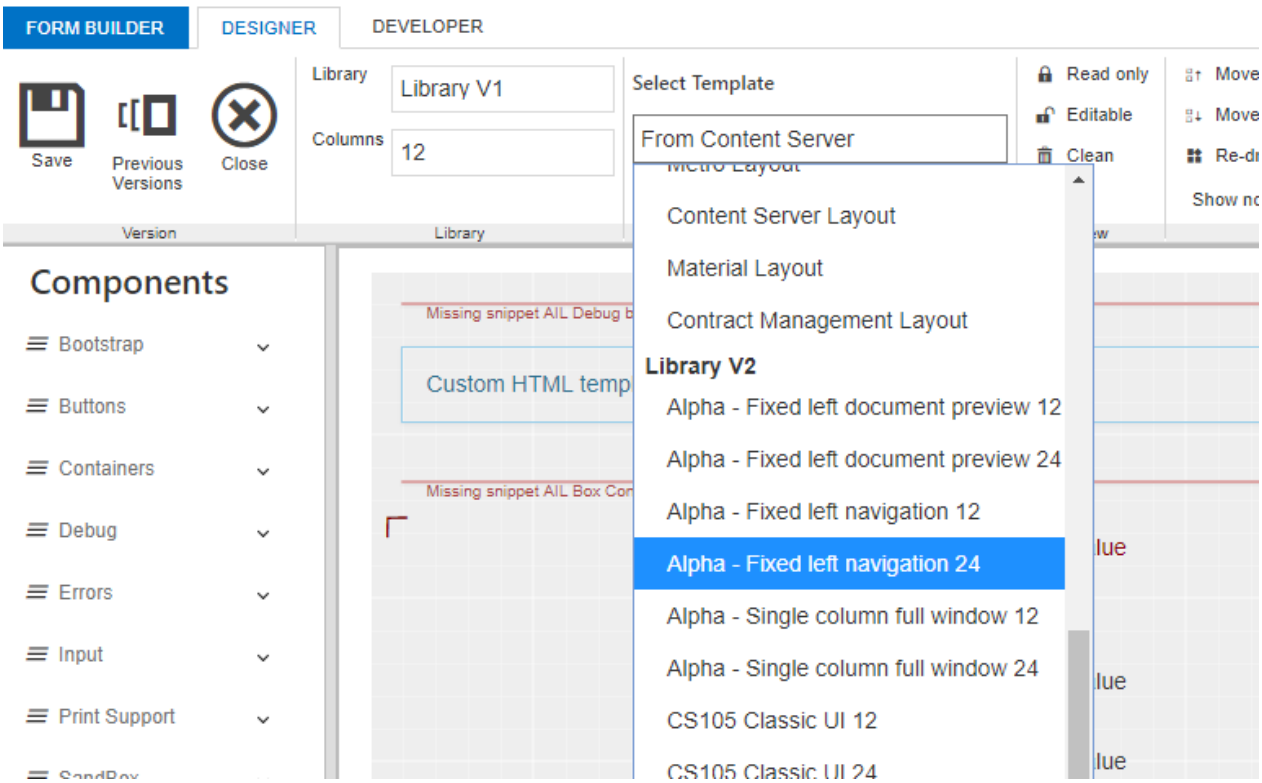
The widget configuration panel¶

When a widget in the Main Working Area is selected, the **Configuration Panel** can be activated through the dedicated menu option or by right-clicking the widget. The content of the panel is specific to the type of widget, and allows to define the widget binding to underlying form fields (in case of input widgets), as well as how the widget will be rendered, what validation rules will be applied to it, and any other setting that could be necessary for the specific widget.

Copyright © 2013-2022 AnswerModules Sagl



Beautiful WebForms View Templates¶



The BWF Framework enforces the Model View Controller paradigm, in fact Beautiful WebForms Views (and Templates) are always processed, before being rendered, from the module's internal Templating engine. At rendering time the BWF framework creates (as Model) for the Form View

Copyright © 2013-2022 AnswerModules Sagl

an Execution Context very similar to the one used by the Content Script Engine. The main difference between the two contexts is the presence of the "form" variable that refers to a server side representation of the Form object to which the Form View has been associated. As discussed each BWF View can be associated to a Form Template. At rendering time the framework executes the following operations:

- Substitutes in the Form Template any occurrences of the tag **<am:form />** with the content of the Form View as defined, for example, using the Form Builder
- Evaluates the result of the previous operation with the internal Templating Engine

The most important consequence of the aforementioned rendering procedure is that **any** valid Templating expression present both in the View and in the Template will be evaluated and eventually substituted by the Templating engine. This feature is widely used by default Form Templates and default Form Snippets.

Default Form Templates make use of these characteristics of the framework to slightly change their aspect, resulting behaviors, or more simply to load the most appropriate static resources (i.e. javascript libraries and CSS stylesheets).

For developers convenience the BWF frameworks defines also a set of macro that simplify the creation of new templates or the management of existing one. In the following section the source code of these macro is listed.

Customize the way validation error messages are rendered¶

In order to customize the way validation error messages related to form's fields are displayed you can leverage the Errors *(/working/bwebforms/widgets/#errors_1)* widget in order to override both the javascript (used to render errors on client side) and Velocity (used to render errors on server side) functions in your view.

```
(function(root, factory){
if (typeof csui !== 'undefined' && typeof csui.require === 'function') {
    csui.require(['jquery','underscore','v3/js/am/am_init','v3/js/am/am_ajaxvalidation'], function($
        factory($, _, amui, amform);
    });
}else if ( typeof require === 'function'){
    require(['jquery','underscore','v3/js/am/am_init','v3/js/am/am_ajaxvalidation'], function($,_, a
        factory($, _, amui, amform);
    });
}  else {
    factory(root.jQuery, root.amui);
}
}(this, function($, _, amui, amform) {

    amform.zcleanFieldValidationError = function (comp){
        var wrapper =comp.closest('.am-form-input-wrap')
        wrapper.removeClass('am-has-error-tooltip')
        wrapper.removeClass('has-error')
        wrapper.data('title', '').attr('title', '');
```

Copyright © 2013-2022 AnswerModules Sagl

```
        try {
            wrapper.tooltip('destroy')
        } catch (e) {
        }
    }

    amform.zcleanFormValidationError = function (form){
        form.find('.help-block.has-error').remove();
        form.find('.am-form-input-wrap').removeClass('has-error');
        form.find('.am-has-error-tooltip').each(
        function() {
            $(this).removeClass('am-has-error-tooltip').data('title', '')
                    .attr('title', '')
            try {
                $(this).tooltip('destroy')
            } catch (e) {
            }
        });
    }

    amform.zdisplayValidationError= function (message, failingElements){
        $(failingElements).each(
                function() {
                    var wrapper = $(this).closest('.am-form-input-wrap')
                    try {
                        wrapper.addClass('am-has-error-tooltip').addClass(
                                'has-error').attr(
                                'title',
                                ((wrapper.data('title') != undefined) ? wrapper
                                        .data('title') : '')
                                        + ' ' + message);
                        wrapper.tooltip('destroy')
                        wrapper.tooltip()
                    } catch (e) {
                    }
                });
    }
}));
```

```
#macro( showErrors $field )
<script>
(function(root, factory) {
    if (typeof csui !== 'undefined' && typeof csui.require === 'function') {
        csui.require(['jquery','v3/js/am/am_init','underscore','regula'], function($,amui,underscore,
            factory($, amui, _, regula);
        });
    }else if (typeof require === 'function') {
        require(['jquery', 'v3/js/am/am_init', 'underscore', 'regula'], function ($, amui, _, regula
             return factory($, amui, _ ,regula);
        });
    } else {
        factory(root.jQuery, root.amui, root._, regula);
    }
}(this, function($, amui, _, regula) {
       #if($field.getValidationStatus().size() gt 0)
       amui.registerInitWidgetCallback(function(){
            $('#$field.id').data('title','');
            #foreach ($error in $field.getValidationStatus() )
                    $('#$field.id').data('title', $('#$field.id').data('title')+' $error.validationE
            #end
            var wrapper = $('#$field.id').closest('.am-form-input-wrap');
            try{
                wrapper.tooltip('destroy')
            }catch(e){
            }
                      wrapper.addClass('am-has-error-tooltip')
                      .data('title', $('#$field.id').data('title'))
```

Copyright © 2013-2022 AnswerModules Sagl

```
                        .attr('title', $('#$field.id').data('title'))
                        .tooltip()
                        .addClass('has-error');
        });
        #end
}));
</script>
#end
```

Display errors in Smart View¶

In order to be compliant with the way SmartView displays error messages the following overrides can be utilized

```
(function(root, factory){
if (typeof csui !== 'undefined' && typeof csui.require === 'function') {
    csui.require(['jquery','underscore','v3/js/am/am_init','v3/js/am/am_ajaxvalidation'], function($,
        factory($, _, amui, amform);
    });
}else if ( typeof require === 'function'){
    require(['jquery','underscore','v3/js/am/am_init','v3/js/am/am_ajaxvalidation'], function($,_, a
        factory($, _, amui, amform);
    });
}  else {
    factory(root.jQuery, root.amui);
}
}(this, function($, _, amui, amform) {

    amform.zdisplayValidationError= function(message, failingElements){
        $(failingElements).each(
            function() {
                var wrapper = $(this);
                try {
                    wrapper.addClass("am-smartui-error");
                    wrapper.closest('.am-form-input-wrap').append("<div class='amsmartui-help-block

                } catch (e) {
                    //jquery compatibility
                }
            });
    }

    amform.zcleanFieldValidationError=function(comp){
        var wrapper =comp
        wrapper.removeClass('am-smartui-error')
        wrapper.closest('.am-form-input-wrap').find(".amsmartui-help-block").remove();

    }

    amform.zcleanFormValidationError = function(form){
        form.find('.help-block.has-error').remove();
        form.find('.am-form-input-wrap').removeClass('has-error');
        form.find('.am-smartui-error').each(
            function() {
                $(this).removeClass('am-smartui-error').closest('.am-form-input-wrap').find(".amsmar
            });
    }
}));
```

```
#macro( showErrors $field )
<script>
(function(root, factory) {
```

Copyright © 2013-2022 AnswerModules Sagl

```
    if (typeof csui !== 'undefined' && typeof csui.require === 'function') {
        csui.require(['jquery','v3/js/am/am_init','underscore','regula'], function($,amui,underscore,
            factory($, amui, _, regula);
        });
    }else if (typeof require === 'function') {
        require(['jquery', 'v3/js/am/am_init', 'underscore', 'regula'], function ($, amui, _, regula)
            return factory($, amui, _ ,regula);
        });
    } else {
        factory(root.jQuery, root.amui, root._, regula);
    }
}(this, function($, amui, _, regula) {
       #if($field.getValidationStatus().size() gt 0)
       amui.registerInitWidgetCallback(function(){
           var wrapper = $('#$field.id');
           wrapper.addClass("am-smartui-error");
           #foreach ($error in $field.getValidationStatus() )
           wrapper.closest('.am-form-input-wrap').append("<div class='amsmartui-help-block form-cont
           #end

       });
       #end
}));
</script>
#end
```

Widgets

Beautiful WebForms Widgets¶

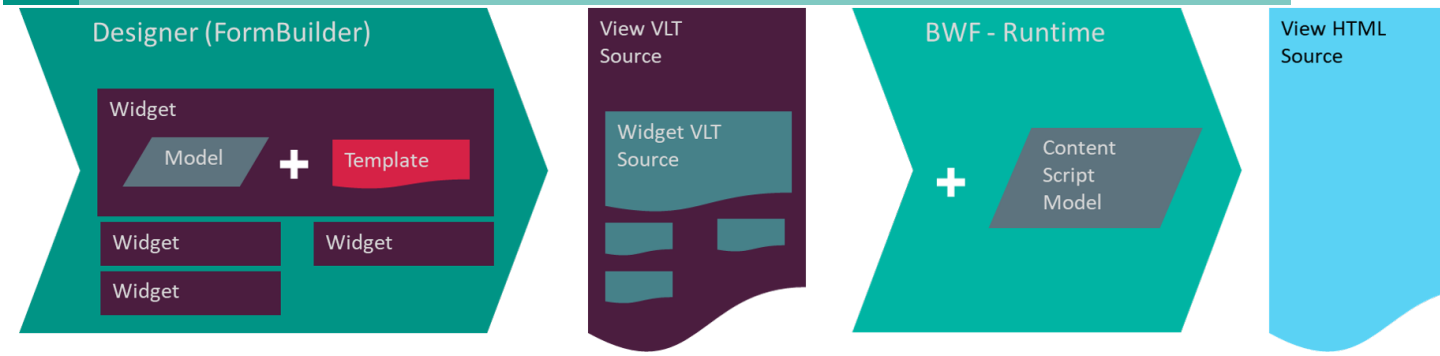
Beautiful WebForms Widgets are the base units a View is composed of (a View is in fact nothing but a collection of Widgets). Beautiful WebForms Widgets are implemented by Module Suite Template objects of type *Beautiful WebForm Snippet* stored under the **CSFormSnippets** folder in the Content Script Volume *(/administration/csvolume/)*.

Widgets are defined by a Model and a Template.

View's Widgets templates and their models are evaluated by the Form Builder[1] to produce the intermediate View Velocity Template Document (VVTD).

At runtime (when a WebForm is rendered) the Beautiful WebForm MVC framework evaluates the VTD against a Content Script Model to produce the final WebForm HTML page.

Copyright © 2013-2022 AnswerModules Sagl



Model and Template¶

The Widget model is implemented in the form of a Javascript object while the template is implemented in the form of an Handlebars *(https://handlebarsjs.com/)* template. The template might contain a set of partials *(https://handlebarsjs.com/guide/partials.html#partials)* defined by Module Suite Template objects of type *Content Script Snippet* stored under the **CSSystem** folder in the Content Script Volume *(/administration/csvolume/)*, partials can be identified because their name is prefixed by the ***Partial*** keyword.

Below an example of a Widget Model and template:

Model

```
{
  "fields":{
    ...
    "h_base"  :{"title":"Basics","type":"_help","help":"oh_baseProperties"},
    "fieldA":{"label":"A Field Label","type":"input","value":"","help":"Field's help message", "i18n
    ...
  }
  ,"title":"My Widget"
  ,"help":{"value":"oh_textInput"}
  ,"order":["fieldA", "fieldB"]
  ,"jsdependencies":[]
  ,"cssdependencies":[]
  ,"nonRendableWidgets":false
  ,"columns":true
  ,"binding":true
  ,"style":true
  ,"validation":true
  ,"readonly":true
  ,"container":false,
  ,"rendered": true
}
```

Template

Copyright © 2013-2022 AnswerModules Sagl

Copyright © 2013-2022 AnswerModules Sagl

```
65
66  {{> _renderedOpen}} {{!-- Manages the "Show-if" configuration option (creates the VTL expressio
67  {{#*inline "_componentClass"}}am-form-text-input{{/inline}}
68
69  {{#if label}}^
70      {{> _labelLeft}}
71      <div class="{{> _colSize}} {{>_componentClass}} {{add_class}}" {{>_amWID}}>  {{!-- {{> _col
        {{> _labelTop}}

    {{else}}
        <div class="{{> _colSize}} {{>_componentClass}} {{#unless label}}{{#if required}}am-form-re
    {{/if}}

            {{#if render}}
            #foreach( $rowField in {{id}} )
            {{> _defaultValue }}
            {{/if}}

            <div class="am-form-input-wrap"  {{> _popover }} >

                {{#if render}}
                #if({{{readonly}}})

                <p class="{{#if bold_body}}am-form-bold{{/if}}" >
                    <span class="form-control-static"> $esc.html({{id}}.value)</span>
                </p>

                #else

                {{/if}}
                <input id="{{id}}.id"
                      name="{{id}}.id"
                      value="{{#if render}}$esc.html({{id}}.value){{else}}{{placeholder}}{{/if}}" t
                      placeholder="{{placeholder}}"
                      class="form-control"
                      style="{{style}}"
                      data-constraints="{{id}}.validation('{{validation}}')"

                      {{#each dataatts}}
                      data-{{label}}="{{{value}}}"
                      {{/each}}
                      />

                    {{> _addDeleteButtons}}

                    {{> _showErrors}}
                {{#if render}}
                #end
                {{/if}}
            </div>

            {{#if render}}
            #end
            {{/if}}


            {{#if helptext}}
            <p class="help-block">{{helptext}}</p>
            {{/if}}

    {{#if label}}

        {{> _labelBottom}}
            </div> {{!-- Close component div --}}
        {{> _labelRight}}

    {{else}}
        </div> {{!-- Close component div --}}
    {{/if}}
```

Copyright © 2013-2022 AnswerModules Sagl

```
<!-- END Text input-->

{{> _renderedClose}}
```

Copyright © 2013-2022 AnswerModules Sagl

Model properties details

Property	Mandatory	Default	Note
fields	YES		A map containing configuraiton options. The options names and values are used to *build* the actual widget's model
title	YES		The widget's title as displayed in the left sidebar of the FormBuilder
help	NO		The help message displayed in in the Form Builder configuration panel, as well as on the FormBuilder's left sidebar
order	NO		A list containing the widget's configuration's options names in the order in which they should be displayed in the configuration panel
jsdependencies	NO		List of static javascript resources the widget depends on
cssdependencies	NO		List of static CSS resources the widget depends on
nonRendableWidgets	NO	false	if true the widget can be resized (if true columns field is automatically injected among the widget's model fields list) (default: true)
columns	NO	true	The help message displayed in in the Form Builder configuration panel
binding	NO	true	if true the widget can be bound to an attribute of the Form Template
style	NO	true	if true the field *Custom Style* is automatically injected among the widget's model fields list.
validation	NO	true	True if the widget support validation (default:true)
readonly	NO	true	if true the field *Read Only* is automatically injected among the widget's model fields list.
container	NO	false	if true the widget will act as a container. The final view source code for all the widgets that are, in the Form Builder's working area, between the container opening and closing widget will result wrapped by the source code generated by the widget itself. When dropped in the Form Builder's main working area the corresponding closing widget will be automatically created and bound to it. The closing widget shall be named after the opening widget and suffixed with **_closed**.
rendered	NO	true	True if the designer should be able to specify a condition under which the widget will be displayed ("Show if" configuration option)

{{#if render}} expression in Widgets templates

As previously discussed, widget templates are mainly used to generate the VVTD, however they are also used to generate the HTML code that represents the widget in the FormBuilder workspace. When the Widget template is evaluated to generate the HTML for the FormBuilder workspace, an additional "render" property is injected into the widget model, so the designer has the possibility to filter elements that should not be rendered in static HTML. (e.g. any Velocity *(https://velocity.apache.org/)* expression).

Designers can modify widgets' models properties using the Form Builder widgets configuration panel. Any a widget's model modification triggers the immediate re-evaluation of the widget's template resulting into an update of the source code.

Copyright © 2013-2022 AnswerModules Sagl

Static Resources Management¶

Beautiful WebForms widgest might depend on static resources (Javascript and CSS files). These dependencies are defined in the widget's model through the properties **jsdependencies** and **cssdependencies**.

The definition of a static-resource dependency is represented by a list of three elements:

- the reltaive [2] path to the static resource file
- the version of the resource to load (a string formatted as "Major.Minor.Revision")
- an optional list of dependency definitions for static resources this library depends on

E.g.

```
["v2/css/select2/select2-bootstrap","3.5.4", [ ["v2/css/select2/select2","3.5.4"] ] ]
```

When a form is rendered the framework computes the list of all the static resources required by the associated view's widgets. The list is optimized to avoid repetitions and to respect the proper loading order. The final list of static dependencies is then automatically injected by the framework in two ViewParams *(/working/bwebforms/views/#viewparams)* variables:`am_CssViewDependecies` and `am_JsViewDependecies`.

Beautiful webForms View Templates utilize the aformentioned variables to render the HTML code required to load the associated static files.

Two Velocity macros have been designed to handle this task:

```
#macro( bwfJsResources $resList $blackList )
#macro( bwfCssResources $resList $blackList)
```

These macros combine the contents of the variables `am_CssViewDependecies` and `am_JsViewDependecies` with the list of dependencies specified as macro arguments (which are typically dependencies specific to View Template *(/working/bwebforms/views/#beautiful-webforms-view-templates)*) to calculate the final list of static resources that must be loaded (producing at the same time the relevant HTML code).

$blacklist resources not to be loaded

It is sometimes desirable that the static resources that need to be loaded to satisfy a widget's dependency are not actually loaded, for example because they have been replaced by other resources already loaded by the View Template *(/working/bwebforms/views/#beautiful-webforms-view-templates)*, in these cases it is possible to pass to the above mentioned macros an additional optional list of resources not to be loaded.

E.g.

Copyright © 2013-2022 AnswerModules Sagl

```
#bwfCssResources([
    ['v2/css/am/am_form', "2.0.0"]
    ,['v2/css/font-awesome.min', "0.0.0"]
    ,['v2/css/metro-bootstrap.min', "0.0.0"]
    ,['v2/css/am/am_gridTable', "2.0.0"]
    ,["v2/css/select2/select2-bootstrap","3.5.4",
        [
            ["v2/css/select2/select2","3.5.4"]
        ]
    ]
],
[ ["v2/css/bootstrap.min","3.3.6"]])
```

There are situations in which it is necessary to load multiple views dependecies when a WebForm is rendered:

- It is necessary whenever the WebForms can programmatically swithc view (e.g. a Webform organized in tabs);
- It is necessary whenever the WebForm's View makes use of SubViews widgets;

In these cases it is possibile to use the Content Script `forms.addResourceDependencies` API in the view OnLoad *(/working/bwebforms/views/#custom-logic-execution-hooks-cleh)* CLEH Script to force the framework to also load static resources dependencies from other Views.

The above mentioned API accepts three parameters: `forms.addResourceDependencies(boolean loadJS, boolean loadCSS, String[] viewNames)`

- A boolean flag indicating if Javascript resources should be loaded;
- A boolean flag indicating if CSS resources should be loaded;
- An optional list of Views from where to load dependecies from, if not specified resources will be loaded for all the Views associated with the parent Form Template object;

View Names

Prior to Module Suite version 2.7 *(/releasenotes/2_0_0/)* Views names had to be specified in single quotes.

E.g.

```
forms.addResourceDependencies(true, true, "'View2'", "'View3'")
```

Starting with Module Suite version 2.7 *(/releasenotes/2_0_0/)* Views names have be specified without quotes.

E.g.

```
forms.addResourceDependencies(true, true, "View2", "View3")
```

Performances-tips: Always load the minimum amout of resources necessary

When a Beautiful WebForm View is created the framework automatically injects in the OnLoad *(/working/bwebforms/views/#custom-logic-execution-hooks-cleh)* CLEH Script the code required to load static resource

Copyright © 2013-2022 AnswerModules Sagl

dependecies from all the other views beloging to the same parent Form Template object. This code works well and has no impact on the performance of WebForm rendering, in most cases because Form Templates usually have very few associated views. However , there are situations in which this behaviour is not desirable (e.g. the Form Template contains many indipendent Views, the Form Template contains *non active* views etc..). loading static resource dependecies from other Views when unnecessary could be expensive and even lead to hardly detectable errors (e.g. a view in the template uses a different version of the widget library).
It's highly recommended, if your Form Template contains more than one view, to review the code automatically injected by the framework and modify it by passing to the `forms.addResourceDependencies` API (line 3) the list of Views from which it is actually necessary to load the resources.

```
1
2  form.viewParams.ajaxEnabled=true
3  if(form.viewParams.ajaxEnabled && !form.viewParams.isResourcesInit){
4      forms.addResourceDependencies( form, true, true)
5      form.viewParams.isResourcesInit = true
6  }
7  if (form.isFirstLoad()){
8    //Code to be executed on first load only
9    // es. form.myField.value = 'my value'
10
11 }
12 else{
13
14 }
```

Widgets libraries¶

A **Widgets library** is defined as an extensible set of Widgets that can be used through the drag & drop visual editor (FormBuilder). To simplify the navigation, the widgets are arranged in families of objects having similar functionalities. Widgets within the same library use the same initialization mechanism, as far as the JavaScript and CSS frameworks are concerned. Whenever it is necessary or convenient to introduce breaking changes, in the way in which the widgets are defined or in the way in which the widgets are managed, a new library is released.

No need to update

Beautiful WebForms is always shipped with a copy of all still supported previous libraries. When a new library is issued, customers are not required to immediatelly upgrade their views to it. They are free to keep working with previous widget libraries.

Do not mix libraries

Given the nature of the differences between different libraries it is of highly recommended not to use widgets on different libraries in the same view. Mixing widgets from different libraries can lead to unpredictable results or errors.

Widget Library V1¶

This is the first version of the widget library shipped with the first version of Module Suite. This widget library has been retired and is no longer supported since Module Suite version 2.6 (/

Copyright © 2013-2022 AnswerModules Sagl

releasenotes/2_6_0/). View Templates designed to work with library V1 are not compatible with any other library. Do not use other libraries' widgets with these View Templates.

Widget Library V2¶

This version of the widget library was first introduced with Module Suite 2.0 *(/releasenotes/2_0_0/)* and is still fully supported. This library is the first using the concept of static resources management. View templates leveraging this library loads their static resource dependencies through standard HTML tags `<link>` and `<script>`. The actual HTML code required to load resources is produced by the two Velocity macros (*bwfCssResources* and *bwfJsResources*) mentioned in the static resources management paragraph. View Templates designed to work with library V2 are not compatible with any other library. Do not use other libraries' widgets with these View Templates.

Widgets of library V2 have two additional model properties: *jsdependencies* and *cssdependencies*, they represent the list of static javascript and css resources the widget depends on:

The definition of a static-resource dependency is represented by a list of three elements:

- the reltaive [2] path to the static resource file
- the version of the resource to load (a string formatted as "Major.Minor.Revision")
- an optional list of dependency definitions for static resources this library depends on

E.g.

```
...
jsdependencies:[ ["v2/css/select2/select2-bootstrap","3.5.4", [ ["v2/css/select2/select2","3.5.4"] ]
...
```

Widget Library V3¶

This version of the widget library was first introduced with Module Suite 2.4 *(/releasenotes/2_4_0/)* and is still fully supported. This library revised the concept of static resources management. View templates leveraging this library loads their static resource dependencies through standard HTML tags as far as CSS resources are concerned and a JavaScript file and module loader Require JS *(https://requirejs.org/)* for Javascript resources. The actual HTML code required to load CSS resources is produced by the the Velocity macro (*bwfCssResources*) mentioned in the static resources management paragraph. View Templates designed to work with library V3 are not compatible with any other library. Do not use other libraries' widgets with these View Templates.

Widgets of library V3 have two additional model properties: *jsdependencies* and *cssdependencies*, they represent the list of static javascript and css resources the widget depends on:

Copyright © 2013-2022 AnswerModules Sagl

CSS dependecies

The definition of a static-resource CSS dependency is represented by a list of three elements:

- the reltaive [2] path to the static resource file
- the version of the resource to load (a string formatted as "Major.Minor.Revision")
- an optional list of dependency definitions for static resources this library depends on

E.g.

```
...
"cssdependencies":[
  ["v3/js/handsontable/handsontable.full","4.0.0", [["v3/js/handsontable/pikaday","1.4.0"]]]
  ,["v3/css/select2/select2","3.5.4"]
]
...
```

JS dependecies

The definition of a static-resource JS dependency is represented by a list of three elements:

- the reltaive [2] path to the static Javascript bundle containing the modules to be loaded
- the version of above mentioned bundle (a string formatted as "Major.Minor.Revision")
- the list of module that are part of the bundle (modules are defined by a list made of their name and version)

```
...
"jsdependencies":[
    ["v3/js/handsontable/am_init","1.0.0",[["Handsontable","4.0.0"], ["pikaday","1.4.0"], ["numbro","
]
...
```

Widget Library V4¶

This version of the widget library was first introduced with Module Suite 2.6 *(/releasenotes/2_6_0/)* and is still fully supported. This library it's an evolution of the previous iteration (library V3) which significantly increases the compatibility with standard Smart View UI. View templates leveraging this library loads their static resource dependencies through standard HTML tags as far as CSS resources are concerned and a JavaScript file and module loader Require JS *(https://requirejs.org/)* for Javascript resources, which is the same AMD library used by native Content Server Smart View framework. The actual HTML code required to load CSS resources is produced by the the Velocity macro (*bwfCssResources*) mentioned in the static resources management paragraph. View Templates designed to work with library V4 are not compatible with any other library. Do not use other libraries' widgets with these View Templates.

Widgets of library V4 have two additional model properties: *jsdependencies* and *cssdependencies*, they represent the list of static javascript and css resources the widget depends on:

Copyright © 2013-2022 AnswerModules Sagl

CSS dependecies

The definition of a static-resource CSS dependency is represented by a list of three elements:

- the reltaive [3] path to the static resource file
- the version of the resource to load (a string formatted as "Major.Minor.Revision")
- an optional list of dependency definitions for static resources this library depends on

E.g.

```
...
"cssdependencies":[
    ["amui/handsontable.full","4.0.0", [["amui/pikaday","1.4.0"]]]
    ,["amui/select2/select2","3.5.4"]
]
...
```

JS dependecies

The definition of a static-resource JS dependency is represented by a list of three elements:

- the name of the Javascript bundle containing the modules to be loaded, the bundles and the names of the modules no longer contain references to the name of the library version
- the version of above mentioned bundle (a string formatted as "Major.Minor.Revision")
- the list of module that are part of the bundle (modules are defined by a list made of their name and version)

```
...
"jsdependencies":[
    ["bwf/handsontable/am_init","1.0.0"]
]
...
```

---

1. FormBuilder acts as a Model View Controller framework with respect to BWF Widgets
2. Paths are relative to the folder **/support/ansbwebform/lib**
3. Paths are relative to the folder **/support/ansbwebform/lib/v4**, paths are defined in the View Template through Velocity expressions

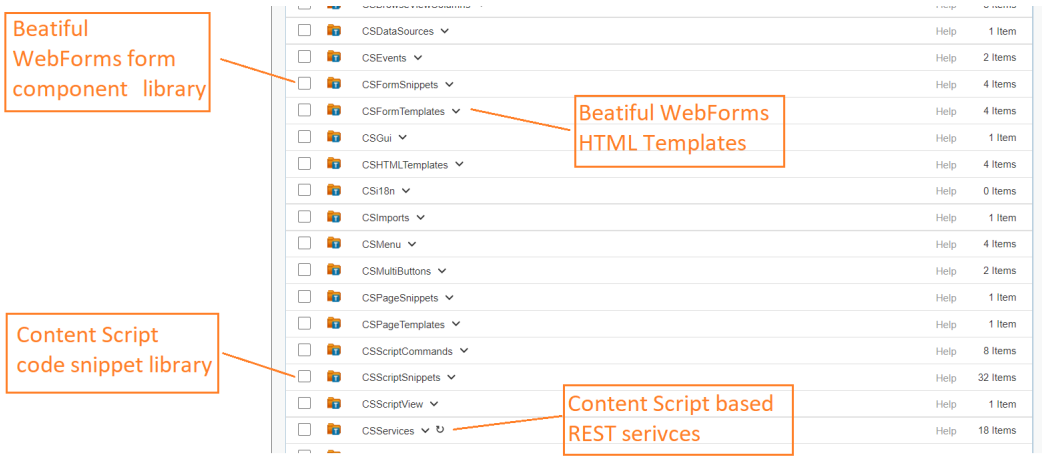
Copyright © 2013-2022 AnswerModules Sagl

Extending BWF

Content Script Volume¶

As for Content Script, Beautiful WebForms makes use of the Content Script Volume to store a set of objects necessary for the correct operation of the framework. These object are stored in specific containers, which will be covered on the following sections:

- CSFormTemplates
- CSFormSnippets
- CSServices
- CSScriptSnippets



CSServices¶

The **CSServices** *(/working/contentscript/rest/)* container is dedicated to Content Scripts that should be accessible as REST services, and has been covered in the previous sections.

Content Script REST services are somehow related to Beautiful WebForms in that some components used to build forms (essentially, the ones with AJAX capabilities) make use of these services to work correctly.

An example is the **getuserbyname** REST service, which backs the user selection components available in the form builder.

Copyright © 2013-2022 AnswerModules Sagl

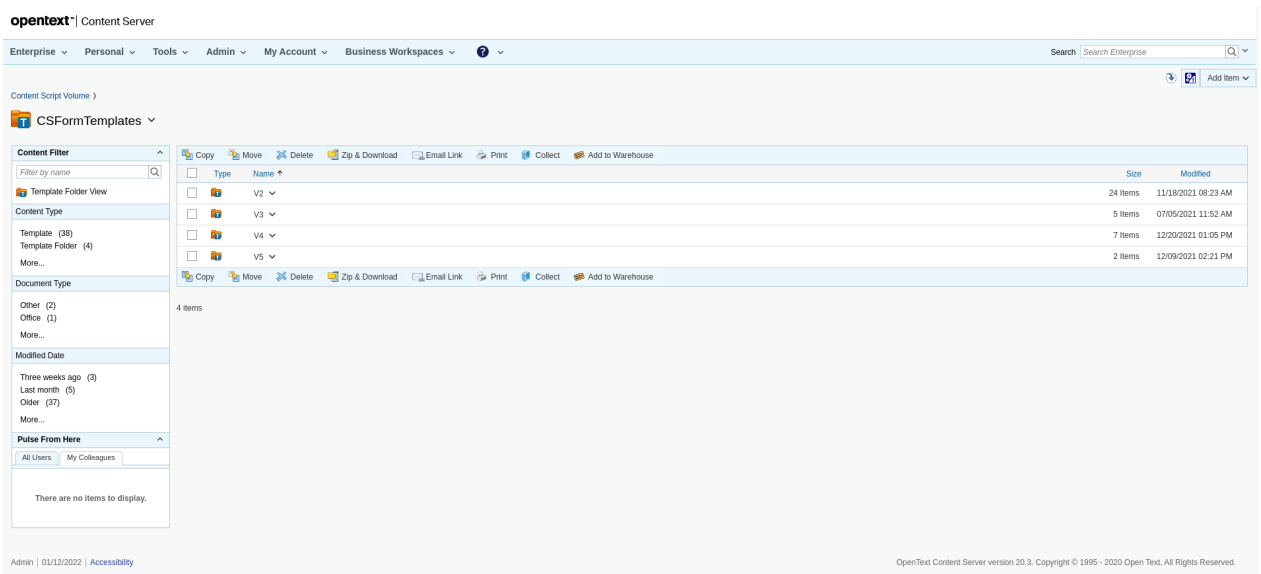
CSFormTemplates¶

The **CSFormTemplates** container is dedicated to HTML templates associated to **Beautiful WebForms Views**.

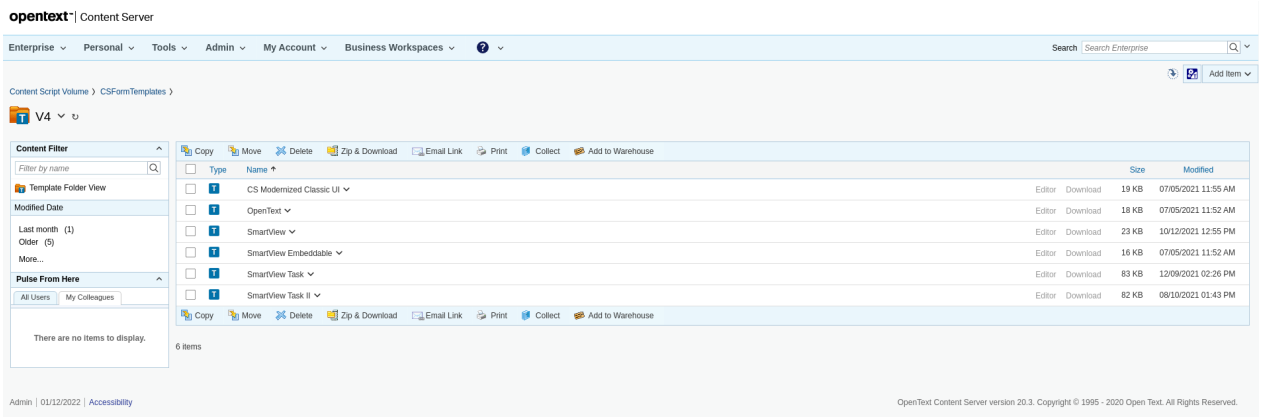
The templates are essentially **Velocity** HTML templates. A placeholder expression indicating where the actual Form Fieldset should be placed, this should usually be present in all Beautiful WebForms Templates.

Beatiful WebForms Templates are grouped by the library version:

- Content Script Volume
  - CSFormTemplates
    - V2
    - V3
    - V4
      - *<custom template A>*
      - *<custom template B>*



Copyright © 2013-2022 AnswerModules Sagl



New templates added to a library version folder will automatically be available in the **template selection** dropdown menu accessible from the Beautiful WebForms Views Specific Properties tab.

CSFormSnippets¶

The **CSFormSnippets** container is dedicated to the libraries of **components** that are available to build Beautiful WebForms views.

The CSFormSnippets container is organized on two levels: the first level is a container and identifies the Component Family, while at the second level there are the actual components.

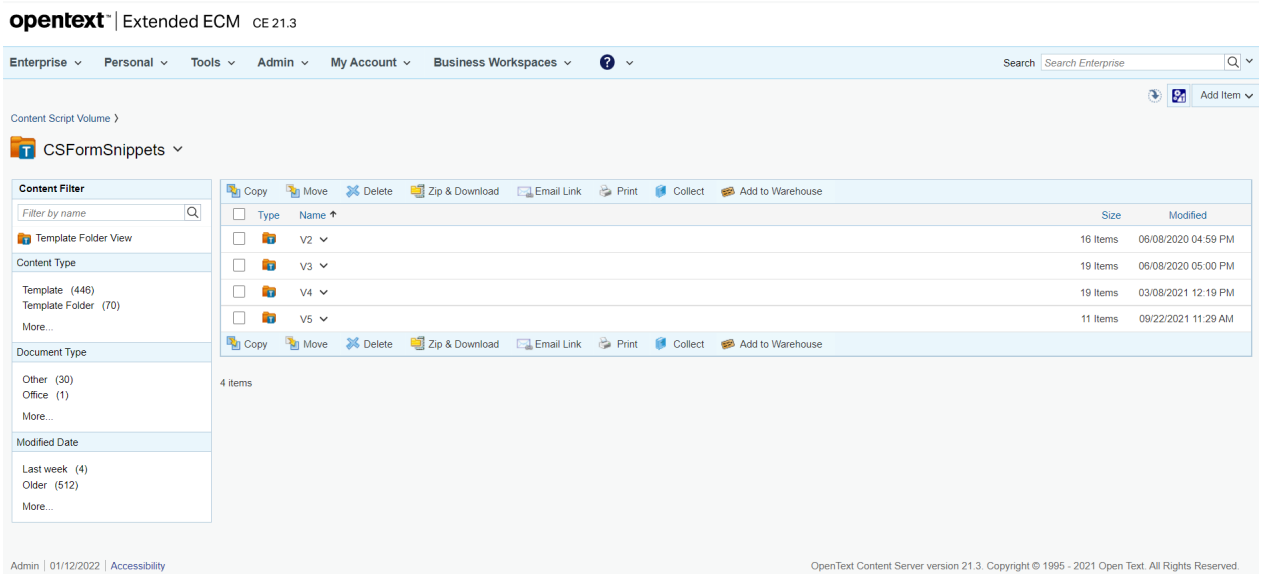
The Beatiful WebForms Snippets are stored in a two levels folders hierarchy: the first level is a library version container, while the second level is a container that identifies the Component Family.

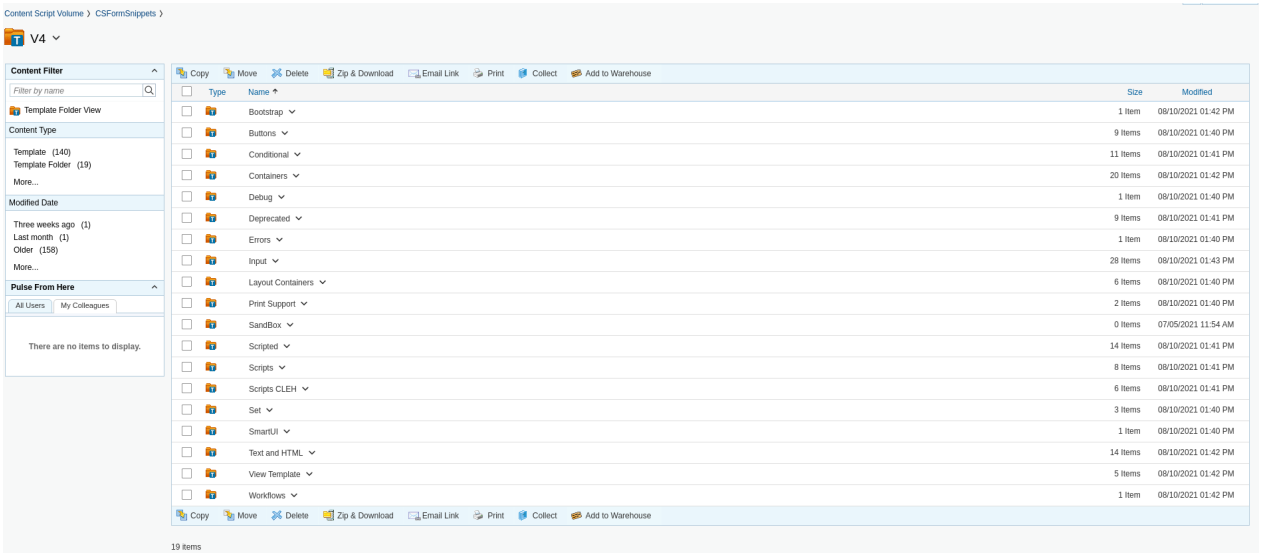
- Content Script Volume
  - CSFormSnippets
    - V2
    - V3
    - V4 - *library version level*
      - Buttons - *Component Family level*
      - Input
        - CheckBox
        - Datepicker
        - Text input
        - *<custom component A>*

Copyright © 2013-2022 AnswerModules Sagl

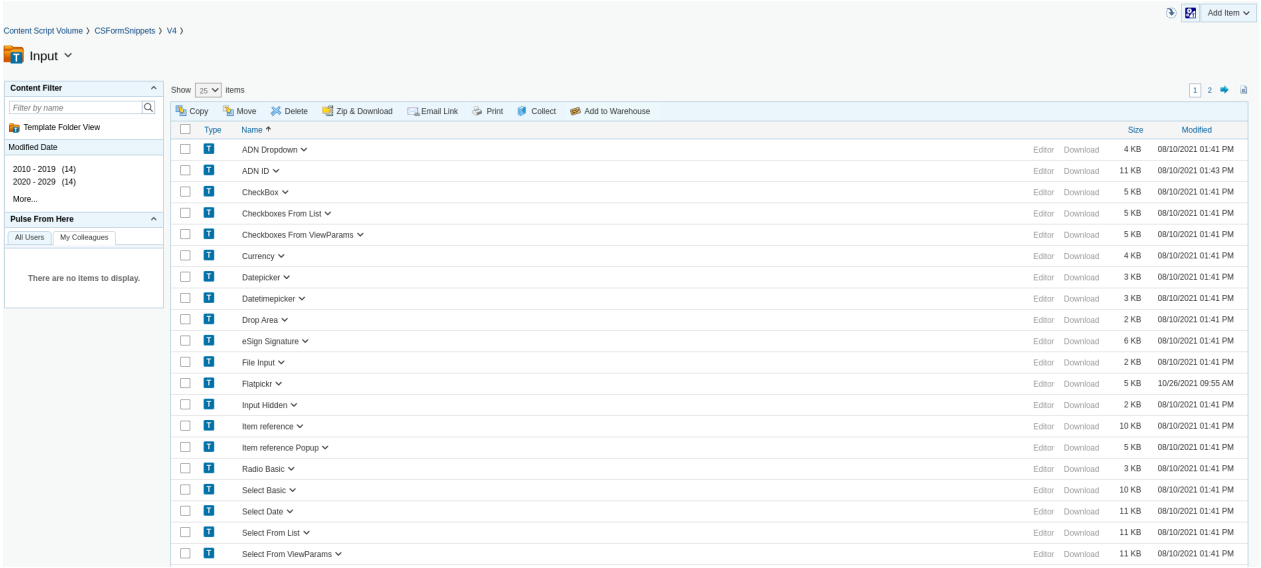
- Set

New component families and components created in this container will automatically be available to the developer in the Beautiful WebForms Form Builder tool.





Copyright © 2013-2022 AnswerModules Sagl



Embed into Smart View¶

Why?¶

The main purpose of embedding BWF views into Smart View's tiles is to leverage the BWF framework as a primary input mechanism for your next EIM applications. Integrating BWF into Smart View wont just enable you to collect and validate user's input but also to perform complex actions and surface the most relevant business information in highly interactive dashboards.

Create an embeddable WebForms¶

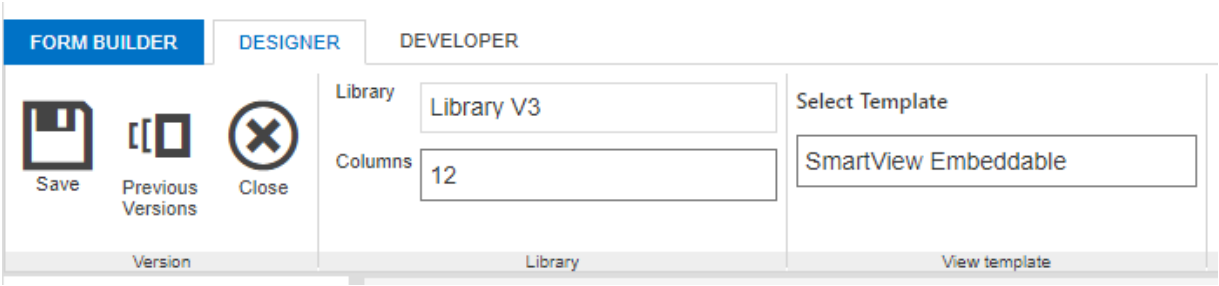
Creating an *embeddable webforms* is not different from creating any other webform on the system. The steps are:

- Create a **Form Template** object
- Create a **Beautiful WebForm View** view associated to the **Form Template** created in the previous step
- Using the Beautiful WebForms Form Builder define your form (structure and layout)

The embeddable view template

Copyright © 2013-2022 AnswerModules Sagl

What makes a Beautiful WebForms view embeddable into the Smart View is the usage of the **V3:SmartView Embeddable** view template



- Create a standard Content Server Form object and associate it to the previously created **Form Template** and **Beautiful WebFomr View**

How to publish a Webform into a Smart View perspective¶

In order to publish a WebForm in a Smart View perspective's tile you need either:

ModuleSuite Smart Pages is installed

1. A Content Script object (for managing the server side initialization of the form)
2. An **AnswerModules ModuleSuite:Content Script Result** perspective tile, configured to use the above script as datasource

or

ModuleSuite Smart Pages is not installed

1. A Content Script object (to mange the server side initialization of the form)
2. A WebReport to encapsulate the above script execution
3. An **Content Intelligence:HTML WebReport** perspective tile, configured to use the above script as Webreport as datasource

ModuleSuite Smart Pages is installed¶

If the ModuleSuite Smart Pages is installed on your system you will be able to leverage the tight integration between ModuleSuite and the OTCS Smart View in order to add WebForms in perspective's tiles.

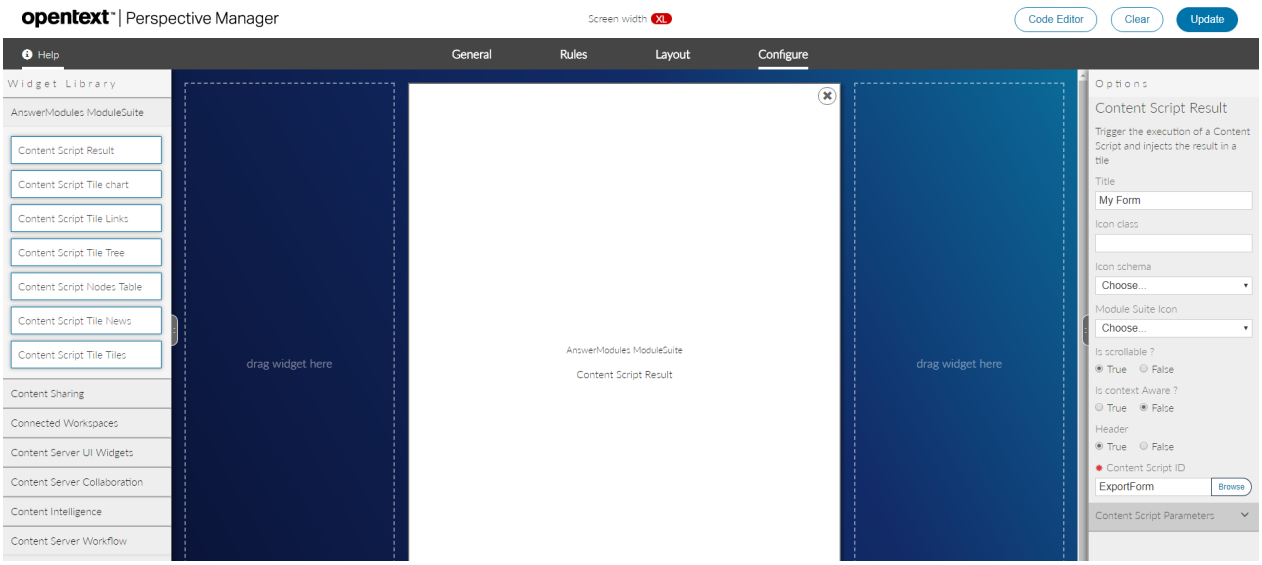
In this case the minimum Content Script required for managing the server side initialization of the form will be:

```
def formNode = docman.getNodeByPath("Path:To:Your:Form")
form         = formNode.getFormInfo()
view         = formNode.view
```

Copyright © 2013-2022 AnswerModules Sagl

```
form.viewParams.uiParentID = params.uiParentID //The perspective current space

json([
    output:view.renderView(binding, form),
            widgetConfig:[
                reloadCommands:["someCommand"],
                tileContentClasses:"am-whitebckg",
                tileLayoutClasses:"am-whitebckg"
            ]
        ]
)
```

The configuration of the associated **AnswerModules ModuleSuite:Content Script Result** will be as simple as:



ModuleSuite Smart Pages is not installed¶

If the ModuleSuite Smart Pages is not installed on your system you will not be able to leverage the tight integration between ModuleSuite and the OTCS Smart View in order to add WebForms in perspective's tiles, thus you will need an additional **WebReport** object in order to encapsulate the execution of the Content Script data source.

In this case the minimum Content Script required for managing the server side initialization of the form will be:

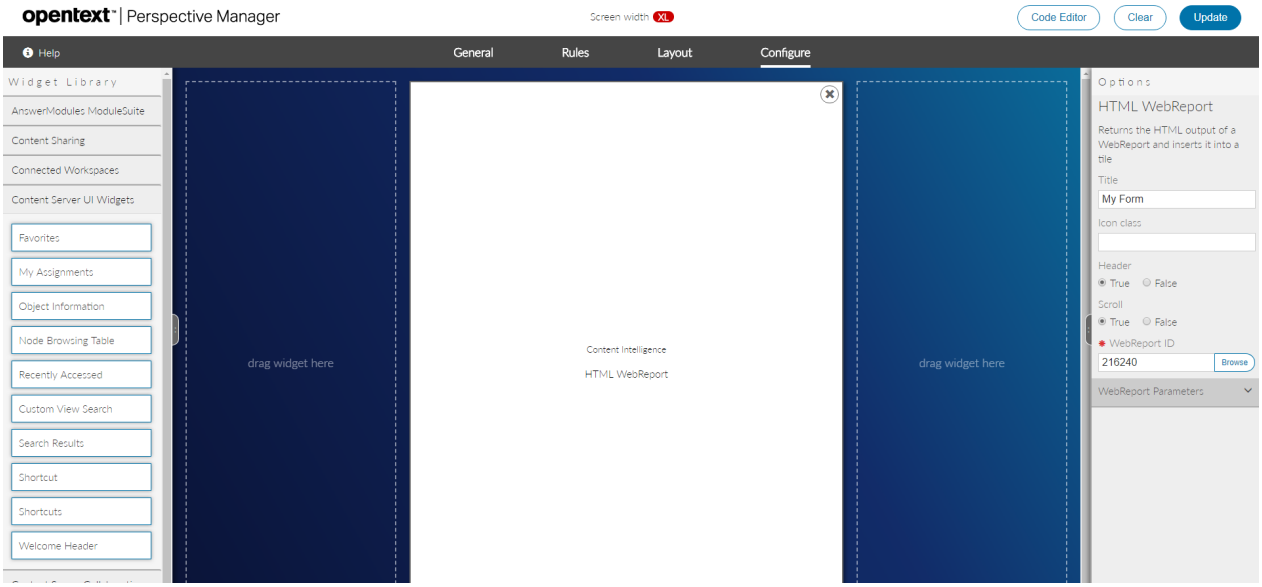
```
gui.gui = false
def formNode = docman.getNodeByPath("Path:To:Your:Form")
form         = formNode.getFormInfo()
view         = formNode.view
out << view.renderView(binding, form)
```

While the minimum WebReport required to encapsulate the execution of the above script will be:

Copyright © 2013-2022 AnswerModules Sagl

```
[LL_REPTAG_'123456' RUNCS /] [// Script ID
[LL_WEBREPORT_STARTROW /]
[LL_WEBREPORT_ENDROW /]
```

The configuration of the associated **Content Intelligence:HTML WebReport** will be as simple as:



Beautiful Webforms views updater¶

What is it?¶

The Beautiful Webforms View Updater (BWVU) is an utility designed to simplify and automate the process of upgrading a webform view designed with a previous version of Module Suite. Module Suite IDEs allows you to keep working with the views created using the widget library shipped with a previous version of Module Suite, nevertheless, in order to leverage the widgets introduced in a newer version of the widget's library an upgrade is required.

This tool aims to simplify the upgrade procedure.

Tool setup¶

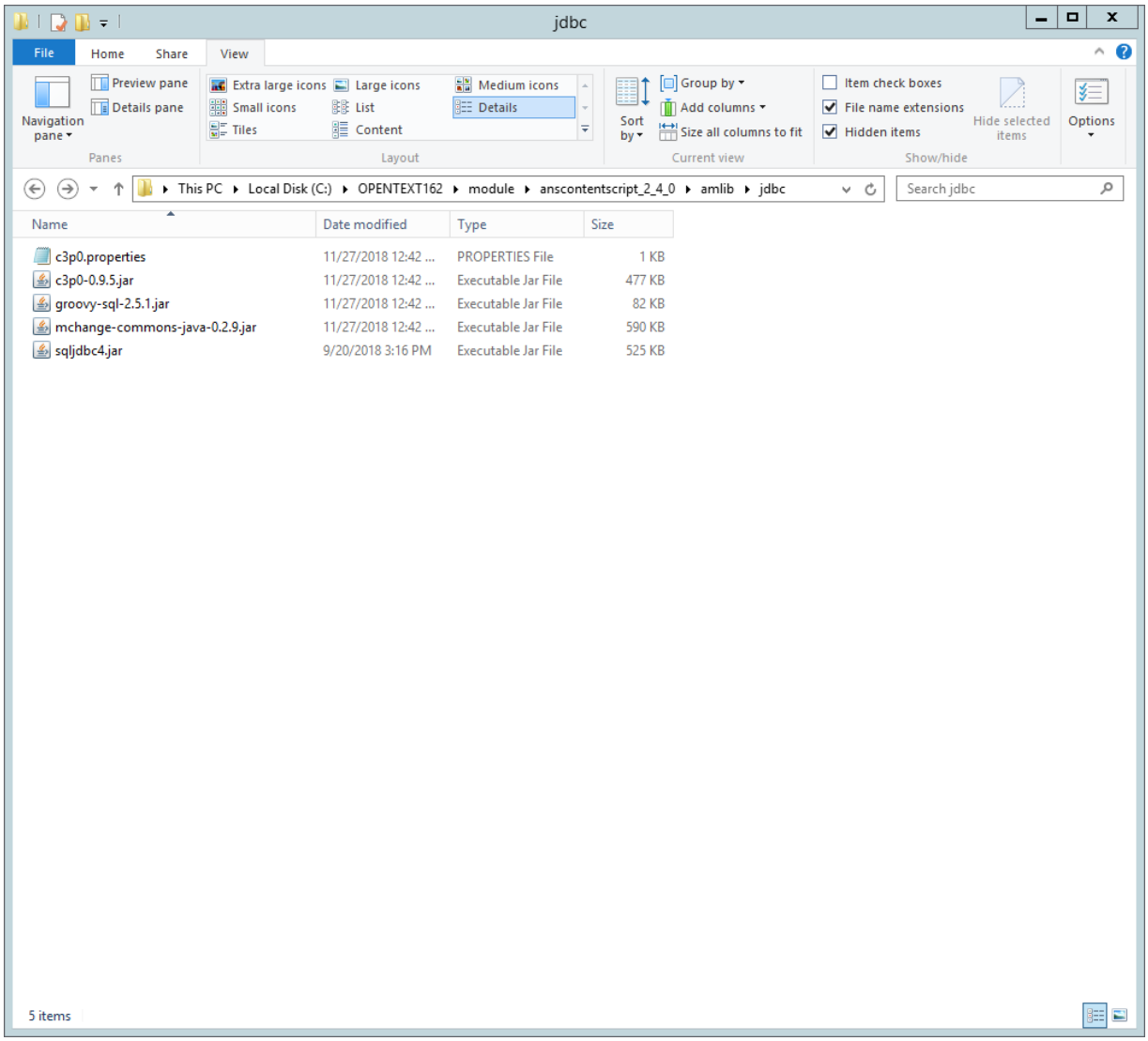
Installing the BWVU, is a straight forward procedure which consists of just two steps:

BWVU dependecies

BWVU is a Module Suite application that leverage extensively the jdbc extension package. It requires you to have it installed and to have the proper JDBC drivers for your Content Server DBMS deployed in the OTHOME/module/

Copyright © 2013-2022 AnswerModules Sagl

anscontentscript_X_y_Z/amlib/jdbc folder.



- [x] Run the BWVU installer on a server on which both Content Server and Module Suite have been installed and configured. The installation program will ask for the location of the Content Server's installation folder.

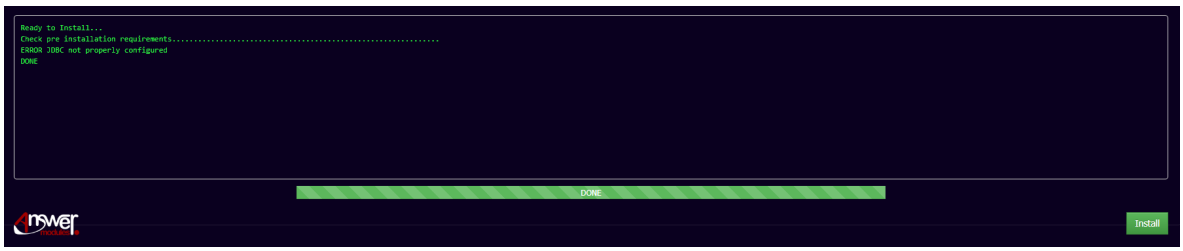
Cluster installation

If you are working with a cluster and want to be able to run the tool from any server of the cluster, you should perform this step on each one of them

- [x] Run the installation script http://your.contentsuite.com/otcs/cs.exe?func=amcs.executeadmcs&script=bwfupdater_install.cs *(http://your.contentsuite.com/otcs/cs.exe?func=amcs.executeadmcs&script=bwfupdater_install.cs)* and click "Install" :

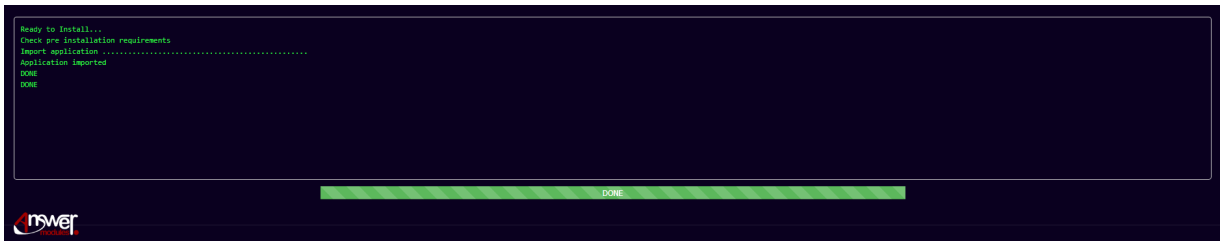
Copyright © 2013-2022 AnswerModules Sagl

The application will perform some pre-installation checks and will eventually inform you regarding errors that prevent the successful completion of the installation procedure.



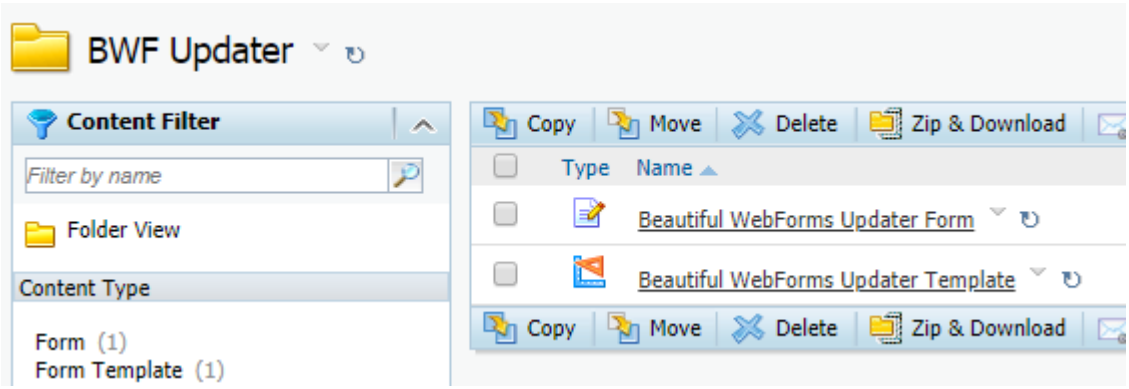
Installation completed

If the installation was completed without errors, you should see a message like the one below



Tool usage¶

The main entry point for using the tool is the webform **Beautiful WebForms Updater Form** created in the OTCS Enterprise Workspace in the **BWF Updater** folder .

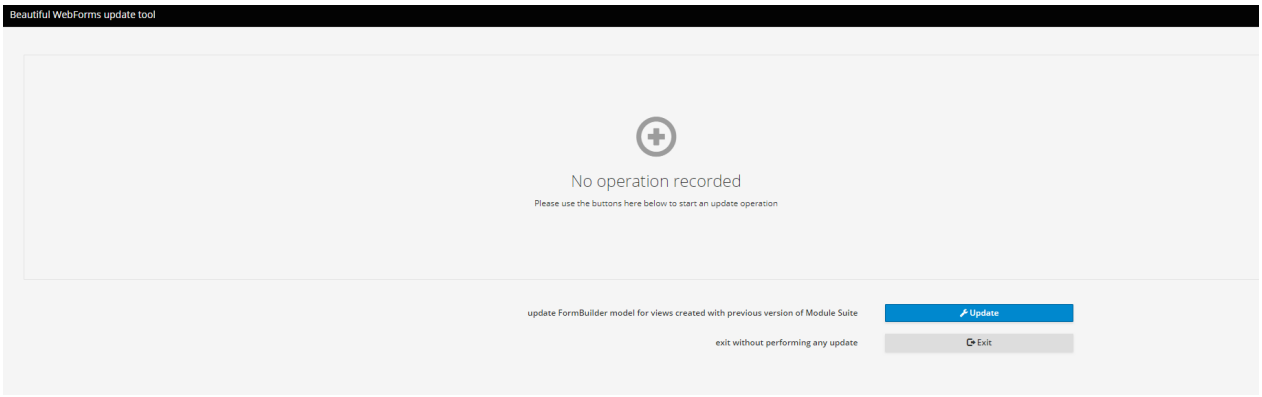


Move and rename

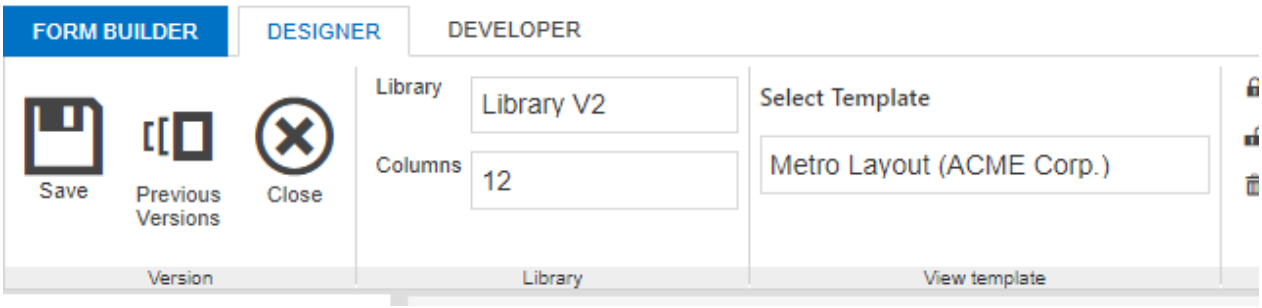
Upon import is completed, the root folder and tool objects (WebForms and Form Template) can be renamed and moved according to your needs without problems.

Copyright © 2013-2022 AnswerModules Sagl

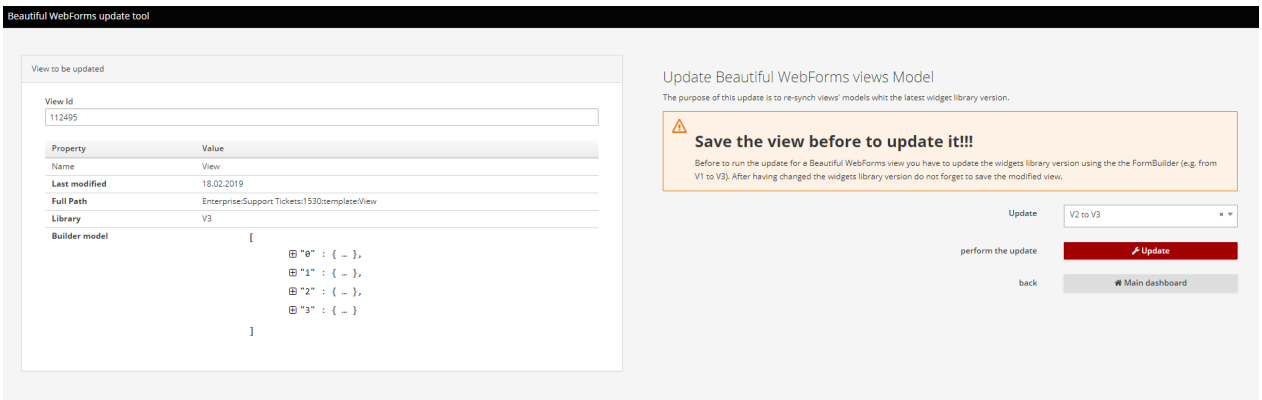
Running the above mentioned WebForm you'll enter the tool's dashboard. From here you can either review the results of previous updates or, by clicking on the **Update** button, run a new one.



Before trying to perform the update of an existing Beautiful WebForms view you have to update the widgets library version using the the FormBuilder (e.g. from V1 to V3). After having changed the widgets library and saved the modified view you can come back to the BWVU tool.



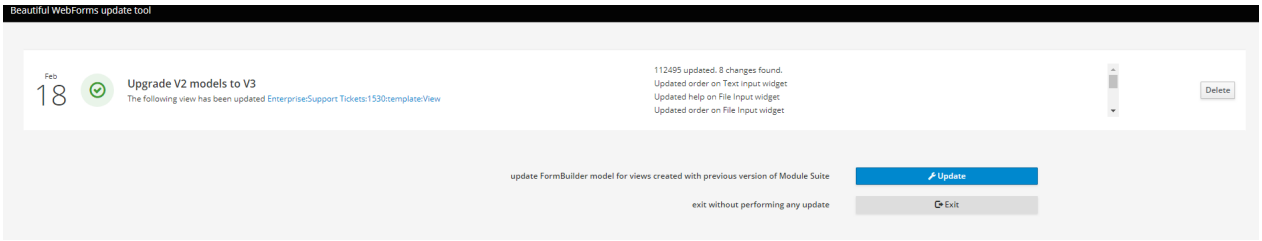
Once the view's widgets library version is up to date you can execute the desired upgrade using the BWVU tool, simply enter the view unique id (DataId) and click **Update**.



The result of the update together with the detailed list of operations performed will be

Copyright © 2013-2022 AnswerModules Sagl

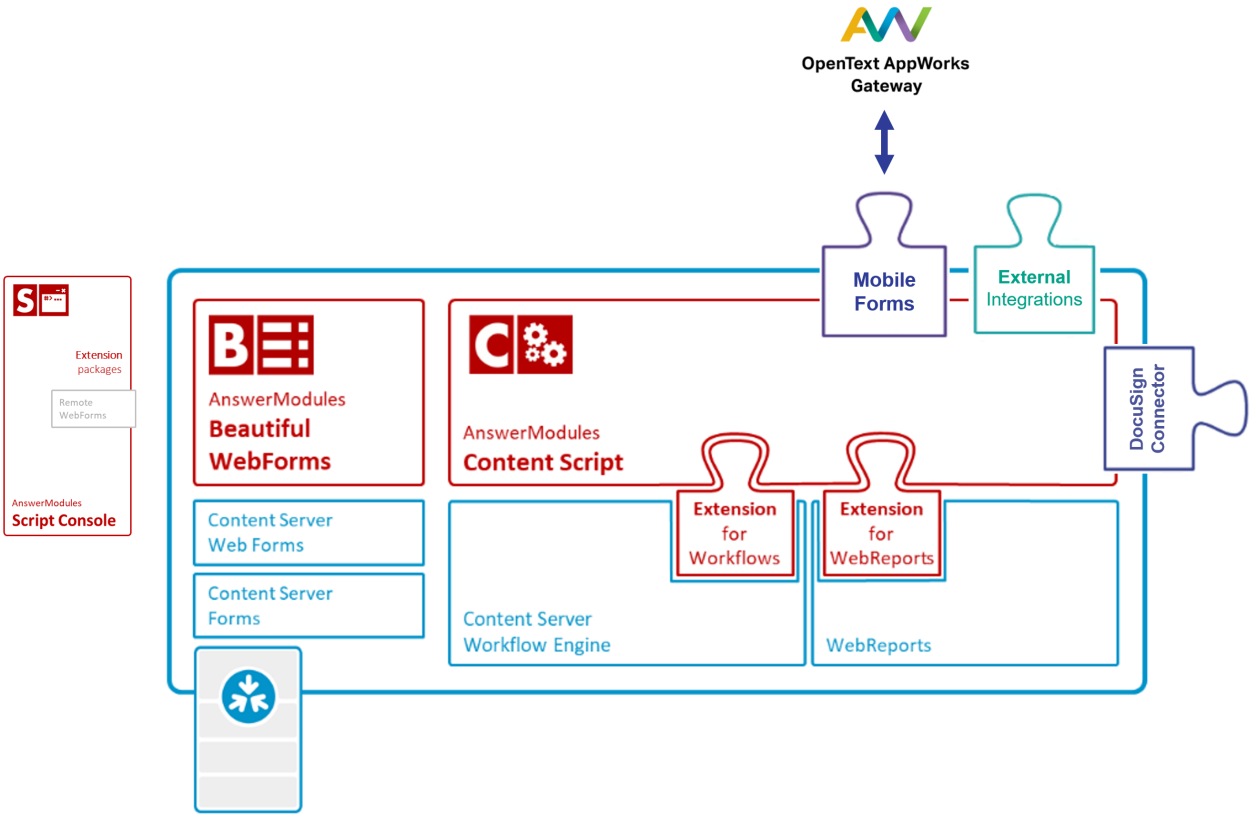
available for review in the tools dashboard.



Extension: Mobile WebForms

What is it?¶

AnswerModules' Mobile WebForms is both: - An add-on solution for CSP/xECM. - A functional extension for Module Suite (AnswerModules' core solution).



AnswerModules' Mobile WebForms consists of three macro components:

AppWorks Mobile Application¶

Every Mobile WebForms is transformed into an AppWorks application so that it can be it distributed to end-users' devices through the AppWorks Gateway. This approach guarantees a

Copyright © 2013-2022 AnswerModules Sagl

very high degree of flexibility in terms of controlling access to the mobile form as well as governing the mobile form's data security. By leveraging the AppWorks technology, a mobile form's lifecycle can be fully managed (versioning, fine-grain user distribution, etc..), support for specific devices may be pre-defined and if necessary saved data could be remotely deleted from a specific device.

Module Suite based extension for REST APIs¶

By extending the CSP/xECM REST APIs a dedicated endpoint for Mobile WebForms has been created. The endpoint can be easily extended or adapted in order to effectively open a potentially infinite number of use cases when it comes to how form data is utilized and persisted once its synchronized onto CSP/xECM. Some possible scenarios for how the form data can be utilized include: starting or updating a workflow, creating Connected Workspaces programmatically, generating documents (PDF, Word, Excel, etc...), transmitting the data to another system (i.e.: CRM, ERP, etc...), and much more.

Mobile WebForms Application Builder¶

This component allows to create new AppWorks applications in a matter of minutes starting from an existing form. An intuitive wizard-like tool guides users in defining all the necessary elements to transform a simple WebForm into a Mobile WebForms. A preview of the process can be viewed at: https://youtu.be/xiBjPMAH-HU *(https://youtu.be/xiBjPMAH-HU)*

Mobile WebForms setup¶

Installing the Mobile WebForms application on your system is a straightforward procedure made of a few simple steps.

> **As administrator**
>
> The installation procedure must be performed using a user with administrative rights on the system (for example, the administrator user)

- Download the Mobile WebForms Installation Package. (You can download it from here)
- Extract the contents of the zip file to a temporary location.
- Copy the contents of the Mobile Components.zip in the <Content_Server_home> directory and then restart the Content Server services.
- Logon to the OpenText Content Server with an administrative account.
- Create a folder that will contain the installation package.
- Upload the mobileWebFormsXML.xml file, in the previously created folder.
- Create a Content Script in the same location for importing the package in the system. (please refer to the snippet below as a reference).

Copyright © 2013-2022 AnswerModules Sagl

```
def source    = docman.getNodeByName(self.parent, "mobileWebFormsXML.xml")
def xmlFolder = docman.getNodeByName(self.parent, "Mobile WebForms")
if(!xmlFolder){
    xmlFolder = admin.importXml(self.parent, source.content.content)
}
redirect "${url}/open/${docman.getNodeByName(xmlFolder, 'Install').ID}?scriptInstall=${self.ID}"
```

The execution of the Content Script will generate a folder in the Enterprise Workspace named “MobileWebForms” and will generate the application’s contents in it.

Pre-requisites

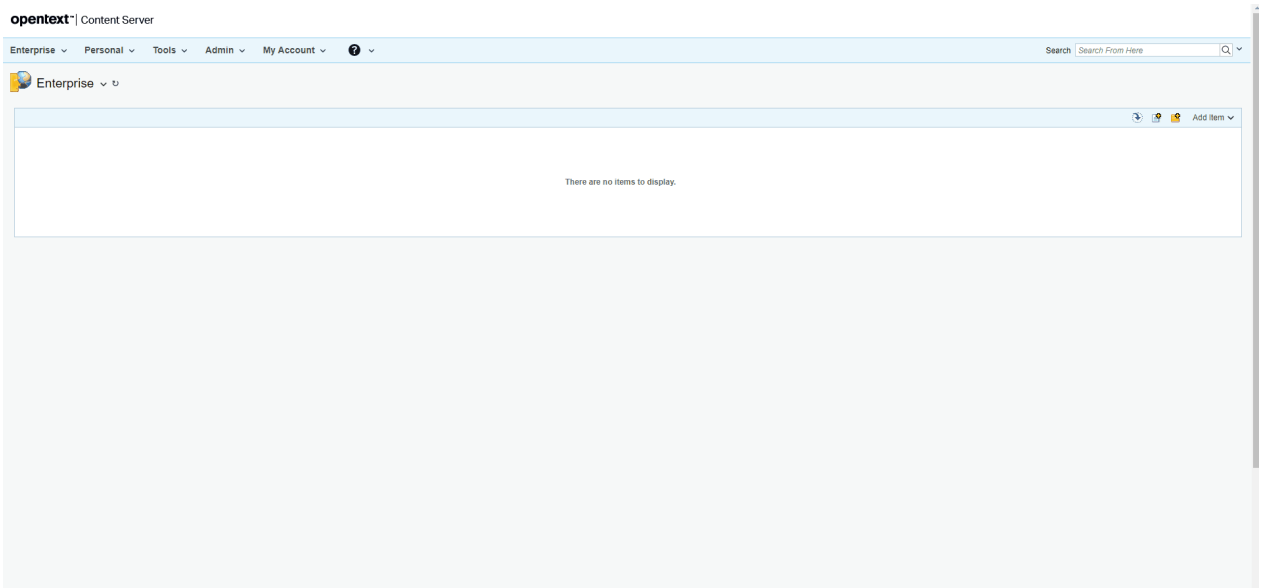
During the setup process the installer, will check if all the prerequisites are met. If the setup process notifies the need of a missing extension package, install the package before continuing.

To install an extension package you can refer to the following guide: http://developer.answermodules.com/manuals/current/installation/extpacks/ *(http://developer.answermodules.com/manuals/current/installation/extpacks/)*

In the case the requested extension is the AnswerModules' Cache Extension Package then after the installation some additional configuration will be needed.

To properly configure the AnswerModules’ Cache Extension Package refer to the below guide:

https://support.answermodules.com/portal/kb/articles/content-script-extension-cache *(https://support.answermodules.com/portal/kb/articles/content-script-extension-cache)*



Using the tool¶

A Mobile WebForms application is composed of three main elements:

- A form for inserting the information.
- An end-point Content Script that will implement the logics to properly manage the data upon synchronization from the OpenText AppWorks Gateway application.

Copyright © 2013-2022 AnswerModules Sagl

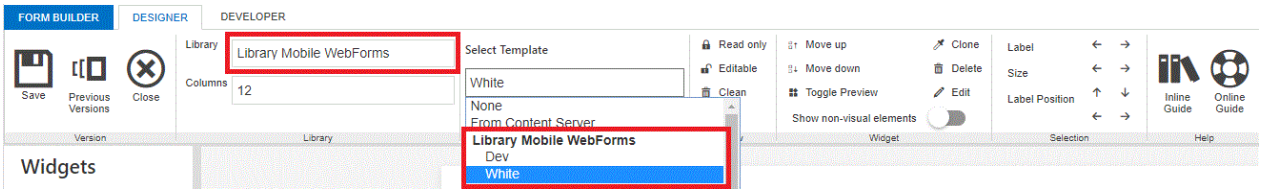
- An OpenText AppWorks Gateway application for distributing the application to the end users.

Creating the form¶

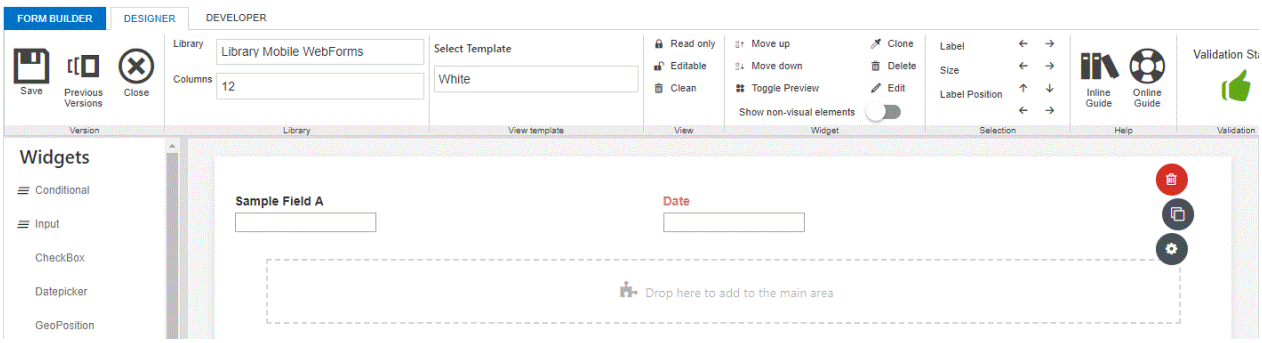
The first step is the creation of the form that will be utilized to gather information from the end users.

When editing the form's view with the Form Builder the widget library must be set to "Library Mobile WebForms". As for the template to use there are two options under the "Library Mobile WebForms" section:

- Dev: this template offers the possibility to verify the look & feel of the form without the need to deploy it on the OpenText AppWorks Gateway. This template should be only utilized during the development phase or for debugging purposes.
- White: this is template to be utilized when the application is ready to be deployed on the OpenText AppWorks Gateway.



When editing a form's view with the Form Builder, the form's view will be pre-populated with the widgets representing the elements inserted in the Form Template. A Mobile WebForms will need to be designed using specific widgets coming from the Mobile WebForms Library, to do so delete the self created widgets derived from the form template, verify that the Library Mobile WebForms is selected, save the form's view and refresh the page. Once the page has refreshed drag&drop the widgets from the left-hand side of the Form Builder to the form's view.



Implementing the Content Script end-point¶

When synchronizing the information back to Content Server, the Mobile WebForms application will make a call to a Content Script.

Copyright © 2013-2022 AnswerModules Sagl

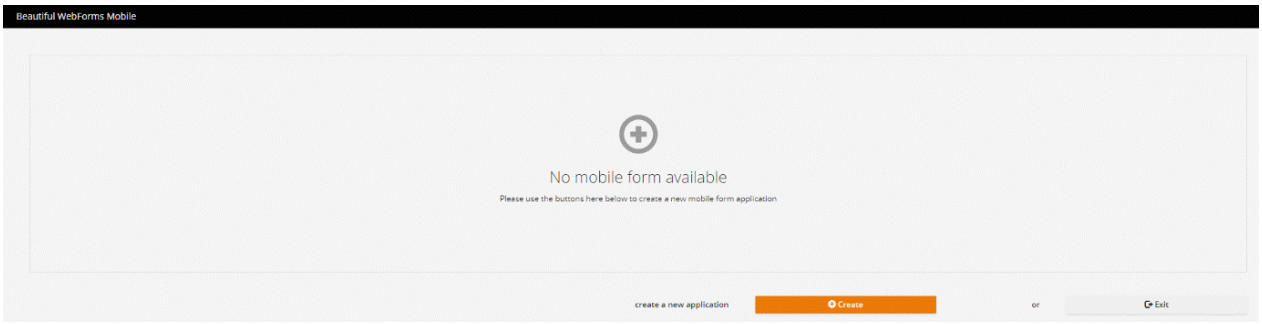
For a detailed explanation on using AnswerModules' Content Scripts please refer to the following guide: http://developer.answermodules.com/manuals/current/working/contentscript/otcsobj/ *(http://developer.answermodules.com/manuals/current/working/contentscript/otcsobj/)*

The Content Script must reside inside the CSServices folder within the Content Script Volume. The script must contain all the business logic needed to properly manage the information that is being synchronized from the OpenText Gateway application. The installation process will create a default end-point called "mobileWebForms" please refer to it as a reference implementation.

Building the OpenText AppWorks Gateway Application¶

To deploy the application on the OpenText AppWorks Gateway it will be necessary to prepare a deployable package compliant with the OpenText AppWorks Gateway. The preparation of the up-said package can be done via the Mobile WebForms application by opening the form “Registered Applications”. The form can be found under Enterprise\MobileWebForms\Application Builder\Builder

Once opened, the form will show the list of registered application. New applications can be created by clicking on the "Create" button at the bottom of the page.

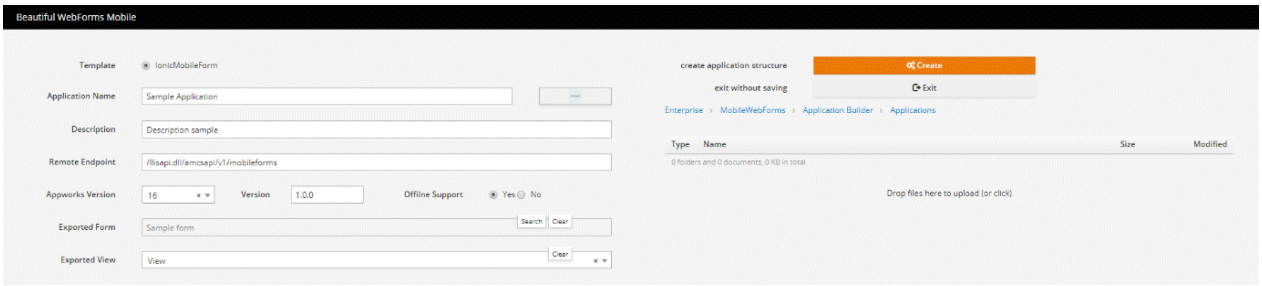


Clicking on the “Create” button will prompt the user for the application's details.

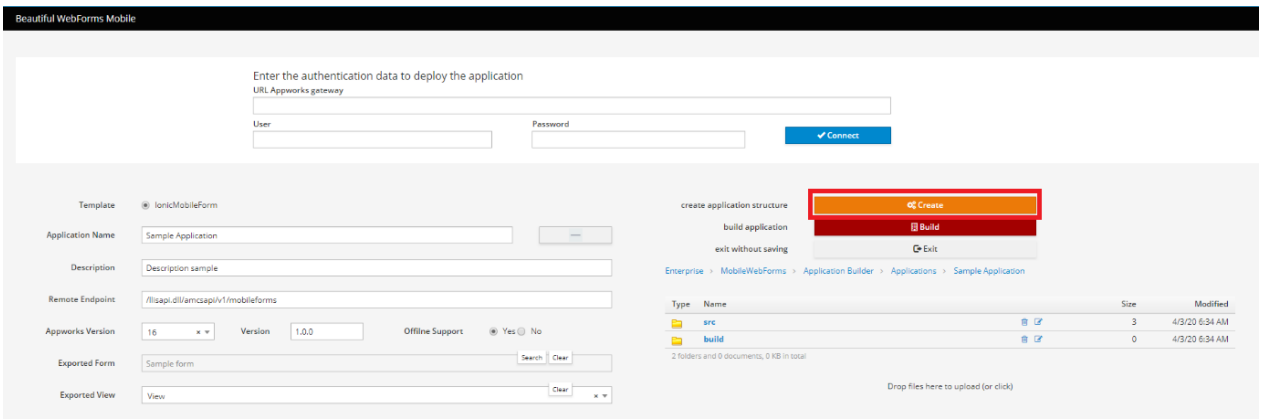
- Application name
- An icon for the application (to be shown as the application's icon on the mobile device)
- A description (to be set as the application's description on the mobile device)
- The remote end-point script name (called when synchronizing the form's data)
- The Appworks Gateway version
- The application’s version (When updating the application the version number must be increased)
- The related form

Copyright © 2013-2022 AnswerModules Sagl

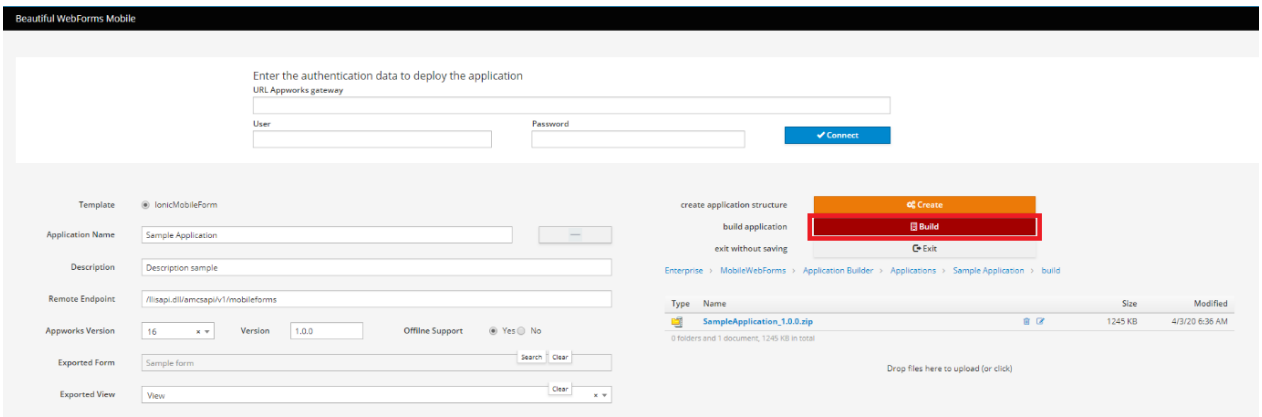
- The specific view to be used



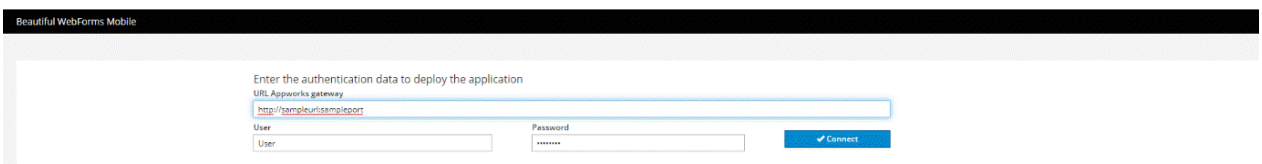
Clicking the “Create” button will automatically create an appropriate folder structure containing all the application's required objects



Once the application's structure has been created it will be possible to create an OpenText AppWorks Gateway deployable package by clicking on the "Build" button.

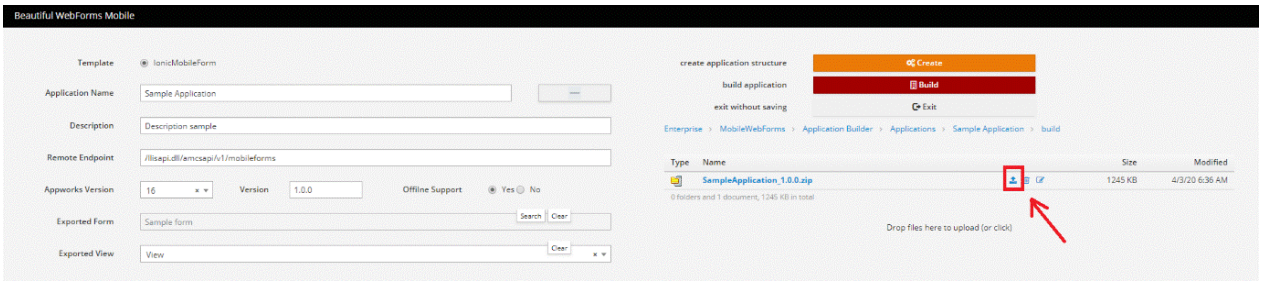


To upload the application to the OpenText AppWorks Gateway, enter the path and the authentication credentials of the destination OpenText AppWorks Gateway and click "connect".

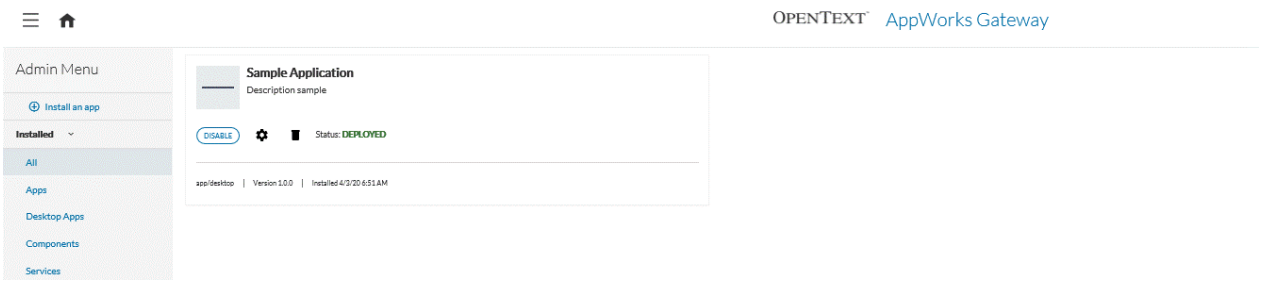


Copyright © 2013-2022 AnswerModules Sagl

Once connected to the OpenText AppWorks Gateway, the system will enable the user to deploy the application. Clicking on the deploy icon will automatically upload, install and enable the application on the OpenTextAppWorks Gateway.



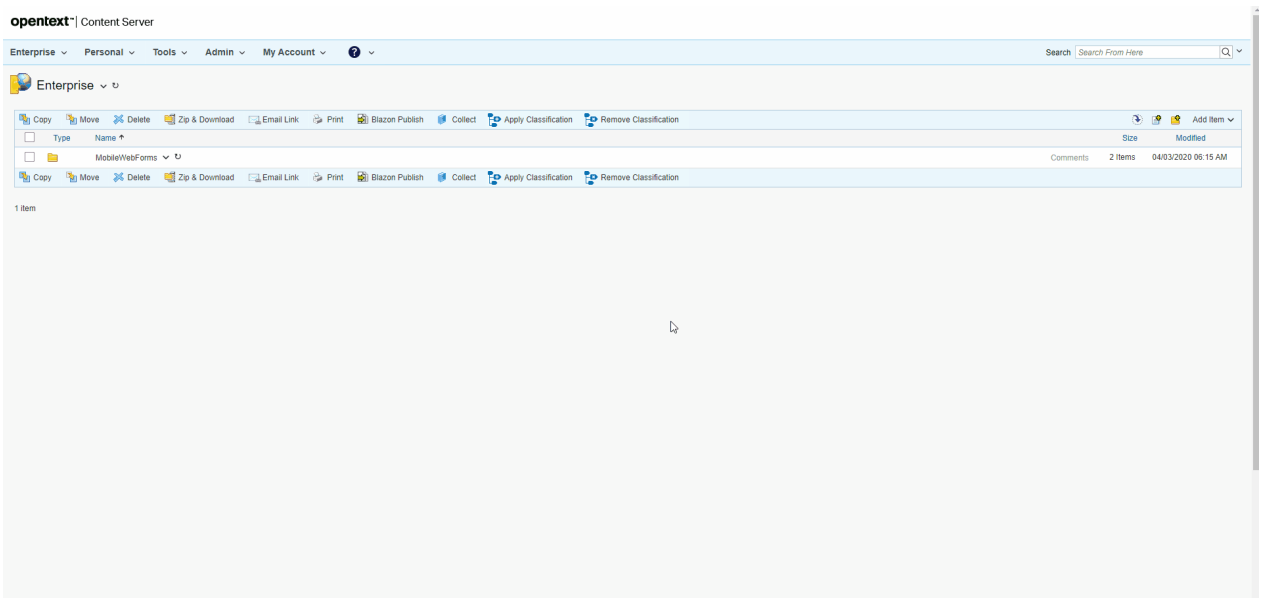
To verify the correctness of the process access the OpenText AppWorks Gateway and verify that in the "Installed" section the application to be distributed is present and enabled.



OpenText AppWorks Gateway

No information will be provided for installing and properly configuring the OpenText AppWorks Gateway. For installing ad configuring the OpenText AppWorks Gateway please refer to the official OpenText documentation.

The complete tour:



Copyright © 2013-2022 AnswerModules Sagl

Extension: Remote WebForms

What is it?¶

Remote Beautiful WebForm is an extension package for Script Console *(/working/scriptconsole/base/)* that allows you to deploy a Beautiful WebForms powered webform created on Content Server on the Script Console engine.

The main purpose of this extension is to simplify the process of gathering the contribution of users that do not have access to Content Server and synchronize these information back on Content Server. An other quite common scenario, is the off-line usage of Content Server webforms: the possibility of accessing, through a locally deployed Script Console instance, a copy of a Content Server webform, even when a connection with Content Server is not available.

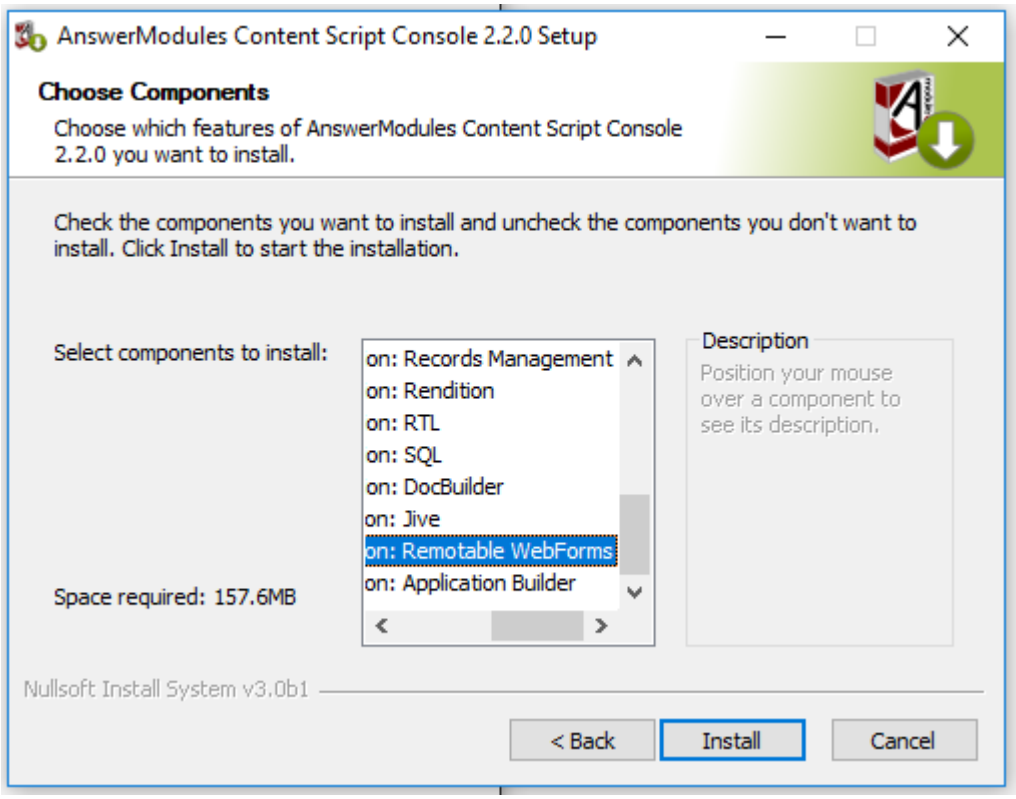
In both the cases the information submitted through the remote webform are stored locally within the Script Console to be later synchronize back towards Content Server.

Extension setup¶

Installing the remote-webform extension package on a Script Console instance, is a straight forward procedure which consists of just two steps:

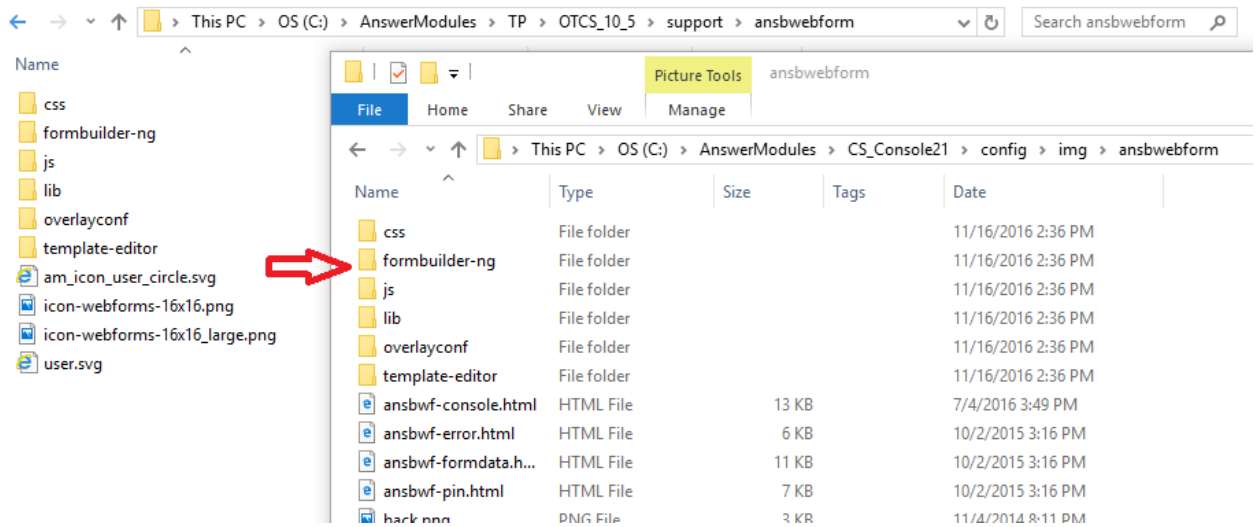
- Run the Script Console master installer and install the **Remotable WebForms** extension package

Copyright © 2013-2022 AnswerModules Sagl



- Copy all the static resources from the Beautiful WebForms Module Support in:

```
<Script Console Home>\config\img\ansbwebform
```



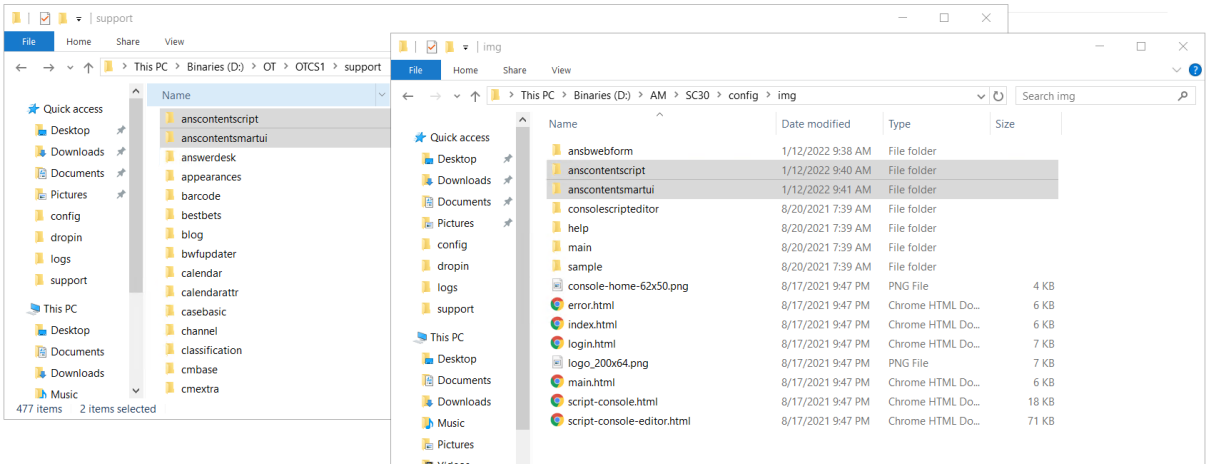
- Copy all the static resources from the Content Script Module Support (\support\anscontentscript) in:

```
<Script Console Home>\config\img\anscontentscript
```

- Copy all the static resources from the Module Suite for SmartUI Module Support (\support\anscontentsmartui) in:

```
<Script Console Home>\config\img\anscontentsmartui
```

Copyright © 2013-2022 AnswerModules Sagl



Create remote package¶

Beautiful WebForms deployable packages can be created either programmatically, using the Content Script `forms` service or manually, through the Beautiful Webforms Studio application.

Using `forms.createExPackage` API¶

Content Script `forms.createExPackage` API can be used to programmatically create a deployable Beautiful WebForms remote package. The API can be used from within a Beautiful WebForms View CLEH script, or from any other Content Script object.

In most of the cases, if used within a stand-alone script, this API is used in conjunction with `forms.getFormInfo` or `forms.listFormData` APIs.

Properly initialize the form object

It's important that you keep in mind that when the `form` object is loaded using the `form` service it is not initialized. You can either initialize it as part of your script or rely on it's OnLoad CLEH for its proper initialization. Here below an example of how properly initialize the form object:

Minimum initialization required

```
def formNode= docman.getNodeByPath("Path:to:your:form")
form = formNode.getFormInfo()
forms.addResourceDependencies( form, true, true)
```

Initialization through the OnLoad script (if any)

Copyright © 2013-2022 AnswerModules Sagl

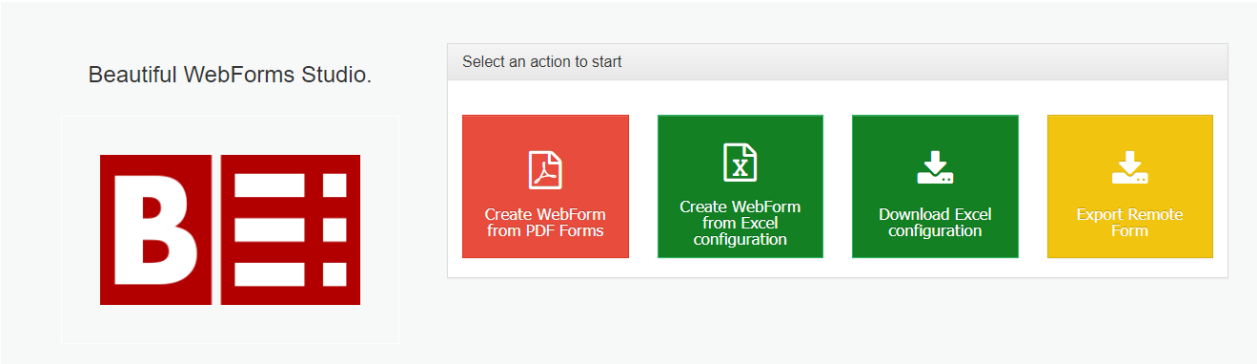
```
def formNode= docman.getNodeByPath("Path:to:your:form")

form = formNode.getFormInfo()
def bwfView = docman.getNode(form.amViewId)
def onLoad = bwfView.childrenFast.find{it.name == "OnLoad"}

if(onLoad){
    docman.runContentScript(onLoad, binding)
}
```

```
forms.createExPackage(
                      Form form,    // The form to export
                      String name, // An alpha-numeric identifier for the package to be created
                      String instructions, // The instruction to be displayed to help the user filli
                      String nextUrl, // Where to redirect the user upon submission
                      Date validUpTo, // A date after which the form should no longer be available (
                      List<String> viewsToExport, // The names of the views you want to export as pa
                                                  // if null all the views will be exported
                      String pin,                 // An optional pin that can be used to protect the
                      CSDocument[] arrayOfDocuments // An optional list of documents to be exported
                     )
```

Using Beautiful Webforms Studio¶

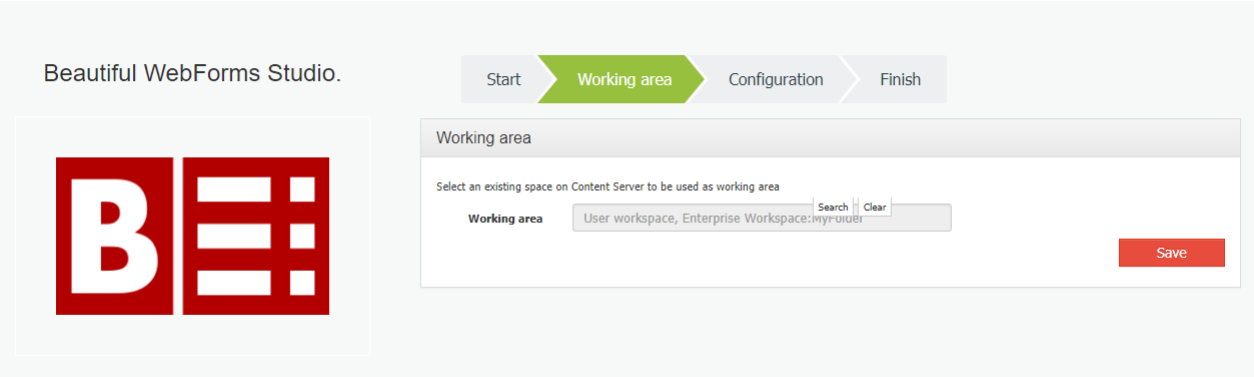
Beautiful Webforms Studio which can be found at the following location: `Content Script Volume:CSTools:Beautiful WebForm Studio`

Among the possibilities offered the studio application can help you leveraging the `forms.createExPackage` through a simplified visual wizard. The first step is to select **Export Remote Form** among the available actions.



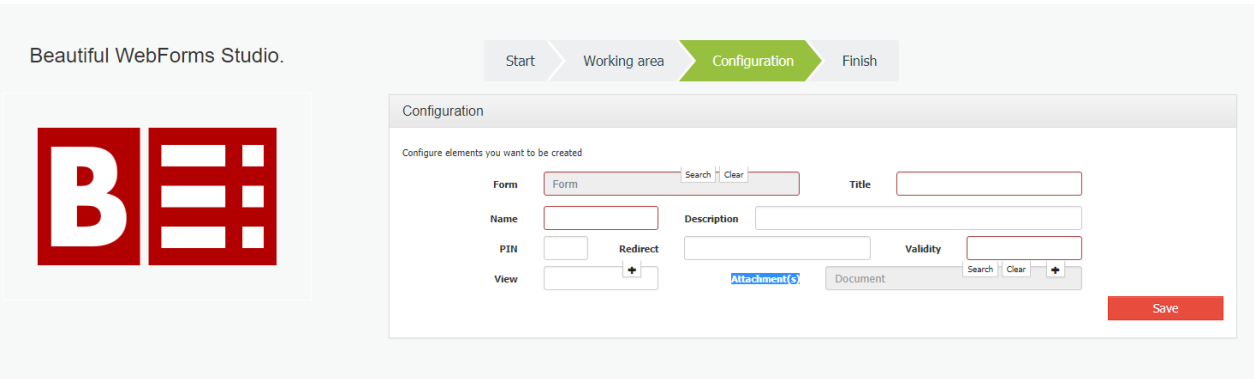
than you'll be asked for a space on Content Server to be used as the wizard workspace (where objects and content will be created):

Copyright © 2013-2022 AnswerModules Sagl



finally you will be asked about export configuration parameters

- **Form**: the form object to be exported
- **Title**: the form's title as it will be displayed on the script console default dashboard
- **Name**: the export package name (should be an alpha-numeric value)
- **Description**: the form's description as it will be displayed on the script console default dashboard
- **PIN**: an optional PIN to be used in order to protect un-authorized access to the form on the console
- **Redirect**: an URL where to redirect user's navigation upon submission
- **View**: an optional list of views names to be exported
- **Attachment(s)**: an optional list of documents to be exported



upon submission the export package file will be created in the selected workspace.

Copyright © 2013-2022 AnswerModules Sagl

How to deploy a Beautiful WebForms remote form package¶

The Beautiful WebForms remote form package is actually a .zip archive containing all objects necessary to the form (view files, scripts, templates, etc.).

You can manually extract its contents in a new folder inside:

```
<Script Console Home>\config\scripts\ext\forms\forms
```

for example:

```
<Script Console Home>\config\scripts\ext\forms\forms\myform
```

at this point, you should be able to access the form via the Script Console Dashboard, or via direct URL.

Synchronize form data back to Content Server¶

Form data submitted on Script Console can be synchronized back to Content Server in different ways which all are based on the same paradigm: the asynchronous exchange of information is based on data files.

Data files can be moved from the Script Console to Content Server no matter which transportation mechanism is used.

In the following paragraphs we will cover the most common scenarios.

Remote data pack files are produced on Script Console and sent over to Content Server¶

> **Script Console and Content Server can be isolated**
>
> In order to implement this scenario there is no need for the two systems to communicate each other.

In this scenario a local script is executed (or scheduled) on the Script Console in order to collect submitted data and prepare the exchange data files to be sent over Content Server.

The Remotable Beautiful WebForms extension for Script Console comes with several exemplar scripts of this kind that can be found at the following location:

```
<Script Console Home>\config\scripts\ext\forms
```

E.g **synchLocal.cs**

Copyright © 2013-2022 AnswerModules Sagl

```
import groovy.json.JsonSlurper
import groovy.io.FileType
import java.util.zip.ZipOutputStream
import java.util.zip.ZipEntry
formsAvailable = []
system = context.getAttribute("system")
formRepository =  system.extensionRepositories.find{
    it.repoHome.name == 'forms'
}
formRepositoryDir = new File(formRepository.getAbsolutePath(), "forms")
formRepositoryDirLocal = new File(formRepository.getAbsolutePath(), "inout")
if(formRepositoryDir && formRepositoryDir.isDirectory()){
    def deleteFile = []
    formRepositoryDirLocal.eachFileRecurse(FileType.FILES){
        if(it.name.endsWith(".amf")){
            File newForm = new File(formRepositoryDir, it.name-'.amf')
            if(!newForm.mkdir()){
                return
            }
            def zipFile = new java.util.zip.ZipFile(it)
            zipFile.entries().each {
                ins = zipFile.getInputStream(it)
                new File(newForm, it.name) << ins
                ins.close()
            }
            zipFile.close();
            deleteFile << it
        }
    }
    deleteFile.each {
        it.delete()
    }
}
if (params.upload == 'true' && params.selform){
    list =[]
    list.addAll( params.selform)
    toBeDeleted = []
    list.each{ form->
        formRepositoryDir = new File(formRepository.getAbsolutePath(), "data/$form")
        if(formRepositoryDir && formRepositoryDir.isDirectory()){
            formRepositoryDir.eachFileRecurse(FileType.FILES){
                if(it.name == "data.amf"){
                    File dataPack = it.getParentFile()
                    String zipFileName = "${dataPack.name}.rpf"
                    File zipFile = new File(new File(formRepository.getAbsolutePath(), "temp"), zipF
                    ZipOutputStream zipOS = new ZipOutputStream(new FileOutputStream(zipFile))
                    zapDir(dataPack.path, zipOS, dataPack.path)
                    zipOS.close()
                    zipFile.renameTo(new File(formRepositoryDirLocal, zipFile.name))
                    toBeDeleted << dataPack
                }
            }
        }
    }
    toBeDeleted.each{
         it.deleteDir()
    }
}
def static zapDir(String dir2zip, ZipOutputStream zos, String stripDir) {
        File zipDir = new File(dir2zip)
        def dirList = zipDir.list()
        byte[] readBuffer = new byte[2156]
        int bytesIn = 0
        dirList.each {
            File f = new File(zipDir, it)
            if(f.isDirectory())
                zapDir(f.path, zos, stripDir)
            else {
```

Copyright © 2013-2022 AnswerModules Sagl

```
                FileInputStream fis = new FileInputStream(f)
                ZipEntry anEntry = new ZipEntry(f.path.substring(stripDir.length()+1))
                zos.putNextEntry(anEntry)
                while((bytesIn = fis.read(readBuffer)) != -1) {
                    zos.write(readBuffer, 0, bytesIn);
                }
                fis.close();
            }
        }
}
redirect params.nextUrl
```

If you want to schedule this kind of scripts to be automatically executed by the Script Console you have to configure the job in the `cs-console-schedulerConfiguration.xml` file, which is a standard Quartz scheduler configuration file. You should find a sample job in there.

Here below a configuration example:

```
<?xml version="1.0" encoding="UTF-8"?>
<job-scheduling-data
    xmlns="http://www.quartz-scheduler.org/xml/JobSchedulingData"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.quartz-scheduler.org/xml/JobSchedulingData http://www.quartz-sche
    version="1.8">
    <pre-processing-commands>
        <delete-jobs-in-group>*</delete-jobs-in-group>  <!-- clear all jobs in scheduler -->
        <delete-triggers-in-group>*</delete-triggers-in-group> <!-- clear all triggers in scheduler
    </pre-processing-commands>
    <processing-directives>
        <!-- if there are any jobs/trigger in scheduler of same name (as in this
            file), overwrite them -->
        <overwrite-existing-data>true</overwrite-existing-data>
        <!-- if there are any jobs/trigger in scheduler of same name (as in this
            file), and over-write is false, ignore them rather then generating an error -->
        <ignore-duplicates>false</ignore-duplicates>
    </processing-directives>
    <schedule>
        <job>
            <name>PollJobSynchronization</name>
            <group>Synchronization</group>
            <job-class>com.answer.modules.cscript.console.scheduler.CommandLauncherJob</job-class>
            <job-data-map>
                <entry>
                    <key>script</key>
                    <value>ext/forms/synchLocal.cs</value>
                </entry>
                <entry>
                    <key>system</key>
                    <value>LOCAL</value>
                </entry>
            </job-data-map>
        </job>
        <trigger>
            <cron>
                <name>LaunchEvery1Minutes</name>
                <group>SynchronizationTriggerGroup</group>
                <job-name>PollJobSynchronization</job-name>
                <job-group>Synchronization</job-group>
                <start-time>2010-02-09T12:26:00.0</start-time>
                <end-time>2020-02-09T12:26:00.0</end-time>
                <misfire-instruction>MISFIRE_INSTRUCTION_SMART_POLICY</misfire-instruction>
                <cron-expression>0 * * ? * *</cron-expression>
                <time-zone>America/Los_Angeles</time-zone>
            </cron>
```

Copyright © 2013-2022 AnswerModules Sagl

```
        </trigger>
    </schedule>
</job-scheduling-data>
```

Later on Content Server the data files are unpacked using the `forms` service from within a Content Script that can be either manually executed or scheduled.

E.g.

```
// remPack is a data pack file, how this file was obtained is not relevant.
// It may have been fetched from an email folder, a ftp server, a shared folder a cloud service,
// or even uploaded on Content Server using web-services, etc...
def packList = forms.getExPackageContent( remPack) // returns a Map<String, CSResource>

if(packList."data.amf"){
    def res = packList.find{it.key == "data.amf"}.value
    def form  = forms.deserializeForm(res.content.getText("UTF-8"))

    // The form object can be used for various purposes
    // Submitting the data back to Content Server
    forms.submitForm(form)

    // Starting a workflow
    def damageInvestigation = docman.getNodeByPath("Fleet Management:Workflows:Damage Ingestigation I

    def inst = forms.startWorkFlow(damageInvestigation, form, "Form", "Damage Ingestigation - Veichl

    // Seding on a running workflow
    def task = workflow.getWorkFlowTask(form.getAmWorkID(), form.getAmSubWorkID(), form.getAmTaskID(

    forms.updateWorkFlowForm(
                            task, //The task
                            "Form Name", //The form name
                            form,  //The form object
                            true   // True if the task should be sent on
                           )

}
```

Form data are submitted directly from Script Console¶

Script Console and Content Server can't be isolated

In order to implement this scenario the two systems shall be able to communicate each other.

This scenario can be implemented executing or scheduling a script similar to the one reported here below on the Script Console:

```
import groovy.io.FileType

log.debug("Running Your Form Synch Job")

formsAvailable = []
system = context.get("system")
formRepository =  system.extensionRepositories.find{
    it.repoHome.name == 'forms'
}
```

Copyright © 2013-2022 AnswerModules Sagl

```
//Synch up
formRepositoryDirParent = new File(formRepository.getAbsolutePath(), "data")
def toBeDeleted = []
formRepositoryDirParent.eachFileRecurse(FileType.DIRECTORIES){ formRepositoryDir->

    if(("yourform").equalsIgnoreCase(formRepositoryDir.name)){
        if(formRepositoryDir && formRepositoryDir.isDirectory()){
            formRepositoryDir.eachFileRecurse(FileType.FILES){
                if(it.name == "data.amf"){
                    formObj = forms.deserializeForm(it.text)
                    File dataPack = it.getParentFile()
                    try{
                        forms.submitForm(formObj)
                        toBeDeleted << dataPack
                    }catch(e){
                        log.error("Unable to synch data back to OTCS",e)
                    }
                }
            }
        }
    }
}
toBeDeleted.each{
    it.deleteDir()
}
```

If you want to schedule this kind of scripts to be automatically executed by the Script Console you have to configure the job in the `cs-console-schedulerConfiguration.xml` file, which is a standard Quartz scheduler configuration file. You should find a sample job in there.

Here below a configuration example:

```
<?xml version="1.0" encoding="UTF-8"?>
<job-scheduling-data
    xmlns="http://www.quartz-scheduler.org/xml/JobSchedulingData"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.quartz-scheduler.org/xml/JobSchedulingData http://www.quartz-sche
    version="1.8">
    <pre-processing-commands>
        <delete-jobs-in-group>*</delete-jobs-in-group>  <!-- clear all jobs in scheduler -->
        <delete-triggers-in-group>*</delete-triggers-in-group> <!-- clear all triggers in scheduler
    </pre-processing-commands>
    <processing-directives>
        <!-- if there are any jobs/trigger in scheduler of same name (as in this
            file), overwrite them -->
        <overwrite-existing-data>true</overwrite-existing-data>
        <!-- if there are any jobs/trigger in scheduler of same name (as in this
            file), and over-write is false, ignore them rather then generating an error -->
        <ignore-duplicates>false</ignore-duplicates>
    </processing-directives>
    <schedule>
        <job>
            <name>PollJobSynchronization</name>
            <group>Synchronization</group>
            <job-class>com.answer.modules.cscript.console.scheduler.CommandLauncherJob</job-class>
            <job-data-map>
                <entry>
                    <key>script</key>
                    <value>ext/forms/submitMyFormLocal.cs</value>
                </entry>
                <entry>
                    <key>system</key>
                    <value>LOCAL</value>
```

Copyright © 2013-2022 AnswerModules Sagl

```
                </entry>
            </job-data-map>
        </job>
        <trigger>
            <cron>
                <name>LaunchEvery1Minutes</name>
                <group>SynchronizationTriggerGroup</group>
                <job-name>PollJobSynchronization</job-name>
                <job-group>Synchronization</job-group>
                <start-time>2010-02-09T12:26:00.0</start-time>
                <end-time>2020-02-09T12:26:00.0</end-time>
                <misfire-instruction>MISFIRE_INSTRUCTION_SMART_POLICY</misfire-instruction>
                <cron-expression>0 * * ? * *</cron-expression>
                <time-zone>America/Los_Angeles</time-zone>
            </cron>
        </trigger>
    </schedule>
</job-scheduling-data>
```

Copyright © 2013-2022 AnswerModules Sagl

Smart Pages

This guide introduces the basic functionalities related to the **Module Suite Smart Pages**.

Basic concepts¶

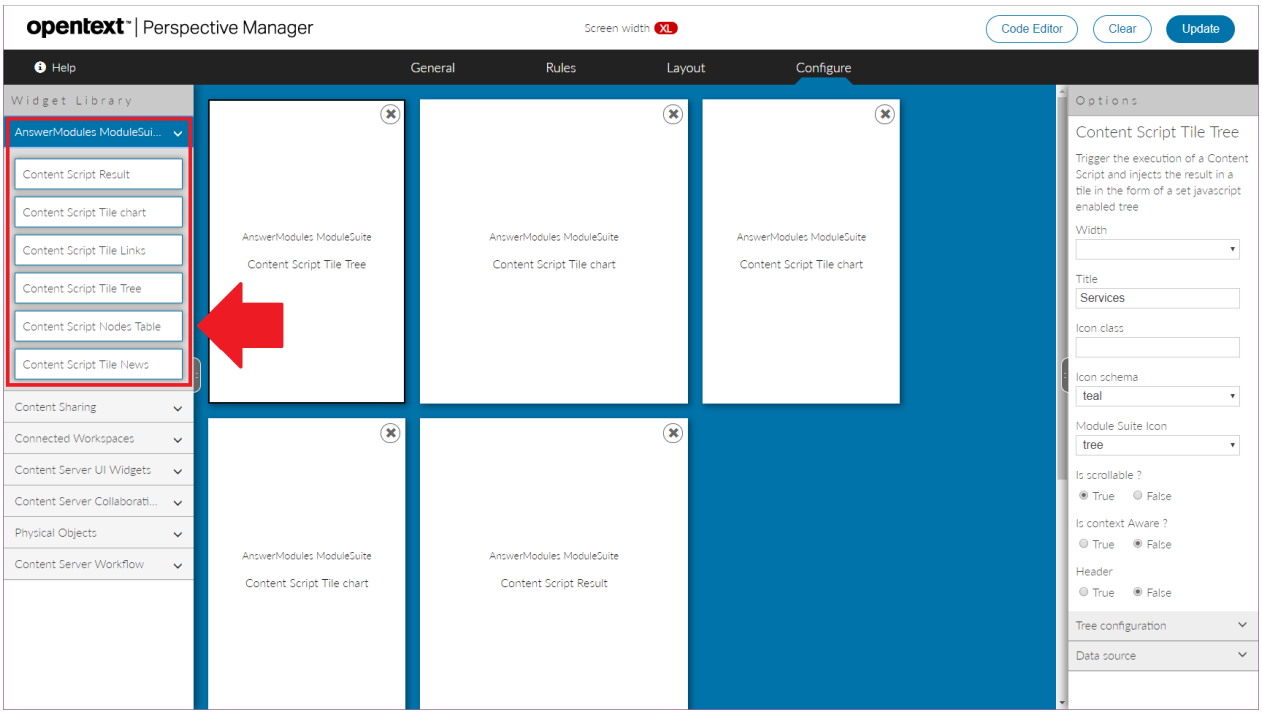
The Module Suite Smart Pages is an optional extension to Module Suite that introduces new features for those users that need an extra level of flexibility when creating customized SmartUI perspectives, and, more broadly, for those who prefer using the SmartUI in place of the Classic UI for their Content Server applications.

The extension includes the following components:

- A new set of SmartUI tiles, available within the Perspective Builder **Widget Library**
- A set of **Content Script snippets** that showcase how to create datasources for the SmartUI tiles
- **Smart Pages** module. **Smart Pages** module that aims to simplify the creation of good-looking functional user interfaces, both as a standalone solution and as part of the Smart View perspectives.
- Low-coding Smart View tailoring capabilities (allows you to custoize several aspect of the Smart View without having to relay on the Smart View SDK and without the need to deploy new artifact on Content Server servers)

Copyright © 2013-2022 AnswerModules Sagl

Module Suite Tiles in the Widget Library¶



The following tiles are available in *AnswerModules Module Suite* section:

- Content Script Result
- Content Script Tile Chart
- Content Script Tile Links
- Content Script Tile Tree
- Content Script Node Table
- Content Script Tile News
- Content Script Tile Tiles

Tile Configuration¶

Module Suite tiles share some common configuration options, while other options are specific to single tiles.

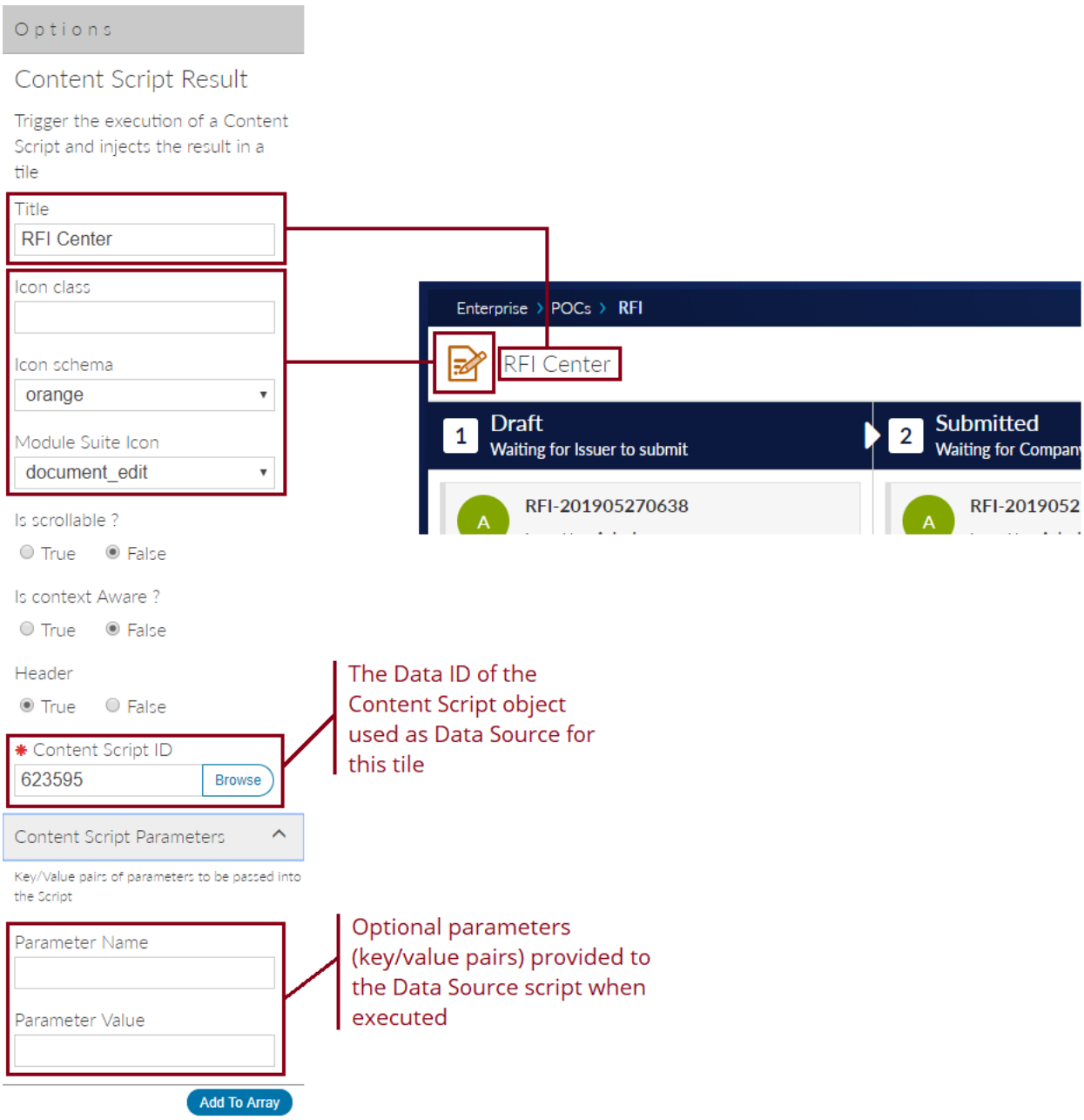
Common options include the configuration of the external frame (header, scrolling content, title, icon) and the configuration of the tile's Data Source. All Module Suite tiles require to specify a Content Script object that will be executed when the tile content is created. This script acts as a Data Source for the tile, and allows to make its content dynamic.

Through the configuration, it is also possible to pass additional parameters to the script. The parameter will be available to the developer within the **params** variable.

Copyright © 2013-2022 AnswerModules Sagl

When configuring the tile's icon, two different approaches are possible:

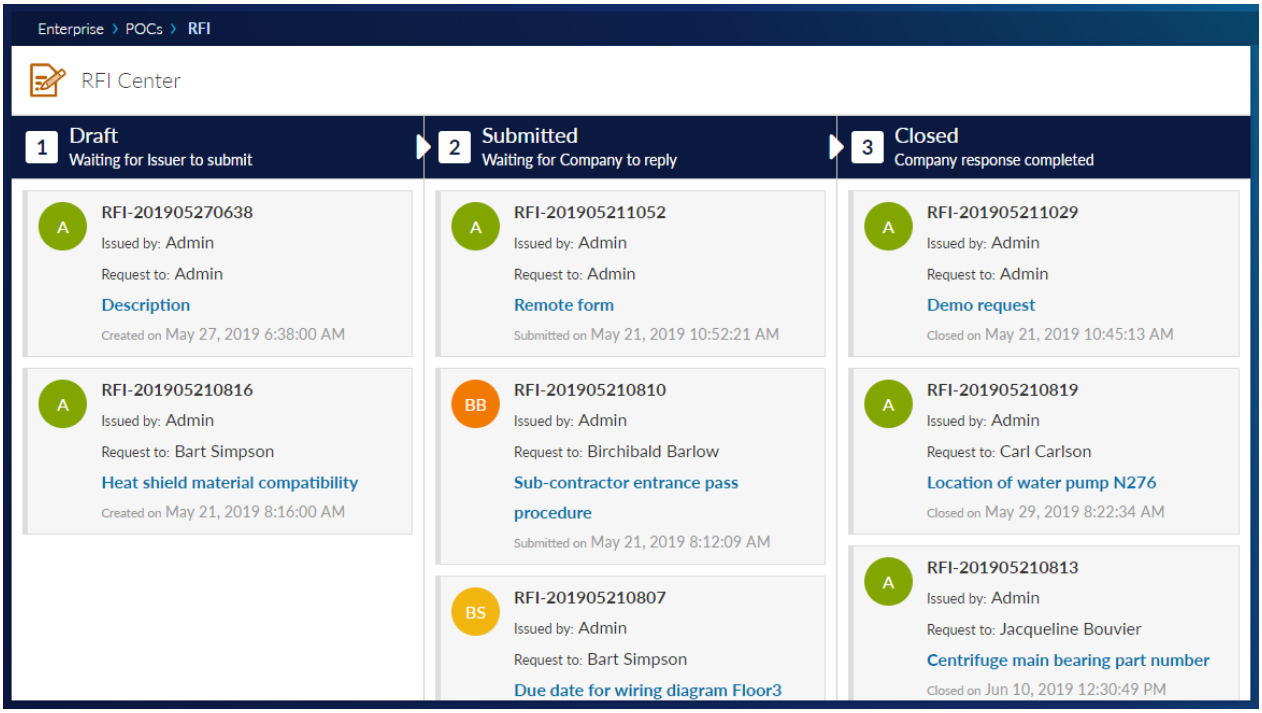
- specify a CSS style class to apply to the icon element. This should define the rules needed to apply the desired icon.
- specify the name (and color scheme) of the desired icon among the ones available in the Module Suite icon set. See the icon reference cheat sheet for a full list of options.



Tile: Content Script Result¶

The **Content Script Result** is a general-purpose tile that can be used to inject any output generated by a Content Script Data source into a SmartUI perspective.

Copyright © 2013-2022 AnswerModules Sagl



Tile: Content Script Tile Chart¶

The **Content Script Tile Chart** is a tile who's purpose is to create interactive charts within the SmartUI. The data shown in the charts will be provided by a Content Script data source.

Copyright © 2013-2022 AnswerModules Sagl

Module Suite Tile

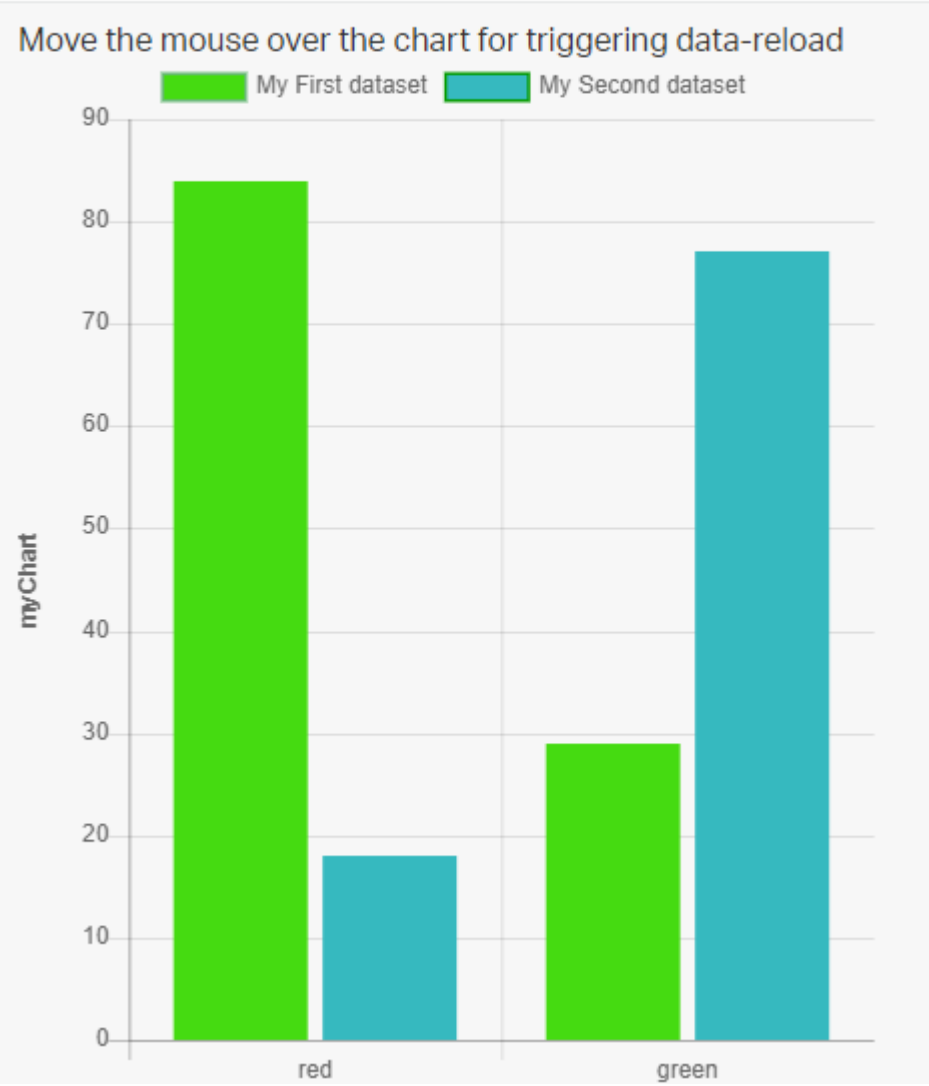


Chart tiles leverage two different javascript libraries:

- Chartist (supported for backward compatibility)
- Chart.js (suggested)

Depending on the selected chart type, the appropriate configuration has to be provided in JSON format. The following sample configuration produces the pie chart in the image above.

```
def rand = new Random()

if(params.widgetConfig){

    json(widgetConfig:[
        reloadCommands:["updateChart"],
        html:"""


<small>Move the mouse over the chart for triggering data-reload</small>
<script>
    csui.onReady2([
```

Copyright © 2013-2022 AnswerModules Sagl

```
        "csui/lib/jquery",
        "csui/lib/underscore",
        "csui/lib/radio"],
                  function(jQ, _, Radio){

                      //Get the page message bus
                      var amChannel  = Radio.channel('ampagenotify');

                      //Get the chart
                      var chart = amChannel.request("ampages:myChart");

                      var canvas = jQ("#myChart");
                      canvas.unbind("click");
                      canvas.on("click", function (evt) {
                        var activePoints = chart.getElementsAtEvent(evt);
                        var vals = _.map(_.pluck(_.filter(chart.legend.legendItems, function(it){ re
                        if(!_.isUndefined(activePoints[0])){
                           var chartData = activePoints[0]['_chart'].config.data;
                           var idx = activePoints[0]['_index'];

                           var label = chartData.labels[idx];
                           var value = chartData.datasets[0].data[idx];
                           amChannel.trigger("updateChart", [ {name:"where_type", value:label} ]);

                         } else {
                           amChannel.trigger("updateChart", [ {name:"where_type", value:vals} ]);
                         }
                        });


                      canvas.hover(function(){
                         var self = jQ(this);
                         //jQ(".myChartLoader").removeClass("binf-hidden");
                         amChannel.trigger("updateChart", [{name:"filter", value:"first"}]);
                      });
                  });
</script>"""
    ])
}else{

    json([

        type:"bar",
        data:
        [

            labels: ["red", "green"],
            datasets: [

                [
                    label: "My First dataset",
                    backgroundColor: "${AMBWFWidgetsLib.getBehaviour("ambwf","generateRandomHTMLColo
                    borderColor: "${AMBWFWidgetsLib.getBehaviour("ambwf","generateRandomHTMLColor",
                    data: [rand.nextInt(100), rand.nextInt(100)],

                ],
                [
                    label: "My Second dataset",
                    borderColor: "${AMBWFWidgetsLib.getBehaviour("ambwf","generateRandomHTMLColor",
                    backgroundColor: "${AMBWFWidgetsLib.getBehaviour("ambwf","generateRandomHTMLColo
                    data: [ rand.nextInt(100), rand.nextInt(100)],

                ]
            ]
        ],
        options: [
            maintainAspectRatio: false,
            title: [
                display: true,
```

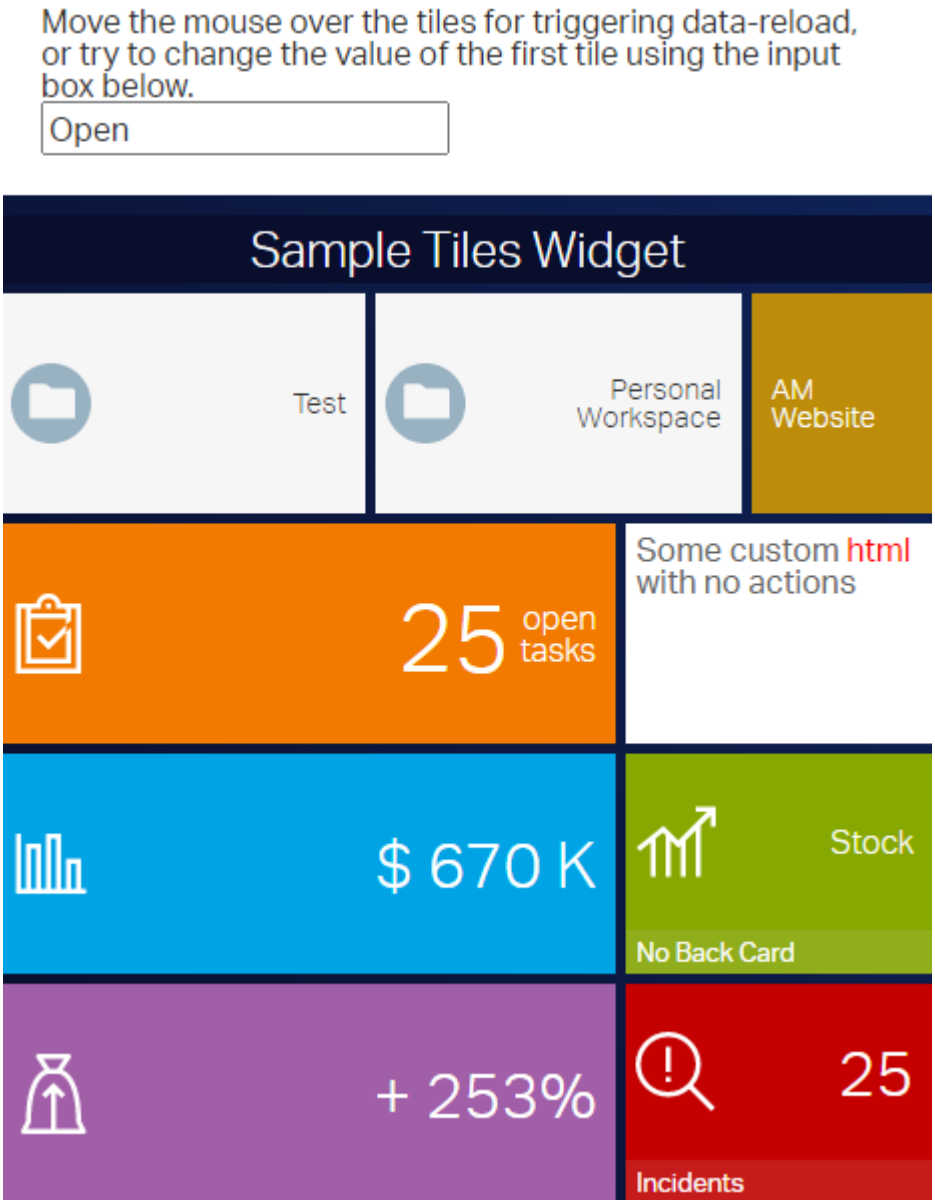
Copyright © 2013-2022 AnswerModules Sagl

```
                text: 'myChart',
                position: 'left'
            ],
            legend: [
                display: true,
                position: 'top'
            ],
            scales: [
                yAxes: [
                    [
                        ticks: [
                            beginAtZero:true
                        ]
                    ]
                ]
            ]
        ]
    ])
}
```

Tile: Content Script Tile Tiles¶

The **Content Script Tile Tiles** is a tile meant to create a customizable list of clickable links and HTML Tiles. The data controlling the links is provided by the backing Content Script data source.

Copyright © 2013-2022 AnswerModules Sagl



The following Content Script sample configuration produces the Links tile shown above.

```
def targetSpaceFilter = 2000


def subtypeFilter = "144".split(",")

if(params.widgetConfig){
    json([
      widgetConfig:[
          reloadCommands:[ "updateData" ],
          columnsWithSearch:[ "Owner", "Name" ]
      ]
     ])
    return
}

if(params.page?.contains("_") && params.page_list){
    if(params.page_list[0].contains("_") && !params.page_list?[1]?.contains("_")){
        params.page = params.page_list[1]
    }else if(!params.page_list[0].contains("_") && params.page_list?[1]?.contains("_")){
```

Copyright © 2013-2022 AnswerModules Sagl

```
        params.page = params.page_list[0]
    }
}

def paging = [actual_count:0,
              limit:((params.limit?:"30") as int),
              page:((params.page?:"1") as int),
              page_total:0,
              range_max:0,
              range_min:0,
              total_count:0,
              total_row_count:0,
              total_source_count:0]

def pageSize = paging.limit
def offset = (paging.limit * (paging.page - 1))
def firstRow = offset + 1
def lastRow = firstRow + paging.limit

nodes = []

def nameFilter = null
if( params.where_name ){
     nameFilter = "%${params.where_name}%"
}

def ownerFilter = null
if( params.where_owner ){
     ownerFilter = "%${params.where_owner}%"
}

def sortingOrderParam     = 'desc'
def sortingColumnParam     = 'name'

def sortingOrder     = 'DESC'
def sortingColumn     = 'DTree.Name'


if( params.sort && params.sort.contains('_') ){

    def sorting = params.sort.split('_')

    sortingOrderParam     = sorting[0]
    sortingColumnParam     = sorting[1]

    sortingOrder = ( sortingOrderParam == 'asc' ) ? 'ASC' : 'DESC'


    switch( sortingColumnParam?.trim() ){

        case 'name' :
            sortingColumn = 'DTree.Name'
            break

        case 'owner' :
            sortingColumn = 'KUAF.ID'
            break

        default :
            sortingColumn = 'DTree.Name'
            break
    }

}


try{
```

Copyright © 2013-2022 AnswerModules Sagl

```
    def queryParams = [targetSpaceFilter as String]
    def queryIndex = 1

    def permExpr = "(exists (select DataID from DTreeACL aclT where aclT.DataID=DTree.DataID and ${u

    sqlCode = """ select DTree.DataID "DID",
                         DTree.Name "NAME",
                         COUNT(*) OVER() as "overall_count"

                    from DTree
                    LEFT JOIN KUAF ON DTree.UserID = KUAF.ID

                    where DTree.ParentID = %1 """

    if(subtypeFilter.size() == 1){
        sqlCode += " and DTree.SubType = %${++queryIndex} "
        queryParams << (subtypeFilter[0] as long)
    } else if( subtypeFilter.size() > 1 ) {
        sqlCode += " and DTree.SubType IN (${subtypeFilter.join(',')}) "
    }

    if(nameFilter){
        sqlCode += " and DTree.Name LIKE %${++queryIndex} "
        queryParams << (nameFilter as String)
    }

    if(ownerFilter){
        sqlCode += " and (KUAF.Name LIKE %${++queryIndex} OR KUAF.LastName LIKE %${queryIndex} ) "
        queryParams << (ownerFilter as String)
    }

    if(!users.current.canAdministerSystem){
        sqlCode += " and ${permExpr} "
    }

    sqlCode += """
                   ORDER BY ${sortingColumn} ${sortingOrder}
                    OFFSET ${offset} ROWS
                    FETCH NEXT ${pageSize} ROWS ONLY

                """



    def queryResults

    if(queryParams){
        queryResults =  sql.runSQLFast(sqlCode, true, true, 100, *queryParams).rows
    } else {
        queryResults =  sql.runSQLFast(sqlCode, true, true, 100 ).rows
    }

    def totalCount = (queryResults) ? queryResults[0].overall_count : 0

    nodes = queryResults?.collect{it.DID as Long}



    paging << [
              actual_count:totalCount,
              page_total:((totalCount%paging.limit)+1),
              range_min:paging.page*paging.limit-paging.limit+1,
              range_max:(paging.limit*(paging.page+1)-totalCount)>0?(paging.limit*(paging.page+1)-to
              total_count:totalCount,
              total_row_count:totalCount,
              total_source_count:totalCount]

}catch(e){
```

Copyright © 2013-2022 AnswerModules Sagl

```
    log.error("Error loading nodes table data",e)
    printError(e)
}



def drawStatusBar = { node ->

    def statusList = ['Draft', 'Under Revision', 'Approved', 'Published']
    def numSteps = statusList.size()
    def currStep = new Random().nextInt(statusList.size())
    def currStepName = statusList[currStep]


    def stepStyle = "height:100%; width:calc(100% / ${numSteps}); float:left; background-color:#F0AD

    def stepsHtml = ""

    (currStep + 1).times{
        stepsHtml += """<span style="${stepStyle}"></span>"""
    }


    return """
    <div style="text-align:center; font-size:.75em">${currStepName}</div>
    <div style="margin:3px 0; padding:0; height:5px; background-color:#eee;">${stepsHtml}</div>"""
}



def slurper = new JsonSlurper()

def processNode = { node, myNode ->

    /* Add your custom node post-processing here */

    //def myNode = asCSNode(node?.data.properties.id as long)

    node.data.amcsproxy =  [
        columns: [:],
        commands:[]
    ]

    //Add custom column: node.data.amcsproxy.colums.sample_column = "My custom Value"

        def owner = myNode.createdBy
        def ownerBox = "<span><img src='/otcs/cs.exe/pulse/photos/userphoto/${owner.ID}/2000' style=
        node.data.amcsproxy.columns.owner = ownerBox

    node.data.amcsproxy.columns.comment = myNode.comment
     node.data.amcsproxy.columns.statusBar = drawStatusBar( myNode )

    return node
}

results = []

def fields = JsonOutput.toJson( [
    'actions': [ 'fields': [] ],
    'properties': [ 'fields': [] ],
    'versions': [ 'fields': [] ],
    'amcsproxy': [ 'fields': [] ],
])

//Identifies actions to be displayed for every node
//Node actions are return together with data request thay may lead to additinal response time
// [] - docman.getNodesRestV2JSon will not process actions.
//      Actions will be processed on a separate call based on the list provided (see returned json o
// null - default list of actions will be returned
```

Copyright © 2013-2022 AnswerModules Sagl

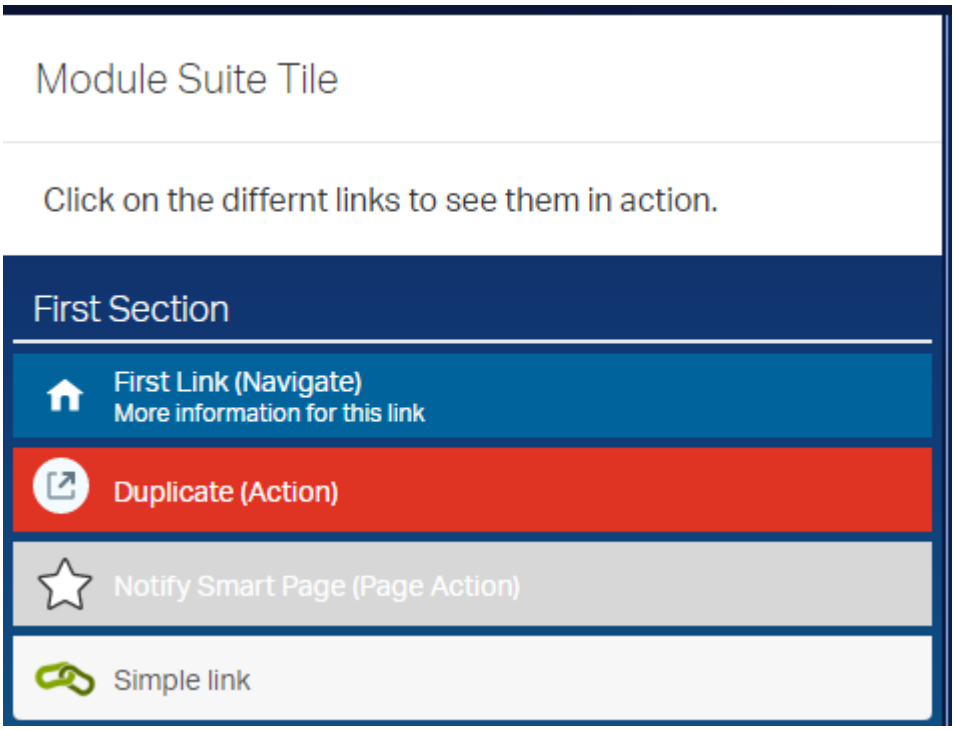
```
// ['open','properties','copy','move','edit'] - sample list of actions
// To ideal actions processing requires you to assign an empty list (see below) to the nodesActions
// using the 'actions' list property of the json object returned by this script (see last line)
def nodesActions = []

if( nodes.size() > 1 ){
    log.error("Nodes ${nodes}")
    temp = slurper.parseText( docman.getNodesRestV2JSon(nodes, fields, '{"properties":{"fields":["pa
    theNodes = docman.getNodesFastWith(nodes, [], params, false, false, false)
    nodes.each{ node ->

        def jsonNode = temp.find{ it.data.properties.id == node }
        results << processNode(jsonNode, theNodes.find{it.ID == node}  )
    }


} else if (nodes.size() == 1 ){

    it = slurper.parseText(docman.getNodesRestV2JSon(nodes, fields, '{"properties":{"fields":["paren
    processNode(it, docman.getNodeFast(nodes[0]))

    results = [it]
}




def columns = [

    type: [
            key:"type",
            name:"Type",
            type:2,
            type_name:"Integer",
            sort:false
          ]

    ,name: [
            key:"name",
            name:"Name",
            type:-1,
            type_name:"String",
            sort:true,
            align:"left"
          ]

    ,owner: [
            key:"owner",
            name:"Owner",
            type:43200,
            type_name:"String",
            sort:true,
            align:"left"
          ]

    ,statusBar: [
            key:"statusBar",
            name:"Doc. Status",
            type:43200,
            type_name:"String",
            sort:false,
            align:"left"
          ]

    ,comment: [
            key:"comment",
            name:"Comment",
            type:-1,
            type_name:"String",
```

Copyright © 2013-2022 AnswerModules Sagl

```
            sort:false,
            align:"left"
          ]
]


// actions - list of commands defined for all the nodes listed in the page
// action=[] - will return all possible actions for a node
json(
    [
      paging:paging,
      columnsWithSearch:[ "name" , "owner" ],
      results:results,
      columns:columns,
      tableColumns:columns,
      widgetConfig:[
          reloadCommands:[ "updateData" ]
      ],
      actions: ['open','properties','copy']
   ]
)
```

Tile: Content Script Tile Links¶

The **Content Script Tile Links** is a tile meant to create a customizable list of clickable links. The data controlling the links is provided by the backing Content Script data source.



The following Content Script sample configuration produces the Links tile shown above.

```
if(params.widgetConfig){

    json(widgetConfig:[
        reloadCommands:["updateLinks"],
        html:"""

<style>
```

Copyright © 2013-2022 AnswerModules Sagl

```
div.ans-tile-content-linkstiles{
    background: linear-gradient(180deg, #122c69 0%, #078db3 100% );
    color:#fff;
    height:100%;
}

div.ans-tile-content-linkstiles > div.binf-list-group > a:nth-child(2),
div.ans-tile-content-linkstiles > div.binf-list-group > a:nth-child(6),
div.ans-tile-content-linkstiles > div.binf-list-group > a:nth-child(10){
    background:#00639b;
    color:#fff;
    border-radius:0px;
}
div.ans-tile-content-linkstiles > div.binf-list-group > a:nth-child(3),
div.ans-tile-content-linkstiles > div.binf-list-group > a:nth-child(7),
div.ans-tile-content-linkstiles > div.binf-list-group > a:nth-child(11){
    background:#df3324;
    color:#fff;
    border-radius:0px;
}
div.ans-tile-content-linkstiles > div.binf-list-group > a:nth-child(4),
div.ans-tile-content-linkstiles > div.binf-list-group > a:nth-child(8),
div.ans-tile-content-linkstiles > div.binf-list-group > a:nth-child(12){
    background:#008485;
    color:#fff;
    border-radius:0px;
}

</style>
<div style="padding:20px; background-color:white;margin-bottom:10px;color:#333" >
Click on the differnt links to see them in action.
</div>
<script>
    csui.onReady2([    'csui/lib/underscore',
        'csui/lib/backbone',
        'csui/lib/jquery',
        'csui/lib/radio'],
        function(_,Backbone, jQ, Radio){
            var amChannel = Radio.channel("ampagenotify");
            amChannel.on("smartPage_action", function(action,param){
                console.log("GOT Page Action request. Action: "+action+ " parameter: "+param);
            });

        });
</script>
"""
    ])
}else{

    retVal =
        [
            data:[
                links:[
                    [
                        issection:true,
                        name:"First Section",
                    ],
                    [
                        issection:false,
                        icon:"csui-icon-home",
                        name:"First Link (Navigate)",
                        desc:"More information for this link",

                        url:"#", //If action != null url must be set equal to #
                        action:"navigate", //Will trigger a browse action of the current view
                        params:"2000", //The DataID of the node you wanto to navigate to

                    ],
                    [
```

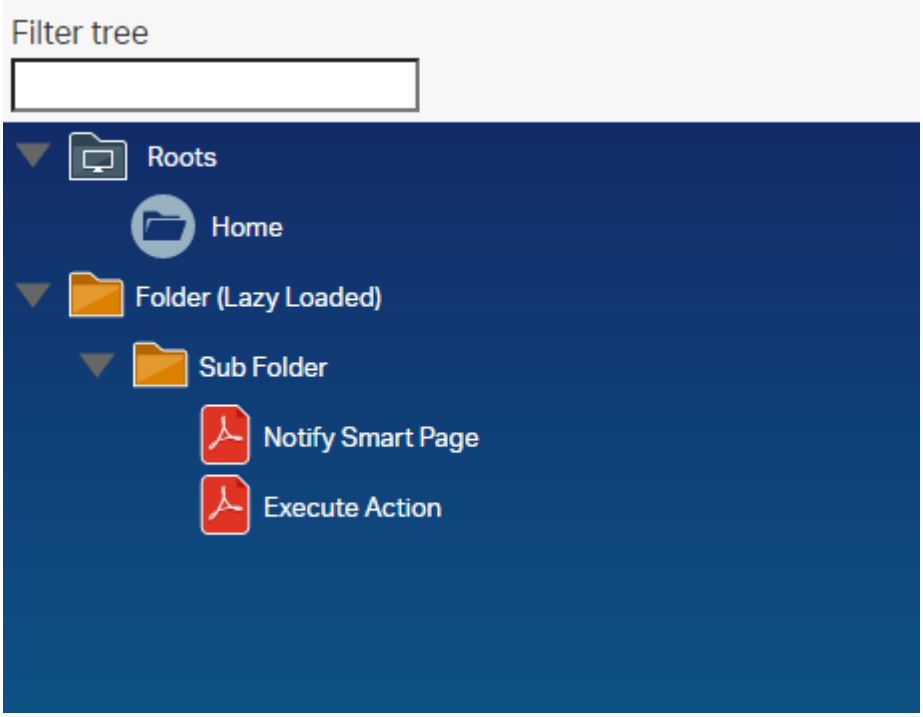
Copyright © 2013-2022 AnswerModules Sagl

```
                        issection:false,
                        icon:"icon-tileExpand icon-perspective-open",
                        name:"Duplicate (Action)",

                        url:"#", //If action != null url must be set equal to #
                        action:"notify", //Will trigger the execution of the command below
                        command:"updateLinks", //The action to execute
                        params:"duplicate", //The action's parameter, this value will be passed to t

                    ],
                    [
                        issection:false,
                        icon:"icon-socialFavOpen",
                        name:"Notify Smart Page (Page Action)",

                        url:"#", //If action != null url must be set equal to #
                        command:"smartPage", //The SmartPage(s) to notify
                        action:"updatePage",   //The action to execute
                        params:"2000" //The action's parameter

                    ],
                    [
                        issection:false,
                        am_icon:"am_icon_link",
                        am_icon_schema:"am_icon_green",
                        name:"Simple link",

                        url:"http://www.answermodules.com",
                        newtab:true

                    ]

                ]
            ]
        ]

    if(params.tile == "duplicate"){
        retVal.data.links += retVal.data.links[-5].clone()
        retVal.data.links += retVal.data.links[-5]
        retVal.data.links += retVal.data.links[-5]
        retVal.data.links += retVal.data.links[-5]

        retVal.data.links[-4].name = "Second Section"
    }else if(params.tile == "triple"){
        retVal.data.links += retVal.data.links[-5].clone()
        retVal.data.links += retVal.data.links[-5]
        retVal.data.links += retVal.data.links[-5]
        retVal.data.links += retVal.data.links[-5]

        retVal.data.links[-4].name = "Second Section"

        retVal.data.links += retVal.data.links[-4].clone()
        retVal.data.links += retVal.data.links[-4]
        retVal.data.links += retVal.data.links[-4]
        retVal.data.links += retVal.data.links[-4]

        retVal.data.links[-4].name = "Third Section"
    }

    json(
        retVal
    )


}
```

Copyright © 2013-2022 AnswerModules Sagl

Tile: Content Script Tile Tree¶

The **Content Script Tile Tree** creates an interactive tree structure with nodes that can be expanded and collapsed. The tree structure uses a Content Script data source for the initial data and for subsequent ajax data load calls.



The following sample configuration generates the tree in the image above.

```
if(params.widgetConfig){
    json( [ id           : 2,
            widgetConfig : [
                tileLayoutClasses  : "",
                tileContentClasses : "",
                reloadCommands     : ["updateTree"],
                root               : 2000,
                plugins            : [ "wholerow" ],
                theme              : [ 'name': 'proton',
                                       'responsive': true ],
                html               : """
<style>
div.ans-tile-tree{
  background: linear-gradient(180deg, #122c69 0%, #078db3 100% );
  color:#fff;
  height:calc(100vh - 222px);
  font-size:13px !important;
}
.binf-widgets .jstree-proton .jstree-icon.csui-icon-node-task {
  background-image:url('${img}csui/themes/carbonfiber/image/icons/mime_task.svg')
}
.binf-widgets .jstree-proton .jstree-icon.mime_pdf{
  background-image:url('${img}csui/themes/carbonfiber/image/icons/mime_pdf.svg')
}
.jstree-anchor small{
  font-size:.9em;
  font-style:italic;
}
</style>
```

Copyright © 2013-2022 AnswerModules Sagl

```
<div class="am-form-text-input" style="margin-top: 1px;padding: 5px 0px;">
    <label class=" control-label col-form-label am-form-text-input-label   am-form-label-top" style='
    <div class="am-form-input-wrap" style="padding: 0 5px;">
        <input id="filter" type="text" placeholder="" class="form-control" style="border-radius: 0px,
    </div>
</div>

<script>
    csui.onReady2([    'csui/lib/underscore',
        'csui/lib/backbone',
        'csui/lib/jquery',
        'csui/lib/radio'],
        function(_,Backbone, jQ, Radio){
            var amChannel = Radio.channel("ampagenotify");
            amChannel.on("printConsole", function(params){
                console.log("GOT request "+JSON.stringify(params));
            });
            amChannel.on("smartPage_action", function(action,param){
                console.log("GOT Page Action request. Action: "+action+ " parameter: "+param);
            });
            jQ("#filter").on("blur", function(){
                amChannel.trigger("updateTree",{'term':jQ(this).val()})
            })

        });
</script>"""
           ]
          ] )
    return
}

data =

    [
        [

            icon     : "csui-icon cs_vfolder", //mime_folder, cs_folder_root, cs_vfolder, cs_folder_
            id       : 1,
            text     : "Roots",
            children :  [
                [
                    action   : "navigate", //Trigger a Smart View navigation
                    icon     : "csui-icon cs_folder_root", //cs_folder_root, cs_vfolder, cs_folder_o
                    id       : 2000, //The node will be used as the action's parameter
                    text     : "Home",
                    children : false
                ]

            ],
            state       : [
                opened    : true
            ]
        ],
        [

            action   : "printConsole", //Trigger a Tile action
            params   : "3", //This value will be passed to the script in a parameter named 'tile'
            icon     : "csui-icon mime_folder",
            id       : 3,
            text     : "Folder (Lazy Loaded)",
            children : true,
            state       : [
                opened    : false
            ]
        ]
    ]


if(params.uiParentID == "3"){
```

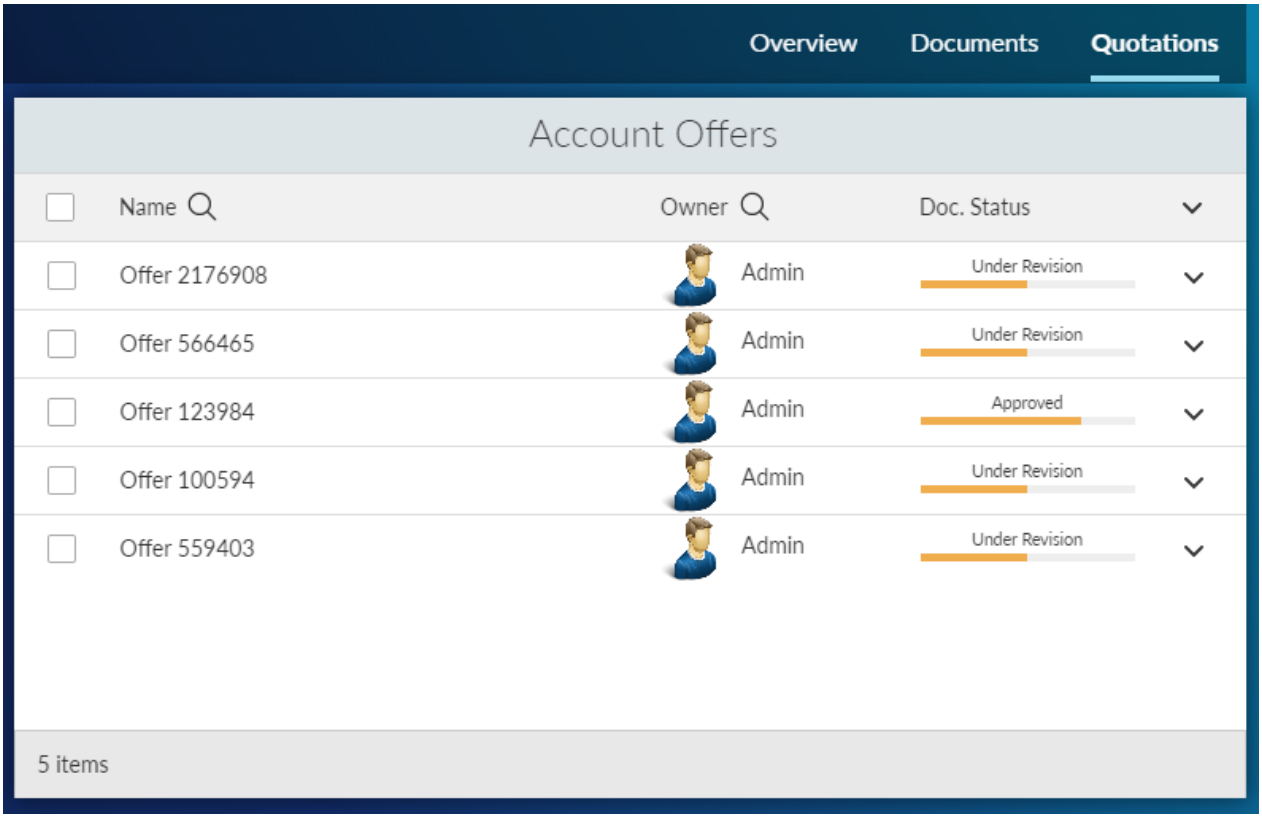
Copyright © 2013-2022 AnswerModules Sagl

```
    data[1].children = [
        [

            icon     : "csui-icon mime_folder",
            id       : 4,
            text     : "Sub Folder",
            children : [
                [
                    notify   : "smartPage",    //Triggers a Smart Page action noifying the provided
                    action   : "customAction", //The action to execute
                    params   : "2000",         //The action's parameter
                    icon     : "csui-icon mime_pdf",
                    id       : 5,
                    text     : "Notify Smart Page",
                    children : false
                ],
                [
                    action   : "printConsole",
                    params   : "2000",
                    icon     : "csui-icon mime_pdf",
                    id       : 6,
                    text     : "Execute Action",
                    children : false
                ]

            ]
        ]

    ]
}
if(params.term){
    data = data.findAll{it.text.startsWith(params.term)}
}
json(data)
```

Tile: Content Script Node Table¶

The **Content Script Node Table** is an enhancement of the standard Node Table tile. The tile uses a Content Script as data source, allowing to set up any custom business logic to generate the list of nodes to be shown.

Copyright © 2013-2022 AnswerModules Sagl




```
def targetSpaceFilter = 2000


def subtypeFilter = "144".split(",")

if(params.widgetConfig){
    json([
      widgetConfig:[
          reloadCommands:[ "updateData" ],
          columnsWithSearch:[ "Owner", "Name" ]
      ]
     ])
    return
}

if(params.page?.contains("_") && params.page_list){
    if(params.page_list[0].contains("_") && !params.page_list?[1]?.contains("_")){
        params.page = params.page_list[1]
    }else if(!params.page_list[0].contains("_") && params.page_list?[1]?.contains("_")){
        params.page = params.page_list[0]
    }
}

def paging = [actual_count:0,
              limit:((params.limit?:"30") as int),
              page:((params.page?:"1") as int),
              page_total:0,
              range_max:0,
              range_min:0,
              total_count:0,
              total_row_count:0,
              total_source_count:0]

def pageSize = paging.limit
def offset = (paging.limit * (paging.page - 1))
def firstRow = offset + 1
def lastRow = firstRow + paging.limit
```

Copyright © 2013-2022 AnswerModules Sagl

```
nodes = []

def nameFilter = null
if( params.where_name ){
     nameFilter = "%${params.where_name}%"
}

def ownerFilter = null
if( params.where_owner ){
     ownerFilter = "%${params.where_owner}%"
}

def sortingOrderParam     = 'desc'
def sortingColumnParam     = 'name'

def sortingOrder     = 'DESC'
def sortingColumn     = 'DTree.Name'


if( params.sort && params.sort.contains('_') ){

    def sorting = params.sort.split('_')

    sortingOrderParam     = sorting[0]
    sortingColumnParam     = sorting[1]

    sortingOrder = ( sortingOrderParam == 'asc' ) ? 'ASC' : 'DESC'


    switch( sortingColumnParam?.trim() ){

        case 'name' :
            sortingColumn = 'DTree.Name'
            break

        case 'owner' :
            sortingColumn = 'KUAF.ID'
            break

        default :
            sortingColumn = 'DTree.Name'
            break
    }

}


try{


    def queryParams = [targetSpaceFilter as String]
    def queryIndex = 1

    def permExpr = "(exists (select DataID from DTreeACL aclT where aclT.DataID=DTree.DataID and ${u


    sqlCode = """ select DTree.DataID "DID",
                         DTree.Name "NAME",
                         COUNT(*) OVER() as "overall_count"

                    from DTree
                    LEFT JOIN KUAF ON DTree.UserID = KUAF.ID

                    where DTree.ParentID = %1 """

    if(subtypeFilter.size() == 1){
        sqlCode += " and DTree.SubType = %${++queryIndex} "
        queryParams << (subtypeFilter[0] as long)
```

Copyright © 2013-2022 AnswerModules Sagl

```
    } else if( subtypeFilter.size() > 1 ) {
        sqlCode += " and DTree.SubType IN (${subtypeFilter.join(',')}) "
    }

    if(nameFilter){
        sqlCode += " and DTree.Name LIKE %${++queryIndex} "
        queryParams << (nameFilter as String)
    }

    if(ownerFilter){
        sqlCode += " and (KUAF.Name LIKE %${++queryIndex} OR KUAF.LastName LIKE %${queryIndex} ) "
        queryParams << (ownerFilter as String)
    }

    if(!users.current.canAdministerSystem){
        sqlCode += " and ${permExpr} "
    }

    sqlCode += """
                   ORDER BY ${sortingColumn} ${sortingOrder}
                    OFFSET ${offset} ROWS
                    FETCH NEXT ${pageSize} ROWS ONLY

                """



    def queryResults

    if(queryParams){
        queryResults =  sql.runSQLFast(sqlCode, true, true, 100, *queryParams).rows
    } else {
        queryResults =  sql.runSQLFast(sqlCode, true, true, 100 ).rows
    }

    def totalCount = (queryResults) ? queryResults[0].overall_count : 0

    nodes = queryResults?.collect{it.DID as Long}



    paging << [
              actual_count:totalCount,
              page_total:((totalCount%paging.limit)+1),
              range_min:paging.page*paging.limit-paging.limit+1,
              range_max:(paging.limit*(paging.page+1)-totalCount)>0?(paging.limit*(paging.page+1)-to
              total_count:totalCount,
              total_row_count:totalCount,
              total_source_count:totalCount]

}catch(e){
    log.error("Error loading nodes table data",e)
    printError(e)
}



def drawStatusBar = { node ->

    def statusList = ['Draft', 'Under Revision', 'Approved', 'Published']
    def numSteps = statusList.size()
    def currStep = new Random().nextInt(statusList.size())
    def currStepName = statusList[currStep]


    def stepStyle = "height:100%; width:calc(100% / ${numSteps}); float:left; background-color:#F0AD

    def stepsHtml = ""
```

Copyright © 2013-2022 AnswerModules Sagl

```
    (currStep + 1).times{
        stepsHtml += """<span style="${stepStyle}"></span>"""
    }


    return """
    <div style="text-align:center; font-size:.75em">${currStepName}</div>
    <div style="margin:3px 0; padding:0; height:5px; background-color:#eee;">${stepsHtml}</div>"""
}



def slurper = new JsonSlurper()

def processNode = { node, myNode ->

    /* Add your custom node post-processing here */

    //def myNode = asCSNode(node?.data.properties.id as long)

    node.data.amcsproxy =  [
        columns: [:],
        commands:[]
    ]

    //Add custom column: node.data.amcsproxy.colums.sample_column = "My custom Value"

        def owner = myNode.createdBy
        def ownerBox = "<span><img src='/otcs/cs.exe/pulse/photos/userphoto/${owner.ID}/2000' style=
        node.data.amcsproxy.columns.owner = ownerBox

    node.data.amcsproxy.columns.comment = myNode.comment
     node.data.amcsproxy.columns.statusBar = drawStatusBar( myNode )

    return node
}

results = []

def fields = JsonOutput.toJson( [
    'actions': [ 'fields': [] ],
    'properties': [ 'fields': [] ],
    'versions': [ 'fields': [] ],
    'amcsproxy': [ 'fields': [] ],
])

//Identifies actions to be displayed for every node
//Node actions are return together with data request thay may lead to additinal response time
// [] - docman.getNodesRestV2JSon will not process actions.
//      Actions will be processed on a separate call based on the list provided (see returned json ob
// null - default list of actions will be returned
// ['open','properties','copy','move','edit'] - sample list of actions
// To ideal actions processing requires you to assign an empty list (see below) to the nodesActions
// using the 'actions' list property of the json object returned by this script (see last line)
def nodesActions = []

if( nodes.size() > 1 ){
    log.error("Nodes ${nodes}")
    temp = slurper.parseText( docman.getNodesRestV2JSon(nodes, fields, '{"properties":{"fields":["pa
    theNodes = docman.getNodesFastWith(nodes, [], params, false, false, false)
    nodes.each{ node ->

        def jsonNode = temp.find{ it.data.properties.id == node }
        results << processNode(jsonNode, theNodes.find{it.ID == node}  )
    }


} else if (nodes.size() == 1 ){
```

Copyright © 2013-2022 AnswerModules Sagl

```
    it = slurper.parseText(docman.getNodesRestV2JSon(nodes, fields, '{"properties":{"fields":["paren
    processNode(it, docman.getNodeFast(nodes[0]))

    results = [it]
}



def columns = [

    type: [
            key:"type",
            name:"Type",
            type:2,
            type_name:"Integer",
            sort:false
          ]

    ,name: [
            key:"name",
            name:"Name",
            type:-1,
            type_name:"String",
            sort:true,
            align:"left"
          ]

    ,owner: [
            key:"owner",
            name:"Owner",
            type:43200,
            type_name:"String",
            sort:true,
            align:"left"
          ]

    ,statusBar: [
            key:"statusBar",
            name:"Doc. Status",
            type:43200,
            type_name:"String",
            sort:false,
            align:"left"
          ]

    ,comment: [
            key:"comment",
            name:"Comment",
            type:-1,
            type_name:"String",
            sort:false,
            align:"left"
          ]
]


// actions - list of commands defined for all the nodes listed in the page
// action=[] - will return all possible actions for a node
json(
    [
      paging:paging,
      columnsWithSearch:[ "name" , "owner" ],
      results:results,
      columns:columns,
      tableColumns:columns,
      widgetConfig:[
          reloadCommands:[ "updateData" ]
      ],
```

Copyright © 2013-2022 AnswerModules Sagl

```
        actions: ['open','properties','copy']
    ]
)
```

Embedding Beautiful WebForms views in SmartUI¶

In order to embed a Beautiful WebForms form in a SmartUI tile, it is possible to use a **Content Script Result Tile** with the following minimal configuration:

```
def formID = 123456 // the dataID of the form to embed
def viewID = 234567 // the dataID of the SmartUI form view, within the Form Template

form = forms.getFormInfo(formID)
view = asCSNode(viewID)

json([ output : view.renderView(binding, form),
       widgetConfig :[
           reloadCommands:[], // any SmartUI commands that will trigger a reload of the form
           tileContentClasses:"am-nobckg",
           tileLayoutClasses:"am-nobckg"
       ]
     ])
```

Form View Template

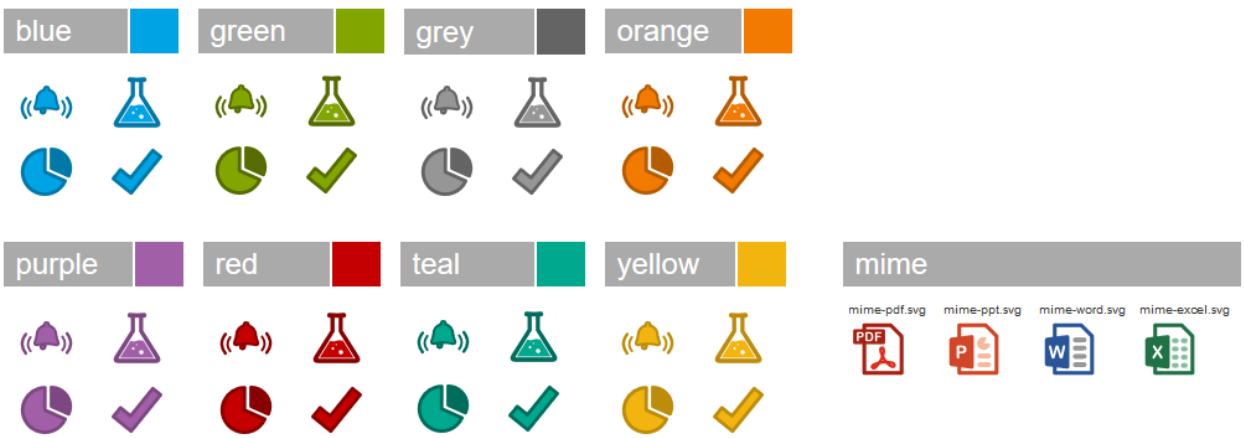
In order for the form to load resources compatible with usage within the SmartUI, you should use the "**SmartView Embeddable**" form template, available within the SmartUI extension libraries.

For additional details, see the dedicated section in the Beautiful WebForms documentation.

Icon reference cheat sheet¶

Iconset Color codes¶

Module Suite icons are available in the following colors:



Copyright © 2013-2022 AnswerModules Sagl

All icons¶

Copyright © 2013-2022 AnswerModules Sagl

A complete list of the currently available icons is shown below:

Copyright © 2013-2022 AnswerModules Sagl

Smart Pages¶

Smart View overrides - general concepts¶

Like many other features in Module Suite, Smart Pages overrides of Smart View features follow a convention on the configuration approach so that for applying a customization to the Smart View UI using one of the supported overrides it is sufficient, in most cases, to create the appropriate script under the appropriate Content Script Volume folder. Smart View overrides are organized as follows:

- Content Script Volume
- CSSmartView
  - Actions *Used to define lazy loaded actions to be displayed in nodes' related actionbars*
  - Commands *Used o define new commands to be displayed in nodes' related actionbars*
  - Columns *Used to define custom dynamic columns to be displayed in Content Server spaces*
  - Overrides *Ovverrides configuration. Its content determines when and where a particular override (above) is used*

Having a possible serious impact on the end user experience, it is important that the system is effective in calculating how, where and when overrides should be applied. For this reason Module Suite uses an elaborate algorithm to determine the **Actual Override Map (AOP)** to use when overrides should be applied. The following is a detailed description of how the AOM is determined.

The content of the **Overrides** folder is used to compute an Override Map (OM), specific to your repository, having the following structure:

```
OM = [
    "globals": [            (1)
        540588
    ],
    "type": [               (2)
        "144": [            (3)
            548066
        ]
    ],
    "tenants": [            (4)
        "497147": [         (5)
            "globals": [    (6)
                548169
            ],
            "type": [       (7)
                "144": [    (8)
                    496932
                ]
            ],
            "ids": [        (9)
                "496931": [ (10)
                    545972
```

Copyright © 2013-2022 AnswerModules Sagl

```
                    ]
                ]
            ]
        ]
    ]
```

where:

- (1) identifies a list of scripts to be always executed
- (2) a list of scripts to be executed only if the current space has at least one node having of the identified type (3)
- (4) scripts to be considered only if the current space is descendant of the specified tenant (5) (a space identified by its DataID)
- (5) is a "tenant" configuration
- (6) identifies a list of scripts that must always be executed if the current space is descendant of the specified tenant (5)
- (7) a list of scripts to be executed only if the current space has at least one node having of the identified type (8) and is descendant of the specified tenant (5)
- (9) a list of scripts to be executed only if the current space has at least one node having of the identified id (10) and is descendant of the specified tenant (5)
- scripts in the OM are executed in the following order (1), (2), (6), (7), (10).

Given the above example and imagining that all the scripts in (3) (8) and (10) return the list ["comm_one","comm_two"], the resulting AOM will contain:

Copyright © 2013-2022 AnswerModules Sagl

```
(3) AOM = [
                ...
                "S144":[commands:["comm_one","comm_two"]],
                ...
            ]
    (8) AOM = [
                ...
                "S144":[commands:["comm_one","comm_two"]],
                ...
            ]
    (10) AOM = [
                ...
                "D496931":[commands:["comm_one","comm_two"]],
                ...
            ]
- scripts in (1), (6), (10) MUST return a Map having entries of the form:
    "SXXXX":[
        commands:["comm_one", "comm_two",...],
        columns: [ //Optional
                    col_name:"col value", //value can be HTML
                    ...
                    ]
    ]
    where XXXX is a valid SubType
    or
    "DYYYY":[
        commands:["comm_one", "comm_two",...],
        columns: [  //Optional
                    col_name:"col value", //value can be HTML
                    ...
                    ]
    ]
```

where YYYY is a valid node's ID.

OM is to be considered a "static" information in productive environments and as such, to guarantee optimal performances, the framework should be allowed to cached it.

This can be done by setting to "true" the " amcs.amsui.volumeCache" parameter int the base configuration.

When a user changes the current space, the OM is evaluated by the framework against the users' permissions and the actual override map (AOM) associated to the space is determined. AOM is determined by executing the relevant scripts in OM in the order described above. The AOM has the following form:

```
AOM = [
    "S144":[                                                                    (1)
            commands:["comm_one", "comm_two",...], //list of commands' command_key  (2)
            columns: [                                                          (3)
                        col_name:"col value", //value can be HTML
                        ...
                    ]
        ],
    "D1234":[                                                                   (4)
            commands:["comm_one", "comm_two",...], //list of commands' command_key
            columns: [
                        col_name:"col value", //value can be HTML
                        ...
                    ]
```

Copyright © 2013-2022 AnswerModules Sagl

```
            ]
    ...
]
```

where: (1) represents commands and columns to be associated to all the nodes having the identified subtype, (3) can be omitted, (4) represents commands and columns to be associated a specific node (identified by its id), (4) takes precedence over (1).

How OM is created ?¶

In order to determine the OM, the content of the **Overrides** folder is evaluated following the logic below:

```
[
    "globals":[              (1)
        540588
    ],
    "type": [                (2)
        "144": [             (3)
            548066
        ]
    ],
    "tenants": [             (4)
        "497147": [          (5)
            "globals": [     (6)
                548169
            ],
            "type": [        (7)
                "144": [     (8)
                    496932
                ]
            },
            "ids": [         (9)
                "496931": [ (10)
                    545972
                ]
            ]
        ]
    ]
]
```

- (1) Contains the list of scripts objects stored directly under "Overrides"
- (2) For each direct subfolder of "Overrides" that has a name starting by the letter "S" an entry is created in "type" map (2). The key of such entry is the target subtype (as specified in the subfolder's name) while the value is the list of scripts contained the aforementioned subfolder.
- (4) For each direct subfolder of "Overrides" that has a name starting by the letter "D" an entry is created in "tenants" map (2). The key of such entry is the tenant's DataID (as specified in the subfolder's name) while the value is the tenant OM configuration.
- (5) For each "tenant" subfolder a sub-Override Map is created (SOM). The structure of SOM is identical to the one of OM with the only difference that subfolders of a tenant subfolder having a name starting with the letter "D" are used in SOM for creating entries in the "ids" map.

Copyright © 2013-2022 AnswerModules Sagl

Below an exemplar content of the Overrides folder

Name	ID	SubType
Overrides	00001	AnsTemplateFolder
- GlobaScript	00002	Content Script
- S144	00003	Content AnsTemplateFolder
- - Document Script	00004	Content Script
- D1234	00005	AnsTemplateFolder
- - S0	00006	AnsTemplateFolder
- - - Folder Script	00007	Content Script
- - D5678	00008	AnsTemplateFolder
- - - Node Script	00009	Content Script

and the resulting OM

```
[
    "globals":[
        00002
    ],
    "type": [
        "144": [
            00004
        ]
    ],
    "tenants": [
        "1234": [
            "globals": [ ],
            "type": [
                "0": [
                    00007
                ]
            },
            "ids": [
                "5678": [
                    00009
                ]
            ]
        ]
    }
]
```

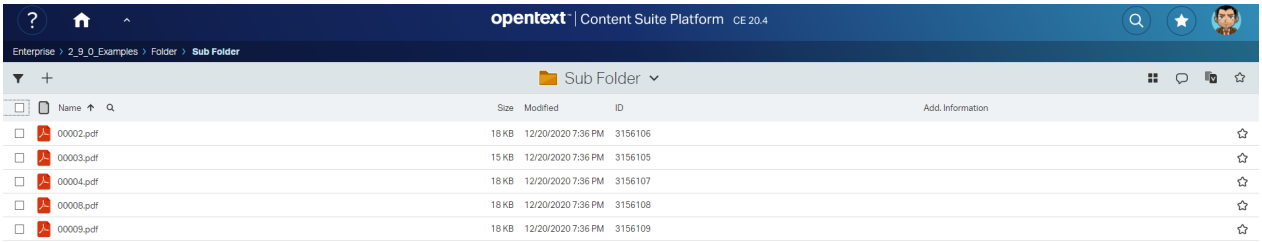
Overrides¶

CSSmartView:Columns¶

It's possible to add/remove columns from/to browsing views using Content Scripts stored in the aforementioned folder. E.g.

Example

Copyright © 2013-2022 AnswerModules Sagl

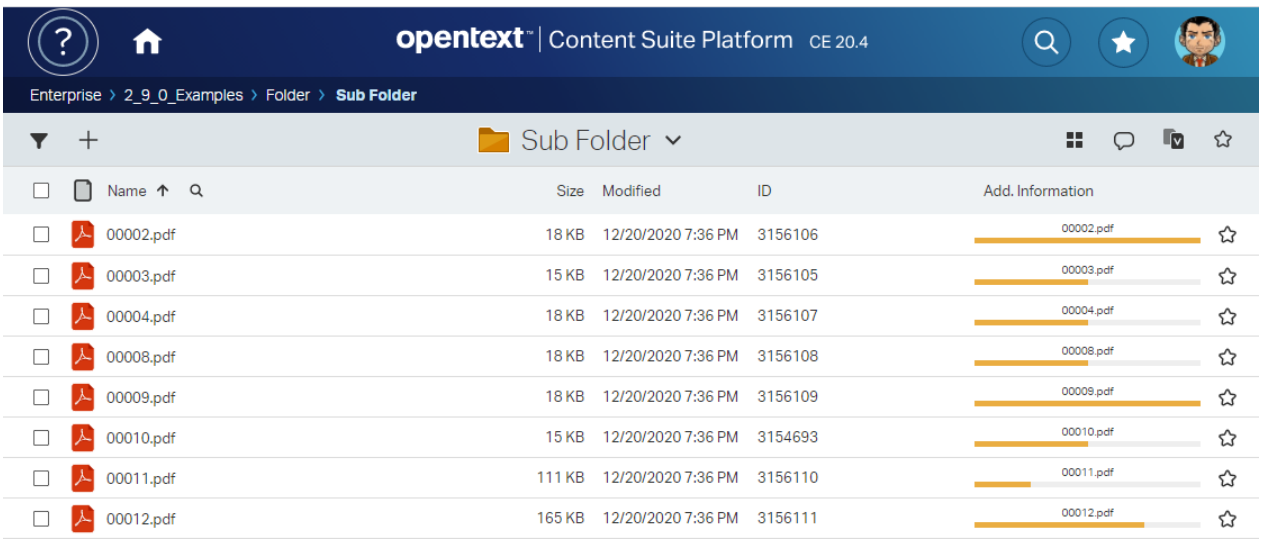


Script

```
//In the execution context of this script:
// - nodesColumns ( a map that associates nodes' ids with their columns definitions). Tipically cont
// - nodes: the list of nodes records. Tipically contains a single item.
// - req: the original REST request record
// - envelope: the current REST API call envelope

nodesColumns[3156087]?.add([type:43200, data_type:43200, name:"Add. Information", sort_key:"type", k

//Must return the revised nodeColumns
return nodesColumns
```

It is possible to enhance the information associated with nodes with column information injected via Module Suite E.g.

Example



Script

```
def drawStatusBar = { node ->

    def statusList = ['Draft', 'Under Revision', 'Approved', 'Published']
    def numSteps = statusList.size()
    def currStep = new Random().nextInt(statusList.size())
    def currStepName = node.name

    def stepStyle = "height:100%; width:calc(100% / ${numSteps}); float:left; background-color:#F0AD

    def stepsHtml = ""
```

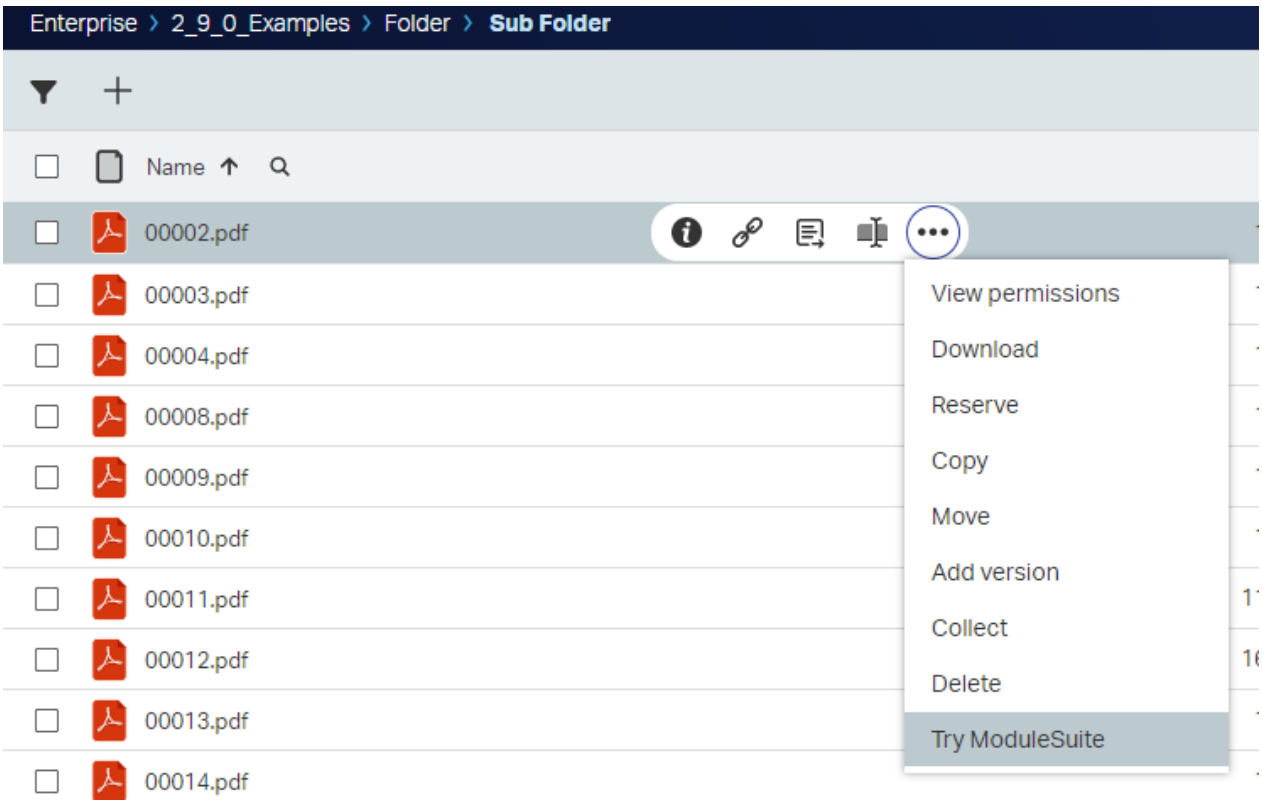
Copyright © 2013-2022 AnswerModules Sagl

```
    (currStep + 1).times{
        stepsHtml += """<span style="${stepStyle}"></span>"""
    }


    return """
    <div style="text-align:center; font-size:.75em">${currStepName}</div>
    <div style="margin:3px 0; padding:0; height:5px; background-color:#eee;">${stepsHtml}</div>""".t
}



retVal =  nodes.collect{
    [
        ("D${it.dataid}" as String):[ //The object returned MUST be made of simple types (no GString
            commands:["am_group", "am_bwf"],
            columns:[
                // Column defined in CSSmartView:Columns as nodesColumns[3156087]?.add([type:43200, 
                // columns of type 43200 can be used to inject HTML
                _am_info:drawStatusBar(it)
            ]
        ]
    ]
}
return retVal
```

CSSmartView:Actions¶

It's now possible to add custom actions to a node's menu lazy loaded set of actions . E.g.

Example



Script

Copyright © 2013-2022 AnswerModules Sagl

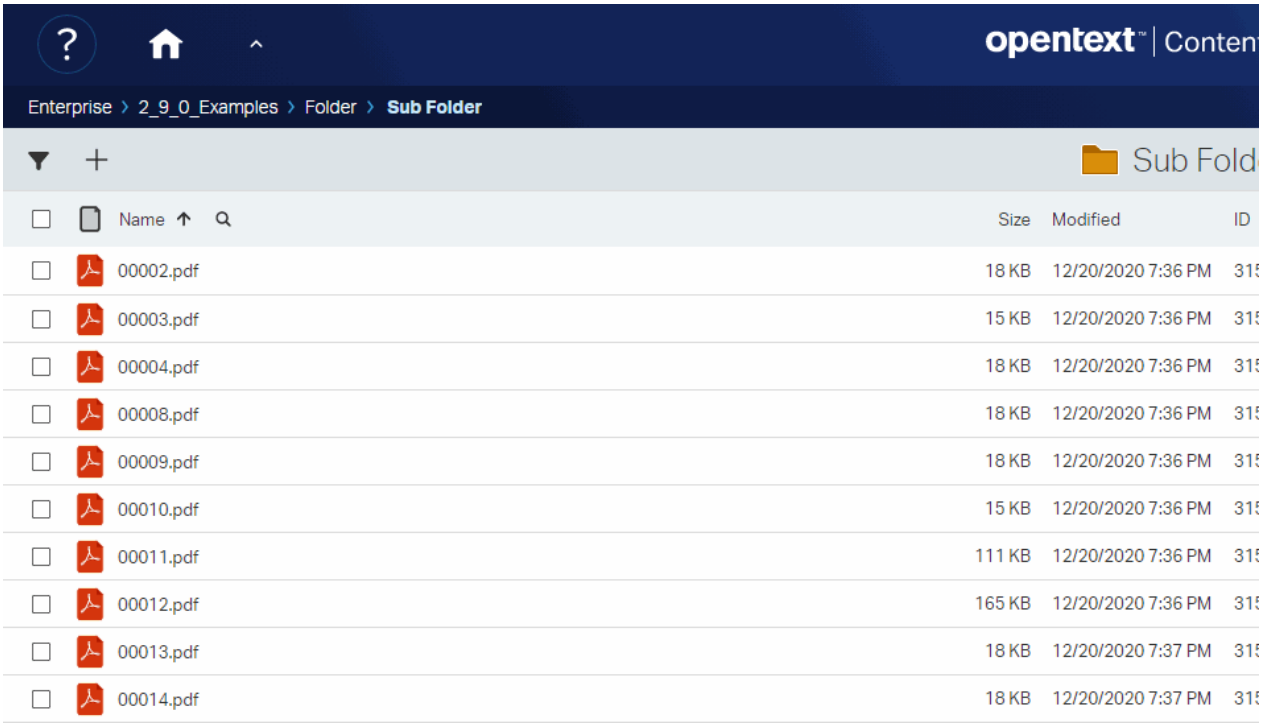
```
/**
This script receives the following variables in the execution context:

- actions: a map that associates the node id to the list of available actions
E.g.
    "12345": {
        "data": {
            "Classify": {
                "content_type": "application/x-www-form-urlencoded",
                "method": "POST",
                "name": "Add RM Classification",
                "href": "/api/v2/nodes/2891606/rmclassifications",
                "body": "{\"displayPrompt\":false,\"enabled\":false,\"inheritfrom\":false,\"managed\'
                "form_href": ""
            },
            "initiatedocumentworkflow": {
                "content_type": "",
                "method": "",
                "name": "",
                "href": "",
                "body": "initiate_in_smartview",
                "form_href": "",
                "wfList": [

                ]
            },
            "zipanddownload": {
                "content_type": "",
                "method": "POST",
                "name": "Zip and Download",
                "href": "/api/v2/zipanddownload",
                "body": "",
                "form_href": ""
            },
            "RemoveClassification": {
                "content_type": "application/x-www-form-urlencoded",
                "method": "POST",
                "name": "Remove Classificfation",
                "href": "/api/v2/nodes/2891606/rmclassifications",
                "body": "",
                "form_href": ""
            }
        },
        "map": {
            "default_action": "open"
        },
        "order": [
            "initiatedocumentworkflow",
            "Classify",
            "RemoveClassification",
            "zipanddownload"
        ]
    }
}

- req: the current HTTP request
- envelope: the REST API request's envelope

By changing the support variable "actions" you can make visible actions defined by scripts in CSVolu

**/

actions[3156106].data["am_release"] = [
    body:"am_release"
]
actions[3156106].order.add("3156106")
```

Copyright © 2013-2022 AnswerModules Sagl

CSSmartView:Commands¶

It's possible to define multiple commands in the same script and group them in the same sub-menu. E.g.

Example



Script

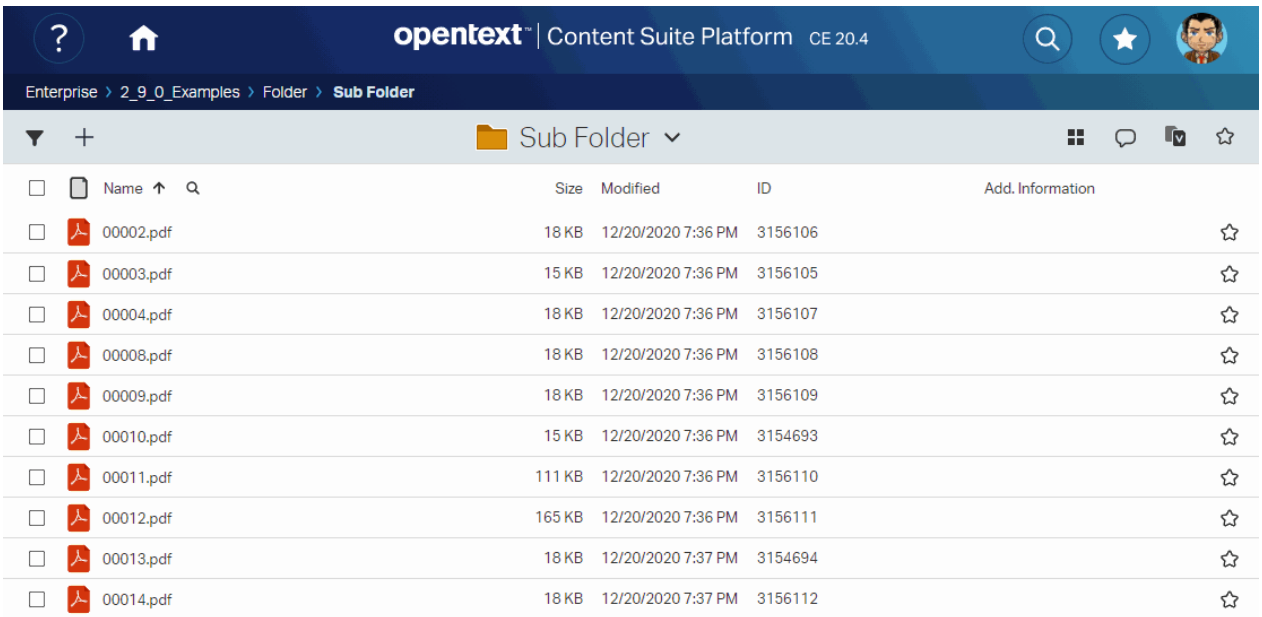
```
//Commands scripts can now return a list
return [
    [
        am:[
            exec:[
                mode:"group"// (1) This command will act as our flyout
            ]
        ]
        ,scope: "multiple"
        ,group: "info"
        ,flyout: "am_group" // (2) This command will act as our flyout
        ,baricon: null
        ,icon: null
        ,name: "Try Module Suite"
        ,command_key: "am_group"
        ,signature: "am_group"
    ],
    [
        am:[

            confirmation:[
                required:false,
                title:"",
                message:""
            ],
            panel:[
                width:40,
                cssClass:"",
```

Copyright © 2013-2022 AnswerModules Sagl

```
                slides:[
                    [
                        title:"",
                        script:null
                    ]
                ]
            ],
            key:[
                code: 83
                ,message:""
                ,nogui:false
            ],
            exec:[
                mode:"script"
                ,script: 2644067
                ,params:[

                ]
                ,refresh_on_success:true
                ,on_success_action:""
                ,newtab:false
                ,url:""
            ]
        ]
        ,baricon: null
        ,icon: null
        ,name: "Content Script"
        ,command_key: "am_content_script"
        ,signature: "am_content_script"
        ,scope: "multiple"
        ,flyout:"am_group"
        ,selfBlockOnly: false
    ]
    ...
]
```

Content Script scripts executed as commands can return execution information to the caller. E.g.

Example



Script

Copyright © 2013-2022 AnswerModules Sagl

```
//Script code...
//Once done...notify caller
json([message:[type:'success', text:"Get the Module Suite. You won't need anything else.", details:"
```

Copyright © 2013-2022 AnswerModules Sagl

Script Console

Working with Script Console

Execution modes¶

Script Console is a runtime environment that features different execution modes (a shell, a script interpreter and a lightweight webserver) therefore it's the perfect solution when it comes to integrate Content Server with external systems. The simplest way to use the Script Console is to start it as a command line shell.

Script Console can run under both a Windows system and a Unix system, being based on a modular Java-based architecture. The main scripts for both the supported platforms are located under the "**bin**" directory in the runtime installation directory.

Command Line Shell Mode¶

In order to start the Script Console as a command line shell you have to execute the following command

```
>app-windows
```

Without any additional parameter. The system should respond you with the Script Console prompt (as shown in the figure below)

The prompt indicates the current system and its connection status. In the case of the figure above the current system has been labeled "TEST" and is currently off-line. New system can be added using the main configuration file of the Script Console. When newly installed a "TEST" system configuration is made available for future references.

Copyright © 2013-2022 AnswerModules Sagl

An online help about the supported commands is available directly from the Script Console shell. Here below the list of all the commands available out-of-the-box:

loadcs

```
usage: loadcs -i 00000

Load a Content Script from a file or from Content Server

-e,--encoding <arg> The file encoding (platform default if not

specified)

-f,--file <arg> The local file to load as a script

-h,--help This help message

-i,--id <arg> The ID of the target script on the system
```

memsrc

```
usage: memsrc -g "MyGroup"

Search members

-@,--col-email COLUMN: Mail address

-a,--all Use a long listing format for results

-c,--match-contains MATCHING: Contains

-e,--match-endswith MATCHING: Ends with

-f,--col-first-name COLUMN: First name

-g,--filter-groups FILTER: Search only groups

-h,--help This help message

-k,--match-like MATCHING: Sounds like

-l,--col-last-name COLUMN: Last name

-m,--filter-members FILTER: Search any member
```

Copyright © 2013-2022 AnswerModules Sagl

```
-n,--col-name COLUMN: Name

-s,--match-startswith MATCHING: Starts with

-u,--filter-users FILTER: Search only users
```

ls

```
usage: ls

List the children of the current node

-h,--help This help message

-l,--long Use a long listing format
```

rm

```
usage: rm "Node to delete" "Another node to delete"

Delete one or more nodes in the working node

-h,--help This help message

-i,--id Reference nodes by ID

-p,--parent <arg> Use specified parent in place of working node

-r,--regexp <arg> Match the node names to delete against the

specified regexp
```

mkdir

```
usage: mkdir "Folder Name"

Create a new folder

-h,--help This help message

-p,--parent <arg> The parent ID of the new folder
```

script

Copyright © 2013-2022 AnswerModules Sagl

```
usage: script

Switch to scripting mode. In script mode you can write and save your one
script one line at a time

-h,--help This help message
```

quit | exit

```
Shutdown and exit
```

whoami

```
Information about the current user
```

loaddocs

```
usage: loaddocs -d /home/user/myDocs -i -r .\*.pdf

Load documents on Content Server

-d,--directory <arg> The local directory to load files from

-h,--help This help message

-i,--interactive Prompt for confirmation for each file

-n,--name Prompt for a new name for each file

-p,--parent <arg> The target directory

-s,--suffix <arg> Match the node names to delete with the

specified suffix
```

system

```
usage: system -options

List systems or switch the current system

-a,--add <arg> Add a new system
```

Copyright © 2013-2022 AnswerModules Sagl

```
-c,--current The current system details

-h,--help This help message

-l,--list List all the available systems

-s,--system <arg> Switch to the target system
```

pwd

```
Print the current working node
```

mkuser

```
usage: mkuser bob -p passwd1 -g "MyGroup, Developers"

Create a new user

-a Public access enabled

-c Can create and update users

-g Can create and update groups

-h,--help This help message

-l Login enabled

-p,--password <arg> The initial password

-s Can administer system

-u Can administer users
```

interactive

```
Switch to interactive console mode. In interactive mode you can enter
Content Script commands and execute them directly.
```

sync

```
usage: sync

Synchronized console command scripts
```

Copyright © 2013-2022 AnswerModules Sagl

```
-c,--commit Commit modified local scripts to Content Server

-h,--help This help message

-n,--name <arg> The single command to sync
```

su

```
usage: su bob

Impersonate a different user

-h,--help This help message

-r,--restore Restore the original logged in user
```

login

```
usage: login -options

Login to the specified system

-h,--help This help message

-i,--interactive Force credential prompt (useful id there are

saved credentials)

-k,--save Save the provided credentials (Crypted)

-p,--password <arg> The user's password

-s,--system <arg> The system to connect to

-u,--username <arg> The username
```

cd

```
usage: cd -i 2000

Change the current working node

-c,--category Switch to category WS
```

Copyright © 2013-2022 AnswerModules Sagl

```
-e,--enterprise Switch to enterprise WS

-h,--help This help message

-i,--id <arg> The ID of the target node

-n,--nickname <arg> The nickname of the target node

-p,--personal Switch to personal WS
```

logout

```
Logout from the current system
```

loadConfig

```
usage: loadConfig -v -m Mode

Loads the current system Base Configuration in the Script Console
Configuration

-h,--help Usage Information

-m,--mode <arg> Mode: either BASE, CUSTOM, ALL

-v,--verbose Verbose
```

Creating new command

New commands can be registered using Content Script to implement them. Script Console comes with a set of example commands implemented through Content Scripts that a developer can use as a reference to create his own.

Script Interpreter Mode¶

The Script Console can also be executed as a Script interpreter (in order to execute a specific Content Script) in this case the Console should be executed specifying both the script to be executed and the system to log in:

```
>app-windows -c Script.cs -s SYSTEM
```

Copyright © 2013-2022 AnswerModules Sagl

In order to be able to execute the Script Console with this Mode valid user's credentials should have been registered using the command:

```
login -k -i -s SYSTEM
```

Server Mode¶

A third way the Script Console can be executed is as a lightweight webserver. In this case the Console should be executed specifying both the port on which to listen for incoming connection and the system to log in:

In order to be able to execute the Script Console with this Mode valid user's credentials should have been registered using the command:

```
login -k -i -s SYSTEM
```

Script repositories¶

The Script Console organizes the registered Content Script in isolated repositories. A Script repository might be dedicated to a specific system (in this case the Scripts stored in this repository will be loaded and made available only when the user decides to login to that system), or to a specific extension.

Script Console extensions' script are made available through all the configured systems.

Script Console features a synchronization command (synch), that can be used, both when the Console is running as a shell as well as when the console is running as a web server, in order to synchronize a system repository with the contents of the corresponding **CSCommands** Template folder in the Content Script Volume of the current system.

Script Console Internal scheduler configuration file ¶

The Script Console features an internal scheduler configurable through an XML configuration file (**cs-console-schedulerConfiguration.xml**) that is stored under the **config** directory.

The internal scheduler allows to plan and execute tasks to be automatically run in the Script Console. It is based on Quartz open source library (a well-known Java Scheduler). For further

Copyright © 2013-2022 AnswerModules Sagl

information please make reference directly to the Quartz documentation http://quartz-scheduler.org/ *(http://quartz-scheduler.org/)* .

Copyright © 2013-2022 AnswerModules Sagl

Extension for DocuSign

Working with DocuSign

This guide includes the basic set of operations that can be used to setup a document signing process using the **Module Suite Extension for DocuSign**.

Creating a signing Envelope¶

One of the core concepts when setting up a DocuSign signing process is the "**Envelope**", which represents the overall container for a transaction.

When defining an envelope, you will be able to provide all details of the transaction. The minimal set of information to provide includes:

- the **documents** to sign
- the **recipients** of the signing request
- the **message** they will receive

See the official DocuSign REST API guide *(https://developers.docusign.com/esign-rest-api/guides/concepts/envelopes)* for more details on this topic.

The **docusign** Content Script service includes methods to programmatically create and send signing envelopes.

EXAMPLE: Creating a simple envelope¶

```
def contract      = docman.getDocument(123456)
String contractID = contract.ID as String

definition = docusign.getNewEnvelopeDefinition()
                     .setEmailSubject("XYZ contract for signature")
                     .setEmailBody("Please sign the contract.")
                     .addRecipient('signers', 'Homer J. Simpson', 'homer@example.com', 'Manager')
                     .addSignHereTab("homer@example.com", contractID, "Sign here", 1, 89, 100)
                     .addDocuments(contract)
                     .notifyOnEnvelopeCompleted()
                     .notifyOnEnvelopeDeclined()
                     .notifyOnEnvelopeVoided()

envelope = docusign.createEnvelopeAndSend(null, definition)

docusign.registerEnvelope(envelope) // This command will register the envelope locally on Content Se
```

Copyright © 2013-2022 AnswerModules Sagl

EXAMPLE: Creating an envelope using a predefined template¶

When creating a new DocuSign envelope, it is possible to provide the envelope configuration in the form of a Map object. The structure of this map is compatible with the JSON format DocuSign uses to define Envelopes and Templates. For this reason, for complex envelope templates, a possible approach is to define the Template within your DocuSign account (using the visual editor to setup Recipients, Signing Tabs, etc.) and then export it and use it within your Content Script app.

```
def documentToSign      = docman.getDocument(123456)
def emailMessageSubject = "XYZ contract for signature"
def emailMessageBody    = "Please sign the contract."
def documentsToSign     = [documentToSign]

def user = users.current


def envDefinition = [

    "documents"         : documentsToSign,
    "emailSubject"      : emailMessageSubject,
    "emailBlurb"        : emailMessageBody,
    "signingLocation"   : "Online",
    "authoritativeCopy" : "false",
    "notification": [
        "reminders": [
            "reminderEnabled"   : "false",
            "reminderDelay"     : "0",
            "reminderFrequency" : "0"
        ],
        "expirations": [
            "expireEnabled" : "true",
            "expireAfter"   : "120",
            "expireWarn"    : "0"
        ]
    ],
    "enforceSignerVisibility" : "false",
    "enableWetSign"           : "true",
    "allowMarkup"             : "false",
    "allowReassign"           : "false",
    "messageLock"             : "false",
    "recipientsLock"          : "false",
    "recipients": [
        "signers": [ user ],

        /* Alternatively, a map structure can be provided to define recipients (required for externa

        "signers": [
            [
                "defaultRecipient"   : "false",
                "signInEachLocation" : "false",
                "name"  : "",
                "email" : "",
                "otuser":[
                            "name"  : user.displayName,
                            "email" : user.email,
                            "ID"    : user.ID
                          ],
                "accessCode"       : "",
                "requireIdLookup"  : "false",
                "routingOrder"     : "1",
                "note"             : "",
                "roleName"         : "Responder",
```

Copyright © 2013-2022 AnswerModules Sagl

```
                "deliveryMethod"   : "email",
                "templateLocked"   : "false",
                "templateRequired" : "false",
                "inheritEmailNotificationConfiguration": "false",
                "tabs": [
                    //"signHereTabs": []
                ]
            ]
         ],
        */

        "agents"              : [],
        "editors"             : [],
        "intermediaries"      : [],
        "carbonCopies"        : [],
        "certifiedDeliveries" : [],
        "inPersonSigners"     : [],
        "recipientCount"      : "1"
    ],

    "envelopeIdStamping" : "true",
    "autoNavigation"     : "true"
]

def envDef = docusign.getNewEnvelopeDefinition(envDefinition)
                     .notifyOnEnvelopeSent()
                     .notifyOnRecipientCompleted()
                     .notifyOnEnvelopeCompleted()

def env =  docusign.createEnvelopeAndSend(null,  envDef)

envelope = docusign.registerEnvelope(env).envelope // Register this envelope on Content Server. This
```

Embedded recipients¶

Module Suite Extension for DocuSign supports **embedded signing** for authenticated OTCS users. When using this pattern, DocuSign delegates the task of identifying the recipients of the signing request to Content Server. Content Server is allowed to request the generation of a pre-signed **signing url**, which can be used by the recipient to sign the documents without having to authenticate with DocuSign. This approach avoids the context switching of the normal flow, which would require to open the system-generated email notification and access the DocuSign signing request from the provided link.

Refer to the official DocuSign REST API Guide - Embedding *(https://developers.docusign.com/esign-rest-api/guides/features/embedding)* for further details on this topic.

When using the **embedded signing** pattern, recipients should be specified using a CSUser object.

EXAMPLE: Get a pre-authenticated signing URL for an OTCS internal user¶

In order to generate a signing URL for an **embedded recipient**, use the *docusign.getRecipientUrl(...)* API.

Copyright © 2013-2022 AnswerModules Sagl

```
String envelopeID = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
String username = 'Admin'


def user = users.getUserByLoginName('Admin')
def env  = docusign.getEnvelope(envelopeID)

String profile               = null
String recipientUserName     = user.name
String recipientEmail        = user.email
String recipientClientUserID = user.ID as String
String recipientID           = env.recipients.find{ it.clientUserID == recipientClientUserID }.recip

String nextUrl               = "http://mycontentserver.example.com/otcs/cs.exe"

String signingUrl            = docusign.getRecipientUrl( profile, envelopeID, recipientUserName, rec

redirect signingUrl
```

Envelope status update and signed document synch back¶

An important action to be performed when a signing workflow is concluded is to retrieve the signed documents and synchronize them back on your Content Server system. Module Suite Extension for DocuSign supports automating this task in different ways:

- Subscribe to DocuSign push notifications when the envelopes change state (webhook pattern)
- Poll the envelope status and update the local instance when a change is detected

The first approach (webhook) relies on the creation of an endpoint that can be invoked from DocuSign when changes happen. This pattern can be implemented by setting up the **Script Console DocuSign Extension**

The second approach (polling) can be implemented by using the *getEnvelopeUpdates(...)* API on the **docusign** service.

EXAMPLE: Poll DocuSign for Envelope updates and synch back documents¶

The following script can be scheduled to periodically update all active DocuSign envelopes.

Correct API usage

DocuSign monitors that the usage of the API is compliant with certain guidelines. Specifically, certain APIs cannot be invoked with a frequency that goes over a certain threshold. When scheduling polling scripts, make sure that the scheduling frequency complies with the DocuSign guidelines.

NOTE: This limitation can be overcome by using the **webhook** pattern, as described earlier.

Copyright © 2013-2022 AnswerModules Sagl

```
res =  sql.runSQLFast("""SELECT AM_DocuSign.EnvelopeID ENVELOPEID
                         from AM_DocuSign where
                              AM_DocuSign.EnvelopeStatus not in ('completed', 'Completed')""", fals
if(res){
    docusign.getEnvelopesUpdates(null, res).each{

        docusign.updateEnvelope(docusign.getEnvelopeDetails(null, it.envelope))

        if(it.envelopeStatus == "completed"){
            docusign.getEnvelopeDocuments(null, it.envelope).each{ doc->
                doc.each{
                    if(it.key > 0){
                        docman.getNodeFast(it.key).addVersion(it.value)
                    }
                }
            }
        }
    }
}
```

Copyright © 2013-2022 AnswerModules Sagl

How to

Content Script: Retrive information

Nodes¶

Getting Content Server nodes¶

All the objects stored on OpentText Content Server are referred as nodes in Content Script.

The base interface representing a node is the CSNode interface. CSNode is the base interface for most of the Content Script API objects *(/working/contentscript/scripts/#content-script-api-objects)*.

Almost all the Content Script API Objects inherit from CSNodeImpl which is the base-class implementing the CSNode interface. As said a *node* represents an object on Content Server.

Different Objects correspond to different implementation of the CSNode interface (e.g. Folders(*SubType=0*) are implemented by CSFolderImpl, Documents(*SubType=144*) correspond to CSDocumentImpl).

A CSNode (more generally speaking any Content Script API Object) features:

- **Properties**: this is information specific to the Content Server object (e.g. name, subtype, size, creation date) and may vary for each CSNode implementation. In order to be recognized as properties the CSNode fields must be decoretad with the `@ContentScriptAPIField`;
- **API Methdos**: these are the APIs used to manipulate and retrieve information associated with objects;
- **Features**: these are additional features that are not strictly related to objects (their are not object's properties) but depend on external factors: the way Content Server is configured (which modules are available, how are they configuration), on object's configuration, on the user's permissions on the objects, on the context in which the features are accessed etc.

The Content Script API service you are going to use the most for retriving nodes is the docman{:style="color:red"} service.

docman{:style="color:red"} features several methods that allows you to retrive a node given:

- its unique numeric identifier;

Copyright © 2013-2022 AnswerModules Sagl

- its path
- its name and the container in which is located;
- its nickname;
- etc..

The Base API *(/working/contentscript/scripts/#api-services)* also features a asCSNode{:style="color:#e7853c;font-size:bold;"} method that serves as a shortcut for the above mentioned use cases.

Performances-tip: Lazy loading

In order to optimize performances, Content Scripts lazy-loads information from OTCS 'database, which means that such information is not available until firstly accessed. docman{:style="color:red"} APIs allow you to specify which information you want to load beforehand. Retriving the minimum amount of information necessary is tipically done using the APIs ending with the Fast suffix and is to be consider a **best practice** and might have a significant impact over your's application performances.

Do

```
def node = docman.getNode(123456)

if((node.Invoice.Status as String) == "Paid"){ // The node is loaded with regular method at l
    ...
}
```

Don't

```
node = docman.getNode(123545)
if(node.parentID == -1 ){ //ParentID is a base property for CSNode and since we are only acce
    ...
}
```

Getting a node given its ID¶

```
def node = docman.getNode(2000, //NodeID (on most of the environments 2000 identifies the Enterprise
                      true, //'true' if Reference information shall be loaded
                      true, //'true' if Reservation information shall be loaeded
                      true, //'true' if Versions information shall be loaeded
                      true, //-true- if Current Version  shall be loaeded
                      true, //'true' if Node's features shall be loaeded
                      true, //'true' if Metadata  shall be loaeded
                      true, //'true' if Permissions information shall be loaeded
                     )

node = docman.getNode(2000) //this is a shortcut for docman.getNode(2000, true, true, false, false,

node = docman.getNodeFast(2000) //this is a shortcut for docman.getNode(2000, false, false, false, f

node = asCSNode(id:2000) //this is a shortcut for docman.getNode(2000)

node = asCSNode(2000)//this is a shortcut for asCSNode(id:2000)
```

Copyright © 2013-2022 AnswerModules Sagl

Get a list of nodes given their IDs¶

```
docman.getNodesFastWith(
                        [2000L, 2006L], // List of nodes IDs
                        ["GIF", "promotedCmds", "defaultLink", "size", "tableName"], // List of addi
                        params, //Current request parameters
                        true, //'true' if Versions information shall be loaeded
                        true, //'true' if Node's features shall be loaeded
                        true //'true' if Permissions information shall be loaeded
                       )

docman.getNodesFast(2000L, 2006L) //this is a shortcut for docman.getNodesFastWith([2000L,2006L], [],

docman.getNodes(2000L, 2006L) //this is a shortcut for docman.getNodesFastWith([2000L, 2006L], [], [
```

Get Volumes¶

The most common volumes can be easily accessed using a dedicated API featured by the docman service. If an API is not available a volume can be retrieved using a simple SQL query based on its subtype. Volumes come in handy when you want to retrieve a node by its path.

```
docman.getEnterpriseWS() //Enterprise Workspace

docman.getPersonalWS()   //Personal Workspace

docman.getCategoryWS()   //Category Workspace

docman.getContentScriptVolume() //Content Script Volume

/*
 161 -- Workflow Volume
 198 -- Classification Volume
 211 -- Reports Volume
 233 -- Database Lookups
 236 -- Database Connections
 274 -- Best Bets
 405 -- Recycle Bin
 541 -- Content Server Templates
 862 -- Connected Workspaces
 863 -- Workspace Types
*/

def node =  docman.getNodeFast(sql.runSQLFast("""Select "DataID"
                FROM DTree
                Where SubType = 161""", false, false, 0
             ).rows[0].DataID)
```

Get Nodes By Path¶

```
def ews = docman.getEnterpriseWS()

node = docman.getNodeByPath(ews, "Training:Folder")

node = docman.getNodeByPath( "Training:Folder") //this is a shortcut for docman.getNodeByPath(docman

node = asCSNode(path:"Training:Folder")//this is a shortcut for docman.getNodeByPath("Training:Folde
```

Copyright © 2013-2022 AnswerModules Sagl

Performances-tip: Use the variable to avoid reloading the same information

In order to optimize performances, you should always assign information you know is not going to change (during your script execution) to Content Script variables so to avoid to reload them everytime they are accessed.

Do

```
def ews = docman.getEnterpriseWS()

node = docman.getNodeByPath(ews, "Training:Folder")

node = docman.getNodeByPath(ews, "An:Other:Path")
```

Don't

```
node = docman.getNodeByPath( docman.getEnterpriseWS(), "Training:Folder")

node = docman.getNodeByPath( docman.getEnterpriseWS(), "An:Other:Path")
```

Users and Groups¶

Getting Content Server Users and Groups¶

Content Server Users and Groups are managed by the *users* service in the Content Script. *users* service operates with CSMember, CSUser and CSGroup classes. CSUser and CSGroup are the classes that are providing API to work with the Content Server Users and Groups correspondingly. CSMember is an abstract class for for CSUser and CSGroup objects. It is used in the API where both Users and Groups classes can be passed as a parameter or return as a method return value. *users* service provides set of methods to retrieve User or a Group:

- get current user (user who is actually executing Content Script)
- get user/group by id
- get group by name
- get user by login name
- list group members
- etc..

Get current User¶

```
def user = users.current // Will return CSUser object of the user that is executing the script
```

Get by member ID¶

```
CSMember member
```

Copyright © 2013-2022 AnswerModules Sagl

```
// Pass User or Group by ID. Method will return CSUser or CSGroup class objects

//Pass ID of the Content Server User
member = users.getMemberById(1000)
out << member instanceof CSUserImpl // will display true

//Pass ID of the Content Server User
member = users.getMemberById(1001)
out << ( member instanceof CSGroupImpl ) // will return true

//Get User by ID
member = users.getUserById(1000) // will return CSUser class object

//Get group by ID
member = users.getGroupById(1001) // will return CSGroup class object
```

Get member by the name¶

```
CSMember member

//Get Member using User Login Name
member = users.getMemberByLoginName("Admin") // Will return CSUser class object

//Get Member using Group Name
member = users.getMemberByLoginName("DefaultGroup") // Will return CSGroup class object

//Get User by UserName
member = users.getUserByLoginName("Admin")

//Get Group by Name
member = users.getGroupByName("DefaultGroup")
```

Get members by ID¶

```
def members

//Get by IDs
members = users.getMembersByID(1000,1001)

//members[0] - is object of CSUser class
//members[1] - is object of CSGroup class
```

Permissions¶

Getting Content Server Node Permissions¶

Content Script *docman* service allows script developers to perform operations with the Content Server permissions model. To get get node permissions:

```
CSNode node = asCSNode(33561)
//Node permissions can be retrieved either
//calling CSNode getRigths() method
CSNodeRights nodeRights = node.getRights()
```

Copyright © 2013-2022 AnswerModules Sagl

```
//or by calling docman method and passing node as an attribute
nodeRights = docman.getRights(node)
```

Content Server permissions model is represented as two classes ***CSNodeRights*** and ***CSNodeRight***. ***CSNodeRights*** class contains all the permissions of the node. It's fields correspond to Content Server node permission type. **ownerRight** - Owner Permissions **ownerGroupRight** - Owner Group Permissions **publicRight** - Public Access Permissions **ACLRights** - list of Assigned permissions Every permission is an ***CSNodeRight*** object, with following fields: **rightID** - ID of the User/Group to whom this Right is assigned **permissions** - list of permissions set. Following options are possible:

```
1   [SEE, SEECONTENTS, MODIFY, EDITATTRIBUTES, RESERVE, ADDITEMS, DELETEVERSIONS, DELETE, EDITPERMIS
```

To get node permissions:

```
//To get Owner Permissions
out << nodeRights.ownerRight.permissions

//To get Assignemt Permissions Users with their permissions
def assignedAccessUsers = [:]

nodeRights.ACLRights.each{ right ->
    def currUser = users.getMemberById(right.rightID);
    assignedAccessUsers[currUser.name] = right.permissions
}

out << assignedAccessUsers
```

There are set of methods to check if current user has special permissions against the node. Methods to check permission are implemented for CSNode and they are prefixed with "has" and than following permissions description:

- hasAddItemPermission()
- hasDeletePermission()
- hasDeleteVersionsPermission()
- hasEditAttributesPermission()
- hasEditPermissionsPermission()
- hasModifyPermission()
- hasReservePermission()
- hasSeeContentsPermission()
- hasSeePermission()

Sample validation:

```
CSNode node = asCSNode(33561)

out << node.hasDeletePermission() //will return TRUE if current user has Delete pemissions on a node
```

Copyright © 2013-2022 AnswerModules Sagl

Categories¶

Getting Node Categories¶

Content Script *docman* service allows to performs full set of actions related to Content Server categories. Below you will find samples how to get Category definition and get Content Server node categories along with its attribute values.

```
def category = docman.getCategory(self.parent, "User Info") // Object of type CSCategory

def attributesMap = category.getAttributes() // Get map with Category Attributes

def firstNameAttr = category.getAttribute(attributesMap[2 as Long]) // get definition of the attribu

out <<  "Attribute ${firstNameAttr.getDisplayName()} has default value set to: ${firstNameAttr.value
```

Get value of the category attributes applied to a node:

```
def node = docman.getNodeByName(self.parent, "Folder With Categoty")

//Get Attribute value
def attrValue = node."User Info"."First Name" as String
out << "The current value of First Name is now ${attrValue} <br/>"

//get first attribute value
attrValue = node."User Info".Phone
out << "Get first Phone attribute value ${attrValue} <br/>"

//get all attribute values
attrValue = node."User Info".Phone as List
out << "Get all Phone attribute values ${attrValue} <br/>"
```

You can always export the category as a map, and later on update it from the very same map:

```
out << node."User Info" as Map
```

Classification¶

Manipulation with a node Classifications in Content Script is performed by the *classification* service. This sections describes how to get classifications applied to a node.

First of all if you need to check if node is classifiable:

```
def node = docman.getNodeByName( self.parent, "Test Folder")

//Check if Classification can be applied to the node
out << "Classification can be applied to a node: ${classification.isClassifiable(node)}"
out << "<br>"

//List classifialbe subtypes
```

Copyright © 2013-2022 AnswerModules Sagl

```
out << "Classification can be applied to following node subtypes:"
out << "<br>"
out << classification.listClassifiebleSubTypes()
```

To get classifications:

```
def node = docman.getNodeByName( self.parent, "Test Folder")

// get node classifications
def classifications = classification.getClassifications(node)

//Will return list of classifications applied to a node
out << classifications.collect { it.name }
```

Executing SQL queries¶

Content Script API allows execution of SQL statements against Content Server database, without the need for creatomg a LiveReport object. *sql* service has a set of methods allowing developer to run SQL queries.

Not all DBMS are equal

Please keep in mind DBMS server SQL specific syntax of the queries used. Adapt provided queries to the DBMS server type in your environment.

Execute a simple SQL query¶

```
out << sql.runSQL("""select * from DTree where %1 and ParentID = %2 and ModifyDate > %3""", //SQL Co
           true, // true if the query must be executed using a cursor
           true, // true if the query must be wrapped in a transaction (required administrative priv
           10,   // numer of records to be returned

                              // Below the list of optional parameters
           "#FilterObject:0", // Parameters can be a LiveReport query template expression
           2000,              // Integers
           1.year.ago).rows   // Dates
                              // Strings
```

The above query is executed with three parameters, specified as %N in the SQL statement.

SQL execution methods are returning *CSReportResult* class object. To get query executing result *rows* feature should be used, as in the example above.

Another option to run SQL queis utilization of the `sql.runSQLFast()` methods. Syntax for *"Fast"* methods is the same. These methods are faster implementation of the SQL execution script, but the compromise is that they are not ThreadSafe (i.e. not to be used in multi-threaded scripts).

Copyright © 2013-2022 AnswerModules Sagl

Execute a SQL query with pagination¶

In some cases it is required to implement queries that return paginated data, e.g. for browsing pages. *sql* exposes a set of methods that allow developers to easily build such queries The example below provides an overview of the usage of `sql.runPaginatedSql()` API:

```
def sqlProjections = "DataID, Name"
def fromClause = "DTree dt"
def whereClause = "SubType = 0"
def pageSize = 5
def transaction = true

def runPaginatedQuery = { firstRow ->

    def sqlResult = sql.runPaginatedSql(sqlProjections, fromClause, whereClause, firstRow, pageSize,

    out << "<br>"
    out << "Start row ${firstRow}"
    sqlResult.rows.each { row ->
        out << "<br>"
        out << "Folder Name: ${row.name}. Name: ${row.dataid}"
    }
}

runPaginatedQuery(1)
runPaginatedQuery(6)
```

Working with Forms¶

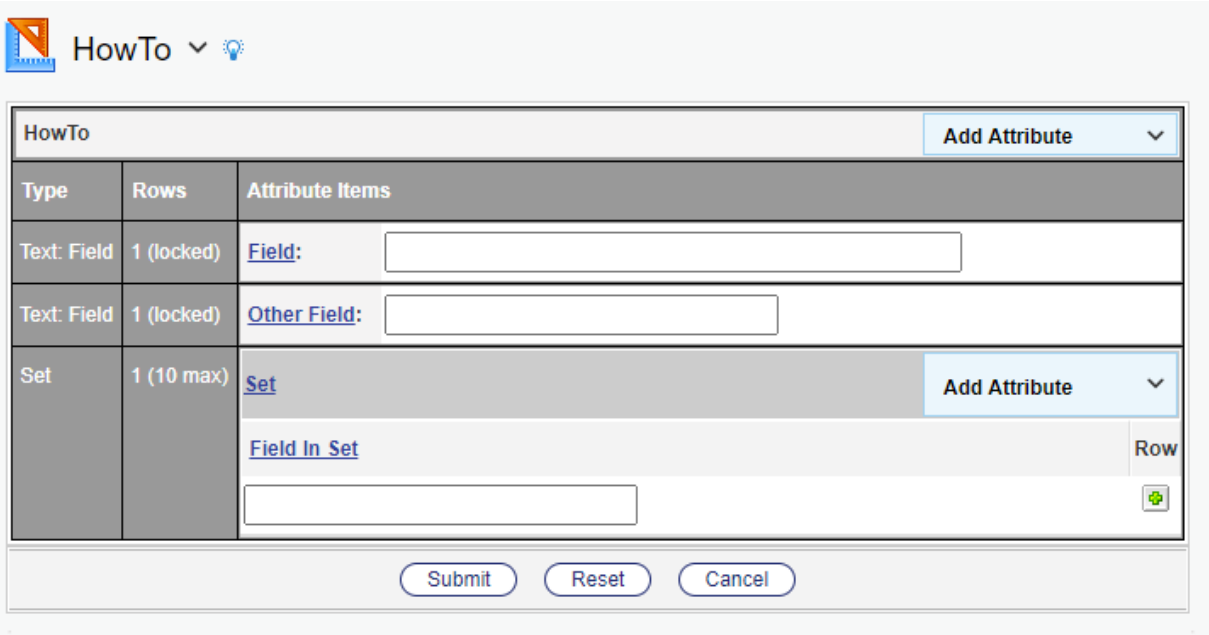
Content Server Forms and Form Templates objects can be manipulated with Content Script through the *forms* service API.
The most important Service API Objects returned by the aformentioned service are: *CSForm*, *CSFormTemplate* and *Form*

While *CSForm* is used to manipulate the Content Server Forms objects (e.g. changing name, applying categories and classifications, changing permissions etc...) the *Form* type is used to represent the data submitted (record) through the form.

Objects used in this paragraph's examples

The examples presented in this paragraph are all making use of a Form Object named **HowTo Form** associated to a FormTemplate object named **HowTo** having the following structure.

Copyright © 2013-2022 AnswerModules Sagl



The FormTemplate object has been configured to be associated to an SQL Table named **Z_HowTo**.
At the time of configuration Content Server produced the following SQL DDL instructions:

```
create table Z_HowTo
(
VolumeID bigint not null,
DataID bigint not null,
VersionNum bigint not null,
Seq bigint null,
RowSeqNum int default 1 not null,
IterationNum int default 1 not null,
Field nvarchar(255) null,
Other_Field nvarchar(255) null
)
/

create index Z_HowTo_Index1
on Z_HowTo ( VolumeID, DataID, VersionNum, Seq )
/

create table Z_HowToSet
(
VolumeID bigint not null,
DataID bigint not null,
VersionNum bigint not null,
Seq bigint null,
SubSeq int null,
RowSeqNum int default 1 not null,
IterationNum int default 1 not null,
Field_In_Set nvarchar(255) null
)
/

create index Z_HowToSet_Index1
on Z_HowToSet ( VolumeID, DataID, VersionNum, Seq )
/
```

Which once executed, resulted in the creation of two tables: **Z_HowTo** and **Z_HowToSet**

Copyright © 2013-2022 AnswerModules Sagl

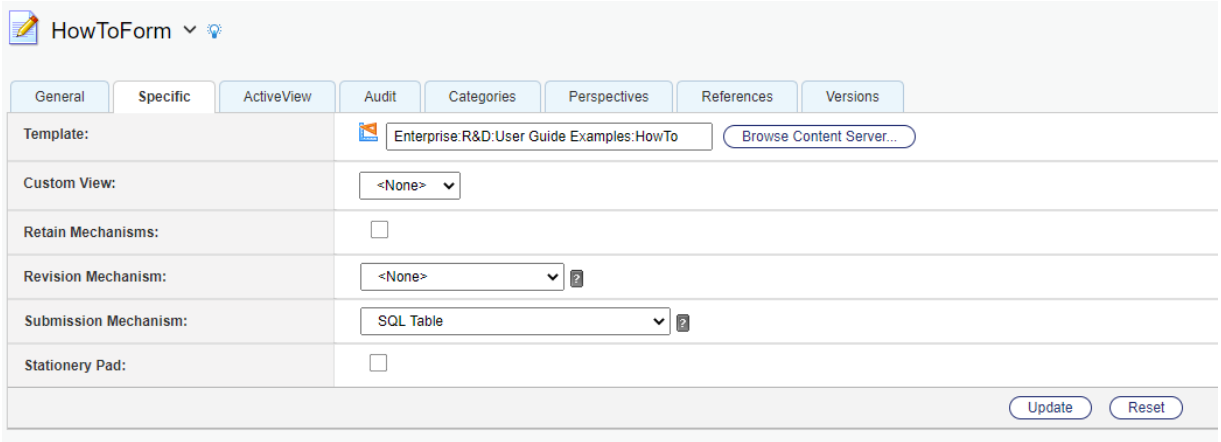
Z_HowTo (otcs)

Column Name	Data Type	Allow Nulls
VolumeID	bigint	☐
DataID	bigint	☐
VersionNum	bigint	☐
Seq	bigint	☑
RowSeqNum	int	☐
IterationNum	int	☐
Field	nvarchar(255)	☑
Other_Field	nvarchar(255)	☑
		☐

Z_HowToSet (otcs)

Column Name	Data Type	Allow Nulls
VolumeID	bigint	☐
DataID	bigint	☐
VersionNum	bigint	☐
Seq	bigint	☑
SubSeq	int	☑
RowSeqNum	int	☐
IterationNum	int	☐
Field_In_Set	nvarchar(255)	☑
		☐

The Form object uses, as a submission mechanism, the SQL Storage option, while no revision mechanism has been associated to it.



Retrive submitted data¶

To get the Content Server Form associated submitted data you can leverage the **listFormData*** APIs, these APIs accept an optional **filters** parameter, which can be used only for Forms having **SQL Table** as associated submission mechanism. Filters are Maps having as keys the names of the tables you want to filter data from and as values a valid SQL **where** clause:

Script

Copyright © 2013-2022 AnswerModules Sagl

```
def writer = new StringWriter()
def html = new MarkupBuilder(writer)
out<< template.evaluateTemplate("#csresource(['bootstrap'])")
html.table(class:"table"){
    thead{
        tr(class:"danger"){
            th("Field")
            th("Other Field")
            th("Set")
        }
    }
    tbody{
        formNode.listFormData(["Z_HowTo":" Seq in (select Seq from Z_HowToSet where Field_In_Se
            tr{
                td(form.field.value)
                td(form.otherField.value)
                td{
                    table(class:"table table-condensed"){
                        thead{
                            tr(class:"danger"){
                                th("Field in Set")
                            }
                        }
                        tbody{
                            form.set.each{ row->
                                tr{
                                    td(row.fieldInSet.value)
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
out << writer.toString()
```

Output

Field	Other Field	Set
one		Field in Set one
two		Field in Set two

```
def formNode = docman.getNodeByName(self.parent, "User Info Form") //returns a CSFormImpl node
def submittedData = formNode.listFormData()
```

Copyright © 2013-2022 AnswerModules Sagl

In the script above *formNode* in CSForm object type that has API implemented to work with Content Server Forms. *submittedData* is a list of `Form` object types that corresponds to certain record of the submitted form data. To access fields of the form:

```
//List sumbitted data
//Access Form fields
submittedData.each {form ->
    out << "User ${form.firstName[0]} ${form.lastName as String}. Age ${form.age as String}"
    out << "<br>"
}
```

In the example above following form attributes are accessed:

Field Name	Normalized
First Name	firstName
Last Name	lastName
Age	age

In scripts, form field values can be accessed using the following notation *form.normalizedname.value*

where normalization is performed by the Content Suite Framework.

```
// Initalize form field values: some examples

form.wordsWithSpaces.value = "TEST VALUE E" // Form template field name: words with spaces

form.camelcase.value = "TEST VALUE D" // Form template field name: camelCase

form.capitalized.value = "TEST VALUE C" // Form template field name: Capitalized

form.uppercase.value = "TEST VALUE B" // Form template field name: UPPERCASE

form.lowercase.value = "TEST VALUE A" // Form template field name: lowercase
```

Also it is possible to represent Form attributed values as a Map. This allows easy access to the form data:

```
out << "List Form data as a Map <br>"

//List all form Records as a Map
submittedData.each {form ->
    out << "<br>"
    out << "${forms.formToMap(form)}"
}
```

Reverse logic is kept as well, meaning Form data cat be set from a Map utilizing `forms.MapToForm(Map map, Form form)`

Copyright © 2013-2022 AnswerModules Sagl

Content Script: Create objects

Coming soon...¶

Training Center

What is it?¶

Module Suite Training Center is a simple Module Suite application that allows you to download and configure on your system a series of simple examples of using the Module Suite. The examples are organized into two main categories: Content Script and Beautiful Webforms and listed in increasing order of complexity.

No Representations or Warranties; Limitations on Liability

The Training Center application (THE APPLICATION) has been created with the sole purpose of showcasing the Module Suite's capabilities. As such, it **should not be utilized in productive environments** and AnswerModules in no way guarantees that included examples are fully functional or free of errors. The information and materials on the Training Center application could include technical inaccuracies or typographical errors. Changes are periodically made to the information contained within it. AnswerModules Sagl MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO ANY INFORMATION, MATERIALS, CODES OR GRAPHICS ON THE APPLICATION, ALL OF WHICH IS PROVIDED ON A STRICTLY "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND AND HEREBY EXPRESSLY DISCLAIMS ALL WARRANTIES WITH REGARD TO ANY INFORMATION, MATERIALS CODES OR GRAPHICS ON THE APPLICATION, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. UNDER NO CIRCUMSTANCES SHALL AnswerModules Sagl BE LIABLE UNDER ANY THEORY OF RECOVERY, AT LAW OR IN EQUITY, FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, SPECIAL, DIRECT, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES (INCLUDING, BUT NOT LIMITED TO LOSS OF USE OR LOST PROFITS), ARISING OUT OF OR IN ANY MANNER CONNECTED WITH THE USE OF INFORMATION OR SERVICE, OR THE FAILURE TO PROVIDE INFORMATION OR SERVICES, FROM THE APPLICATION.

Training Center setup¶

Installing the Training Center application on your system is a straightforward procedure made of just two simple steps.

As administrator

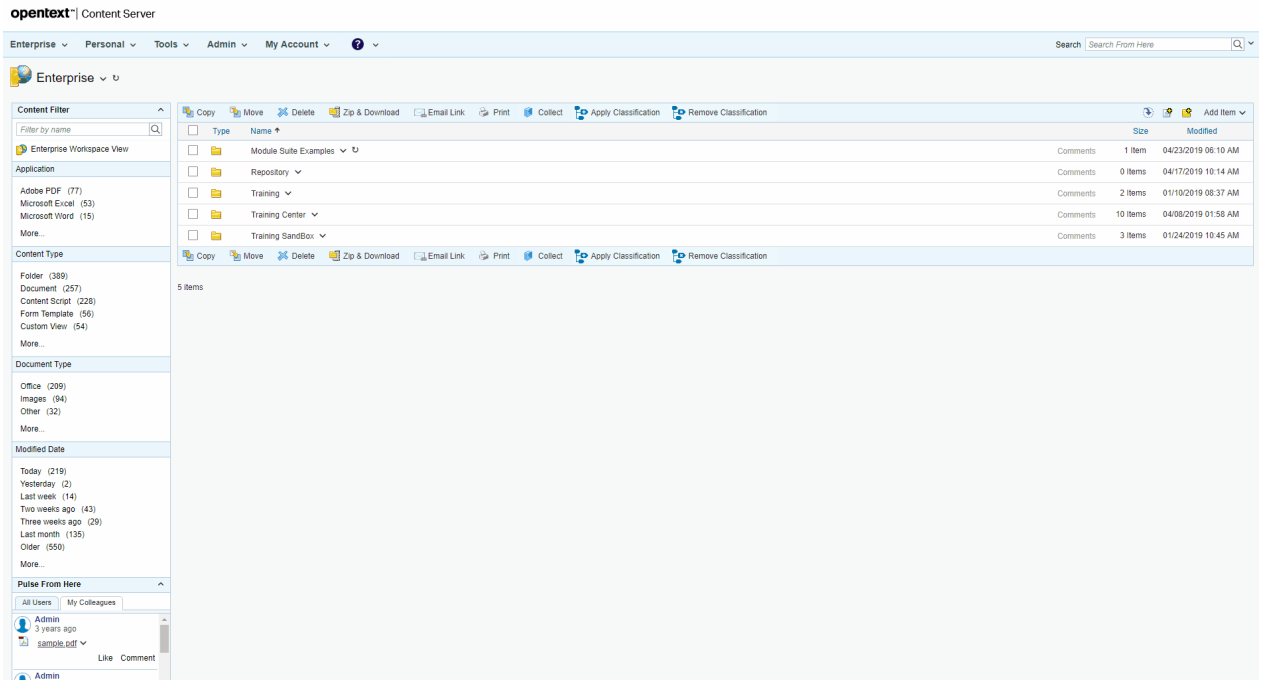
The installation procedure must be performed using a user with administrative rights on the system (for example, the administrator user)

Copyright © 2013-2022 AnswerModules Sagl

First you upload on your Content Server instance the TrainingCenterXML Package (you can download it from here *(/resources/trainingCenterXML.xml)*) than you create a Content Script to import it in the right location (please refer to the snippet below as a reference).

```
def targetSpace =  docman.getNodeByPath(docman.getContentScriptVolume(), "CSTools")
def source = docman.getNodeByName(self.parent, "trainingCenterXML.xml")
admin.importXml(targetSpace, source.content.content)
self.delete()
source.delete()
redirect "${url}/open/${targetSpace.ID}"
```

Once imported the Training Center application can be accessed from the Content Script Volume. The application entry point is a Content Script named "Dashboard" located under: `Content Script Volume:CSTools:Training Center`



Using the tool¶

Internet access required

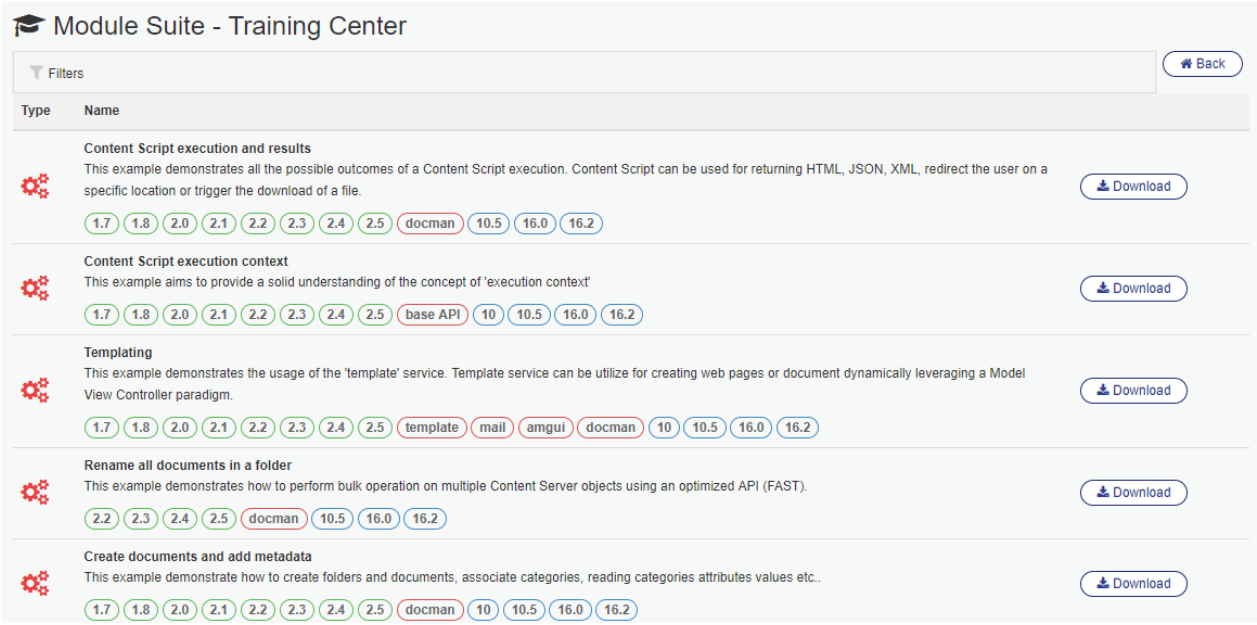
Your browser must have access to the Internet in order to properly execute the application of the Training Center.

As administrator

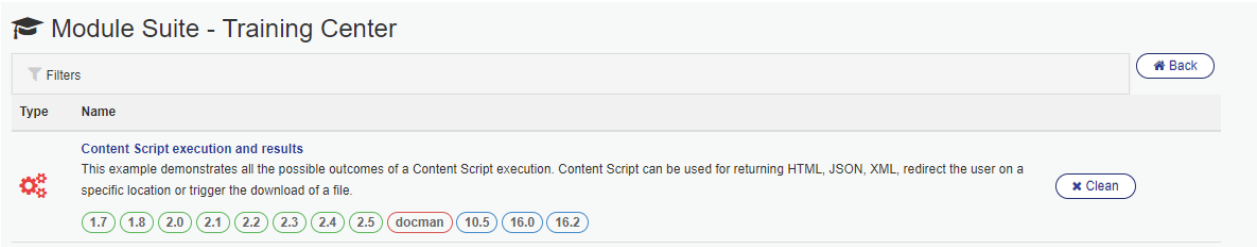
The examples must be imported using a user with administrative rights on the system (for example, the administrator user). Your browser is required to have access to the internet in order to be able to properly run the Training Center application.

Copyright © 2013-2022 AnswerModules Sagl

To download and configure an example on your system just press the "download" button associated with it. The application automatically downloads the required resources from the `developer.answermodules.com` and installs / configures them on your system.



Once imported the example will be available under `Enterprise:Module Suite examples`.



Do not manually delete imported examples

We strongly advise you not to manually delete any imported examples with the Training Center application. If you want to remove the example from your system, press the "clean" button associated with it (the application will perform the necessary cleaning service on your behalf)

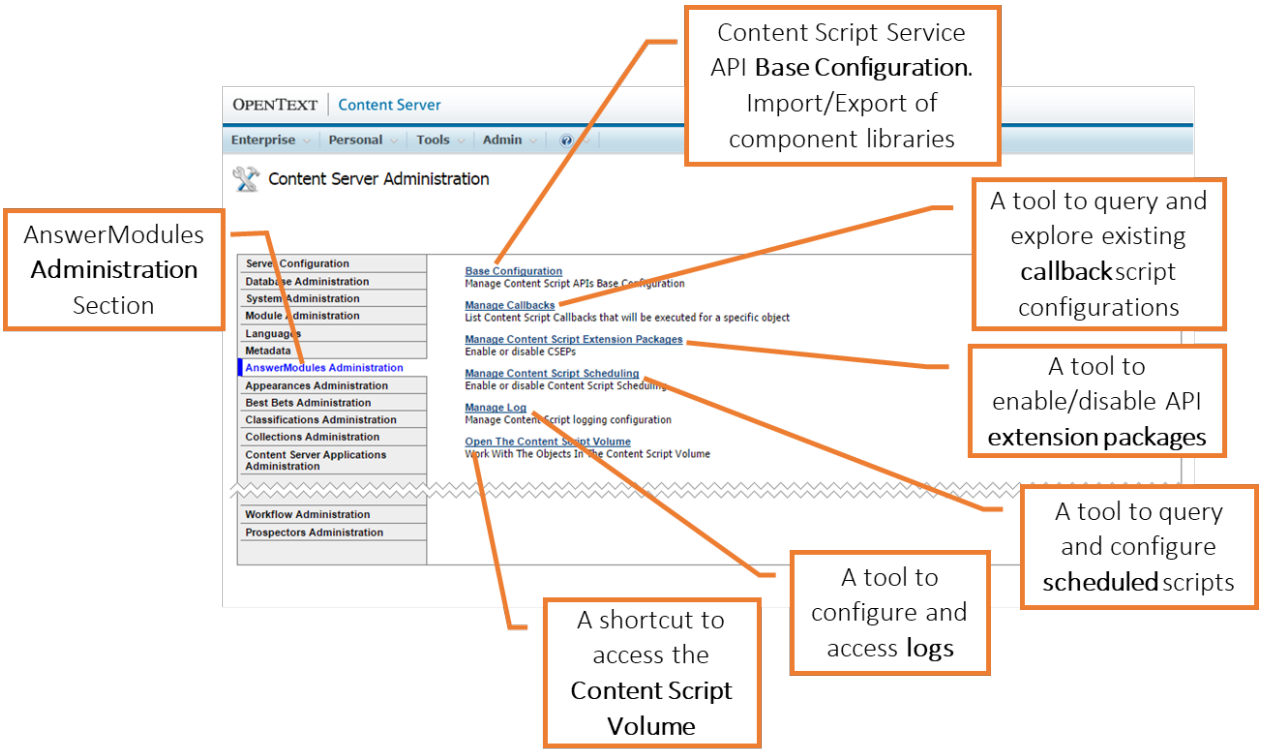
Copyright © 2013-2022 AnswerModules Sagl

Adminisration

Admnistrative pages

Settings and administration tools specific to ModuleSuite components can be accessed from the Content Server Administration pages.

Detailed information related to the single tools and configuration pages is provided in the following sections.



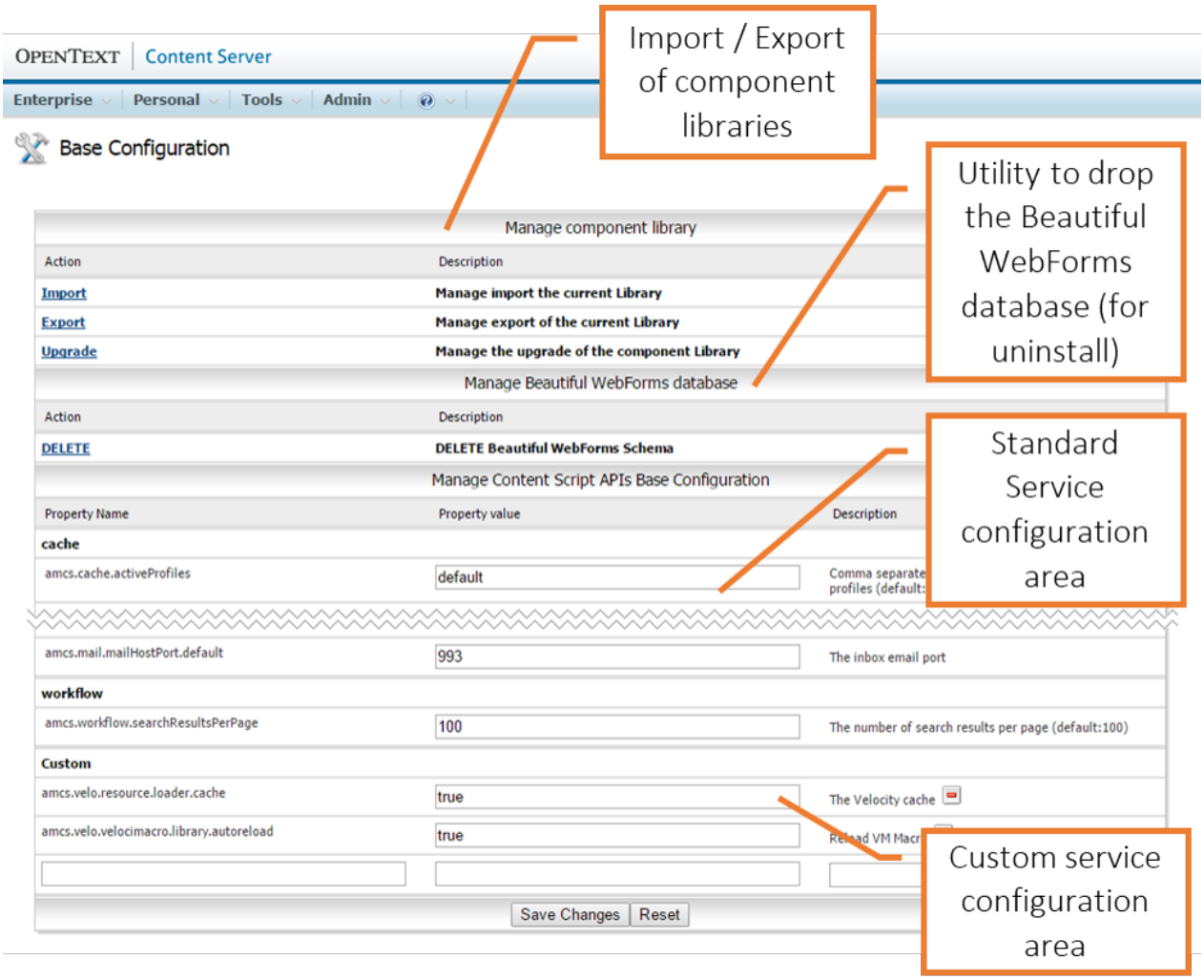
Base Configuration¶

The Base Configuration page provides access to:

- utilities to perform initial import/upgrade/backup of the Content Script Volume contents
- licensing status
- ModuleSuite database maintenance utilities
- global configuration of the Content Script engine, and configuration of the single API services

Copyright © 2013-2022 AnswerModules Sagl

configuration of custom Content Script extension modules



Restart required

Every time changes are made to the Base Configuration, a Content Server restart is required.

Select default IP address¶

It is quite common for Content Server services to be installed on servers that have multiple newtork interfaces associated with different IP addresses.

Sometimes it is desirable to control which interface or IP address Module Suite uses for external communication (for example with the Content Script extension's **csws** service). In such a situation you can use an additional custom configuration parameter in the base configuration to control interface binding. In fact, the Custom property ***amcs.loopback.cIPs*** allows you to bind an IP address to its server host address. Multiple mappings are supported.

Copyright © 2013-2022 AnswerModules Sagl

IP Mapping Configuration

```
hostname.domain.com=192.168.100.100.
```

Keep in mind that IP addresses must be valid and assigned to one of the server's network iterfaces

Enable / Disable Module Suite features¶

The amcs.core.debugEnabled is a "core" configuration flag you can use to customize your Module Suite instance enabling/disabling ModuleSuite core features. Each bit in the mask represent a different feature that can enabled (0) or disabled (1).

Here below a reference for the meaning of each bit in the mask.

Position	Meaning	Valid values	Decimal value
1	RESERVED	0	
2	Enable/Disable Module Suite internal cache (CSVolume, Form Templates, SubViews, Localization etc)	0 (default)= cache enabled, 1=cache disabled	2
3	Callback script execution context mode	0(default)= single execution context for each script of the chain, 1= shared execution context (same for all the scripts in the chain)	4
4	Content Script objects indexing	0(default)= Content Script objects are not indexed by the search engine, 1=Content Script objects are indexed by the search engine	8
5	Track in the audit trail when a Content Script is executed	0(default)= Do not track in the audit trail the execution of Content Scripts, 1=Track in the audit trail the execution of Content Scripts	16
6	Enable/Disable Asynch events management	0(default)= Asynch events management is enabled, 1=Asynch events management is disabled	32

Copyright © 2013-2022 AnswerModules Sagl

Position	Meaning	Valid values	Decimal value
7	Perform the lookup to determined if there are script to be executed asynchronously when the event is raised	0(default)= Any "interesting" event for Asynch events management is tracked in the Distributed Agent queue and the lookup required to determined if there are scripts to be executed is performed later on by the same DA worker that manages script execution, 1=The lookup required to determined if there are scripts to be executed asynchronously given the registered event is executed when the event is raised. The information is passed to the DA queue only if the lookup finds that there are scripts that need to be executed	64
8	Enable the Content Script Sandbox		128
9	Enables the Content Script Sandbox (disabled by default)		256
10	Enables the View Template Cache (The system is no longer going to check for the version of the Beautiful Webforms View Templates associated to the view when a WebForm is rendered)		512

Copyright © 2013-2022 AnswerModules Sagl

Position	Meaning	Valid values	Decimal value
11	For every Content Script, it is possible to define a set of static, precompiled variables whose values will be available when the script is executed. The framework supports the definition of these variables by means of a second script, whose outcome is the data map containing the values. For performance reasons, this second script is executed only when it changes (or when execution is explicitly forced by an editor), and the results are stored as part of the script object. The information is retrieved from the Database upon execution when you execute a subscript using the runCS API or you retrive this information using the getCSVars API on a CSScript object. In situation in which the Database is under stress or the retrieval of this information does not perform as expected it is possible to configure the framework so that this information is cached in memory.	0(default)=cache disabled. 1=cache enabled	1024
12	In environments where ACL evaluation is quite expensive it is possible to change strategy used by the API that fetches the information related to the list of nodes in a container	0(default)=standard strategy (should work well in most of the cases), 1=Alternative strategy (due to complex and convoluted ACLs)	2048

Example of valid configuration values:

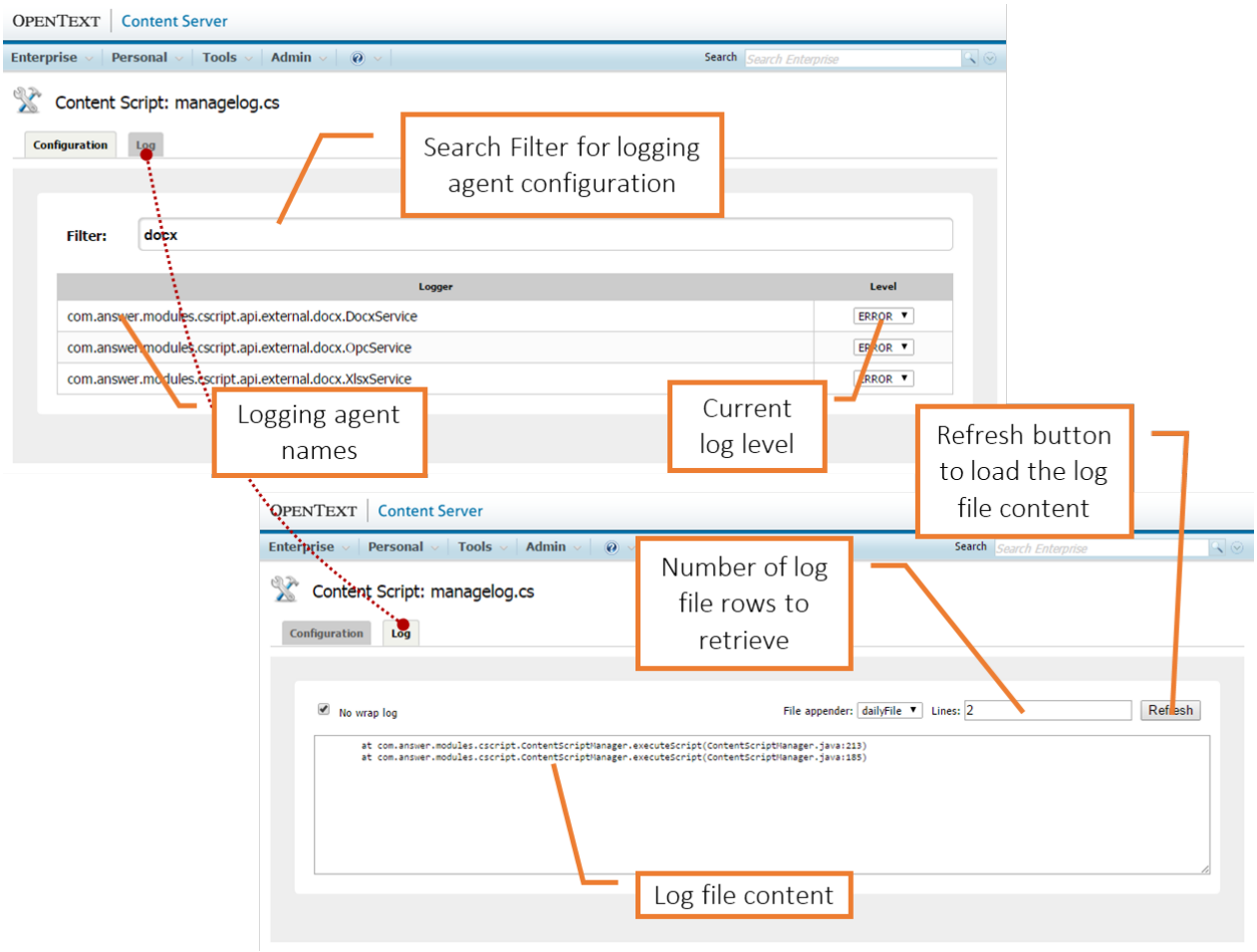
- Enable Content Script indexing while disabling Module Suite cache: 8+2 = 2
- Enable Content Script execution audit trail while disabling Asynch events management: 16 + 32 = 48

Logging administration¶

The Content Script logging page allows administrators to configure the logging level of Content Script services and objects, and also provides quick access to the log file. Logging level can be changed at runtime without restarting the Content Server.

Copyright © 2013-2022 AnswerModules Sagl

Each Content Script is identified by the two capital letters "CS" followed by the Content Script's DataID (e.g. CS12345).



Where is my log ?

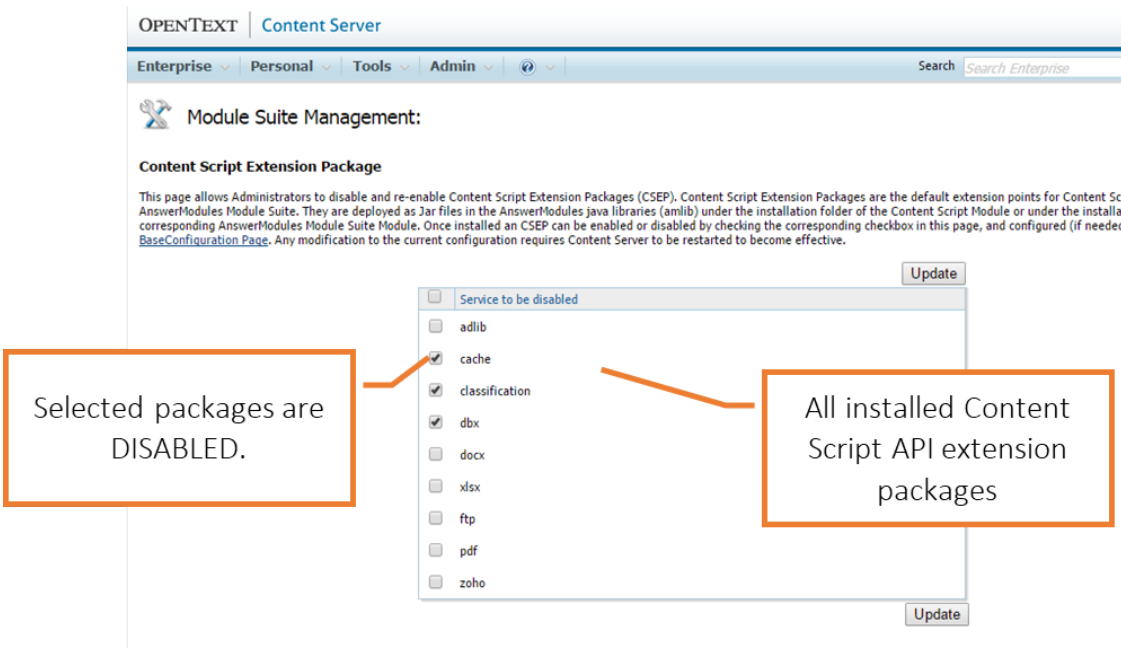
Logging configuration is a server-instance configuration, please keep it in consideration when trying to set the logging level of a scheduled script (in clustered environments), since it could actually be executed on any server on which Distributed Agents are activated.

Manage API Services¶

The Content Script Extension Package management page allows to configure the availability of Content Script API services during script execution.

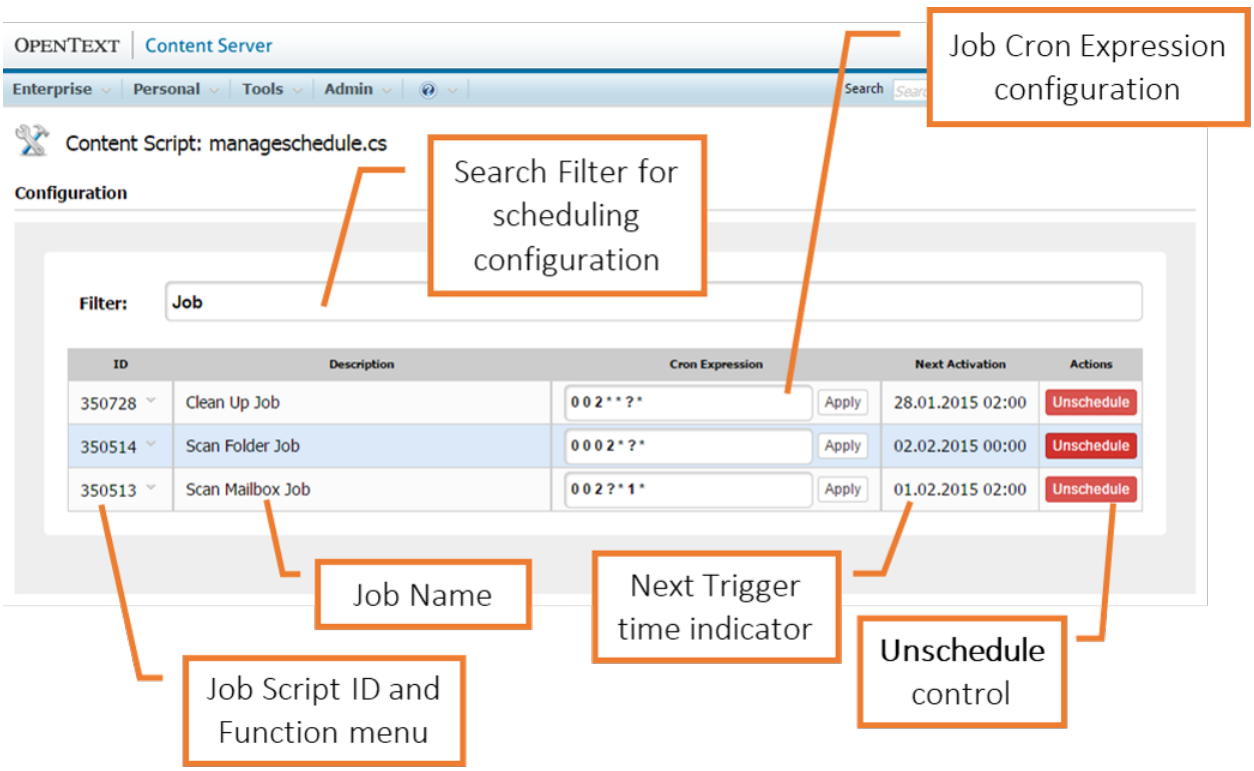
When installed, services are active by default. A Content Server restart is required whenever changes are made to Content Script API extension package availability.

Copyright © 2013-2022 AnswerModules Sagl



Scheduling¶

The Content Script Scheduling administration panel provides a quick overview of the Content Script objects that are queued for scheduled execution, together with the next fire time, the expression used to calculate the execution schedule, and generic information related to the object itself. The object menu allows to easily access the node standard functions.



An **unschedule** utility allows to stop the scheduling of the corresponding script.

Copyright © 2013-2022 AnswerModules Sagl

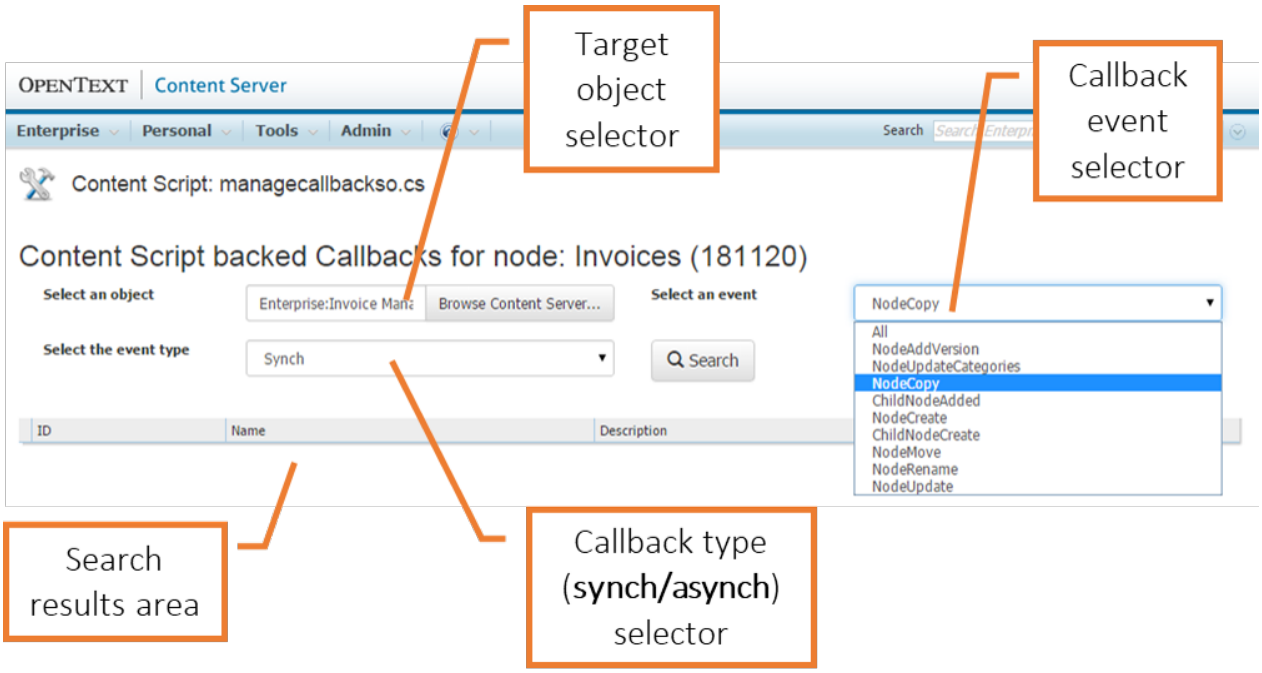
Configuration

The complete set of configuration options for Content Script scheduling (as well as impersonation settings) are available through the Content Script editor Administration *(/working/contentscript/otcsobj/#scheduling)* tab

Manage Callbacks¶

The Callbacks management panel provides a tool to verify in every moment what Content Script callbacks will be executed for specific objects in response to specific event types.

Details on how to configure Content Script Callbacks are provided in the following sections.



Content Script Volume

The **Content Script Volume** is a Content Server volume automatically created upon module installation.

The volume is used to store objects for various purposes. Among others, in the Content Script volume we may find:

- **System Objects:** Objects necessary for the correct execution of different Module Suite components. These objects should not require modification in normal cases.
- **Configuration Objects**: Objects used to configure specific functionalities
    - standard UI customization

Copyright © 2013-2022 AnswerModules Sagl

    - event callback configuration
    -
    - custom column data sources

- **Template Objects**: Various sorts of objects to be used as templates, such as:
    - Content Script code snippets
    - Beautiful WebForms form templates
    - Beautiful WebForms form components
    - HTML view templates
    - ...
- **Service Scripts**: Scripts executed as service endpoints
    - Content Script backing REST services
    - ...

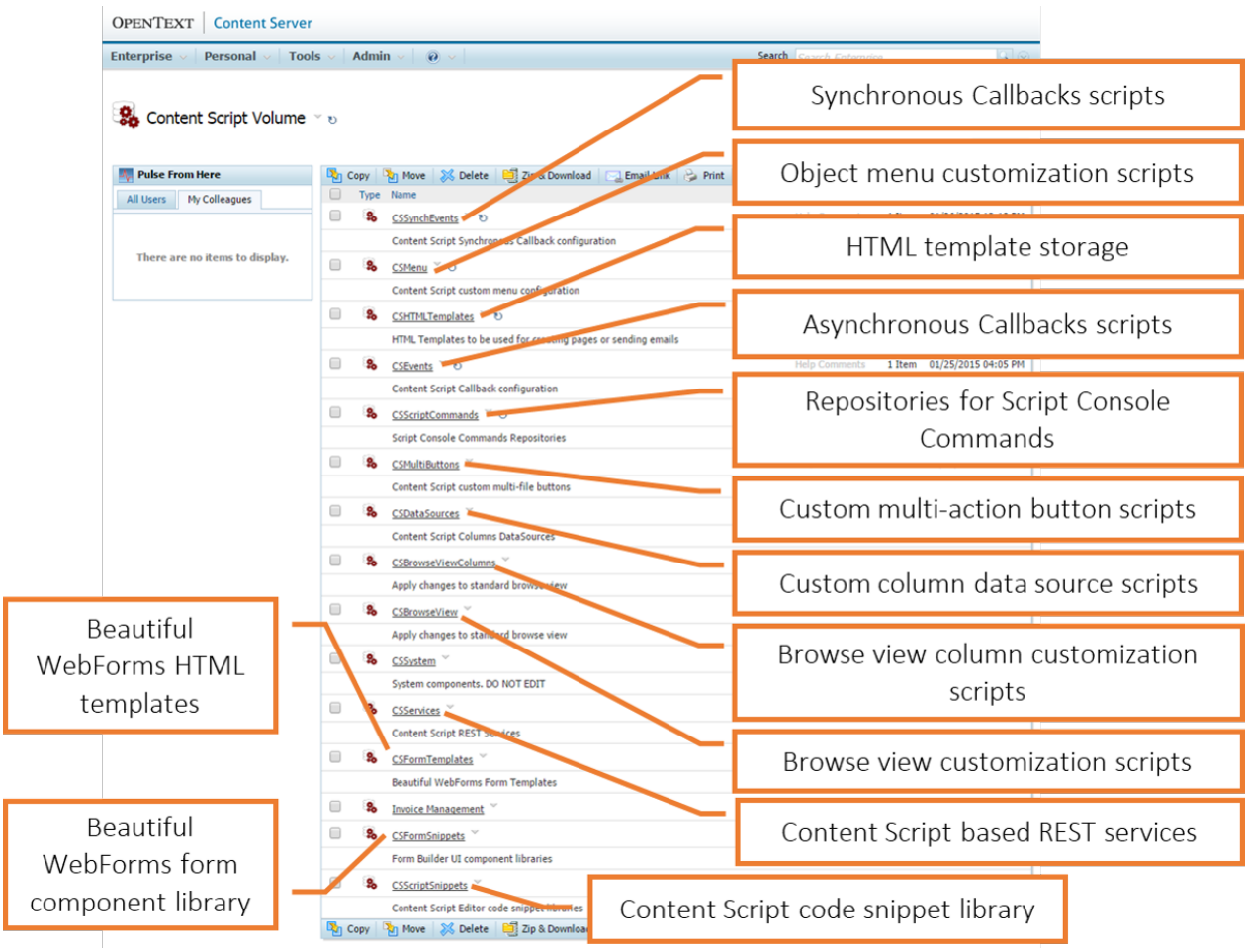
Whenever possible, a **convention-over-configuration** approach is adopted in the Content Script Volume: simply placing a specific object in a specific position will be enough to alter in some way the behavior of some functionalities.

For this reason, a set of **predefined containers** is available in the volume, each one meant for a specific purpose. Here after is a view of the Content Server Volume.

The following sections will explain the purpose of each of the Containers.

> **How should I organize my volume ?**
>
> Even though the Content Script Volume has a predefined container structure, it is not unusual to have custom user data to be stored in the volume. Users are encouraged to use the volume to store custom templates and configurations, for example.

Copyright © 2013-2022 AnswerModules Sagl



CSSystem¶

The **CSSystem** container is dedicated to Module Suite system components. The contents in this location should not require editing except for very specific reasons.

CSFormTemplates¶

This container is dedicated to HTML templates associated to **Beautiful WebForms Views**.

It will be covered in detail in the sections dedicated to Beautiful WebForms *(/working/bwebforms/sdk/#csformtemplates)*.

CSHTMLTemplates¶

The **CSHTMLTemplates** is a container dedicated to general-purpose HTML templates that could be necessary throughout Content Script applications.

As previously seen, Content Script can be used to create various types of output, including web pages and document. Additionally, a few services (such as the **mail** service) can use templates to perform their job.

Copyright © 2013-2022 AnswerModules Sagl

It is usually discouraged to place HTML templating code directly within Scripts: the suggested approach is to separate the presentation templates from the underlying business logic, and to store it somewhere else on Content Server, where it can be reused across applications.

The **CSHTMLTemplates** container is available for developers as a common storage for templates necessary in their applications.

CSFormSnippets¶

The **CSFormSnippets** container is dedicated to the libraries of components that are available to build Beautiful WebForms views.

It will be covered in detail in the sections dedicated to Beautiful WebForms.

CSScriptSnippets¶

The **CSScriptSnippets** container features a two-level structure identical to the one described for the CSFormSnippets container, except that the objects stored here are not form components but Code Snippets to be used to simplify the creation of new scripts in the Content Script Editor.

As for the Form Snippets, new families and components added in this container will automatically be available in the Code Snippet library of the Content Script Editor.

Module Suite components and widgets library ¶

Module Suite's components behaviour and functionalities can be modified and extended by manipulating the content of the Content Script Volume (a Content Server's Volume created when installing the Content Script module).

The purpose of most of the structure and content of the Content Script Volume can be easily understood by simply navigating the volume thanks to the "convention over configuration" paradigm that has been adopted. That means that most of the time, simply creating the right Content Script, Template Folder or Template in the right place will be enough to activate a specific feature. The default configuration (i.e. the default Content Script Volume's structure) should be imported as part of the installation procedure of the Content Script module.

In the next sections we will refer to specific locations in the Content Script Volume content as "Component Library" or simply "Library". This directory contains the default initial version of the Library and will be used later on to manage Library's backups and upgrades. The Library can always be imported, exported or upgraded directly from the Module Suite's administrative pages.

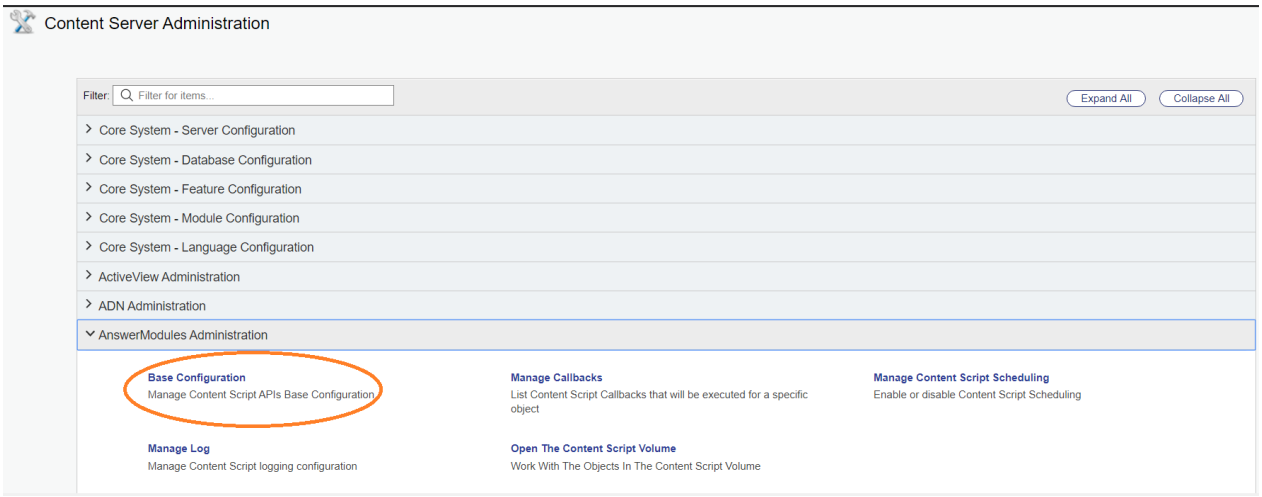
Copyright © 2013-2022 AnswerModules Sagl

Import and upgrade tool¶

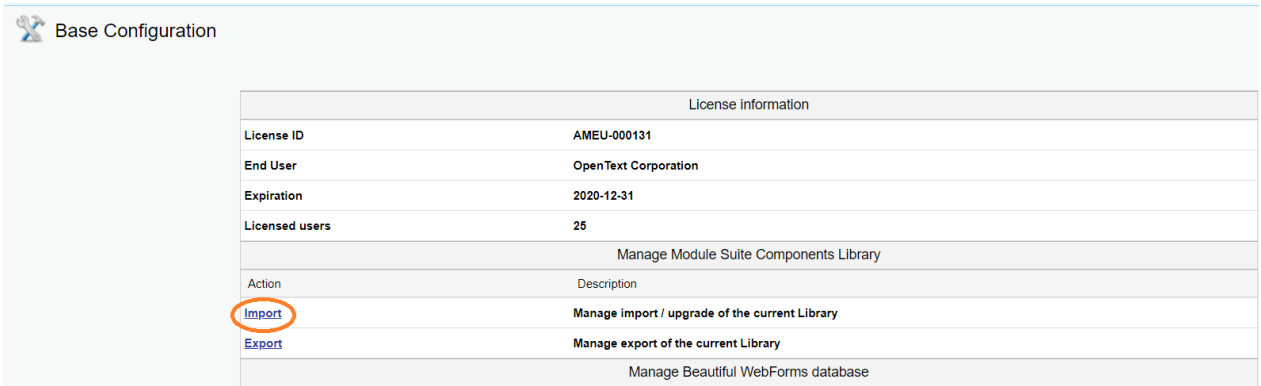
The import and upgrade tool is a Content Script script, shipped with the module and available through the Content Server administrative pages, that will allow you to manage and mantain your Module Suite Library

To open the tool:

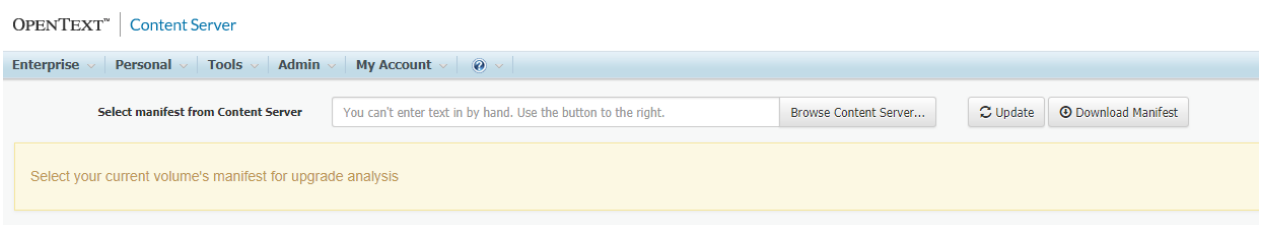
Login as Administrator and access the Module administration panel, then from the Administration Home, select **AnswerModules Administration > Base Configuration**



At the top of the page, click the link **Import - Manage import / upgrade of the current Library**



The import and upgrade tool will be displayed.



Copyright © 2013-2022 AnswerModules Sagl

Load a Library's manifest file¶

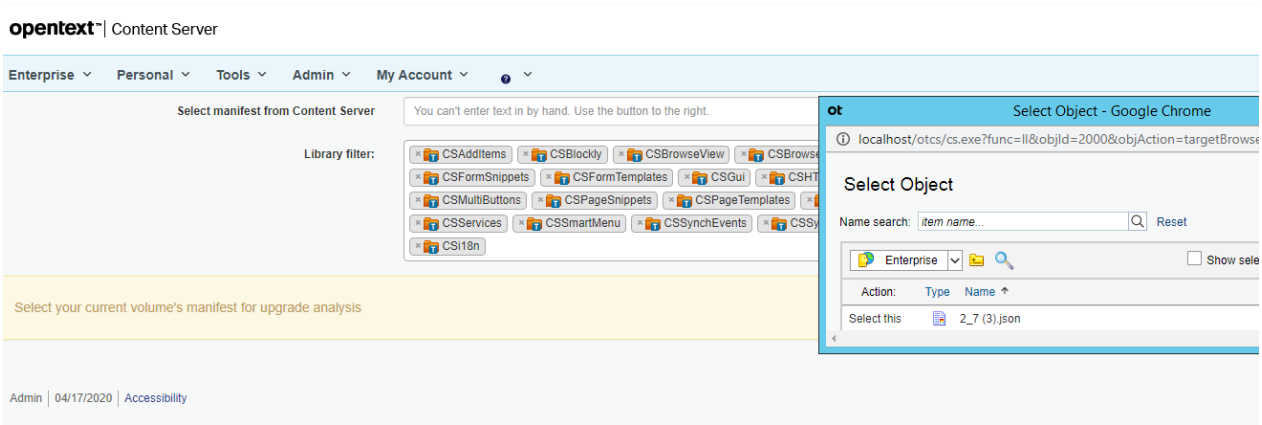
Any operation, such as the import/upgrade of a new library or of a subset of it, should be performed using the provided import and upgrade tool, which will validate it against the relative manifest file.

Module Suite volume manifest file

The manifest file is a JSON file containing information related to the components and widgets that are contained within the newly installed module's library files that will need to be imported or updated.

The manifest file can be generated directly from the import and upgrade tool clicking on the **Generate Manifest** button.

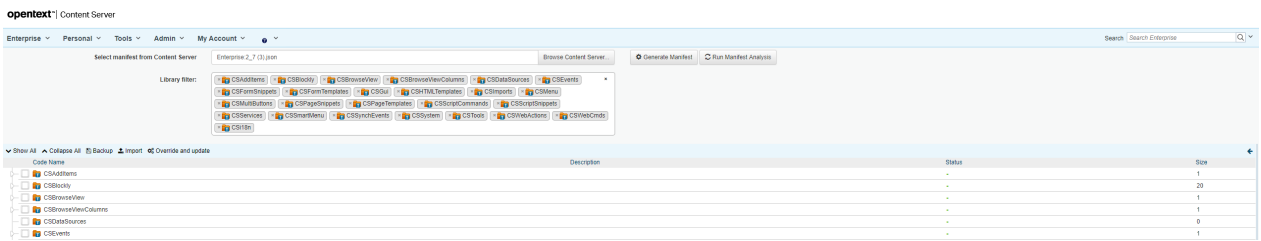
Once generated, the manifest file shall be uploaded in a Content Server's space or folder and loaded from there in the import and upgrade tool using the **Browse Content Server** button.



Analysing the incoming changes and the current Library version¶

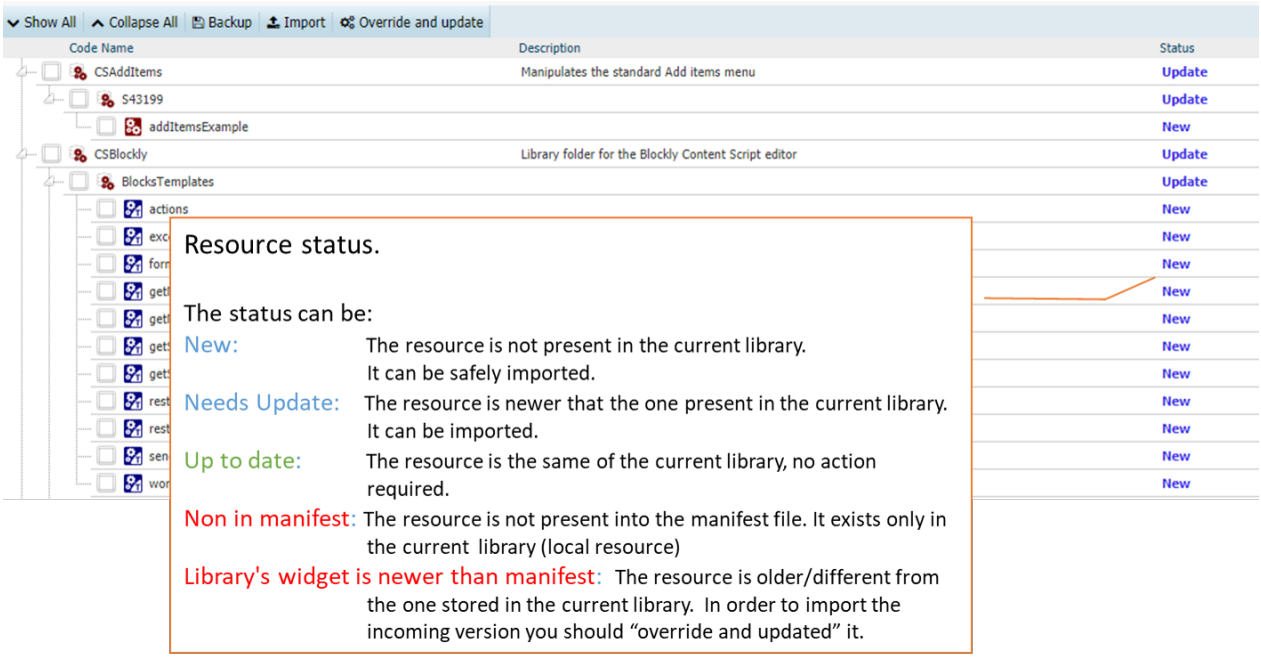
The import and upgrade tool can perform an analysis comparing the incoming library's manifest file with your currently installed library (if any), at the end of the analysis a detailed report will be displayed.

In order to perform the analysis, once the manifest file has been loaded just click on the **Run Manifest Analysis** button.

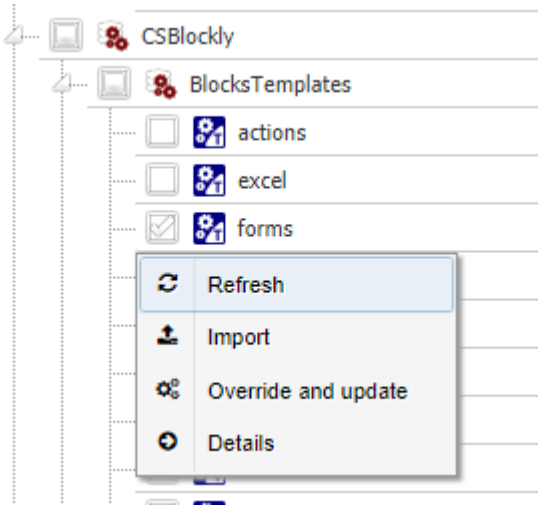


For each resource you can review the analysis status, and take the proper action.

Copyright © 2013-2022 AnswerModules Sagl

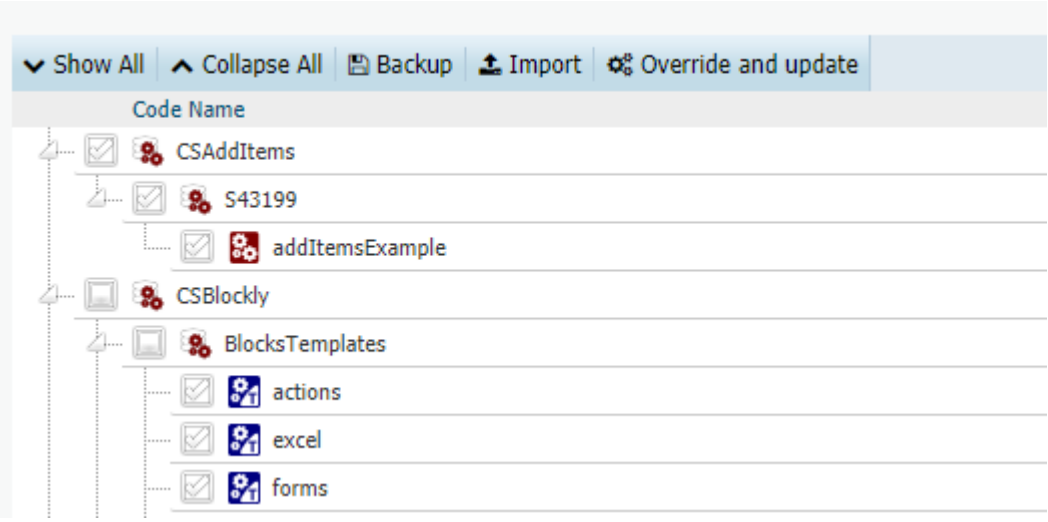


Actions can be taken either one resource at a time, using the contextual menu associated to each one of them:

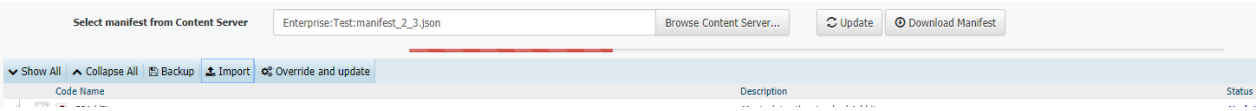


or for all the selected ones at the same time, using the multi-item buttons at the head of the analysis result table:

Copyright © 2013-2022 AnswerModules Sagl



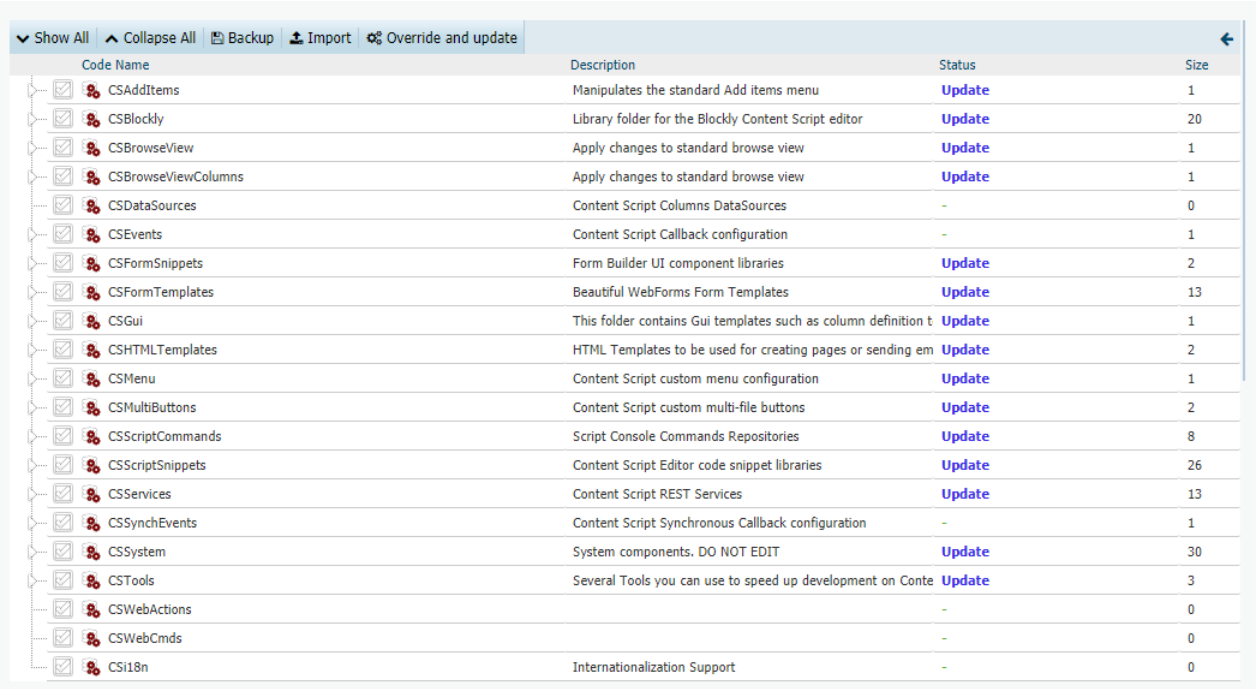
A progress bar will keep you informed regarding the level of completion of the triggered action



Perform the initial library import¶

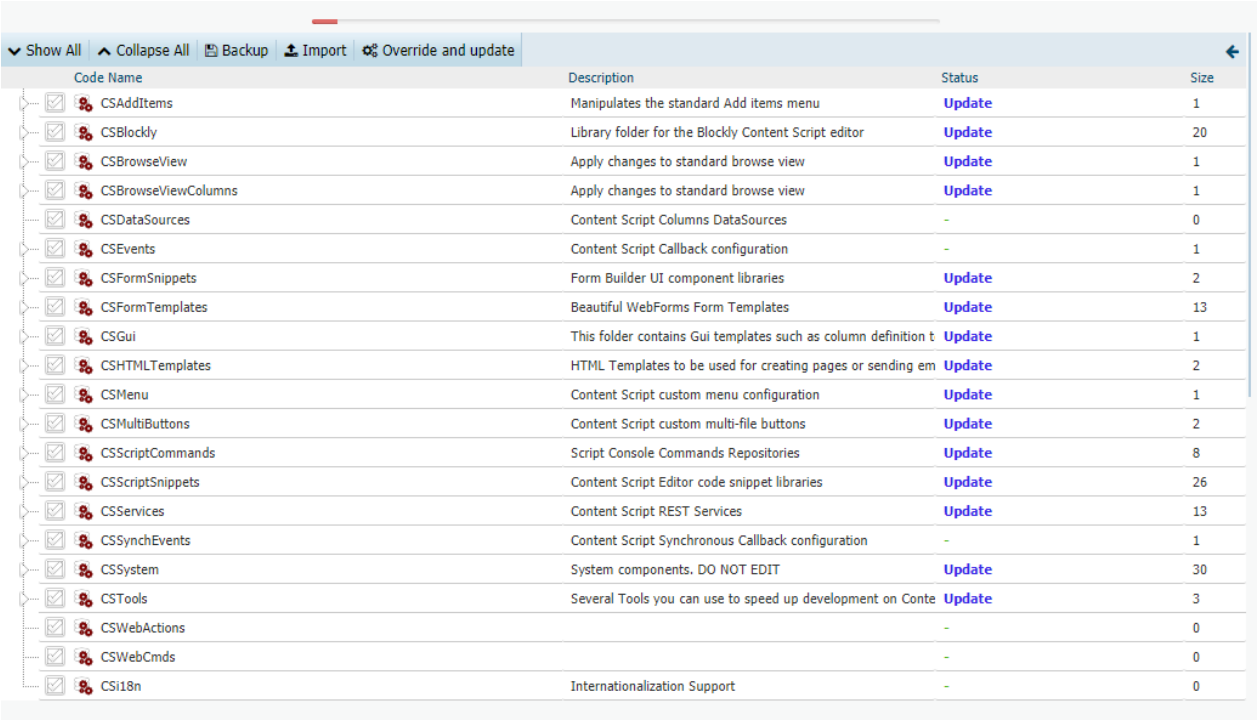
Open the Import and upgrade tool *(/installation/library/#import-and-upgrade-tool)*, load the library manifest file for your ModuleSuite version

Select all the available resources and click on the **Import** multi-item button.



Wait until the progress bar indicator reaches the 100% of completeness

Copyright © 2013-2022 AnswerModules Sagl



Code Name	Description	Status	Size
CSAddItems	Manipulates the standard Add items menu	Update	1
CSBlockly	Library folder for the Blockly Content Script editor	Update	20
CSBrowseView	Apply changes to standard browse view	Update	1
CSBrowseViewColumns	Apply changes to standard browse view	Update	1
CSDataSources	Content Script Columns DataSources	-	0
CSEvents	Content Script Callback configuration	-	1
CSFormSnippets	Form Builder UI component libraries	Update	2
CSFormTemplates	Beautiful WebForms Form Templates	Update	13
CSGui	This folder contains Gui templates such as column definition t	Update	1
CSHTMLTemplates	HTML Templates to be used for creating pages or sending em	Update	2
CSMenu	Content Script custom menu configuration	Update	1
CSMultiButtons	Content Script custom multi-file buttons	Update	2
CSScriptCommands	Script Console Commands Repositories	Update	8
CSScriptSnippets	Content Script Editor code snippet libraries	Update	26
CSServices	Content Script REST Services	Update	13
CSSynchEvents	Content Script Synchronous Callback configuration	-	1
CSSystem	System components. DO NOT EDIT	Update	30
CSTools	Several Tools you can use to speed up development on Conte	Update	3
CSWebActions		-	0
CSWebCmds		-	0
CSi18n	Internationalization Support	-	0

Copyright © 2013-2022 AnswerModules Sagl

Tags

Copyright © 2013-2022 AnswerModules Sagl