



Asbru Ltd.
www.asbrusoft.com
info@asbrusoft.com

Asbru Ltd.



Asbru Web Content Management System

Programming API Guide

*Easily & Inexpensively
Create, Publish & Manage Your Websites*



Copyright and Proprietary Information

Copyright Asbru Ltd 1999–2024. This user guide constitutes proprietary information of Asbru Ltd. No part of this user guide may be reproduced, transmitted, transcribed, stored in a retrieval system or translated into any language in any form, by any means, without the written permission of Asbru Ltd.

Notice

Asbru Ltd. reserves the right to make changes in this user guide at any time and without notice. Asbru Ltd. makes no warranties, express or implied, in this user guide. In no event shall Asbru Ltd. be liable for any indirect, special, incidental or consequential damages arising out of purchase or use of this user guide or the information contained herein.

Licenses and Trademarks

Asbru Web Content Management and the Asbru logo are trademarks or registered trademarks of Asbru Ltd. in the United Kingdom and other countries. All other company, product, or trade names are trademarks or registered trademarks of their respective holders.

Asbru Web Content Management includes and uses the wz_dragdrop.js library, Copyright (c) 2002-2003 Walter Zorn (www.walterzorn.com), licensed under the terms of the GNU Lesser General Public License (LGPL) (<http://www.gnu.org/copyleft/lesser.html>).

Asbru Web Content Management includes and uses the wz_jsgraphics.js library, Copyright (c) 2002-2004 Walter Zorn (<http://www.walterzorn.com>), licensed under the terms of the GNU Lesser General Public License (LGPL) (<http://www.gnu.org/copyleft/lesser.html>).

Asbru Web Content Management includes and uses the Dynarch DHTML Calendar library, Copyright (c) 2002-2005 Mihai Bazon (<http://www.bazon.net/mishoo> - <http://www.dynarch.com/projects/calendar>), licensed under the terms of the GNU Lesser General Public License (LGPL) (<http://www.gnu.org/licenses/lgpl.html>).

Asbru Web Content Management includes and uses the Kryogenix sortable library, Copyright (c) 1997-2005 Stuart Langridge (<http://www.kryogenix.org/code/browser/sortable/>), licensed under the terms of the MIT License (<http://www.kryogenix.org/code/browser/license.html>).

Asbru Web Content Management includes and uses the SWFupload component and library, Copyright (c) 2006-2007 Lars Huring, Olov Nilzén and Mammon Media, and Copyright (c) 2007-2008 Jake Roberts (<http://www.swfupload.org/>), licensed under the terms of the MIT License (<http://www.opensource.org/licenses/mit-license.php>).

Asbru Web Content Management includes and uses the Prototype library, Copyright (c) 2005 Sam Stephenson (<http://prototype.conio.net/>), licensed under the terms of an MIT-style License (<http://prototype.conio.net/>).

Asbru Web Content Management includes and uses the Scriptaculous library, Copyright (c) 2005 Thomas Fuchs (<http://script.aculo.us/>), licensed as free software.

Asbru Web Content Management includes and uses parts of the Rico library, Copyright (c) 2005 Sabre Airline Solutions (<http://openrico.org/>), licensed under the terms of the Apache License, Version 2.0.

Asbru Web Content Management includes and uses the Rico Livegrid Plus library, Copyright (c) 2006 Matt Brown (<http://dowdybrown.com/>), licensed under the terms of the Apache License, Version 2.0.

Asbru Web Content Management includes and uses the Lightbox library, Copyright (c) 2006 Lokesh Dhakar (<http://www.huddletogether.com/>), licensed under the Creative Commons Attribution 2.5 License (<http://creativecommons.org/licenses/by/2.5/>).

Asbru Web Content Management includes and uses the TableKit library, Copyright (c) 2007 Andrew Tetlaw & Millstream Web Software (<http://www.millstream.com.au/view/code/tablekit/>), licensed as free software.

Asbru Web Content Management includes and uses the jQuery library, Copyright (c) 2011 John Resig (<http://jquery.org/>), licensed under the terms of the MIT License (<http://jquery.org/license/>).

Asbru Web Content Management includes and uses the jsTree library, Copyright (c) 2010 Ivan Bozhanov (<http://jstree.com/>), licensed under the terms of the MIT License (<http://www.opensource.org/licenses/mit-license.php>).

Asbru Web Content Management includes and uses the JavaBeans Activation Framework library, Copyright (c) Sun Microsystems (<http://www.sun.com/>), licensed under the terms of the Sun Microsystems, Inc. Binary Code License Agreement.



Asbru Web Content Management includes and uses the JavaMail library, Copyright (c) 2009 Sun Microsystems (<http://www.sun.com/>), licensed under the terms of the Sun Microsystems, Inc. Binary Code License Agreement.

Asbru Web Content Management includes and uses the Apache Jakarta JCS library, Copyright (c) 2001-2007 The Apache Software Foundation (<http://www.apache.org/>), licensed under the terms of the Apache License, Version 2.0.

Asbru Web Content Management includes and uses the Apache Commons Logging library, Copyright (c) 2003-2007 The Apache Software Foundation (<http://www.apache.org/>), licensed under the terms of the Apache License, Version 2.0.

Asbru Web Content Management includes and uses the Apache log4j library, Copyright (c) 2010 The Apache Software Foundation (<http://www.apache.org/>), licensed under the terms of the Apache License, Version 2.0.

Asbru Web Content Management includes and uses the concurrent library, Copyright (c) Doug Lea (<http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html>), licensed as free software.

Asbru Web Content Management includes and uses the slidetabs library, Copyright (c) WebStack (<http://www.slidetabs.com/>).

Asbru Web Content Management includes and uses the ContentBuilder.js library, Copyright (c) InnovaStudio (<http://www.innovastudio.com/>).

Asbru Web Content Management includes and uses the CodeMirror library, Copyright (c) 2017 Marijn Havebeke (marijnh@gmail.com) (<http://www.codemirror.net/>), licensed under the terms of the MIT License (<https://codemirror.net/LICENSE>).

Asbru Web Content Management includes and uses the Tiny Colorpicker library, Copyright (c) 2013 Maarten Baijs (<http://www.baijs.com/>), licensed under the terms of the MIT License.

Asbru Web Content Management includes and uses the HTML5 FormData Polyfill, Copyright (c) 2016 Jimmy Karl Roland Wärtling (<https://github.com/jimmywarting/FormData>), licensed under the terms of the MIT License.

Asbru Ltd

Asbru Ltd. provides Internet/Web services, consultancy and solutions for businesses and individuals. Registered in England - Company Registration No. 3865324 - www.asbrusoft.com



Asbru Web Content Management System

*Easily & Inexpensively
Create, Publish & Manage Your Websites*

Introduction

This document is the Programming API guide for the Asbru Web Content Management System. The user guide describes how you, the website programmer, install, configure and use your own program scripts to be used in and with the Asbru Web Content Management System to create, publish and manage your websites.

This user guide is divided into five main parts:

Part 1 describes how to develop, install and configure your own add-on modules with additional website administration sections, menu items and functionality.

Part 2 describes how to develop, install and configure your own extension program scripts with additional programmed website content for your website pages.

Part 3 describes how to develop, install and configure your own automated product availability and product delivery program scripts for the E-Commerce Add-On Module.

Part 4 describes how to develop, install and configure your own workflow action program scripts for your website content administration workflow processes.

Part 5 describes how to develop, install and configure your own web content editor program scripts for your website content administration editing.

Part 6 describes how to develop, install and configure your own program scripts to customize/extend some of the website administration functionality such as publishing and unpublishing content items; handling of uploaded images and files; validating content details; and validating user account details.

Part 7 describes how to use content from the web content management system in your own custom website content delivery scripts/templates; and with third-party web applications.

Part 8 describes how to develop, install and configure your own program scripts for additional media files cloud storage services.

Part 9 describes how to develop and install your own program scripts to customize/extend the web content management system One-Time Password Login functionality.

Part 10 describes how to access the website content and web content administration functionality through the web content management system “headless” REST API.



Table of Contents

INTRODUCTION	4
TABLE OF CONTENTS	5
1 CUSTOM / THIRD-PARTY ADD-ON MODULES	16
1.1 Installation and Configuration	16
1.2 Development.....	16
1.3 Payment Processing.....	22
2 CUSTOM / THIRD-PARTY EXTENSIONS	23
2.1 Installation and Configuration	23
2.2 Development.....	23
2.3 Usage.....	24
3 PRODUCT AVAILABILITY AND DELIVERY CUSTOM /THIRD-PARTY EXTENSIONS	25
3.1 Product Availability Custom/Third-Party Extensions	25
3.1.1 Installation and Configuration	25
3.1.2 Development.....	25
3.2 Product Delivery Custom/Third-Party Extensions.....	26
3.2.1 Installation and Configuration	26
3.2.2 Development.....	26
3.3 Usage.....	27
4 WORKFLOW ACTION CUSTOM /THIRD-PARTY EXTENSIONS	28
4.1 Installation and Configuration	28
4.2 Development.....	28
4.3 Usage.....	28
5 WEB CONTENT EDITOR CUSTOM /THIRD-PARTY EXTENSIONS	29



5.1	Installation and Configuration	29
5.2	Development.....	29
5.3	Usage.....	29
6	WEBSITE ADMINISTRATION FUNCTIONALITY EXTENSIONS	30
6.1	External Website Publishing/Archiving Programming API.....	30
6.2	External Website User Registration, Subscribe/Unsubscribe Programming API	30
6.3	External Website User Shopcart Order Programming API.....	31
6.4	File Upload Programming API.....	31
6.5	Validate Content Data Programming API	33
6.6	Validate User Data Programming API.....	33
6.7	External Media Digital Asset Management API.....	34
6.7.1	Media Categories	34
6.7.2	Media Files	34
6.7.3	Media Search	35
7	WEB APPLICATION INTEGRATION AND TEMPLATE PROGRAMMING API	36
8	CLOUD DEPLOYMENT AND MEDIA STORAGE	39
8.1	Media Cloud Storage API.....	39
8.2	Cloud Deployment API	40
9	ONE-TIME PASSWORD LOGIN	41
10	HEADLESS REST API.....	42
10.1	General	42
10.1.1	REST API Login and Authorization Access Tokens	42
10.1.1.1	HTTP Authorization Bearer Header.....	42
10.1.1.2	JSON Data Parameter.....	42
10.1.1.3	HTML FORM POST Parameter	42
10.1.1.4	HTTP URL Parameter	43
10.1.1.5	Expiration	43
10.1.2	REST API Methods	43
10.1.3	REST API Request Parameters.....	43
10.1.4	REST API Cookies.....	43
10.1.5	GET Requests with URL Parameters	43



10.1.6	POST, PUT and DELETE Requests with JSON Parameters.....	44
10.1.7	POST, PUT and DELETE Requests with HTML FORM Parameters.....	44
10.2	Website Content and Functionality.....	44
10.2.1	Login.....	44
10.2.2	Page	46
10.2.3	Product Page.....	46
10.2.4	Data Page.....	47
10.2.5	Content Item	47
10.2.6	Data Item	47
10.2.7	Script.....	48
10.2.8	Style Sheet.....	48
10.2.9	Image.....	49
10.2.10	File.....	49
10.2.11	Link.....	49
10.2.12	Search	50
10.2.12.1	Website Content Search	50
10.2.12.1	Content Database Search.....	50
10.2.13	Contact.....	51
10.2.14	Post	51
10.2.14.1	Website Content	51
10.2.14.2	Database Content.....	52
10.2.15	Register User	52
10.2.16	Unregister User.....	53
10.2.17	Subscribe	53
10.2.18	Unsubscribe	54
10.2.19	Shopping Cart.....	54
10.2.20	Heatmap.....	55
10.2.21	Password.....	56
10.2.21.1	Retrieve Password	56
10.2.21.2	Expired Password.....	57
10.2.21.3	Retrieve Superadmin Password.....	57
10.2.22	Personal	58
10.2.22.1	Personal Page	58
10.2.22.2	Personal Page Content and Preferences.....	58
10.2.23	Logout.....	59
10.3	Website Administration Functionality.....	59
10.3.1	Login.....	60
10.3.1.1	Login Settings	60
10.3.2	Content	61
10.3.2.1	List.....	61
10.3.2.1.1	Personal Workspace	62
10.3.2.1.2	Delete Confirmation	63
10.3.2.1.3	Unpublish Confirmation.....	63
10.3.2.1.4	Create Options.....	64
10.3.2.1.5	Archived.....	65
10.3.2.1.6	Scheduled	66
10.3.2.1.7	Archived and Scheduled.....	68
10.3.2.1.8	Livegrid (DEPRECATED).....	69
10.3.2.2	Search.....	70



10.3.2.2.1	Livegrid (DEPRECATED).....	71
10.3.2.3	Search and Replace	72
10.3.2.3.1	Search	72
10.3.2.3.2	Replace	74
10.3.2.4	Read.....	75
10.3.2.5	Create	79
10.3.2.6	Update	80
10.3.2.7	Delete and Unpublish	81
10.3.2.8	Archive	82
10.3.2.9	Checkin	83
10.3.2.10	Checkout.....	83
10.3.2.11	Copy	83
10.3.2.12	Delete (DEPRECATED)	84
10.3.2.13	Move	85
10.3.2.14	Publish.....	86
10.3.2.15	Unpublish (DEPRECATED).....	86
10.3.2.16	Heatmap	87
10.3.2.17	Usersegment.....	88
10.3.2.18	Ustertest.....	88
10.3.2.19	Checkloops	89
10.3.2.20	Checklinks	90
10.3.2.21	Dependencies	90
10.3.2.22	Content Version Device Options.....	91
10.3.2.23	Content Preview Simulator Options.....	92
10.3.2.24	Content Editor Options.....	92
10.3.2.25	Metadata	92
10.3.2.26	Preview.....	93
10.3.2.27	Validate Markup.....	94
10.3.2.28	Stylesheets	95
10.3.2.29	Product Stock	96
10.3.2.30	Workflow Username Options.....	96
10.3.2.31	Structure	96
10.3.2.31.1	List.....	96
10.3.2.31.2	Update	97
10.3.2.32	Create (DEPRECATED)	98
10.3.2.33	Post (DEPRECATED)	98
10.3.2.34	Print	100
10.3.2.35	Count (Contents To Be Deleted)	100
10.3.2.36	Export Static Files	101
10.3.2.37	Folders (File System)	101
10.3.2.38	Import (Text File Contents).....	104
10.3.2.39	Spell Check	104
10.3.3	Data.....	104
10.3.3.1	List.....	104
10.3.3.1.1	Livegrid (DEPRECATED).....	105
10.3.3.2	Search.....	106
10.3.3.3	Read.....	107
10.3.3.4	Create	107
10.3.3.5	Update	109
10.3.3.6	Delete	110
10.3.3.1	Export.....	110



10.3.3.2	Import.....	111
10.3.4	Users.....	111
10.3.4.1	List.....	111
10.3.4.1.1	Livegrid (DEPRECATED).....	112
10.3.4.2	Search.....	113
10.3.4.2.1	Livegrid (DEPRECATED).....	114
10.3.4.3	Read.....	115
10.3.4.4	Create	118
10.3.4.5	Update	118
10.3.4.6	Delete	119
10.3.4.7	Export.....	120
10.3.4.8	Import.....	121
10.3.4.9	Email	121
10.3.4.10	Password Update	123
10.3.4.11	Update All User Passwords	123
10.3.5	Micro-Websites	124
10.3.5.1	List.....	124
10.3.5.2	Read.....	124
10.3.5.3	Create	125
10.3.5.4	Update	126
10.3.5.5	Delete	126
10.3.6	Content Elements.....	126
10.3.6.1	List.....	126
10.3.6.2	Read.....	127
10.3.6.3	Create	128
10.3.6.4	Update	128
10.3.6.5	Delete	129
10.3.7	Content Groups.....	129
10.3.7.1	List.....	129
10.3.7.2	Read.....	130
10.3.7.3	Create	131
10.3.7.4	Update	131
10.3.7.5	Delete	132
10.3.7.6	Exists	133
10.3.8	Content Types.....	133
10.3.8.1	List.....	133
10.3.8.2	Read.....	133
10.3.8.3	Create	134
10.3.8.4	Update	135
10.3.8.5	Delete	136
10.3.8.6	Exists	136
10.3.9	Image Formats	136
10.3.9.1	List.....	136
10.3.9.2	Read.....	137
10.3.9.3	Create	137
10.3.9.4	Update	138
10.3.9.5	Delete	138
10.3.10	Image Groups	138
10.3.10.1	List.....	138
10.3.10.2	Read.....	139
10.3.10.3	Create	140



10.3.10.4	Update	140
10.3.10.5	Delete	141
10.3.10.6	Exists	142
10.3.11	Image Types	142
10.3.11.1	List.....	142
10.3.11.2	Read.....	142
10.3.11.3	Create	143
10.3.11.4	Update	144
10.3.11.5	Delete	144
10.3.11.6	Exists	145
10.3.12	File Formats	145
10.3.12.1	List.....	145
10.3.12.2	Read.....	146
10.3.12.3	Create	146
10.3.12.4	Update	147
10.3.12.5	Delete	147
10.3.13	File Groups	147
10.3.13.1	List.....	147
10.3.13.2	Read.....	148
10.3.13.3	Create	149
10.3.13.4	Update	149
10.3.13.5	Delete	150
10.3.13.6	Exists	151
10.3.14	File Types	151
10.3.14.1	List.....	151
10.3.14.2	Read.....	151
10.3.14.3	Create	152
10.3.14.4	Update	153
10.3.14.5	Delete	153
10.3.14.6	Exists	154
10.3.15	Link Groups	154
10.3.15.1	List.....	154
10.3.15.2	Read.....	155
10.3.15.3	Create	156
10.3.15.4	Update	156
10.3.15.5	Delete	157
10.3.15.6	Exists	157
10.3.16	Link Types	158
10.3.16.1	List.....	158
10.3.16.2	Read.....	158
10.3.16.3	Create	159
10.3.16.4	Update	160
10.3.16.5	Delete	160
10.3.16.6	Exists	161
10.3.17	Content Versions	161
10.3.17.1	List.....	161
10.3.17.2	Read.....	162
10.3.17.3	Create	162
10.3.17.4	Update	162
10.3.17.5	Delete	163
10.3.18	Content Bundles	163



10.3.18.1	List.....	163
10.3.18.2	Read.....	164
10.3.18.3	Update	165
10.3.18.4	Delete	165
10.3.19	Content Packages.....	166
10.3.19.1	List.....	166
10.3.19.2	Read.....	166
10.3.19.3	Update	167
10.3.19.4	Delete	168
10.3.20	Content Metadata (DEPRECATED)	168
10.3.20.1	List (DEPRECATED)	168
10.3.21	User Groups.....	169
10.3.21.1	List.....	169
10.3.21.2	Read.....	169
10.3.21.3	Create	170
10.3.21.4	Update	171
10.3.21.5	Delete	171
10.3.21.6	Exists	172
10.3.22	User Types.....	172
10.3.22.1	List.....	172
10.3.22.2	Read.....	173
10.3.22.3	Create	174
10.3.22.4	Update	174
10.3.22.5	Delete	175
10.3.22.6	Exists	175
10.3.23	User Segments	176
10.3.23.1	List.....	176
10.3.23.2	Read.....	176
10.3.23.3	Create	178
10.3.23.4	Update	179
10.3.23.5	Delete	180
10.3.23.6	Clear User Segments Data.....	180
10.3.24	User Tests	180
10.3.24.1	List.....	180
10.3.24.2	Read.....	181
10.3.24.3	Create	181
10.3.24.4	Update	182
10.3.24.5	Delete	182
10.3.24.6	Exists	183
10.3.24.7	Variants	183
10.3.24.8	Clear User Tests Data.....	184
10.3.25	User Test Results	184
10.3.26	Workflows	185
10.3.26.1	List.....	185
10.3.26.2	Read.....	185
10.3.26.3	Create	186
10.3.26.4	Update	186
10.3.26.5	Delete	187
10.3.26.6	Workflow States	187
10.3.27	Workflow Actions	187
10.3.27.1	List.....	187



10.3.28	Workflow Programs.....	189
10.3.28.1	List.....	189
10.3.29	Comments.....	189
10.3.29.1	List.....	189
10.3.29.2	Read.....	190
10.3.29.3	Create	191
10.3.29.4	Update	191
10.3.29.5	Delete	192
10.3.30	Projects	192
10.3.30.1	List.....	192
10.3.30.1.1	Livegrid (DEPRECATED).....	193
10.3.30.2	Read.....	193
10.3.30.3	Create	195
10.3.30.4	Update	195
10.3.30.5	Delete	196
10.3.31	Project Tasks.....	196
10.3.31.1	List.....	196
10.3.31.2	Read.....	197
10.3.31.3	Create	198
10.3.31.4	Update	199
10.3.31.5	Delete	200
10.3.32	Currencies.....	200
10.3.32.1	List.....	200
10.3.32.2	Read.....	201
10.3.32.3	Create	201
10.3.32.4	Update	202
10.3.32.5	Delete	202
10.3.32.6	Export.....	202
10.3.32.7	Import.....	203
10.3.33	Products	203
10.3.33.1	Export.....	203
10.3.33.2	Import.....	203
10.3.34	Product Groups.....	204
10.3.34.1	List.....	204
10.3.34.2	Read.....	204
10.3.34.3	Create	205
10.3.34.4	Update	206
10.3.34.5	Delete	206
10.3.34.6	Exists	207
10.3.35	Product Types	207
10.3.35.1	List.....	207
10.3.35.2	Read.....	208
10.3.35.3	Create	208
10.3.35.4	Update	209
10.3.35.5	Delete	210
10.3.35.6	Exists	210
10.3.36	Product Availability Programs.....	211
10.3.36.1	List.....	211
10.3.37	Product Delivery Programs.....	211
10.3.37.1	List.....	211
10.3.38	Discounts	212



10.3.38.1	List.....	212
10.3.38.2	Read.....	212
10.3.38.3	Create	213
10.3.38.4	Update	213
10.3.38.5	Delete	214
10.3.39	Shipping.....	214
10.3.39.1	List.....	214
10.3.39.2	Read.....	215
10.3.39.3	Create	216
10.3.39.4	Update	216
10.3.39.5	Delete	217
10.3.40	Tax.....	217
10.3.40.1	List.....	217
10.3.40.2	Read.....	217
10.3.40.3	Create	218
10.3.40.4	Update	219
10.3.40.5	Delete	219
10.3.41	Orders	220
10.3.41.1	List.....	220
10.3.41.1.1	Livegrid (DEPRECATED).....	220
10.3.41.1.2	Delete Confirmation	220
10.3.41.2	Read.....	221
10.3.41.3	Create	222
10.3.41.4	Update	223
10.3.41.5	Delete	223
10.3.41.6	Checkin	224
10.3.41.7	Checkout.....	224
10.3.41.8	Delete (DEPRECATED).....	225
10.3.41.9	Move (workflow)	225
10.3.41.10	Open	226
10.3.41.11	Close.....	226
10.3.41.12	Email	226
10.3.41.13	Export.....	228
10.3.41.14	Workflow Username Options.....	228
10.3.42	Order Items	228
10.3.42.1	List.....	228
10.3.42.2	Read.....	229
10.3.42.3	Create	230
10.3.42.4	Update	231
10.3.42.5	Delete	231
10.3.42.6	Export.....	232
10.3.43	Sales.....	232
10.3.43.1	Sales Reports	232
10.3.43.2	Request Parameters	233
10.3.43.1	Report Data	234
10.3.44	Databases	234
10.3.44.1	List.....	234
10.3.44.2	Read.....	235
10.3.44.3	Create	240
10.3.44.4	Update	241
10.3.44.5	Delete	241



10.3.44.6	Exists	242
10.3.45	Usage Analytics	242
10.3.45.1	Usage Reports	242
10.3.45.2	Request Parameters	244
10.3.45.3	Report Data	245
10.3.45.4	Summarise	246
10.3.46	Usalog Configuration Options	247
10.3.46.1	List – Web Browsers	247
10.3.46.2	List – Web Browser Devices	247
10.3.46.3	List – Web Browser Device Types	247
10.3.46.4	List – Web Browser Device Versions	248
10.3.46.5	List – Web Browser Operating Systems	248
10.3.46.6	List – Search Engines	248
10.3.47	Hosting Clients	249
10.3.47.1	List	249
10.3.47.1.1	Livegrid (DEPRECATED)	249
10.3.47.1.2	Delete Confirmation	249
10.3.47.2	Read	250
10.3.47.3	Create	251
10.3.47.4	Update	251
10.3.47.5	Delete	252
10.3.47.6	Delete (DEPRECATED)	252
10.3.47.7	Move	253
10.3.48	Hosting Licenses	253
10.3.49	Hosting Groups	254
10.3.49.1	List	254
10.3.49.2	Read	254
10.3.49.3	Create	255
10.3.49.4	Update	256
10.3.49.5	Delete	256
10.3.49.6	Exists	257
10.3.50	Hosting Types	257
10.3.50.1	List	257
10.3.50.2	Read	258
10.3.50.3	Create	258
10.3.50.4	Update	259
10.3.50.5	Delete	260
10.3.50.6	Exists	260
10.3.51	Configuration Settings	260
10.3.51.1	Read	260
10.3.51.2	Update	262
10.3.51.3	Systemcheck	262
10.3.51.4	Database Connection(s)	263
10.3.51.5	Database Initialise	264
10.3.51.6	Database Import	265
10.3.51.7	Database Website Import	266
10.3.51.8	Cache	267
10.3.51.8.1	Clear (DEPRECATED)	267
10.3.1	Database	268
10.3.1.1	Upgrade	268
10.3.1.2	Content Dependencies	268



10.3.1.3	Export.....	269
10.3.1.4	Backup.....	271
10.3.1.4.1	List.....	271
10.3.1.4.2	Download.....	272
10.3.1.4.3	Delete.....	272
10.3.1.5	Encrypt (DEPRECATED).....	272
10.3.1.6	Decrypt (DEPRECATED).....	272
10.3.2	Website Administrator Settings.....	273
10.3.2.1	Read.....	273
10.3.2.2	Update.....	275
10.3.3	Superadmin Password.....	276
10.3.3.1	Read.....	276
10.3.4	Logout.....	276
10.3.5	Custom Modules.....	276
10.3.6	Custom Extension Programs.....	279
10.3.6.1	List.....	279
10.3.7	API Programs.....	279
10.3.7.1	List.....	279
10.3.7.2	Media (external).....	279
10.3.8	Publish Scheduled.....	280



1 Custom / Third-Party Add-On Modules

The Asbru Web Content Management system enables you to create your own custom add-on modules and to use third-party developed add-on modules.

Custom / third-party add-on modules can be integrated with the Asbru Web Content Management system administration pages and can have their own administration section and/or add menu-items to the existing administration sections.

Custom /third-party add-on modules can also be payment service provider modules for use with the E-Commerce Add-On module

1.1 Installation and Configuration

To install an additional add-on module the module files must be copied to the web server (as default as a new folder under the "/webadmin/module/" folder).

The Asbru Web Content Management system must be configured to load the additional add-on module by editing the "/webadmin/module/config.xxx" file (where "xxx" is "aspx", "jsp" or "php" depending on which version of the Asbru Web Content Management system you are using) and adding an additional line to the bottom of the file. For example to activate the "example" module add the following lines:

- .NET:
`<!-- #include file="example/config.aspx" -->`
- JSP:
`<%@ include file=" example /config.jsp" %>`
- PHP:
`<?php include " example /config.php"; ?>`

If multiple add-on modules are configured the order of them determines the order their administration sections and menu items are displayed on the web content management system administration pages.

1.2 Development

A custom add-on module can be almost anything:

- Additional functionality tightly integrated with the Asbru Web Content Management system.
- A separate application with its own functionality and administration lightly integrated with the Asbru Web Content Management system for easy access through a single administration system.
- An external web service with its own functionality and administration lightly integrated with the Asbru Web Content Management system for easy access through a single administration system.



The Asbru Web Content Management system defines a simple interface for adding additional sections and/or menu items and/or content administration tabs to the web content management system administration pages.

A custom add-on module must include a module configuration file, which defines if, where and how additional sections and menu items and content administration tabs should be integrated with the web content management system administration pages.

Please see the `"/webadmin/module/example/"` example module for details on a module configuration file. The module configuration file definitions are:

- **module**
The module name and version number.
- **Toolbar section**
 - **moduleToolbarTitle**
The module title to be displayed in the web content management system administration pages toolbar.
 - **moduleToolbarImage**
The module image to be displayed in the web content management system administration pages toolbar.
 - **moduleToolbarLink**
The web address which the module's web content management system administration pages toolbar title and image should link to.
- **Home**
 - **moduleHomeIntroTitle**
The module introduction title to be displayed on the web content management system Home page.
 - **moduleHomeIntroText**
The module introduction text to be displayed on the web content management system Home page.
 - **moduleHomeIntroImage**
The module introduction image to be displayed on the web content management system Home page.
 - **moduleHomeIntroLink**
The web address which the module's Home introduction title, text and image should link to.
 - **moduleHomeMenuTitle**
The module menu item title to be displayed on the web content management system Home page.



- `moduleHomeMenuLink`
The web address which the module's Home menu item should link to.
- Content administration - Website Content
 - `moduleContentIntroTitle`
The module introduction title to be displayed on the web content management system content administration page.
 - `moduleContentIntroText`
The module introduction text to be displayed on the web content management system content administration page.
 - `moduleContentMenuTitle`
The module menu item title to be displayed on the web content management system content administration page.
 - `moduleContentMenuLink`
The web address which the module's content administration menu item should link to.
- Content administration - Website Content – Administration Tab
 - `moduleContentTab`
The module HTML/Javascript id to be used on the web content management system content administration page.
 - `moduleContentTabTitle`
The module title to be displayed on the web content management system content administration tab.
 - `moduleContentTabContent`
The module HTML/text content to be displayed on the web content management system content administration page tab, initially.
 - `moduleContentTabURL`
The web address of the HTML/text content to be loaded and displayed on the web content management system content administration page tab when the tab is selected.
 - `moduleContentTabScript`
The Javascript code to be executed on the web content management system content administration page tab when the tab is selected.
- Library administration - Media Library
 - `moduleLibraryIntroTitle`
The module introduction title to be displayed on the web content management system library administration page.



- moduleLibraryIntroText
The module introduction text to be displayed on the web content management system library administration page.
 - moduleLibraryMenuTitle
The module menu item title to be displayed on the web content management system library administration page.
 - moduleLibraryMenuLink
The web address which the module's library administration menu item should link to.
- E-Commerce administration - Products & Orders (E-Commerce Add-On only)
 - moduleEcommerceIntroTitle
The module introduction title to be displayed on the web content management system e-commerce administration page.
 - moduleEcommerceIntroText
The module introduction text to be displayed on the web content management system e-commerce administration page.
 - moduleEcommerceMenuTitle
The module menu item title to be displayed on the web content management system e-commerce administration page.
 - moduleEcommerceMenuLink
The web address which the module's e-commerce administration menu item should link to.
- Community administration (Community Add-On only) (for future use)
 - moduleCommunityIntroTitle
The module introduction title to be displayed on the web content management system community administration page.
 - moduleCommunityIntroText
The module introduction text to be displayed on the web content management system community administration page.
 - moduleCommunityMenuTitle
The module menu item title to be displayed on the web content management system community administration page.
 - moduleCommunityMenuLink
The web address which the module's community administration menu item should link to.
- Databases administration (Databases Add-On only) (for future use)



- `moduleDatabasesIntroTitle`
The module introduction title to be displayed on the web content management system databases administration page.
 - `moduleDatabasesIntroText`
The module introduction text to be displayed on the web content management system databases administration page.
 - `moduleDatabasesMenuTitle`
The module menu item title to be displayed on the web content management system databases administration page.
 - `moduleDatabasesMenuLink`
The web address which the module's databases administration menu item should link to.
- Statistics administration (Statistics Add-On only) (for future use)
 - `moduleStatisticsIntroTitle`
The module introduction title to be displayed on the web content management system statistics administration page.
 - `moduleStatisticsIntroText`
The module introduction text to be displayed on the web content management system statistics administration page.
 - `moduleStatisticsMenuTitle`
The module menu item title to be displayed on the web content management system statistics administration page.
 - `moduleStatisticsMenuLink`
The web address which the module's statistics administration menu item should link to.
- Users administration - User Database
 - `moduleUsersIntroTitle`
The module introduction title to be displayed on the web content management system users administration page.
 - `moduleUsersIntroText`
The module introduction text to be displayed on the web content management system users administration page.
 - `moduleUsersMenuTitle`
The module menu item title to be displayed on the web content management system users administration page.
 - `moduleUsersMenuLink`
The web address which the module's users administration menu item should link to.



- Configuration
 - `moduleConfigIntroTitle`
The module introduction title to be displayed on the web content management system configuration page.
 - `moduleConfigIntroText`
The module introduction text to be displayed on the web content management system configuration page.
 - `moduleConfigMenuTitle`
The module menu item title to be displayed on the web content management system configuration page.
 - `moduleConfigMenuLink`
The web address which the module's configuration menu item should link to.
- Hosting administration - Hosting Clients (Hosting Edition only)
 - `moduleHostingIntroTitle`
The module introduction title to be displayed on the web content management system hosting administration page.
 - `moduleHostingIntroText`
The module introduction text to be displayed on the web content management system hosting administration page.
 - `moduleHostingMenuTitle`
The module menu item title to be displayed on the web content management system hosting administration page.
 - `moduleHostingMenuLink`
The web address which the module's hosting administration menu item should link to.
- Payment Processing configuration (E-Commerce Add-On module only)
 - `modulePaymentTitle`
The module introduction title to be displayed on the E-Commerce Add-On module Payment Processing configuration page.
 - `modulePaymentOptions`
The payment service provider introduction text and options to be displayed on the E-Commerce Add-On module Payment Processing configuration page. Any number and type of payment processing options can be configured. Please see the included PayPal module for an example.

Please note that all module configuration file definitions must be specified even if they are not used. Set unused module configuration file definitions to "".



1.3 Payment Processing

For payment processing custom / third-party add-on modules, an additional program file must also be provided. For a payment processing custom / third-party add-on module titled “PayPal”, a “/webadmin/module/PayPal/payment.aspx”, “/webadmin/module/PayPal/payment.jsp” or “/webadmin/module/PayPal/payment.php” (depending on your programming language version of the web content management system) program file must exist and handle the actual payment processing.

The order details for completed orders are available to the payment processing module. The payment processing module should use the order details to process or prepare the payment for the order, and return payment instructions or payment confirmation to the E-Commerce Add-On module for display on the Order Completed shopping cart / checkout page. Please see the included PayPal module for an example.



2 Custom / Third-Party Extensions

The Asbru Web Content Management system enables you to create your own custom extensions and to use third-party developed extensions.

Custom / third-party extensions can be integrated with the Asbru Web Content Management system content to include content from other applications and sources than the web content management system.

2.1 Installation and Configuration

To install an additional extension the extension file must be copied to the web server (as default as a new file under the "/webadmin/extension/" folder). No configuration is required.

2.2 Development

A custom extension can be almost anything:

- A simple text or HTML file.
- An ASP, .NET, JSP or PHP program script, which generates content programmatically.
- A .NET, JSP or PHP program script, which reads content from a file or a database.
- A .NET, JSP or PHP program script, which reads content from an external web service.

The custom extension is simply included/executed when its special code is used in the web content management system's content, and the output from the custom extension is included in the web content management system's content.

Please see the "/webadmin/extension/" example extension "hello" for details on a simple extension file. The extension file is simply a .NET, JSP or PHP file depending on which programming language you are using and it must generate some output. Parameters (if any) are passed from the web content management system to the extension as a simple session string variable named "extension":

- .NET - hello.aspx:
`<%@ Page Language="C#" Debug="true" ValidateRequest="false" %>`
`<%= "Hello " + Session["extension"] %>`
- JSP - hello.jsp:
`<%= "Hello " + session.getValue("extension") %>`
- PHP - hello.php:
`<?php echo "Hello " . $_SESSION["extension"] ?>`

In addition to the "extension" session parameter, the extension can use the standard web server request variables etc. such as the .NET "Request", the JSP "request" and the PHP "\$_GET" and "\$_POST".



Typically, custom extensions simply process some input parameters and outputs some content to be displayed as part of the viewed web page.

Optionally, a custom extension can also set a server-side cookie session variable for the website user. To do this the custom extension must output/return a specially formatted session command as its only output: "WCM:SESSION:NAME=VALUE" (replace "NAME" with a session variable name and replace "VALUE" with the session variable value).

Optionally, a custom extension can also redirect (web browser) or forward (web server) the website user to another web page. To do this the custom extension must output/return a specially formatted redirect or forward command as its only output: "WCM:REDIRECT:URL" or "WCM:FORWARD:URL" (replace "URL" with a web address such as "http://www.asbrusoft.com/" or "/index.html").

2.3 Usage

Custom extensions are included in the web content management system content using simple special codes similar to the other special codes used in templates etc:

<code>@@@extension:hello@@@</code>	Defines where the output generated by the extension named "hello" is to be inserted. No parameter is passed to the "hello" extension.
<code>@@@extension:hello(World)@@@</code>	Defines where the output generated by the extension named "hello" is to be inserted. "World" is passed as a parameter to the "hello" extension.



3 Product Availability and Delivery Custom /Third-Party Extensions

The Asbru Web Content Management system E-Commerce Add-On module enables you to create your own product availability and delivery custom extensions and to use third-party developed extensions.

Product availability and delivery custom / third-party extensions can be integrated with the Asbru Web Content Management system E-Commerce Add-On module to include content from other applications and sources than the web content management system.

3.1 Product Availability Custom/Third-Party Extensions

Product availability custom/third-party extensions can be used on product, shopping cart and checkout pages to check if a product is available. For example, to check your own external stock inventory system to see if a product is in stock; or to check a supplier's external stock inventory system to see if a product is in stock and can be ordered; or to check if non-physical products such as usernames, email addresses and Internet domain names are available or have already be registered.

3.1.1 Installation and Configuration

To install an additional product availability extension the extension file must be copied to the web server (as default as a new file under the "/webadmin/productavailability/" folder). No configuration is required.

3.1.2 Development

A product availability custom extension can be almost anything:

- A simple text or HTML file.
- A .NET, JSP or PHP program script, which reads content from a file or a database.
- A .NET, JSP or PHP program script, which reads content from an external web service.

The product availability custom extension is simply included/executed when a product page or a shopping cart/checkout page with products for which the product delivery custom extension is selected, and the output from the product availability custom extension can be included on the page displayed to the customer.

Please see the "/webadmin/productavailability/" example extension "hello" for details on a simple product availability extension file. The extension file is simply a .NET, JSP or PHP file depending on which programming language you are using and it may generate some output:

- .NET - hello.aspx:

```
<% Session("productavailability") = "+1 123 In Stock" %>  
<% Session("productavailability") = "0 Unavailable" %>  
<% Session("productavailability") = "-1 345 Available From Supplier" %>
```



- JSP - hello.jsp:

```
<% session.setValue("productavailability ", "+1 123 In Stock "); %>
<% session.setValue("productavailability ", "0 Unavailable "); %>
<% session.setValue("productavailability ", "-1 345 Available From Supplier "); %>
```
- PHP - hello.php:

```
<?php $_SESSION["productavailability "] = "+1 123 In Stock "; ?>
<?php $_SESSION["productavailability "] = "0 Unavailable "; ?>
<?php $_SESSION["productavailability "] = "-1 345 Available From Supplier "; ?>
```

In addition to the "extension" session parameter, the extension can use the standard web server request variables etc. such as the .NET "Request.QueryString", the JSP "request" and the PHP "\$_GET" and "\$_POST".

The output from the product availability custom extension must be in a specific format – a number followed by a blank followed by some text. The number is not displayed by the web content management system but indicates if a product is available or not:

- A positive number indicates that the product is in stock. The product can be ordered and the customer will be charged for it immediately when the order is placed.
- A negative number indicates that the product is not in stock but that it can still be (back/pre)ordered. The product can be ordered but the customer will not be charged for it immediately when the order is placed (i.e. you will charge the customer when you process the order, manually or through an external system, at a later time).
- The number 0 indicates that the product is unavailable. The product cannot be ordered and it will be removed from the order if an order is placed and the customer will not be charged for it.

The remaining text output will be displayed if/where a @@@availability@@@ special code is used on the product, shopping cart and checkout pages. If the number of products in stock etc. should be displayed then the number should be repeated/included in the text – the first number is not displayed, but only used to indicate the availability.

3.2 Product Delivery Custom/Third-Party Extensions

Product delivery custom/third-party extensions can be used to automatically generate digital products and to update external systems when a product has been ordered. For example, to update your own external stock inventory system; or place an order with a supplier; or to generate/deliver non-physical products such as usernames, email addresses and Internet domain names.

3.2.1 Installation and Configuration

To install an additional product delivery extension the extension file must be copied to the web server (as default as a new file under the "/webadmin/productdelivery/" folder). No configuration is required.

3.2.2 Development

A product delivery custom extension can be almost anything:

- A simple text or HTML file.



- A .NET, JSP or PHP program script, which generates content programmatically.
- A .NET, JSP or PHP program script, which reads content from a file or a database.
- A .NET, JSP or PHP program script, which reads content from an external web service.

The product delivery custom extension is simply included/executed upon successful order and payment completion of a product for which the product delivery custom extension is selected, and the output from the product delivery custom extension can be included in the product delivery page displayed to the customer and/or in the email sent to the customer.

Please see the "/webadmin/productdelivery/" example extension "hello" for details on a simple product delivery extension file. The extension file is simply a .NET, JSP or PHP file depending on which programming language you are using and it may generate some output:

- .NET - hello.aspx:
`<% Session("productdelivery") = "Hello " %>`
- JSP - hello.jsp:
`<% session.setValue("productdelivery", "Hello"); %>`
- PHP - hello.php:
`<?php $_SESSION["productdelivery"] = "Hello"; ?>`

In addition to the "extension" session parameter, the extension can use the standard web server request variables etc. such as the .NET "Request.QueryString", the JSP "request" and the PHP "\$_GET" and "\$_POST".

3.3 Usage

The product availability/delivery custom extensions are selected in the web content management system through the Custom Extension Program input field on the Product Delivery tab when adding and updating products in the Products & Orders administration section of the web content management system.



4 Workflow Action Custom /Third-Party Extensions

The Asbru Web Content Management system enables you to create your own workflow action custom extensions and to use third-party developed extensions.

Workflow action custom / third-party extensions can be integrated with the Asbru Web Content Management system to make and log content administration actions through other applications and sources than the web content management system.

4.1 Installation and Configuration

To install an additional workflow action extension the extension file must be copied to the web server (as default as a new file under the "/webadmin/workflowaction/" folder). No configuration is required.

4.2 Development

A workflow action custom extension can be almost anything:

- A .NET, JSP or PHP program script, which reads/writes content from/to a file or a database.
- A .NET, JSP or PHP program script, which reads/writes content from/to an external web service.

The workflow action custom extension is simply executed when a workflow action for which the workflow action custom extension is configured is applied to a content item or an e-commerce module order in the web content management system.

Please see the "/webadmin/workflowaction/" example extensions "Workflow Action Module Template" and "Test Workflow Action" for details on a simple workflow action extension file. The "Workflow Action Module Template" simply shows how to read the relevant data from the web content management system. The "Test Workflow Action" logs the workflow action taken to the content item's Revision History data.

The extension file is simply a .NET, JSP or PHP file depending on which programming language you are using and it should read and eventually update the relevant workflow and content or order data from the web content management system as shown in the example extensions.

4.3 Usage

The workflow action custom extensions are selected in the web content management system through the Content Changes - Custom Extension Program input field when adding and updating workflow actions in the Configuration administration section of the web content management system. The workflow action custom extension is then automatically executed when the workflow action is applied to a content item or an e-commerce module order in the web content management system.



5 Web Content Editor Custom /Third-Party Extensions

The Asbru Web Content Management system enables you to create your own web content editor custom extensions and to use third-party developed extensions.

Web content editor custom / third-party extensions can be integrated with the Asbru Web Content Management system to use another web content editor than the Asbru Web Content Editor included with the web content management system.

5.1 Installation and Configuration

To install an additional web content editor extension the extension file must be copied to the web server (as default as a new file under the "/webadmin/webeditors/EXTENSION NAME/" folder). No configuration is required.

5.2 Development

A web content editor custom extension is a .NET, JSP or PHP program script, which defines the Javascript functions required and used for the content administration by the web content management system.

The web content editor custom extension is simply included when a content item or an e-commerce module product is added or updated through the administration pages in the web content management system.

Please see the "/webadmin/webeditors/" example "WebEditorAPI" extension (as well as the default Asbru Web Content Editor "HardCore", "HardCore1", "HardCore2" and "textarea" extensions) for details on web content editor extension files.

The extension file is simply a .NET, JSP or PHP file depending on which programming language you are using and it should define the Javascript functions required and used for the content administration by the web content management system as well as load other program scripts required by the custom / third-party web content editor.

5.3 Usage

The web content editor custom extension is selected in the web content management system through the Configuration / Features / Content Editing / Content Editor administration page in the web content management system. The web content editor custom extension is then automatically included when a content item or an e-commerce module order is added or updated in the web content management system.



6 Website Administration Functionality Extensions

The Asbru Web Content Management system supports customization/extension of some of the website administration functionality through your own custom “/webadmin/api/” program scripts when certain functionality on your website and on your web content management administration pages is used.

Please see the “/webadmin/api/EXAMPLE.xxxxx.xxx” program scripts for basic program examples/templates.

6.1 External Website Publishing/Archiving Programming API

As default the Asbru Web Content Management system runs on the actual website and delivers the website content dynamically. However, for special requirements you may want to program your own program scripts to be executed when a content item is published or unpublished in the web content management system - for example to copy the file to another web server or an archive/backup server.

When a new or updated content item with a “static filename” is published the web content management system will check if a “/webadmin/api/published”, “/webadmin/api/published.bat”, “/webadmin/api/published.sh” or “/webadmin/api/published.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”) file exists and execute it with the published content item’s static filename as parameter. Any “/webadmin/api/published/xxxxx.bat|sh|aspx|jsp|php” program scripts will also be executed.

When a content item with a “static filename” is unpublished the web content management system will check if a “/webadmin/api/unpublished”, “/webadmin/api/unpublished.bat”, “/webadmin/api/unpublished.sh” or “/webadmin/api/unpublished.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”) file exists and execute it with the published content item’s static filename as parameter. Any “/webadmin/api/unpublished/xxxxx.bat|sh|aspx|jsp|php” program scripts will also be executed.

6.2 External Website User Registration, Subscribe/Unsubscribe Programming API

As default the Asbru Web Content Management system handles user registration and subscribe/unsubscribe functionality. However, for special requirements you may want to program your own program scripts to be executed when a new user registers or subscribes to /unsubscribes from a user group/type in the web content management system - for example to notify third-party systems.

When a new user registers on the website the web content management system will check if a “/webadmin/api/register”, “/webadmin/api/register.bat”, “/webadmin/api/register.sh” or “/webadmin/api/register.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”) file exists and execute it with the registered user’s username, password, email, name, scheduled activation, scheduled notification, scheduled expiration as parameters. Any “/webadmin/api/register/xxxxx.bat|sh|aspx|jsp|php” program scripts will also be executed.



When a website user subscribes to a user group/type on the website the web content management system will check if a “/webadmin/api/subscribe”, “/webadmin/api/subscribe.bat”, “/webadmin/api/subscribe.sh” or “/webadmin/api/subscribe.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”) file exists and execute it with the user’s username, “usergroup” or “usertype”, user group/type name as parameters. Any “/webadmin/api/subscribe/xxxxx.bat|sh|aspx|jsp|php” program scripts will also be executed.

When a website user unsubscribes from a user group/type on the website the web content management system will check if a “/webadmin/api/unsubscribe”, “/webadmin/api/unsubscribe.bat”, “/webadmin/api/unsubscribe.sh” or “/webadmin/api/unsubscribe.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”) file exists and execute it with the user’s username, “usergroup” or “usertype”, user group/type name as parameters. Any “/webadmin/api/unsubscribe/xxxxx.bat|sh|aspx|jsp|php” program scripts will also be executed.

6.3 External Website User Shopcart Order Programming API

As default the Asbru Web Content Management system handles user shopcart ordering functionality. However, for special requirements you may want to program your own program scripts to be executed when a user completes a shopcart order in the web content management system - for example to notify third-party systems.

When a user completes a shopcart order on the website the web content management system will check if a “/webadmin/api/shopcart_complete”, “/webadmin/api/shopcart_complete.bat”, “/webadmin/api/shopcart_complete.sh” or “/webadmin/api/shopcart_complete.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”) file exists and execute it with the order id and username as parameters. Any “/webadmin/api/shopcart_complete/xxxxx.bat|sh|aspx|jsp|php” program scripts will also be executed.

6.4 File Upload Programming API

As default the Asbru Web Content Management system simply adds uploaded images and other files as content items in the web content management system. However, for special requirements you may want to program your own program scripts to be executed when an “image” or a “file” is uploaded to the web content management system – for example to check files for virus infections or to convert the files to other formats or sizes.

When an “image” or a “file” is uploaded the web content management system will check if a “/webadmin/api/image”, “/webadmin/api/image.bat”, “/webadmin/api/image.sh”, “/webadmin/api/image.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”), “/webadmin/api/file”, “/webadmin/api/file.bat”, “/webadmin/api/file.sh” or “/webadmin/api/file.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”) file exists and execute it with the uploaded file’s filename as parameter. Depending on what the program script does and what the web content management system should do the program script must return/output:

- The same filename as passed to the program script as a parameter
If the program script has not renamed, moved or deleted the uploaded file.



- The uploaded file's new filename
If the program script has renamed or moved the uploaded file. The web content management system will then update the content item with the new filename.
- Nothing
If the program script has deleted the uploaded file. The web content management system will then also delete the content item.

Any `"/webadmin/api/image/xxxxx.bat|sh|aspx|jsp|php"` and
`"/webadmin/api/file/xxxxx.bat|sh|aspx|jsp|php"` program scripts will also be executed.

When an "image" or a "file" is uploaded the web content management system will also check if a `"/webadmin/api/image1"`, `"/webadmin/api/image1.bat"`, `"/webadmin/api/image1.sh"`, `"/webadmin/api/image1.xxx"` (replace "xxx" with your programming language extensions: "aspx", "jsp" or "php"), `"/webadmin/api/image2"`, `"/webadmin/api/image2.bat"`, `"/webadmin/api/image2.sh"`, `"/webadmin/api/image2.xxx"` (replace "xxx" with your programming language extensions: "aspx", "jsp" or "php"), `"/webadmin/api/image3"`, `"/webadmin/api/image3.bat"`, `"/webadmin/api/image3.sh"`, `"/webadmin/api/image3.xxx"` (replace "xxx" with your programming language extensions: "aspx", "jsp" or "php"), `"/webadmin/api/file1"`, `"/webadmin/api/file1.bat"`, `"/webadmin/api/file1.sh"`, `"/webadmin/api/file1.xxx"` (replace "xxx" with your programming language extensions: "aspx", "jsp" or "php"), `"/webadmin/api/file2"`, `"/webadmin/api/file2.bat"`, `"/webadmin/api/file2.sh"`, `"/webadmin/api/file2.xxx"` (replace "xxx" with your programming language extensions: "aspx", "jsp" or "php"), `"/webadmin/api/file3"`, `"/webadmin/api/file3.bat"`, `"/webadmin/api/file3.sh"` or `"/webadmin/api/file3.xxx"` (replace "xxx" with your programming language extensions: "aspx", "jsp" or "php") file exists and execute it with the uploaded file's filename as parameter. Depending on what the program script does and what the web content management system should do the program script must return/output:

- The filename of new, alternative copy of the uploaded file
If the program script has created a new, alternative copy of the uploaded file – for example a small resolution version of an image, or a PDF version of a Microsoft Word document, or a compressed version of a program file etc. The web content management system will then create an additional content item for the new file. The original uploaded file's content item's corresponding Additional Content / Image 1 / Image 2 / Image 3 / File 1 / File 2 / File 3 attribute will point to the new, alternative file's content item. The new, alternative file's content item's Content Relations / Page Up attribute will point to the original uploaded file's content item.
- Nothing
If the program script has not created a new, alternative copy of the uploaded file. No additional content item will be created by the web content management system.

Any `"/webadmin/api/image1/xxxxx.bat|sh|aspx|jsp|php"`,
`"/webadmin/api/image2/xxxxx.bat|sh|aspx|jsp|php"`,
`"/webadmin/api/image3/xxxxx.bat|sh|aspx|jsp|php"`,
`"/webadmin/api/file1/xxxxx.bat|sh|aspx|jsp|php"`,
`"/webadmin/api/file2/xxxxx.bat|sh|aspx|jsp|php"` and
`"/webadmin/api/file3/xxxxx.bat|sh|aspx|jsp|php"` program scripts will also be executed.



6.5 Validate Content Data Programming API

If you have special requirements for the website content you can program your own program scripts to validate content when/before it is saved to the web content management system. If a `"/webadmin/api/validatecontent.xxx"` (replace `"xxx"` with your programming language extensions: `"aspx"`, `"jsp"` or `"php"`) program script exists then that will be executed when an added or updated content item is saved. The content item's data will be posted to the program script as standard HTML POST form data.

The posted content item data can then be validated and the program script should return a structured response to the web content management system:

- **"OK"**
The content item data are ok and the content item will be saved.
- **"OK:ALERT:MESSAGE"**
The content item data are ok and the content item will be saved and the **"MESSAGE"** will be displayed to the website administrator (replace **"MESSAGE"** with your own text).
- **"ERROR:CONFIRM:MESSAGE"**
There is a potential problem with the content item data and the website administrator will be prompted with the **"MESSAGE"** to confirm to save or cancel / re-edit the content item (replace **"MESSAGE"** with your own text).
- **"ERROR:ALERT:MESSAGE"**
There is a problem with the content item data and the content item will not be saved. The **"MESSAGE"** will be displayed to the website administrator and the website administrator must re-edit the content before it can be saved (replace **"MESSAGE"** with your own text).

6.6 Validate User Data Programming API

If you have special requirements for the website user accounts you can program your own program scripts to validate user data when/before they are saved to the web content management system. If a `"/webadmin/api/validateuser.xxx"` (replace `"xxx"` with your programming language extensions: `"aspx"`, `"jsp"` or `"php"`) program script exists then that will be executed when an added or updated user account is saved as well as when a user registers on the website. The user account's data will be posted to the program script as standard HTML POST form data.

The posted user account data can then be validated and the program script should return a structured response to the web content management system:

- **"OK"**
The user account data are ok and the user account will be saved.
- **"OK:ALERT:MESSAGE"**
The user account data are ok and the user account will be saved and the **"MESSAGE"** will be displayed to the website administrator (replace **"MESSAGE"** with your own text). For website user registrations the **"MESSAGE"** will be displayed to the website user.
- **"ERROR:CONFIRM:MESSAGE"**
There is a potential problem with the user account data and the website administrator will



be prompted with the “MESSAGE” to confirm to save or cancel / re-edit the user account (replace “MESSAGE” with your own text). For website user registrations the “MESSAGE” will be displayed to the website user.

- “ERROR:ALERT:MESSAGE”
There is a problem with the user account data and the user account will not be saved. The “MESSAGE” will be displayed to the website administrator and the website administrator must re-edit the user account before it can be saved (replace “MESSAGE” with your own text). For website user registrations the “MESSAGE” will be displayed to the website user.

6.7 External Media Digital Asset Management API

If you are using a separate Digital Asset Management system to manage media which you would like to use from/in the web content management system you can program your own program script to access and list the media categories and files from external media storage and services.

If a “/webadmin/api/media.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”) program script exists then that will be executed when the web content editor Insert Media and Insert Hyperlink toolbar functions are used and media categories(/folders) and files listed by the program script will be available in the Insert Media and Insert Hyperlink pop-up windows. Any “/webadmin/api/media/xxxxx.bat|sh|aspx|jsp|php” program scripts will also be executed.

6.7.1 Media Categories

If no parameter is passed to the program script then a list of media category(/folder) names should be listed (separated by linebreaks):

- Aaa
Bbb
Ccc
....
Xxx
Yyy
Zzzz

The listed media category(/folder) names will be available through the left-hand panel of the Insert Media pop-up window.

6.7.2 Media Files

If a category(/folder) name parameter is passed to the program script it will be one of the listed media category(/folder) names and then a list of media file names and URLs should be listed (separated by linebreaks, and the media file names and URLs separated by tabs):

- Aaaaa http://somemedialibrary.net/abc/aaaaa.gif
Bbbbb http://somemedialibrary.net/abc/logo.png
Ccccc http:// somemedialibrary.net/abc/zzzzz.png
....
Xxxxx http://somemedialibrary.net/abc/xxx123.png
Yyyyy http://somemedialibrary.net/abc/xyz/xxx123.png
Zzzzz http://somemedialibrary.net/xyz/z789.png



The listed media file names will be available through the center panel of the Insert Media and Insert Hyperlink pop-up windows.

6.7.3 Media Search

If a search parameter (and optionally a category(/folder) name parameter) is passed to the program script it will be one or more search words entered by the website administrator and then a list of matching media file names and URLs should be listed (separated by linebreaks, and the media file names and URLs separated by tabs):

- Aaaaa http://somemedialibrary.net/abc/aaaaa.gif
Bbbbb http://somemedialibrary.net/abc/logo.png
Ccccc http:// somemedialibrary.net/abc/zzzzz.png
.....
Xxxxx http://somemedialibrary.net/abc/xxx123.png
Yyyyy http://somemedialibrary.net/abc/xyz/xxx123.png
Zzzzzz http://somemedialibrary.net/xyz/z789.png

The listed media file names will be available through the center panel of the Insert Media and Insert Hyperlink pop-up windows.



7 Web Application Integration and Template Programming API

As default all content and templates etc. are created in and served by the web content management system with no programming required. However, for special requirements you may want to program your own templates, or you may want to integrate content from the web content management system with your own or third-party applications. An easy to use web content management system programming API can be used to access content from the web content management system.

The web content management system programming API functions are located in the "/webadmin.xxx" files, which your web application should include. The web content management system programming API functions are:

- **ReadContent(id)**
Reads the content item with the given id.
- **ReadPage(id)**
Reads the page with the given id including its template and additional content items.
- **ReadProduct(id)**
Reads the product with the given id including its template and additional content items.
- **ReadData(database, id)**
Reads the data with the given id from the custom content database with the given database name.
- **ContentHeader(id, attribute)**
Returns the value of the given attribute name for the content item with the given id.
- **PageDOCTYPE(id)**
Returns the DOCTYPE HTML code for the page with the given id for output at the top of the HTML page.
- **PageHeader(id, attribute, "")**
Returns the value of the given attribute name for the page with the given id for output in the HTML HEAD.
- **PageHeader(id, element, attribute)**
Returns the value of the given attribute name for the additional content element of the given element class name for the page with the given id for output in the HTML HEAD.
- **PageStyleSheet(id)**
Returns the style sheet HTML code for the page with the given id for output in the HTML HEAD.
- **ProductDOCTYPE(id)**
Returns the DOCTYPE HTML code for the product with the given id for output at the top of the HTML page.



- **ProductHeader(id, attribute, "")**
Returns the value of the given attribute name for the product with the given id for output in the HTML HEAD.
- **ProductHeader(id, element, attribute)**
Returns the value of the given attribute name for the additional content element of the given element class name for the product with the given id for output in the HTML HEAD.
- **ProductStyleSheet(id)**
Returns the style sheet HTML code for the product with the given id for output in the HTML HEAD.
- **StyleSheetHeader()**
Outputs the HTTP style sheet header ("Content-Type: text/css").
- **DisplayContent(id, attributename)**
Returns the value of the given attribute name for the page with the given id for output in the HTML BODY.
- **DisplayPage(id, attribute, "")**
Returns the value of the given attribute name for the page with the given id for output in the HTML BODY.
- **DisplayPage(id, element, attribute)**
Returns the value of the given attribute name for the additional content element of the given element class name for the page with the given id for output in the HTML BODY.
- **DisplayProduct(id, attribute, "")**
Returns the value of the given attribute name for the product with the given id for output in the HTML BODY.
- **DisplayProduct(id, element, attribute)**
Returns the value of the given attribute name for the additional content element of the given element class name for the product with the given id for output in the HTML BODY.
- **OutputContent(content)**
Executes "@@extension:...@@" special codes in the content and returns the content with the output from the program extensions.
- **CMSHeader(id)**
Returns the Browse & Edit mode HTML header code for the page with the given id.
- **CMSDisplay(id)**
Returns the Browse & Edit mode web page header code for the page with the given id.
- **CMSStyleSheet(id)**
Returns the Browse & Edit mode style sheet web page header code for the page with the given id.



- `CMSTemplate(id)`
Returns the Browse & Edit mode template web page header code for the page with the given id.
- `CMSLog(id, class, "")`
Logs a request for the content item of the given class name with the given id.
- `CMSLog(id, "data", database)`
Logs a request for the data item with the given id from the custom content database with the given database name.

Please see the `"/page.xxx"`, `"/product.xxx"` and `"/data.xxx"` web content management system program files for examples on how the web content management system programming API functions are used.



8 Cloud Deployment and Media Storage

8.1 Media Cloud Storage API

As default the Asbru Web Content Management system stores the website images and files on the website server. If the website and the Asbru Web Content Management system run on a cluster of website servers they must be setup to use shared or mirrored/replicated file storage to make the website images and files available on all the website servers. Alternatively, you may want to use a cloud storage service (or some other type of shared storage service) for your website images and files.

When the web content management system is configured to use cloud storage through Configuration / System / Website / Media Storage / Cloud Storage the web content management system will execute a number of “/webadmin/api/” program scripts when website images and files are uploaded, copied, moved/renamed, deleted and downloaded:

- /webadmin/api/exists.xxx” (replace “xxx” with: “aspx”, “jsp” or “php”)
The web content management system needs to know if a given website image/file exists on the cloud storage.
- /webadmin/api/upload.xxx” (replace “xxx” with: “aspx”, “jsp” or “php”)
A new website image/file has been uploaded to the website and should be uploaded to the cloud storage.
- /webadmin/api/copy.xxx” (replace “xxx” with: “aspx”, “jsp” or “php”)
A website image/file has been copied and should be copied on/to the cloud storage.
- /webadmin/api/move.xxx” (replace “xxx” with: “aspx”, “jsp” or “php”)
A website image/file has been moved/renamed and should be moved/renamed on the cloud storage.
- /webadmin/api/delete.xxx” (replace “xxx” with: “aspx”, “jsp” or “php”)
A website image/file has been deleted and should be deleted from the cloud storage.
- /webadmin/api/download.xxx” (replace “xxx” with: “aspx”, “jsp” or “php”)
A modified or non-existing website image/file has been accessed and should be downloaded from the cloud storage.

These “/webadmin/api/” program scripts are included with the web content management system with support for a number of cloud storage service providers as well as placeholder comments in the program code for you to add your own custom programming for your own or other third-party cloud storage service providers. Please see the included “/webadmin/api/” program scripts for details.

The Media Cloud Storage API can be used on its own, or in conjunction with the Cloud Deployment API (which requires the use of the Media Cloud Storage API). Please see 8.2 Cloud Deployment API for details.



8.2 Cloud Deployment API

The Asbru Web Content Management system can be installed on your own servers locally or with a hosting service provider of your own choice as well as cloud hosting services.

For a cloud hosted installation of the web content management system with dynamic scalability through addition of additional web servers, the web content management system installation may need to automatically detect and connect to the configured database server and media cloud storage when a new web server is added.

As default the Asbru Web Content Management system includes support for a number of cloud hosting services and database servers with automatic detection and connection to the database server. To use the web content management with other cloud hosting services and/or databases than the ones supported as default, you may need to add your own program code to automatically detect the configured database server and generate the database connection string to be used by the web content management system to connect to the configured database server as well as the media cloud storage configuration settings.

The cloud deployment detection and database connection and media cloud storage is handled by the `"/config.cloud.xxx"` (replace `"xxx"` with: `"aspx"`, `"jsp"` or `"php"`) special configuration program script. To add support for other cloud hosting service providers and database servers, simply edit this program script and add your own program code to:

- Detect the cloud deployment settings through the server environment variables or any other way these settings are made available by the cloud hosting services.
- Set the `"database"` to the database connection string for the configured database server as it would be entered into the web content management system's Configuration / System / Database / Database Connection configuration page.
- Set the `"database_init"` configuration setting to any special SQL commands, which may be required to create and initialise the database server.
- Set the `"csservice"` configuration setting to one of the supported media cloud storage service providers or to any other unique id/name for your cloud storage service provider as also used in your Media Cloud Storage API program scripts.
- Set the `"csusername"`, `"cspassword"`, `"csrootpath"` and `"csURLrootpath"` configuration settings to the relevant values as they would be entered into the web content management system's Configuration / System / Website / Media Storage configuration page.

Please see the `"/config.cloud.xxx"` (replace `"xxx"` with: `"aspx"`, `"jsp"` or `"php"`) web content management system program files for examples on how the web content management system programming API functions are used

The Cloud Deployment API must be used in conjunction with the Media Cloud Storage API which is used to store media files. Please see 8.1 Media Cloud Storage API for details.



9 One-Time Password Login

The Asbru Web Content Management system supports customization/extension of one-time password login functionality through your own custom “/webadmin/api/” program scripts when website users and/or website administrators are required to login.

As default the one-time password login supports use of random one-time password codes generated by the web content management system and emailed to the website users/administrators; as well as standard RFC 6238 Time-Based One-Time Password compliant codes using third-party apps and devices.

A custom login script can be used to send the one-time password codes generated by the web content management system to the website user/administrator using other methods than email.

A custom login script can also be used to generate its own one-time password codes and send them to the website user/administrator using email or other methods.

When a one-time password code is generated or sent to a website user/administrator the web content management system will check if a “/webadmin/api/login”, “/webadmin/api/login.bat”, “/webadmin/api/login.sh” or “/webadmin/api/login.xxx” (replace “xxx” with your programming language extensions: “aspx”, “jsp” or “php”) file exists and execute it with the website user/administrator details and the generated one-time password code.

Depending on what the program script does and what the web content management system should do the program script must return/output:

- The same one-time password code as passed to the program script as a parameter
If the program script has sent that one-time password code to the website user/administrator.
- A new one-time password code
If the program script has generated a new one-time password code and sent that to the website user/administrator.
- Nothing
If the program script has not done anything. The web content management system should use its own generated one-time password code and email it to the website user/administrator.



10 Headless REST API

The Asbru Web Content Management system supports “headless” access from third-party and custom content delivery and administration systems through REST API functionality. REST APIs are available for both published website content and functionality and for website administration functionality.

10.1 General

10.1.1 REST API Login and Authorization Access Tokens

Access to the web content management system administration REST API and optionally the published website content REST API requires login to the web content management system for an authorization “access token” which must be provided for all other REST API functions.

10.1.1.1 HTTP Authorization Bearer Header

It is recommended that the issued authorization “access token” is provided with REST API requests as a HTTP Authorization Bearer header for all types of request (GET, POST, PUT, DELETE).

HTTP Authorization Bearer Header

```
GET /webadmin/rest/content/?id=123
Authorization: Bearer XXXXX.YYYYY.ZZZZZ
```

10.1.1.2 JSON Data Parameter

Alternatively, the issued authorization “access token” should be provided as a JSON data parameter.

JSON Data Parameter

```
POST /webadmin/rest/content/
Content-Type: application/json
{
  "access_token": "XXXXX.YYYYY.ZZZZZ",
  "id": "123",
  ....
}
```

10.1.1.3 HTML FORM POST Parameter

Alternatively, the issued authorization “access token” can be provided as a HTML FORM POST parameter.

HTML FORM POST Parameter

```
<form action="/webadmin/rest/content/" method="POST">
<input type="hidden" name="access_token" value="XXXXX.YYYYY.ZZZZZ">
<input type="hidden" name="id" value="123">
....
</form>
```



10.1.1.4 HTTP URL Parameter

Eventually, the issued authorization “access token” can be provided as a HTML URL parameter, but this is not recommended for security reasons as such requests may be cached or logged etc. which may give unauthorized users access to the issued authorization token and the REST API functionality.

HTTP URL Parameter
GET /webadmin/rest/content/?id=ID&access_token=XXXXX.YYYYY.ZZZZZ

10.1.1.5 Expiration

Issued authorization access tokens may expire after a given period of time as configured for the web content management system.

Any REST API request using an expired authorization access token will fail with a “HTTP 401 Unauthorized” error. A new REST API login request is then required for a new authorization “access token”.

10.1.2 REST API Methods

The REST API requests may require/support GET, POST, UPDATE and DELETE methods. Please note that all methods are not supported by all REST API functions.

GET methods are used to read content/data from the web content management system.

POST methods are used to create new content/data in the web content management system as well as for login etc.

UPDATE methods are used to update existing content/data in the web content management system.

DELETE methods are used to delete existing content/data from the web content management system.

10.1.3 REST API Request Parameters

REST API requests may require and support a number of different parameters to identify and filter the requested content/data and for other content/data attributes. Parameters can/must be given in different ways depending on the REST API functions and methods used/required.

10.1.4 REST API Cookies

Some REST API functions may use cookies to store and transfer some data. For example, content delivery functions may read/set “usersegments”, “usertests” and “userteststags” cookies for the Experience Management functionality; and the shopping cart function reads/sets “shopcart” and “discount” cookies for the shopping cart contents.

10.1.5 GET Requests with URL Parameters

REST API GET requests to read content/data from the web content management system may require and support a number of different parameters to identify and filter the requested content/data. Parameters for GET requests should be given as URL parameters as for standard web browser URL addresses.



GET Request URL Parameters

GET /rest/page/?id=ID&version=VERSION

10.1.6 POST, PUT and DELETE Requests with JSON Parameters

REST API POST, PUT and DELETE requests to create, update and delete content/data in the web content management system may require and support different parameters to identify the content/data and for the content/data attributes to be created/updated. Parameters for POST, PUT and DELETE requests can be given as JSON data.

JSON Data Parameters

POST /webadmin/rest/content/ Content-Type: application/json { "title": "abc", "content": "xyz", }

10.1.7 POST, PUT and DELETE Requests with HTML FORM Parameters

Alternatively, instead of as JSON parameters, parameters for POST, PUT and DELETE requests can be given as HTML FORM data. However, please note that web browsers may only support HTML FORM GET and POST requests - not HTML FORM PUT and DELETE requests.

HTML FORM POST Parameter

<form action="/webadmin/rest/content/" method="POST"> <input type="text" name="title" value="abc"> <input type="text" name="content" value="xyz"> </form>

10.2 Website Content and Functionality

Published website content and functionality is available through
"http://www.yourwebsite.com/rest/xxxxx/" REST API functions. Basically, these REST API functions give access to the same content and functionality as the various
"http://www.yourwebsite.com/xxxxx.aspx|jsp|php" program scripts.

10.2.1 Login

Typically, the REST API website content functions should be configured to require login. Optionally, the REST API website content functions may be configured to be public, in which login is not required and REST API authorization access tokens are not required.

If REST API website administrator login is required, a valid username and password for a website administrator in the web content system user database must be provided. If REST API website user login is required, a valid username and password for a registered website user or a website administrator in the web content system user database must be provided.



Optionally, an API Key may also be required.

The response to a valid REST API login will be an authorization “access token” in JSON data format. The authorization “access token” must be provided with other REST API requests for access.

Optionally, an authorization “refresh token” in JSON data format may also be provided, which can be used to request a new authorization “access token”.

The REST API function also reads/sets a “login_code” cookie for login code validation if the web content management system is configured to use randomly generated one-time passwords emailed (or otherwise sent) to the website user. In such case: First, login using username and password parameters, which will fail and return a “login_code” cookie and email (or otherwise send) a login code to the website user; then login again using username and password parameters and a code parameter with the emailed one-time password code (and the “login_code” cookie returned by the first failed login).

POST /rest/login/	
username	Username for registered website user or website administrator depending on the web content management system configuration.
password	Password for registered website user or website administrator depending on the web content management system configuration.
code	One-Time Password code for registered website user or website administrator depending on the web content management system configuration.
apikey	API key as configured for the web content management system (if any).
Response	
{ “refresh_token”: “XXXXXXXXXX.YYYYYYYYYY.ZZZZZZZZZZ”, “access_token”: “XXXXXXXXXX.YYYYYYYYYY.ZZZZZZZZZZ” }	

If a refresh token has been provided for a login in addition to an access token then the refresh token can be used instead of username/password/code to request a new access token when the access token has expired. Typically, the refresh token will be valid for a significantly longer time (hours/days) than the access token (minutes/hours).

POST /rest/login/	
refresh_token	Valid refresh token provided for previous login.
apikey	API key as configured for the web content management system (if any).
Response	
{ “access_token”: “XXXXXXXXXX.YYYYYYYYYY.ZZZZZZZZZZ” }	



10.2.2 Page

The “/webadmin/rest/page/” REST API function returns the full, parsed web page including its template etc. split into the individual HTML page parts as JSON data.

GET /rest/page/?id=ID	
GET /rest/page/?id=ID&version=VERSION	
id	Page id number
version	Content version name
device	Content device name
Response	
<pre>{ "doctype": "<!DOCTYPE html>", "html": "", "head": "", "metainfo": "<meta name=\"language\" content=\"EN\">", "author": "", "description": "What we do.", "keywords": "The homepage for My Business.", "title": "My Business", "stylesheets": "<link rel=\"stylesheet\" type=\"text/css\" href=\"/stylesheet.jsp?id=1\">\r\n\r\n", "scripts": "", "htmlheader": "", "contentschema": "", "htmlbodyonload": "", "body": "<div id=\"wrapper_outer\">\r\n <div id=\"wrapper_inner\">\r\n </div>\r\n </div>" }</pre>	

10.2.3 Product Page

The “/rest/product/” REST API function returns the full, parsed product web page including its template etc. split into the individual HTML page parts as JSON data.

GET /rest/product/?id=ID	
GET /rest/product/?id=ID&version=VERSION	
id	Product id number
version	Content version name
device	Content device name
Response	
<pre>{ "doctype": "<!DOCTYPE html>", "html": "", "head": "", "metainfo": "<meta name=\"language\" content=\"EN\">", "author": "", "description": "What we do.", "keywords": "The homepage for My Business.", "title": "My Business", "stylesheets": "<link rel=\"stylesheet\" type=\"text/css\" href=\"/stylesheet.jsp?id=1\">\r\n\r\n", "scripts": "", "htmlheader": "", "contentschema": "", "htmlbodyonload": "", "body": "<div id=\"wrapper_outer\">\r\n <div id=\"wrapper_inner\">\r\n </div>\r\n </div>" }</pre>	



10.2.4 Data Page

The “/rest/data/” REST API function returns the full, parsed data web page including its template etc. split into the individual HTML page parts as JSON data.

GET /rest/data/?database=DATABASE&id=ID	
GET /rest/data/?database=DATABASE&id=ID&version=VERSION	
database	Content database name
id	Content database item id number
version	Content version name
device	Content device name
Response	
<pre>{ "doctype": "<!DOCTYPE html>", "html": "", "head": "", "metainfo": "<meta name=\"language\" content=\"EN\">", "author": "", "description": "What we do.", "keywords": "The homepage for My Business.", "title": "My Business", "stylesheets": "<link rel=\"stylesheet\" type=\"text/css\" href=\"\"/stylesheet.jsp?id=1\">\r\n\r\n", "scripts": "", "htmlheader": "", "contentschema": "", "htmlbodyonload": "", "body": "<div id=\"wrapper_outer\">\r\n <div id=\"wrapper_inner\">\r\n </div>\r\n </div>" }</pre>	

10.2.5 Content Item

The “/rest/contentitem/” REST API function returns just the parsed primary content of a content item excluding its template etc.

GET /rest/contentitem/?id=ID	
GET /rest/contentitem/?id=ID&version=VERSION	
id	Content item id number
version	Content version name
device	Content device name
Response	
<pre>{ "content": "<div id=\"wrapper_outer\">\r\n <div id=\"wrapper_inner\">\r\n </div>\r\n </div>" }</pre>	

10.2.6 Data Item

The “/rest/dataitem/” REST API function returns just the parsed primary content of a content database item excluding its template etc.

```
GET /rest/dataitem/?database=DATABASE&id=ID
GET /rest/dataitem/?database=DATABASE&id=ID&version=VERSION
```



10.2.7 Script

```
GET /rest/script/?id=ID
GET /rest/script/?id=ID&version=VERSION
```

10.2.8 Style Sheet

```
GET /rest/stylesheet/?id=ID
GET /rest/stylesheet/?id=ID&version=VERSION
```

Page 48 of 280



10.2.9 Image

GET /rest/image/?id=ID

Response

GET /rest/file/?id=ID

Response

Page 49 of 280



GET /rest/link/?id=ID	
GET /rest/link/?id=ID&version=VERSION	
id	Link id number
version	Content version name
device	Content device name
Response	
{ "url": "http:\\\\www.somewebsite.com\\" }	

10.2.12 Search

The “/rest/search/” REST API function returns website content or content database search results as JSON data.

10.2.12.1 Website Content Search

If no “database” parameter is given the published website content will be searched. The matching website content items’ title, author, description, keywords, contentclass and id attributes data will be returned.

GET /rest/search/?search=SEARCH	
search	Search words (required).
Response	
[{ "title": "ABC", "author": "", "description": ".....", "keywords": "", "contentclass": "page", "id": "123" }, { "title": "XYZ", "author": "", "description": ".....", "keywords": "", "contentclass": "page", "id": "789" }]	

10.2.12.1 Content Database Search

If a “database” parameter is given that content database will be searched. The content database items’ title and id attributes data will be returned.

GET /rest/search/?database=DATABASE&search=SEARCH	
search	Search words (required).
database	Content database name (required).
Response	
[{ "title": "ABC",	



```
    "id": "123"
  }
  .....
  {
    "title": "XYZ",
    "id": "789"
  }
]
```

10.2.13 Contact

The “/rest/contact/” REST API function emails contact form data to website administrators.

This REST API function mirrors the “/contact.xxx” program script functionality. Please see the Website Developer Guide “4.1 Contact Form” section for details.

The POST request will return an empty response.

POST /rest/contact/	
from	Email address.
to	Email address.
cc	Email address.
bcc	Email address.
subject	Email subject text.
.....	Contact form data.
Request	
{ "from": "johndoe@example.com", "to": "help@yourwebsite.com", "cc": "archive@yourwebsite.com", "subject": ".....", "message": "....." }	
Response	
{ }	

10.2.14 Post

The “/rest/post/” REST API function posts content to the website or a content database.

10.2.14.1 Website Content

This REST API function mirrors the “/post.xxx” program script functionality. Please see the Website Developer Guide “7.5 User Posted Content” section for details.

The POST request will return the created content.

POST /rest/post/	
id	Content item id number (required).
.....	Content data attributes.
publish	“yes” (optional).
ready_to_publish	“yes” (optional).
email_template	Content item id number (optional).
Request	



```
{
  "id": "123",
  .....
  "publish": "yes",
  "email_template": "456"
}
```

Response

```
{
  "content": "....."
}
```

10.2.14.2 Database Content

This REST API function mirrors the “/post.xxx” program script functionality. Please see the Website Developer Guide “8.1.2 Posting Database Content” section for details.

The POST request will return the created content database data.

POST /rest/post/	
database	Content database id name (required).
id	Content database record id number (optional).
.....	Content database data attributes.
email_template	Content item id number (optional).
email_notification	“yes” (optional).
Request	
<pre>{ "database": "XYZ", "id": "123", "email_template": "456", "email_notification": "yes" }</pre>	
Response	
<pre>{ "database": "XYZ", "id": "789", }</pre>	

10.2.15 Register User

The “/rest/register/” REST API function registers a website visitor for a new user account.

This REST API function mirrors the “/register.xxx” program script functionality. Please see the Website Developer Guide “7.2 User Registration” section for details.

The POST request returns an “error” message with the status of the user registration (“[[register.created]]” if successful).

POST /rest/register/	
id	Content page id number.
user_id	User account id number.
content_id	Content item id number.
from	Email address.
cc	Email address.
bcc	Email address.



email	Email address.
username	Email address.
password	Email address.
name	User name.
.....	User data.
Request	
{ "user_id": "1", "email": "johndoe@example.com", "username": "johndoe", "password": "secret", "name": "John Doe", }	
Response	
{ "error": "[register.created]" }	

10.2.16 Unregister User

The “/rest/unregister/” REST API function deactivates/deletes the logged in website user’s user account.

The POST request returns an empty response.

POST /rest/unregister/	
Request	
{ }	
Response	
{ }	

10.2.17 Subscribe

The “/rest/subscribe/” REST API function subscribes the logged website user account to the given user group or user type.

This REST API function mirrors the “/subscribe.xxx” program script functionality. Please see the Website Developer Guide “7.4 User Group/Type Subscriptions” section for details.

The POST request returns an empty response.

POST /rest/subscribe/	
usergroup	User group name.
usertype	User type name.
Request	
{ "usergroup": "ABC" }	
Response	
{ }	



<pre>"usertype": "XYZ" }</pre>
Response
<pre>{ }</pre>

10.2.18 Unsubscribe

The “/rest/unsubscribe/” REST API function unsubscribes the logged website user account from the given user group or user type.

This REST API function mirrors the “/unsubscribe.xxx” program script functionality. Please see the Website Developer Guide “7.4 User Group/Type Subscriptions” section for details.

The POST request returns an empty response.

POST /rest/unsubscribe/	
usergroup	User group name.
usertype	User type name.
Request	
<pre>{ "usergroup": "ABC" }</pre>	
Request	
<pre>{ "usertype": "XYZ" }</pre>	
Response	
<pre>{ }</pre>	

10.2.19 Shopping Cart

The “/rest/shopcart/” REST API function handles the website shopping cart functionality.

This REST API function mirrors the “/shopcart.xxx” program script functionality. Please see the Website Developer Guide “9.2 Special Links” section for details.

The POST request returns the full, parsed shopping cart web page including its template etc. split into the individual HTML page parts as JSON data.

The REST API function also reads/sets “shopcart” and “discount” cookies for the shopping cart contents.

POST /rest/shopcart/	
add	Product id number.
ID	Product id quantity.
checkout	“checkout”
confirm	“confirm”
complete	“complete”
Request – add product id 123 to shopping cart	
<pre>{ "add": "123" }</pre>	



10.2.20 Heatmap

This REST API function mirrors the “/heatmap.xxx” program script functionality.

Page 55 of 280



Response
<pre>{ "id": "123", "action": "click", "logged": "2020-12-31 23:59:59", "x": "535", "y": "63", "w": "800", "h": "600", "usersegments": "", "usertests": "" }</pre>

10.2.21 Password

The “/rest/password/” REST API function retrieves and updates a website user’s password.

10.2.21.1 Retrieve Password

This REST API function mirrors the “/password/index.xxx” program script functionality. Please see the Website Developer Guide “5.8 Retrieve Password Page” section for details.

The POST request returns the full, parsed retrieve password web page including its template etc. split into the individual HTML page parts as JSON data.

POST /rest/password/	
email	Email address (email or username required).
username	Website user account username (email or username required).
from	Email address (optional).
cc	Email address (optional).
bcc	Email address (optional).
Request	
<pre>{ "username": "johndoe", "email": "johndoe@example.com", "from": "password@yourwebsite.com", "cc": "support@yourwebsite.com", "bcc": "archive@yourwebsite.com" }</pre>	
Response	
<pre>{ "doctype": "<!DOCTYPE html>", "html": "", "head": "", "metainfo": "", "author": "", "description": "", "keywords": "", "title": "Forgot your login details?", "stylesheets": "<link rel=\"stylesheet\" type=\"text/css\" href=\"/stylesheet.jsp?id=1\">\r\n\r\n", "scripts": "", "htmlheader": "", "contentschema": "", "htmlbodyonload": "", "body": "<div id=\"wrapper_outer\">\r\n </div>" }</pre>	



10.2.21.2 Expired Password

This REST API function mirrors the “/password/expired.xxx” program script functionality. Please see the Website Developer Guide “5.12 Expired Password Page” section for details.

The PUT request will return an empty response or an error message.

PUT /rest/password/	
old_password	Logged in website user’s existing password (required).
new_password	Logged in website user’s new password (required).
new_password2	Logged in website user’s new password (required).
Request	
{ "old_password": "secret", "new_password": "alsosecret", "new_password2": "alsosecret" }	
Response	
{ }	

10.2.21.3 Retrieve Superadmin Password

This REST API function mirrors the “/webadmin/password/index.xxx” program script functionality. Please see the Configuration Guide “5.8 Retrieve Superadmin Username and Password” section for details.

The GET request returns the configured superadmin email address to which the superadmin username and password has been emailed, or an error message, as JSON data.

GET /rest/password/webadmin/	
GET /rest/password/webadmin/?username=USERNAME	
GET /rest/password/webadmin/?email=EMAIL	
GET /rest/password/webadmin/?username=USERNAME&email=EMAIL	
email	Email address (email and/or username may be required).
username	Website user account username (email and/or username may be required).
Request	
{ "username": "johndoe", "email": "johndoe@example.com" }	
Response	
{ "error": "....." }	
Response	
{ "error": "....." }	



10.2.22 Personal

The “/rest/personal/” REST API function retrieves and updates a website user’s personal website page.

10.2.22.1 Personal Page

This REST API function mirrors the “/personal/?USERNAME” program script functionality. Please see the Website Developer Guide “7.3 Personal Page and Administration” section for details.

The GET request returns the full, parsed personal web page including its template etc. split into the individual HTML page parts as JSON data.

GET /rest/personal/?USERNAME	
GET /rest/personal/?username=USERNAME	
GET /rest/personal/?id=USERNAME	
username	Website user account username (required).
Response	
<pre>{ "doctype": "<!DOCTYPE html>", "html": "", "head": "", "metainfo": "", "author": "", "description": "", "keywords": "", "title": "John Smith", "stylesheets": "<link rel=\"stylesheet\" type=\"text/css\" href=\"\"/stylesheet.jsp?id=1\">\r\n\r\n", "scripts": "", "htmlheader": "", "contentschema": "", "htmlbodyonload": "", "body": "<div id=\"wrapper_outer\">\r\n </div>" }</pre>	

10.2.22.2 Personal Page Content and Preferences

This REST API function mirrors the “/personal/admin.xxx” program script functionality. Please see the Website Developer Guide “7.3 Personal Page and Administration” section for details.

A PUT request will update the logged in website user’s Personal Page content and preferences and website user account details.

The PUT request returns the updated full, parsed personal web page including its template etc. split into the individual HTML page parts as JSON data.

PUT /rest/personal/	
title	Personal Page title.
content	Personal Page content.
template	Template content item id number.
stylesheet	Stylesheet content item id number.
personal_ELEMENT_content	ELEMENT content item id number.



publish	"publish"
contentclass	"page"
.....	Personal Details data.
Request	
{ "title": "ABC" "content": "XYZ" "template": "123" "stylesheet": "456" "personal_news_content": "789" "email": "johndoe@example.com", "name": "John Doe", "password": "secret", "title": "ABC", "publish": "publish" }	
Response	
{ "doctype": "<!DOCTYPE html>", "html": "", "head": "", "metainfo": "", "author": "", "description": "", "keywords": "", "title": "John Smith", "stylesheets": "<link rel=\"stylesheet\" type=\"text/css\" href=\"\"/stylesheet.jsp?id=1\">\r\n\r\n", "scripts": "", "htmlheader": "", "contentschema": "", "htmlbodyonload": "", "body": "<div id=\"wrapper_outer\">\r\n </div>" }	

10.2.23 Logout

The “/rest/logout/” REST API function logs the logged in website user account out.

The POST request returns an empty response.

POST /rest/logout/	
Request	
{ }	
Response	
{ }	

10.3 Website Administration Functionality

Full website administration functionality is available through

“http://www.yourwebsite.com/webadmin/rest/xxxxx/” REST API functions. Basically, these REST API functions give access to the same content and functionality as the various

“http://www.yourwebsite.com/webadmin/xxxxx/” web content system administration pages.



10.3.1 Login

The REST API website administration functions require login.

A valid username and password for a website administrator in the web content system user database must be provided.

Optionally, an API Key may also be required.

The response to a valid REST API login will be an authorization “access token” in JSON data format. The authorization “access token” must be provided with other REST API requests for access.

Optionally, an authorization “refresh token” in JSON data format may also be provided, which can be used to request a new authorization “access token”.

POST /webadmin/rest/login/	
username	Username for website administrator user account.
password	Password for website administrator user account.
code	One-Time Password code for website administrator depending on the web content management system configuration.
apikey	API key as configured for the web content management system (if any).
Response	
{ "refresh_token": "XXXXXXXXXX.YYYYYYYYYY.ZZZZZZZZZZ", "access_token": "XXXXXXXXXX.YYYYYYYYYY.ZZZZZZZZZZ" }	

If a refresh token has been provided for a login in addition to an access token then the refresh token can be used instead of username/password/code to request a new access token when the access token has expired. Typically, the refresh token will be valid for a significantly longer time (hours/days) than the access token (minutes/hours).

POST /webadmin/rest/login/	
refresh_token	Valid refresh token provided for previous login.
apikey	API key as configured for the web content management system (if any).
Response	
{ "access_token": "XXXXXXXXXX.YYYYYYYYYY.ZZZZZZZZZZ" }	

10.3.1.1 Login Settings

The “/webadmin/rest/login/” REST API function returns all supported login methods and credentials for the web content management system administration.

GET /webadmin/rest/login/	
Response	



```
{
  "openid": "yes",
  "facebook": "no",
  "google": "no",
  "credentials": "password"
}
```

10.3.2 Content

The “/webadmin/rest/content/” REST API function returns all data for one or more content items and creates, updates and deletes content items. Additional REST API functions are also available for specific archive, checkin, checkout, copy, delete, move, publish, unpublish, checkloops, checklinks, dependencies, heatmap, usersegment, usertest functionality.

10.3.2.1 List

If no “id” and no “archive” and no “search” and no “personal” and no “index” parameter is given then a list of content items which match the other given parameters will be returned.

GET /webadmin/rest/content/?contentclass=CONTENTCLASS&contentgroup=CONTENTGROU P&contenttype=CONTENTTYPE&contentbundle=CONTENTBUNDLE&contentpackage=CO NTENTPACKAGE&version=VERSION&status=STATUS&stock=STOCK&pagesize=PAG ESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
contentclass	Content class name (“page”, “template”, “stylesheet”, “script”, “image”, “file”, “link”, “product” or any configured content element class name).
contentgroup	Content group name or “-“ for none.
contenttype	Content type name or “-“ for none.
contentbundle	Content bundle name.
contentpackage	Content package name.
version	Content version name or “-“ for none.
status	Content status code (“new”, “updated”, “scheduled”, “published”, “unpublished”, “expiring”, “expired”, “checkedout”).
stock	Product stock status code (“unlimited”, “in”, “low”, “no”, “orderable”, “ordered”).
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content



	items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC" }, { "id": "789", "title": "XYZ" }]</pre>	

10.3.2.1.1 Personal Workspace

If a “personal” parameter is given then a list of content items for the given “personal” parameter personal workspace section will be returned.

GET /webadmin/rest/content/?personal=SECTION&pagesize_SECTION=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
personal	Personal workspace section name (“checkedout”, “updated”, “created”, “expired”, “workflow”, “projects”, “comments”).
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC" }, { "id": "789",</pre>	



```
.....
  "title": "XYZ"
}
```

10.3.2.1.2 Delete Confirmation

If a “index=delete” parameter is given then a list of content items which match the “id” parameter will be returned.

GET /webadmin/rest/content/?index=delete&id=ID,ID,ID&pagesize=PAGESIZE&offset=OFFSET &sort_col=ATTRIBUTE&sort_dir=DESC	
index	“delete”.
id	Content item ids separated by commas.
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC" }, { "id": "789", "title": "XYZ" }]</pre>	

10.3.2.1.3 Unpublish Confirmation

If a “index=unpublish” parameter is given then a list of content items which match the “id” parameter will be returned.

```
GET
/webadmin/rest/content/?index=unpublish&id=ID,ID,ID&pagesize=PAGESIZE&offset=OFF
```



SET&sort col=ATTRIBUTE&sort dir=DESC	
index	“unpublish”.
id	Content item ids separated by commas.
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC" } , , { "id": "789", "title": "XYZ" }]</pre>	

10.3.2.1.4 Create Options

If a “index=create” parameter is given then a list of content items for which the website administrator has “create permissions” and which match the other given parameters will be returned.

GET /webadmin/rest/content/?index=create&contentclass=CONTENTCLASS&contentgroup=CONTENTGROUP&contenttype=CONTENTTYPE&contentbundle=CONTENTBUNDLE&contentpackage=CONTENTPACKAGE&version=VERSION&status=STATUS&stock=STOCK &pagesize=PAGESIZE&offset=OFFSET&sort col=ATTRIBUTE&sort dir=DESC	
index	“create”.
contentclass	Content class name (“page”, “template”, “stylesheet”, “script”, “image”, “file”, “link”, “product” or any configured content element class name).
contentgroup	Content group name or “-” for none.
contenttype	Content type name or “-” for none.



contentbundle	Content bundle name.
contentpackage	Content package name.
version	Content version name or “-“ for none.
status	Content status code (“new”, “updated”, “scheduled”, “published”, “unpublished”, “expiring”, “expired”, “checkedout”).
stock	Product stock status code (“unlimited”, “in”, “low”, “no”, “orderable”, “ordered”).
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC" } , , { "id": "789", "title": "XYZ" }]</pre>	

10.3.2.1.5 Archived

If a “archive=archive” parameter is given then a list of content items which are archived and which match the other given parameters will be returned.

GET /webadmin/rest/content/?archive=archive&contentclass=CONTENTCLASS&contentgroup=CONTENTGROUP&contenttype=CONTENTTYPE&contentbundle=CONTENTBUNDLE&contentpackage=CONTENTPACKAGE&version=VERSION&status=STATUS&stock=STOCK&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
archive	“schedule”.
contentclass	Content class name (“page”, “template”, “stylesheet”, “script”, “image”, “file”, “link”,



	“product” or any configured content element class name).
contentgroup	Content group name or “-“ for none.
contenttype	Content type name or “-“ for none.
contentbundle	Content bundle name.
contentpackage	Content package name.
version	Content version name or “-“ for none.
status	Content status code (“new”, “updated”, “scheduled”, “published”, “unpublished”, “expiring”, “expired”, “checkedout”).
stock	Product stock status code (“unlimited”, “in”, “low”, “no”, “orderable”, “ordered”).
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC" }, { "id": "789", "title": "XYZ" }]</pre>	

10.3.2.1.6 Scheduled

If a “archive=schedule” parameter is given then a list of content items which are scheduled for publishing and which match the other given parameters will be returned.

GET /webadmin/rest/content/?archive=schedule&contentclass=CONTENTCLASS&contentgroup=



CONTENTGROUP&contenttype=CONTENTTYPE&contentbundle=CONTENTBUNDLE&contentpackage=CONTENTPACKAGE&version=VERSION&status=STATUS&stock=STOCK&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
archive	“schedule”.
contentclass	Content class name (“page”, “template”, “stylesheet”, “script”, “image”, “file”, “link”, “product” or any configured content element class name).
contentgroup	Content group name or “-” for none.
contenttype	Content type name or “-” for none.
contentbundle	Content bundle name.
contentpackage	Content package name.
version	Content version name or “-” for none.
status	Content status code (“new”, “updated”, “scheduled”, “published”, “unpublished”, “expiring”, “expired”, “checkedout”).
stock	Product stock status code (“unlimited”, “in”, “low”, “no”, “orderable”, “ordered”).
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC" }, { "id": "789", "title": "XYZ" }]</pre>	



10.3.2.1.7 Archived and Scheduled

If a “archive=*” parameter is given then a list of content items which are archived or scheduled for publishing and which match the other given parameters will be returned.

GET /webadmin/rest/content/?archive=*&contentclass=CONTENTCLASS&contentgroup=CONTENTGROUP&contenttype=CONTENTTYPE&contentbundle=CONTENTBUNDLE&contentpackage=CONTENTPACKAGE&version=VERSION&status=STATUS&stock=STOCK&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
archive	“schedule”.
contentclass	Content class name (“page”, “template”, “stylesheet”, “script”, “image”, “file”, “link”, “product” or any configured content element class name).
contentgroup	Content group name or “-“ for none.
contenttype	Content type name or “-“ for none.
contentbundle	Content bundle name.
contentpackage	Content package name.
version	Content version name or “-“ for none.
status	Content status code (“new”, “updated”, “scheduled”, “published”, “unpublished”, “expiring”, “expired”, “checkedout”).
stock	Product stock status code (“unlimited”, “in”, “low”, “no”, “orderable”, “ordered”).
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC" }, { "id": "789", "title": "XYZ" }]</pre>	



```
}  
]
```

10.3.2.1.8 Livegrid (DEPRECATED)

This duplicates the general list functionality as described above but with output in Rico Livegrid XML format as used by current and older versions of the web content management system administration pages.

Note: DEPRECATED – use GET /webadmin/rest/content/ (Please see section 10.3.2.1 List).

GET /webadmin/rest/content/livegrid/?contentclass=CONTENTCLASS&contentgroup=CONTENTGROUP&contenttype=CONTENTTYPE&contentbundle=CONTENTBUNDLE&contentpackage=CONTENTPACKAGE&version=VERSION&status=STATUS&stock=STOCK&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
contentclass	Content class name (“page”, “template”, “stylesheet”, “script”, “image”, “file”, “link”, “product” or any configured content element class name).
contentgroup	Content group name or “-” for none.
contenttype	Content type name or “-” for none.
contentbundle	Content bundle name.
contentpackage	Content package name.
version	Content version name or “-” for none.
status	Content status code (“new”, “updated”, “scheduled”, “published”, “unpublished”, “expiring”, “expired”, “checkedout”).
stock	Product stock status code (“unlimited”, “in”, “low”, “no”, “orderable”, “ordered”).
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
XML code	



10.3.2.2 Search

If a “search” parameter is given then a search results list of content items which match the “search” parameter and all other given parameters will be returned.

GET /webadmin/rest/content/?search=SEARCH&id=ID,ID,ID§ion=SECTION&class=CONTENTCLASS&group=CONTENTGROUP&type=CONTENTTYPE&bundle=CONTENTBUNDLE&package=CONTENTPACKAGE&version=VERSION&device=DEVICE&usersegment=USERSEGMENT&usertest=USERTEST&status=STATUS&filename=FILENAME&attribute=ATTRIBUTE&search_datetime=DATETIME&search_datetime_attribute=DATETIMEATTRIBUTE&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
search	Search word(s).
id	Content item ids separated by commas.
section	Content section name (“content”, “page”, “template”, “stylesheet”, “script”, “library”, “image”, “file”, “link”, “ecommerce”, “product” or any configured content element class name).
class	Content class name (“page”, “template”, “stylesheet”, “script”, “image”, “file”, “link”, “product” or any configured content element class name).
group	Content group name or “-” for none.
type	Content type name or “-” for none.
bundle	Content bundle name.
package	Content package name.
version	Content version name or “-” for none.
device	Content device name.
usersegment	Content user segment name.
usertest	Content user test name.
status	Content status code (“new”, “updated”, “scheduled”, “published”, “unpublished”, “expiring”, “expired”, “checkedout”, “workflow” (for any non-blank workflow status) or any configured workflow status name).
filename	Content filename or part of content filename.
attribute	Content attribute names to be searched (“title”, “content”, “summary”, “description”, “keywords”, “metainfo”, “htmlheader”) separated by commas or “” for all the attributes.
search_datetime	Date/time format string or substring.
search_datetime_attribute	“created”, “updated”, “published”, “unpublished”, “scheduled”, “expiring”, “deleted” or “” for all the attributes.
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content



	items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC" }, { "id": "789", "title": "XYZ" }]</pre>	

10.3.2.2.1 Livegrid (DEPRECATED)

This duplicates the general search functionality as described above but with output in Rico Livegrid XML format as used by current and older versions of the web content management system administration pages.

Note: DEPRECATED – use GET /webadmin/rest/content/?search=SEARCH (Please see section 10.3.2.2 Search).

GET /webadmin/rest/content/livegrid/?search=SEARCH&id=ID,ID,ID§ion=SECTION&class=CONTENTCLASS&group=CONTENTGROUP&type=CONTENTTYPE&bundle=CONTENTBUNDLE&package=CONTENTPACKAGE&version=VERSION&device=DEVICE&usersegment=USERSEGMENT&usertest=USERTEST&status=STATUS&filename=FILENAME&attribute=ATTRIBUTE&search_datetime=DATETIME&search_datetime attribute=DATE TIMEATTRIBUTE&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
search	Search word(s).
id	Content item ids separated by commas.
section	Content section name (“content”, “page”, “template”, “stylesheet”, “script”, “library”, “image”, “file”, “link”, “ecommerce”, “product” or any configured content element class name).
class	Content class name (“page”, “template”, “stylesheet”, “script”, “image”, “file”, “link”, “product” or any configured content element class name).
group	Content group name or “-“ for none.



type	Content type name or “-“ for none.
bundle	Content bundle name.
package	Content package name.
version	Content version name or “-“ for none.
device	Content device name.
usersegment	Content user segment name.
usertest	Content user test name.
status	Content status code (“new”, “updated”, “scheduled”, “published”, “unpublished”, “expiring”, “expired”, “checkedout”, “workflow” (for any non-blank workflow status) or any configured workflow status name).
filename	Content filename or part of content filename.
attribute	Content attribute names to be searched (“title”, “content”, “summary”, “description”, “keywords”, “metainfo”, “htmlheader”) separated by commas or “” for all the attributes.
search_datetime	Date/time format string or substring.
search_datetime_attribute	“created”, “updated”, “published”, “unpublished”, “scheduled”, “expiring”, “deleted” or “” for all the attributes.
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example “offset=0&pagesize=10” for first 10 content items (1-10) and “offset=10&pagesize=10” for next 10 content items (11-20) and “offset=20&pagesize=10” for next 10 content items (21-30) etc. As default all matching content items will be listed.
sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
XML code	

10.3.2.3 Search and Replace

The “/webadmin/rest/content/searchreplace/” REST API can search for words/phrases in various content attributes and replace those words/phrases with other words/phrases.

10.3.2.3.1 Search

If “search” and “replace” parameters are given then a search results list of content items which match the “search” parameter and all other given parameters, as well as potential search and replace details for those content items will be returned. Note: This will not make any changes to any content items in the web content management system database – this will only show which search and replace changes can be made.



GET /webadmin/rest/content/searchreplace/?search=SEARCH&paste=PASTE&id=ID,ID,ID§ion=SECTION&class=CONTENTCLASS&group=CONTENTGROUP&type=CONTENTTYPE&bundle=CONTENTBUNDLE&package=CONTENTPACKAGE&version=VERSION&device=DEVICE&usersegment=USERSEGMENT&usertest=USERTEST&status=STATUS&filename=FILENAME&attribute=ATTRIBUTE&search_datetime=DATETIME&search_datetime_attribute=DATETIMEATTRIBUTE&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
search	Search word(s).
replace	Replace with word(s).
id	Content item ids separated by commas.
section	Content section name ("content", "page", "template", "stylesheet", "script", "library", "image", "file", "link", "ecommerce", "product" or any configured content element class name).
class	Content class name ("page", "template", "stylesheet", "script", "image", "file", "link", "product" or any configured content element class name).
group	Content group name or "-" for none.
type	Content type name or "-" for none.
bundle	Content bundle name.
package	Content package name.
version	Content version name or "-" for none.
device	Content device name.
usersegment	Content user segment name.
usertest	Content user test name.
status	Content status code ("new", "updated", "scheduled", "published", "unpublished", "expiring", "expired", "checkedout", "workflow" (for any non-blank workflow status) or any configured workflow status name).
filename	Content filename or part of content filename.
attribute	Content attribute names to be searched ("title", "content", "summary", "description", "keywords", "metainfo", "htmlheader") separated by commas or "" for all the attributes.
search_datetime	Date/time format string or substring.
search_datetime_attribute	"created", "updated", "published", "unpublished", "scheduled", "expiring", "deleted" or "" for all the attributes.
pagesize	Maximum number of content items to list. As default all matching content items will be listed.
offset	First content item number of listed content items to return. For example "offset=0&pagesize=10" for first 10 content items (1-10) and "offset=10&pagesize=10" for next 10 content items (11-20) and "offset=20&pagesize=10" for next 10 content items (21-30) etc. As default all matching content items will be listed.



sort_col	Content item attribute name to sort the listed content items by (“id”, “title”, “created”, “updated”, “published” etc).
sort_dir	“ASC” or “DESC” to sort the listed content items in ascending or descending order. As default content items will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "title": "ABC", "search_id": 1, "search_id_position_offset": "123_594_2_108_20", "search": ".....SEARCH.....", "replace": ".....REPLACE.....", "search_position": "594", "search_position_line": "2", "search_position_char": "108", "search_position_offset": "20", "_search_position": "2:108", "_search": ".....SEARCH.....", "_replace": ".....REPLACE.....", } , , { "_search": "SEARCH", "_replace": "REPLACE", "_replaced_count": "12", /* replace 12 times */ "_replaced_ids": "7" /* in 7 content items */ }]</pre>	

10.3.2.3.2 Replace

If “search” and “replace” parameters are given then a search results list of content items which match the “search” parameter and all other given parameters will be returned.

POST /webadmin/rest/content/searchreplace/	
search	Search word(s) (required).
replace	Replace with word(s) (required).
attribute	Content attribute names to be searched (“title”, “content”, “summary”, “description”, “keywords”, “metainfo”, “htmlheader”).
Request	
<pre>{ "search": "SEARCH", "replace": "REPLACE", "attribute": "content", "id": ["123", "456", "789"], "id_123": "123_594_2_108_20", "search_123": ".....SEARCH.....", "replace_123": ".....REPLACE.....", }</pre>	



<pre>"id_456": "456_365_6_45_32", "search_456": ".....SEARCH.....", "replace_456": ".....REPLACE.....", "id_789": "123_75_7_18_26", "search_789": ".....SEARCH.....", "replace_789": ".....REPLACE....." }</pre>
Response
<pre>{ "error": "" }</pre>

10.3.2.4 Read

If an “id” or “archive” parameter and no “search” parameter is given then a GET request will return that content item’s data. The returned data includes all the content item attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content item. Supplementary content titles, user names, stylesheets, elements, scripts, related contents, editors/publishers/administrators emails, admin status, “_webeditor_format”, “_webeditor”, “_webeditor_customscript”, “server_filename_extension”, “server_filename_path”, “server_filename_file”, product program options, hidden, versions, scheduled revisions, permissions flags, dependents count, timestamp attributes are also provided for website administration usage.

GET /webadmin/rest/content/?id=ID	
GET /webadmin/rest/content/?archive=ID	
id	Content item id number.
archive	Archived content item revision id number.
Response	
<pre>{ "id": "32", "created": "2009-11-30 09:20:27", "updated": "2013-08-08 13:42:37", "published": "2013-08-08 13:42:37", "unpublished": "", "deleted": "", "first_published": "2013-08-08 13:42:37", "created_by": "admin", "updated_by": "admin", "published_by": "admin", "unpublished_by": "", "deleted_by": "", "first_published_by": "admin", "revision": "", "archived": true, "scheduled": false, "title": "My Business", "_title": "My Business (32)", /* title + version + device + user segment + user test + id */ "content": "<div class=\"featureimage\">\r\n </div>\r\n<br class=\"clearalign\">", "summary": "", "contentformat": "",</pre>	



```
"searchable": "",
"menuitem": "",
"upload_filename": "",
"server_filename": "",
"server_filename_extension": "",
"server_filename_path": "",
"server_filename_file": "",

"bytesize": "1024",
"filesize": "1.00 KB",
"imagesize": "300x200",

"contentdelivery": "",
"url": "",
  "url": "", /* url for media library content item */
"_mediaclass": "", /* "gflash", "quicktime", "video", "audio", "file",
"image" */
  "_mediatype": "", /* "video/mp4", "video/ogg", "video/webm", "audio/mpeg",
"audio/wav", "audio/ogg" */

"template": "34",
"stylesheet": "",

"element": {
  "banner": "161",
  "breadcrumbs": "-1",
  "featurebox1": "-1",
  "featurebox2": "-1",
  "footer": "-1",
  "logo": "-1",
  "menu": "-1",
  "personal": "-1",
  "toolbar": "-1",
  "utilities": "-1"
},

"image1": "",
"image2": "",
"image3": "",
"file1": "",
"file2": "",
"file3": "",
"link1": "",
"link2": "",
"link3": "",

"author": "",
"keywords": "The homepage for My Business.",
"description": "What we do.",
"metainfo": "<language>EN</language>",
"metainfo_language": "EN",
"metainfos": {
  "language"; "EN"
}
"contentschema": "",
"segmentation": "",

"scripts": "",
"doctype": "",
"htmlattributes": "",
"headattributes": "",
"htmlheader": "",
"htmlbodyonload": "",

"contentpackage": "",
```



```
"contentbundle": "",
"contentclass": "page",
"contentgroup": "Home",
"contenttype": "",

"version_master": "0",
"version": "",
"device": "",
"usersegment": "",
"usertest": "",

"users_users": "",
"users_type": "",
"users_group": "",

"creators_users": "",
"creators_type": "",
"creators_group": "",
"editors_users": "",
"editors_type": "",
"editors_group": "",
"publishers_users": "",
"publishers_type": "",
"publishers_group": "",
"developers_users": "",
"developers_type": "",
"developers_group": "",
"administrators_users": "",
"administrators_type": "",
"administrators_group": "",

"page_top": "",
"page_top_title": "",
"page_up": "",
"page_up_title": "",
"page_previous": "0",
"page_previous_title": "",
"page_next": "531",
"page_next_title": "My Business",
"page_first": "32",
"page_first_title": "My Business",
"page_last": "570",
"page_last_title": "Online Butik",
"related": "",

"scheduled_publish": "",
"requested_publish": "",
"scheduled_unpublish": "",
"requested_unpublish": "",

"checkedout": "",
"_checkedout": false, /* checked out by currently logged in website
administrator */
"status": "",
"status_text": "",
"status_icon": "",
"status_by": "",
"status_user": "admin",

"workflowactionsoptions": "",
"workflowactions": [
]

"locked": 0,
"locked_user": 0,
```



```
"locked_creator": 0,
"locked_developer": 0,
"locked_editor": 0,
"locked_publisher": 0,
"locked_administrator": 0,
"locked_schedule": 0,
"locked_unschedule": 0,

"product_code": "",
"product_currency": "",
"product_currency_symbol": "",
"product_cost": "",
"product_price": "",
"product_period": "",
"product_weight": "",
"product_volume": "",
"product_width": "",
"product_height": "",
"product_depth": "",
"product_email": "",
"product_url": "",
"product_brand": "",
"product_colour": "",
"product_size": "",
"product_info": "",
"product_infos": {
}
"product_options": "",
"product_delivery": "",
"product_stock_amount": "0",
"product_stock_text": "",
"product_restocked_amount": "0",
"product_lowstock_amount": "",
"product_lowstock_text": "",
"product_nostock_order": "",
"product_nostock_text": "",
"product_location": "",
"product_user": "",
"product_page": "",
"product_program": "",
"product_hosting": "",
"product_content": "",
"product_contentgroup": "",
"product_contenttype": "",
"product_imagegroup": "",
"product_imagetype": "",
"product_filegroup": "",
"product_filetype": "",
"product_linkgroup": "",
"product_linktype": "",
"product_productgroup": "",
"product_producttype": "",
"product_usergroup": "",
"product_usertype": "",
"_product_stockcost": "",
"_product_restockcost": "",
"_product_stockvalue": "",
"_product_stockprofit": "",

"heatmap": "",
"heatmapalign": "",
"usagelog": "",

"_create": true,
"user": true,
```



```
"creator": true,  
"editor": true,  
"publisher": true,  
"developer": true,  
"administrator": true  
}
```

10.3.2.5 Create

A POST request will create a new content item with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the content item with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created content item’s data. The returned data includes all the content item attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content item.

POST /webadmin/rest/content/	
id	Content item id number to be copied (optional).
.....	All or any number and combination of content item attributes as used by the web content management system (Please see the 10.3.2.4 Read GET request example response).
file_data	Image or other file contents in Base64 encoded “data:” URL format. If a “file_data” parameter is given then “upload_filename” and “server_filename” parameters should also be given.
publish	“publish” If a “publish” parameter with a non-blank value is given then the created content item will be published.
archive	“archive” If an “archive” parameter with a non-blank value is given then the created content item will be archived.
Request	
{ "id": "123", "contentclass": "page", "contentgroup": "Home", "contenttype": "", "title": "My New Page", "content": "My new page content", "publish": "publish" }	
Request	
{ "contentclass": "image", "contentgroup": "", "contenttype": "Special", "title": "Blank image", "upload_filename": "blank.jpg", }	



```
"server_filename": "image/blank.jpg",
"file_data":
"data:image/jpeg;base64,\/9j\/4AAQSkZJRgABAQAAQABAAD\/2wBDAAMCAgICAgMCAgIDAw
MDBAYEBAQEBAgGBgUGCQgKCgkICQkKDA8MCgsOCwkJDRENDg8QEBEQCgwSExIQEw8QEBD\/2wBDAQMD
AwQDBAgEBAgQCwkLEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBA
AQEBD\/wAARCAABAAEDAREAAhEBAxEB\/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL\/
8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2Jyggk
KFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJi
pKtLJWl15iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLTlNXW19jZ2uHi4+Tl5ufo6erx8vP
09fb3+Pn6\/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL\/8QAtREAAgECBAQDBAcFBAQ
AAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2N
zg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaa
nqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6\/9oADAMBAAIR
AxEAPwD9U6AP\/9k=",
"publish": "publish"
}
```

Response

```
{
  "id": "789",
  "created": "2020-01-31 23:59:59",
  ....
  "user": true,
  "creator": true,
  "editor": true,
  "publisher": true,
  "developer": true,
  "administrator": false
}
```

10.3.2.6 Update

A PUT request will update the existing content item with the given “id” number with the given data attributes.

The PUT request will return the updated content item’s data. The returned data includes all the content item attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content item.

PUT /webadmin/rest/content/	
id	Content item id number to be updated (required).
.....	All or any number and combination of content item attributes as used by the web content management system (Please see the 10.3.2.4 Read GET request example response).
file_data	Image or other file contents in Base64 encoded “data:” URL format. If a “file_data” parameter is given then “upload_filename” and “server_filename” parameters should also be given.
publish	“publish” If a “publish” parameter with a non-blank value is given then the updated content item will be published.
archive	“archive” If an “archive” parameter with a non-blank value is



	given then the updated content item will be archived.
Request	
<pre>{ "id": "123", "contentclass": "page", "contentgroup": "Home", "contenttype": "", "title": "My New Page", "content": "My new page content", "publish": "publish" }</pre>	
Request	
<pre>{ "id": "456", "contentclass": "image", "contentgroup": "", "contenttype": "Special", "title": "Blank image", "upload_filename": "blank.jpg", "server_filename": "image/blank.jpg", "file_data": "data:image/jpeg;base64,\/9j\/4AAQSkZJRgABAQAAQABAAQ\/2wBDAAMCAgICAgMCAgIDAw MDBAYEBAQEBAgGBgUGCQgKCgkICQkKDA8MCgsOCwkJDRENDg8QEBEQCgwSExIQEw8QEBD\/2wBDAQMD AwQDBAgEBAgQCwkLEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBA AQEBD\/wAARCAABAEDAREAAhEBAxEB\/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL\/ 8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2Jyggk KFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJi pKtLJWJW15iZmgKjpKWWp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP 09fb3+Pn6\/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL\/8QAtREAAgECBAQDBAcFBAQ AAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2N zg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eH16goOEhYaHiImKkpOUlZaXmJmaoqOkpaa nqKmsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6\/9oADAMBAAIR AxEAPwD9U6AP\/9k=", "publish": "publish" }</pre>	
Response	
<pre>{ "id": "789", "created": "2020-01-01 00:00:00", "updated": "2020-01-31 23:59:59", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": false }</pre>	

10.3.2.7 Delete and Unpublish

A DELETE request will delete or unpublish the existing content item with the given “id” number.

The DELETE request will return the deleted/unpublished content item’s data. The returned data includes all the content item attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content item.



DELETE /webadmin/rest/content/	
archive	Content item archive id number If a content item archive id is given then that archived content item is deleted. The published and the draft and other archived revisions of the content item are not deleted.
id	Content item id number If a content item id is given then that content item is deleted. As default both the published and the draft and all archived revisions of the content item are deleted.
unpublish	“unpublish” If a non-blank “unpublish” parameter is given then the given content item id is only unpublished. The draft and all archived revisions of the content item are not deleted.
Request	
<pre>{ "id": "123" }</pre>	
Request	
<pre>{ "id": "123", "unpublish": "unpublish" }</pre>	
Request	
<pre>{ "archive": "456" }</pre>	
Response	
<pre>{ "id": "123", "created": "2020-01-01 00:00:00", "updated": "2020-01-31 23:59:59", "unpublished": "2020-12-31 23:59:59", , "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": false }</pre>	

10.3.2.8 Archive

The “/webadmin/rest/content/archive/” REST API function archives the given content items.

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/archive/	
id	Content item id number to be copied (required).
Request	
<pre>{ "id": "123" }</pre>	



Request
<pre>{ "id": ["123", "456", "789"] }</pre>
Response
<pre>{ "error": "XXXXX" }</pre>

10.3.2.9 Checkin

The “/webadmin/rest/content/checkin/” REST API function checkins the given content items.

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/checkin/	
id	Content item id number to checkin (required).
Request	
<pre>{ "id": "123" }</pre>	
Request	
<pre>{ "id": ["123", "456", "789"] }</pre>	
Response	
<pre>{ "error": "XXXXX" }</pre>	

10.3.2.10 Checkout

The “/webadmin/rest/content/checkout/” REST API function checkouts the given content items.

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/checkin/	
id	Content item id number to checkout (required).
Request	
<pre>{ "id": "123" }</pre>	
Request	
<pre>{ "id": ["123", "456", "789"] }</pre>	
Response	
<pre>{ "error": "XXXXX" }</pre>	

10.3.2.11 Copy

The “/webadmin/rest/content/copy/” REST API function copies the given content items.



The POST request returns an error response or “OK” and a list of the copied content items ids.

POST /webadmin/rest/content/copy/	
id	Content item id number to be copied (required).
contentbundle	Content bundle name for copied content (optional).
contentversion	Content version name for copied content (optional).
contentgroup	Content group name for copied content (optional).
contenttype	Content type name for copied content (optional).
contentpackage	Content package name for copied content (optional).
folder	Content folder name for copied content (optional).
publish	“publish” If a “publish” parameter with a non-blank value is given then the copied content items will be published.
archive	“archive” If an “archive” parameter with a non-blank value is given then the copied content items will be archived.
scheduled_publish	Scheduled publish date/time stamp (optional).
scheduled_unpublish	Scheduled unpublish date/time stamp (optional).
status	Workflow action id number (optional).
revision	Content revision text (optional).
Request	
<pre>{ "id": "123", "contentclass": "page", "contentgroup": "Home", "contenttype": "", "publish": "publish" }</pre>	
Request	
<pre>{ "id": ["123", "456", "789"], "contentclass": "page", "contentgroup": "Home", "contenttype": "", "publish": "publish" }</pre>	
Response	
<pre>{ "error": "XXXXXX", "copied": [{ "id": "123", "copy": "1001" } , { "id": "456", "copy": "1002" } , { "id": "789", "copy": "1003" }] }</pre>	

10.3.2.12 Delete (DEPRECATED)

The “/webadmin/rest/content/delete/” REST API function deletes the given content items.

The POST request returns an error response or “OK”.



Note: DEPRECATED – use DELETE /webadmin/rest/content/ (Please see section 10.3.2.7 Delete).

POST /webadmin/rest/content/delete/	
id	Content item id number to be deleted (required).
delete	“delete” If a “delete” parameter with a non-blank value is given then the given content items will be deleted (required).
Request	
<pre>{ "id": "123", "delete": "delete" }</pre>	
Request	
<pre>{ "id": ["123", "456", "789"] "delete": "delete" }</pre>	
Response	
<pre>{ "error": "XXXXX" }</pre>	

10.3.2.13 Move

The “/webadmin/rest/content/move/” REST API function moves the given content items.

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/move/	
id	Content item id number to be moved (required).
contentbundle	Content bundle name for moved content (optional).
contentversion	Content version name for moved content (optional).
contentgroup	Content group name for moved content (optional).
contenttype	Content type name for moved content (optional).
contentpackage	Content package name for moved content (optional).
folder	Content folder name for moved content (optional).
publish	“publish” If a “publish” parameter with a non-blank value is given then the copied content items will be published.
archive	“archive” If an “archive” parameter with a non-blank value is given then the copied content items will be archived.
scheduled_publish	Scheduled publish date/time stamp (optional).
scheduled_unpublish	Scheduled unpublish date/time stamp (optional).
status	Workflow action id number (optional).
revision	Content revision text (optional).
Request	
<pre>{ "id": "123", "contentclass": "page", "contentgroup": "Home", "contenttype": "", </pre>	



<pre>"publish": "publish" }</pre>
Request
<pre>{ "id": ["123", "456", "789"], "contentclass": "page", "contentgroup": "Home", "contenttype": "", "publish": "publish" }</pre>
Response
<pre>{ "error": "XXXXX" }</pre>

10.3.2.14 Publish

The “/webadmin/rest/content/publish/” REST API function publishes the given content items.

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/publish/	
id	Content item id number to be published (required).
scheduled_publish	Scheduled publish date/time stamp (optional).
scheduled_unpublish	Scheduled unpublish date/time stamp (optional).
status	Workflow action id number (optional).
revision	Content revision text (optional).
Request	
<pre>{ "id": "123", "delete": "delete" }</pre>	
Request	
<pre>{ "id": ["123", "456", "789"] "unpublish": "unpublish" }</pre>	
Response	
<pre>{ "error": "XXXXX" }</pre>	

10.3.2.15 Unpublish (DEPRECATED)

The “/webadmin/rest/content/unpublish/” REST API function unpublishes the given content items.

The POST request returns an error response or “OK”.

Note: DEPRECATED – use DELETE /webadmin/rest/content/?unpublish=unpublish (Please see section 10.3.2.7 Delete).

POST /webadmin/rest/content/delete/



id	Content item id number to be deleted or unpublished (required).
unpublish	“unpublish” If an “unpublish” parameter with a non-blank value is given then the given content items will be unpublished (required).
Request	
{ "id": "123", "unpublish": "unpublish" }	
Request	
{ "id": ["123", "456", "789"] "unpublish": "unpublish" }	
Response	
{ "error": "XXXXX" }	

10.3.2.16 Heatmap

The “/webadmin/rest/content/heatmap/” REST API function configures heatmap functionality for the given content items.

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/heatmap/	
id	Content item id number to be deleted or unpublished (required).
log	“delete” Delete all current heatmap log data for the given content items (optional).
log	“click”, “mouse”, “fold”, “reach” in any combination separated by commas. If a “log” parameter with a non-blank and non-“delete” value is given then the given content items will be configured with the given log value (optional).
align	“c”, “l”, “r” or “f” (for center, left, right or full). If an “align” parameter with a non-blank value is given then the given content items will be configured with the given align value (optional).
Request	
{ "id": "123", "log": "click,mouse,fold,reach", "align": "l" }	
Request	
{ "id": ["123", "456", "789"] "log": "click", "align": "f" }	



Response
<pre>{ "error": "XXXXXX" }</pre>

10.3.2.17 Usersegment

The “/webadmin/rest/content/usersegment/” REST API function gets and sets the user segment for the given content items.

The GET request returns the given content item id’s user segment and other content variant details.

GET /webadmin/rest/content/usersegment/	
id	Content item id number to be read (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "XXXXXX", "version_master": "XXXXXX", "version": "XXXXXX", "device": "XXXXXX", "usersegment": "XXXXXX", "usertest": "XXXXXX", "title": "XXXXXX" }</pre>	

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/usersegment/	
id	Content item id number to be updated (required).
usersegment	User segment name for the given content items (required).
Request	
<pre>{ "id": "123", "usersegment": "XXXXXX" }</pre>	
Request	
<pre>{ "id": ["123", "456", "789"], "usersegment": "XXXXXX" }</pre>	
Response	
<pre>{ "error": "XXXXXX" }</pre>	

10.3.2.18 Usertest

The “/webadmin/rest/content/usertest/” REST API function gets and sets the user test for the given content items.



The GET request returns the given content item id's user test and other content variant details.

GET /webadmin/rest/content/usertest/	
id	Content item id number to be read (required).
Request	
{ "id": "123" }	
Response	
{ "id": "XXXXX", "version_master": "XXXXX", "version": "XXXXX", "device": "XXXXX", "usersegment": "XXXXX", "usertest": "XXXXX", "title": "XXXXX" }	

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/usertest/	
id	Content item id number to be updated (required).
usertest	User test name for the given content items (required).
Request	
{ "id": "123", "usertest": "XXXXX" }	
Request	
{ "id": ["123", "456", "789"], "usertest": "XXXXX" }	
Response	
{ "error": "XXXXX" }	

10.3.2.19 Checkloops

The “/webadmin/rest/content/checkloops/” REST API function checks for dependencies and special code loops in the given content item or HTML code.

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/checkloops/	
id	Content item id number (optional).
content	HTML code (optional)
Request	
{ "id": "123" }	
Request	
{	



<pre>"id": ["123", "456", "789"] }</pre>
Request
<pre>{ "content": "<html>....</html>" }</pre>
Response
<pre>{ "error": "XXXXX" }</pre>

10.3.2.20 Checklinks

The “/webadmin/rest/content/checklinks/” REST API function checks links in the given content items or HTML code.

The POST request returns an error response or “OK”.

POST /webadmin/rest/content/checklinks/	
id	Content item id number (optional).
content	HTML code (optional)
Request	
<pre>{ "id": "123" }</pre>	
Request	
<pre>{ "id": ["123", "456", "789"] }</pre>	
Request	
<pre>{ "content": "<html>....</html>" }</pre>	
Response	
<pre>{ "error": "XXXXX" }</pre>	

10.3.2.21 Dependencies

The “/webadmin/rest/content/dependencies/” REST API function lists the content dependencies for the given content item.

The GET request returns an error response or “OK”.

GET /webadmin/rest/content/dependencies/	
id	Content item id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "settings": [], "specialpages": [</pre>	



```
    { "id": "default_page", "specialpage":
"[[config.website.special.page]]" }
  ],
  "users": [ ],
  "usergroups": [ ],
  "usertypes": [ ],
  "websites": [ ],
  "pages": [
    { "id": "723", "title": "About Us" },
    { "id": "724", "title": "About Us" }
  ],
  "products": [ ],
  "images": [ ],
  "files": [ ],
  "links": [ ],
  "elements": [ ],
  "templates": [ ],
  "stylesheets": [ ],
  "scripts": [ ],
  "contentgroups": [ ],
  "contenttypes": [ ],
  "productgroups": [ ],
  "producttypes": [ ],

  "settings_html": "",
  "specialpages_html": "<p>[[config.website.special.page]]</p>\r\n",
  "users_html": "",
  "usergroups_html": "",
  "usertypes_html": "",
  "websites_html": "",
  "pages_html": "<p>About Us [723]</p>\r\n<p>About Us [724]</p>\r\n",
  "products_html": "",
  "images_html": "",
  "files_html": "",
  "links_html": "",
  "elements_html": "",
  "templates_html": "",
  "stylesheets_html": "",
  "scripts_html": "",
  "contentgroups_html": "",
  "contenttypes_html": "",
  "productgroups_html": "",
  "producttypes_html": "",

  "dependencies": [
    { "id": "0", "dependencies": "|default_page|" },
    { "id": "723", "dependencies": "|page_top|page_up|page_top|page_up|"
  },
    { "id": "724", "dependencies": "|page_top|page_up|page_top|page_up|"
  }
  ]
}
```

10.3.2.22 Content Version Device Options

The “/webadmin/rest/devices/” REST API functions returns content version device options available for the configuration of content version variant content items etc.

GET /webadmin/rest/devices/	
Response	



```
[
  { "id": "", "name": "----- Devices -----" },
  { "id": "Device", "name": "Device" },
  { "id": "Phone", "name": "Phone" },
  { "id": "Tablet", "name": "Tablet" },
  { "id": "iPad", "name": "Apple iPad" },
  { "id": "iPhone", "name": "Apple iPhone" },
  .....
]
```

10.3.2.23 Content Preview Simulator Options

The “/webadmin/rest/simulators/” REST API functions returns simulator options available for previewing content items etc.

GET /webadmin/rest/simulators/	
Response	
[{ "id": "100 iPad", "name": "iPad" }, { "id": "110 iPad Mini", "name": "iPad Mini" }, { "id": "200 iPhone 4-4S", "name": "iPhone 4 \\/ 4S" }, { "id": "210 iPhone 5-5S-5C-SE", "name": "iPhone 5 \\/ 5S \\/ 5C \\/ SE" }, { "id": "220 iPhone 6-6S-7-8-SE2", "name": "iPhone 6 \\/ 6S \\/ 7 \\/ 8 \\/ SE2" }, { "id": "230 iPhone 6-6S-7-8 Plus", "name": "iPhone 6 \\/ 6S \\/ 7 \\/ 8 Plus" }, { "id": "240 iPhone X-XS-11 Pro", "name": "iPhone X \\/ XS \\/ 11 Pro" }, { "id": "250 iPhone XS-11 Pro Max", "name": "iPhone XS \\/ 11 Pro Max" }]	

10.3.2.24 Content Editor Options

The “/webadmin/rest/webeditors/” REST API functions returns web content editor options available for editing content items etc.

GET /webadmin/rest/webeditors/	
all	“all” If an “all” parameter with a non-blank value is given then all web content editors including the standard included web content management system web content editors will be listed. Otherwise, only custom/third-party web content editors will be listed (optional).
Response	
[{ "webeditor": "HardCore" }, { "webeditor": "HardCore1" }, { "webeditor": "HardCore2" }, { "webeditor": "textarea" }, { "webeditor": "WebEditorAPI" }]	

10.3.2.25 Metadata

If no “contentclass” parameter is given then metadata for all content will be returned.



GET /webadmin/rest/content/metadata/?contentclass=CONTENTCLASS	
contentclass	“page” or “product”
Response	
<pre>[{ "id": "154", "contentclass": "page", "contentgroup": "Store Locator", "contenttype": "", "version": "", "title": "@@include:database=Stores:id###id###:Store Name@@@", "author": "", "description": "", "keywords": "", "metainfo": "....." }, { "id": "657", "contentclass": "page", "contentgroup": "Store Locator", "contenttype": "", "version": "", "title": "@@include:database=Stores:id###id###:Store Name@@@", "author": "", "description": "", "keywords": "", "metainfo": "....." },]</pre>	

GET /webadmin/rest/content/metadata/csv/?contentclass=CONTENTCLASS	
contentclass	“page” or “product”
Response	
<pre>id,contentclass,contentgroup,contenttype,title,author,description,keywords,met ainfo 154,page,Store Locator,,@@include:database=Stores:id###id###:Store Name@@@,,,,..... 657,page,Store Locator,,@@include:database=Stores:id###id###:Store Name@@@,,,,.....</pre>	

10.3.2.26 Preview

The “/webadmin/rest/content/preview/” REST API function previews the posted content data as a web page and returns the full, parsed web page including its template etc. split into the individual HTML page parts as JSON data; optionally as HTML code.

POST /webadmin/rest/content/preview/	
.....	All or any number and combination of content item attributes as used by the web content management system (Please see the 10.3.2.4 Read GET request example response).
Request	
<pre>{ "id": "123", "contentclass": "page", "title": "My New Page",</pre>	



.....

<pre>"content": "My new page content", "publish": "publish" }</pre>	
Response	
<pre>{ "doctype": "<!DOCTYPE html>", "html": "", "head": "", "metainfo": "<meta name=\"language\" content=\"EN\">", "author": "", "description": "", "keywords": "", "title": " My New Page ", "stylesheets": "<link rel=\"stylesheet\" type=\"text/css\" href=\"\"/stylesheet.jsp?id=1\">\r\n\r\n", "scripts": "", "htmlheader": "", "contentschema": "", "htmlbodyonload": "", "body": "<div id=\"wrapper_outer\">\r\n <div id=\"wrapper_inner\">\r\n My new page content </div>\r\n </div>" }</pre>	

POST /webadmin/rest/content/preview/html/	
.....	All or any number and combination of content item attributes as used by the web content management system (Please see the 10.3.2.4 Read GET request example response).
Request	
<pre>{ "id": "123", "contentclass": "page", "title": "My New Page", "content": "My new page content", "publish": "publish" }</pre>	
Response	
HTML code	

10.3.2.27 Validate Markup

The “/webadmin/rest/content/validatemarkup/html/” REST API function validates HTML and CSS code using the W3C validator services and returns the generated web page report(s) from the W3C validator services. Optionally, content can simply be prepared/previewed for validation to be posted to the W3C validator services (or other) manually or programmatically.

POST /webadmin/rest/content/validatemarkup/proxy/	
.....	All or any number and combination of content item attributes as used by the web content management system (Please see the 10.3.2.4 Read GET request example response).
Request	
{	



<pre>"id": "123", "contentclass": "page", "title": "My New Page", "content": "My new page content", "publish": "publish" }</pre>
Response
HTML code (generated web page report from W3C validator service).

GET /webadmin/rest/content/validatemarkup/?id=ID&id=ID&id=ID	
id	One or more content item id numbers (required).
Request	
<pre>{ "id": ["123", "456", "789"] }</pre>	
Response	
HTML code (generated web page report from W3C validator service).	

POST /webadmin/rest/content/validatemarkup/	
.....	All or any number and combination of content item attributes as used by the web content management system (Please see the 10.3.2.4 Read GET request example response).
Request	
<pre>{ "id": "123", "contentclass": "page", "title": "My New Page", "content": "My new page content", "publish": "publish" }</pre>	
Response	
HTML code (from web content management system to be posted to W3C validator service).	

10.3.2.28 Stylesheets

The “/webadmin/rest/content/stylesheets/” REST API function gets the stylesheet ids to be used for a content item.

GET /webadmin/rest/content/stylesheets/?id=ID&contentclass=CONTENTCLASS&contentgroup=CONTENTGROUP&contenttype=CONTENTTYPE	
id	Content item number (optional).
contentclass	“page”, “product” or “template”.
contentgroup	Content group name.
contenttype	Content type name.
Response	
<pre>{ "ids": "123,456,789" }</pre>	



10.3.2.29 Product Stock

The “/webadmin/rest/content/stock/” REST API function sets, adds to or subtracts from a product content item’s stock amount and returns the new stock amount, status and comment for the product content item.

POST /webadmin/rest/content/stock/?id=ID&product_stock_amount_update=PRODUCTSTOCKAMOUNTUPDATE&product_stock_amount=PRODUCTSTOCKAMOUNT	
id	Content item number (required).
product_stock_amount_update	“set”, “add” or “subtract” (required).
product_stock_amount	Stock amount number (required).
Response	
{ "id": "123", "amount": "456", "status": "in", "comment": "In Stock", }	

10.3.2.30 Workflow Username Options

The “/webadmin/rest/content/workflow_username_options/html/” REST API function returns workflow username options in HTML format for a given content item id and workflow action.

GET /webadmin/rest/content/workflow_username_options/html/?id=ID&contentclass=CONTENTCLASS&contentgroup=CONTENTGROUP&contenttype=CONTENTTYPE&version=VERSION&workflow=WORKFLOW	
id	Content item id number (optional).
contentclass	Content class name.
contentgroup	Content group name.
contenttype	Content type name.
version	Content version name.
workflow	Workflow action id number.
Response	
HTML code	

10.3.2.31 Structure

The “/webadmin/rest/content/structure/” REST API function returns and updates the page relations for one or more content items as for example reorganised using the web content management website structure administration functionality.

10.3.2.31.1 List

The “/webadmin/rest/content/structure/html/” REST API function returns the “root” or “child” content items from the website structure page relations as HTML code list items.

GET /webadmin/rest/content/structure/html/?mode=structure&id=ID



mode	“structure” (exclude version and device and other variants as well as additional content).
id	Parent content item id (for example “content_123”) for sub-content items of the given content item id.
GET /webadmin/rest/content/structure/html/?mode=structure&id=&root=ROOT	
mode	“structure” (exclude version and device and other variants as well as additional content).
id	“” or “0” or “-“ for “root” content items.
root	Root content item id number; otherwise the configured “default page” configuration setting.
GET /webadmin/rest/content/structure/html/?mode=structure&id=000	
id	“000” for “orphan” content items.
Response	
HTML code list items	

10.3.2.31.2 Update

The “/webadmin/rest/content/structure/” REST API function updates the page relations for one or more content items as used for the website structure.

Note: In addition to the given website structure page relations changes, other necessary cascading changes to the website structure page relations may also be made for consistency with the given website structure page relations changes.

POST /webadmin/rest/content/structure/	
page_top_ID	Content item id number to be set as the “page_top” of the content item with the given ID content item id number.
page_up_ID	Content item id number to be set as the “page_up” attribute of the content item with the given ID content item id number.
page_first_ID	Content item id number to be set as the “page_first” attribute of the content item with the given ID content item id number.
page_previous_ID	Content item id number to be set as the “page_previous” attribute of the content item with the given ID content item id number.
page_next_ID	Content item id number to be set as the “page_next” attribute of the content item with the given ID content item id number.
page_last_ID	Content item id number to be set as the “page_last” attribute of the content item with the given ID content item id number.
archive	“*” to “archive” the content items with/after the given page relations changes.
publish	“*” to “publish” the given page relations changes; otherwise the given page relations changes will only be made for the “current” draft content items.
Request	
{ "page_top_123": "789", "page_up_123": "789",	



<pre>"page_next_123": "456", "page_last_123": "456", "page_top_456": "789", "page_up_456": "789", "page_previous_456": "123", "page_first_456": "123" "archive": "*", "publish": "*" }</pre>
Response
<pre>{ "error": "", "output": "....." }</pre>

10.3.2.32 Create (DEPRECATED)

A GET/POST request will return the HTML code OPTIONS for the web content management system's Add New content item functionality.

Note: DEPRECATED – use GET /webadmin/rest/content/?index=create (Please see section 10.3.2.1.4 Create Options).

POST /webadmin/rest/content/create/?contentclass=CONTENTCLASS&contentgroup=CONTENTGROUP&contenttype=CONTENTTYPE&contentbundle=CONTENTBUNDLE&contentpackage=CONTENTPACKAGE&version=VERSION&status=STATUS&stock=STOCK	
contentclass	Content class name (“page”, “template”, “stylesheet”, “script”, “image”, “file”, “link”, “product” or any configured content element class name).
contentgroup	Content group name or “-“ for none.
contenttype	Content type name or “-“ for none.
contentbundle	Content bundle name.
contentpackage	Content package name.
version	Content version name or “-“ for none.
status	Content status code (“new”, “updated”, “scheduled”, “published”, “unpublished”, “expiring”, “expired”, “checkedout”).
stock	Product stock status code (“unlimited”, “in”, “low”, “no”, “orderable”, “ordered”).
Response	
HTML code	

10.3.2.33 Post (DEPRECATED)

A POST request will update the existing content item with the given “id” number with the given data attributes.



The POST request returns an error response or “OK”.

Note: DEPRECATED – use PUT /webadmin/rest/content/ (Please see section 10.3.2.6 Update).

POST /webadmin/rest/content/post/	
id	Content item id number to be updated (required).
.....	All or any number and combination of content item attributes as used by the web content management system (Please see the 10.3.2.4 Read GET request example response).
file_data	Image or other file contents in Base64 encoded “data:” URL format. If a “file_data” parameter is given then “upload_filename” and “server_filename” parameters should also be given.
publish	“publish” If a “publish” parameter with a non-blank value is given then the updated content item will be published.
archive	“archive” If an “archive” parameter with a non-blank value is given then the updated content item will be archived.
Request	
<pre>{ "id": "123", "contentclass": "page", "contentgroup": "Home", "contenttype": "", "title": "My New Page", "content": "My new page content", "publish": "publish" }</pre>	
Request	
<pre>{ "id": "456", "contentclass": "image", "contentgroup": "", "contenttype": "Special", "title": "Blank image", "upload_filename": "blank.jpg", "server_filename": "image/blank.jpg", "file_data": "data:image/jpeg;base64,\\9j\\4AAQSkZJRgABAQAAQABAAD\\2wBDAAMCAgICAgMCAgIDAw MDBAYEBAQEBAgGBGUGCQgKCgkICQkKDA8MCgsOCwkJDRENDg8QEBEQCgwSEwIQEw8QEBD/2wBDAQMD AwQDBAgEBAGQCwkLEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBA AQEBD\\wAARCAABAAEDAREAAhEBAxEB\\8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL\\8QAt RAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcY GRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXQhIiWGH4iJi pKtLJ JWw15iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+ Pn6\\8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL\\8QAtREAAgECBAQDBAcFBAQAAQJ3AAECA xEEBSExBhJBUCdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSEl KU1RVVldYWVpjZGVmZ2hpanN0dXZ3eH16goOEhYaHiImKkpOUlZaXmJmaoqOkpaa nqKmqsrO0tba3uLm6 wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6\\9oADAMBAAIRAxEAPwD9U6AP\\9k=", "publish": "publish" }</pre>	



Response
<pre>{ "error": "OK" }</pre>

10.3.2.34 Print

The “/webadmin/rest/content/print/” REST API function generates a “printing” special page report as a web page and returns the full, parsed web page including its template etc. split into the individual HTML page parts as JSON data; optionally as HTML code.

POST /webadmin/rest/content/print/?print=PRINT	
print	Page content item id number (required).
Response	
<pre>{ "doctype": "<!DOCTYPE html>", "html": "", "head": "", "metainfo": "<meta name=\"language\" content=\"EN\">", "author": "", "description": "", "keywords": "", "title": " My Report Page ", "stylesheets": "<link rel=\"stylesheet\" type=\"text/css\" href=\"\"/stylesheet.jsp?id=1\">\r\n\r\n", "scripts": "", "htmlheader": "", "contentschema": "", "htmlbodyonload": "", "body": "<div id=\"wrapper_outer\">\r\n <div id=\"wrapper_inner\">\r\n My report page content </div>\r\n </div>" }</pre>	

POST /webadmin/rest/content/print/html/?print=PRINT	
print	Page content item id number (required).
Response	
HTML code	

10.3.2.35 Count (Contents To Be Deleted)

The “/webadmin/rest/content/count/” REST API function returns the number of content items that will be deleted given various configuration settings (as used for Archiving Deletion configuration in the web content management system).

GET /webadmin/rest/content/count/	
count	“archived”, “deleted”, “expired”, “new”, “published” (required).
Request	
<pre>{ "count": "archived", "use_delete_archived_days": "365" }</pre>	
Request	
<pre>{ "count": "deleted",</pre>	



<pre>"use_delete_deleted_days": "365" }</pre>
Request
<pre>{ "count": "expired", "used_delete_expired_days": "365" }</pre>
Request
<pre>{ "count": "new", "use_delete_new_days": "365" }</pre>
Request
<pre>{ "count": "published", "use_delete_published_days": "365", "use_delete_published_groups": ["Forum", "...", "Contact Forms"], "use_delete_published_types": ["Messages", "...", "Feedback"] }</pre>
Response
<pre>{ "count": "123" }</pre>

10.3.2.36 Export Static Files

The “/webadmin/rest/content/export/html/” REST API function exports all content items as static HTML, CSS, Javascript and other files to the configured “export_rootpath” configuration setting (if any).

GET /webadmin/rest/content/export/html/	
Response	
HTML code	

10.3.2.37 Folders (File System)

The “/webadmin/rest/content/folders/” REST API function returns a list of folders and/or files under the website “home”/“root” folder.

GET /webadmin/rest/content/folders/	
path	“” or folder/path name (for example “image/icons”).
recursive	“yes” to also list sub-folders and sub-sub-folders etc.
folders	“yes” to list folders.
files	“yes” to list files.
GET /webadmin/rest/content/folders/?path=&files=yes	
Response	
<pre>{ "entries": [{ "path": "AAA.html", "file": "AAA.html" }, { "path": "BBB.html", "file": "BBB.html" }, { "path": "CCC.html", "file": "CCC.html" }, ] }</pre>	



<pre>{ "path": "XXX.gif", "file": "XXX.gif" }, { "path": "YYY.gif", "file": "YYY.gif" }, { "path": "ZZZ.gif", "file": "ZZZ.gif" }] }</pre>
GET /webadmin/rest/content/folders/?path=image%2F&files=yes
Response
<pre>{ "entries": [{ "path": "image\Calendaricon_V1.gif", "file": "Calendaricon_V1.gif" }, { "path": "image\Feedback_small.gif", "file": "Feedback_small.gif" }, { "path": "image\Google_maps.gif", "file": "Google_maps.gif" }, { "path": "image\sharechart_small_1.jpg", "file": "sharechart_small_1.jpg" }, { "path": "image\templateshadow_left.jpg", "file": "templateshadow_left.jpg" }, { "path": "image\templateshadow_right.jpg", "file": "templateshadow_right.jpg" }] }</pre>
GET /webadmin/rest/content/folders/?path=&recursive=yes&folders=yes
Response
<pre>{ "entries": [{ "path": "AAA/", "folder": "AAA", "entries": [{ "path": "AAA\BBB/", "folder": "BBB", "entries": [{ "path": "AAA\BBB\CCC/", "folder": "CCC", "entries": null }] }, { "path": "EXPORT/", "folder": "EXPORT", "entries": [{ "path": "EXPORT\file/", "folder": "file", "entries": null }, { "path": "EXPORT\image/", "folder": "image", "entries": null }], { "path": "file/", "folder": "file", "entries": null }, { "path": "image/", "folder": "image", "entries": null }] }</pre>
GET /webadmin/rest/content/folders/?path=&recursive=yes&folders=yes&files=yes
Response
<pre>{ "entries": [{ "path": "EXPORT/", "folder": "EXPORT", "entries": [{ "path": "EXPORT\file/", "folder": "file", "entries": [{ "path": "EXPORT\file\Agenda_AGM_2021_10_27.pdf", "file": "Agenda_AGM_2021_10_27.pdf" }, { "path": "EXPORT\file\Annual Report 2020.pdf", "file": "Annual Report 2020.pdf" }, { "path": "EXPORT\file\Annual Report 2021.pdf", "file": "Annual Report 2021.pdf" }, { "path": "EXPORT\file\Product1 Installation.pdf", "file": "Product1 Installation.pdf" }, { "path": "EXPORT\file\Product1 Manual.pdf", "file": "Product1 Manual.pdf" }, { "path": "EXPORT\file\productadriver.zip", "file": "productadriver.zip" }] }, { "path": "EXPORT\image/", "folder": "image", "entries": [{ "path": "EXPORT\image\Calendaricon_V1.gif", "file": "Calendaricon_V1.gif" }, { "path": "EXPORT\image\Feedback_small.gif", "file": "Feedback_small.gif" },]] }</pre>



```
{ "path": "EXPORT\\image\\Google_maps.gif", "file": "Google_maps.gif"
},
.....
{ "path": "EXPORT\\image\\sharechart_small_1.jpg", "file":
"sharechart_small_1.jpg" },
{ "path": "EXPORT\\image\\templateshadow_left.jpg", "file":
"templateshadow_left.jpg" },
{ "path": "EXPORT\\image\\templateshadow_right.jpg", "file":
"templateshadow_right.jpg" }
] },
{ "path": "EXPORT\\page_104.html", "file": "page_104.html" },
{ "path": "EXPORT\\page_108.html", "file": "page_108.html" },
{ "path": "EXPORT\\page_109.html", "file": "page_109.html" },
.....
{ "path": "EXPORT\\script_362.js", "file": "script_362.js" },
{ "path": "EXPORT\\stylesheet_1.css", "file": "stylesheet_1.css" },
{ "path": "EXPORT\\stylesheet_634.css", "file": "stylesheet_634.css" }
] },
{ "path": "archive.xml", "file": "archive.xml" },
{ "path": "atom.jsp", "file": "atom.jsp" },
.....
{ "path": "error.html", "file": "error.html" },
{ "path": "error.jsp", "file": "error.jsp" },
{ "path": "favicon.ico", "file": "favicon.ico" },
{ "path": "file/", "folder": "file", "entries": [
{ "path": "file\\Agenda_AGM_2021_10_27.pdf", "file":
"Agenda_AGM_2021_10_27.pdf" },
{ "path": "file\\Annual Report 2020.pdf", "file": "Annual Report
2020.pdf" },
{ "path": "file\\Annual Report 2021.pdf", "file": "Annual Report
2021.pdf" },
.....
{ "path": "file\\Product1 Installation.pdf", "file": "Product1
Installation.pdf" },
{ "path": "file\\Product1 Manual.pdf", "file": "Product1 Manual.pdf" },
{ "path": "file\\productadriver.zip", "file": "productadriver.zip" }
] },
{ "path": "file.jsp", "file": "file.jsp" },
{ "path": "guestbook.jsp", "file": "guestbook.jsp" },
{ "path": "heatmap.js.jsp", "file": "heatmap.js.jsp" },
{ "path": "heatmap.jsp", "file": "heatmap.jsp" },
{ "path": "image/", "folder": "image", "entries": [
{ "path": "image\\Calendaricon V1.gif", "file": "Calendaricon V1.gif" },
{ "path": "image\\Feedback_small.gif", "file": "Feedback_small.gif" },
{ "path": "image\\Google_maps.gif", "file": "Google_maps.gif" },
.....
{ "path": "image\\sharechart_small_1.jpg", "file":
"sharechart_small_1.jpg" },
{ "path": "image\\templateshadow_left.jpg", "file":
"templateshadow_left.jpg" },
{ "path": "image\\templateshadow_right.jpg", "file":
"templateshadow_right.jpg" }
] },
{ "path": "image.jsp", "file": "image.jsp" },
{ "path": "index.jsp", "file": "index.jsp" },
.....
{ "path": "post.jsp", "file": "post.jsp" },
{ "path": "product.jsp", "file": "product.jsp" },
.....
{ "path": "webadmin.jsp", "file": "webadmin.jsp" },
{ "path": "xml.jsp", "file": "xml.jsp" }
}
}
```



10.3.2.38 Import (Text File Contents)

The “/webadmin/rest/content/import/” REST API function outputs the contents of an uploaded file - which can then be copied into a web content editor input field etc.

Note: Only text content such as plain text, HTML, CSS and Javascript etc. is supported.

POST /webadmin/rest/content/import/	
file	HTML FORM INPUT type=”file” name=”file
Response	
Text	

10.3.2.39 Spell Check

The “/webadmin/rest/content/spellcheck/” REST API function spell checks text content and returns a list of misspelled words with suggestions for corrections.

Note: Only text content such as plain text and HTML etc. is supported.

POST /webadmin/rest/content/import/	
content	Text content to be spell checked (required).
dictionary	Spell checking dictionary id/name; otherwise default.
Response	
<pre>{ "error": "", "content": ".....", "misspelled": [{ "offset": "OFFSET", "text": "MISSPELLED", "suggestions": "SUGGESTIONS" }, { "offset": "OFFSET", "text": "MISSPELLED", "suggestions": "SUGGESTIONS" }, { "offset": "OFFSET", "text": "MISSPELLED", "suggestions": "SUGGESTIONS" }], }</pre>	

10.3.3 Data

The “/webadmin/rest/data/” REST API function returns all data for one or more content database data items and creates, updates and deletes content database data items.

10.3.3.1 List

If no “id” and no “search” parameter is given then a list of content database data items which match the other given parameters will be returned.

GET /webadmin/rest/data/?database=DATABASE&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
database	Content database id or name (database or databasename required).
databasename	Content database name (database or databasename required).



columns	Content database attribute names to be returned. As default all content database attributes will be returned.
pagesize	Maximum number of data items to list. As default all matching data items will be listed.
offset	First data item number of listed data items to return. For example “offset=0&pagesize=10” for first 10 data items (1-10) and “offset=10&pagesize=10” for next 10 data items (11-20) and “offset=20&pagesize=10” for next 10 data items (21-30) etc. As default all matching data items will be listed.
sort_col	Content database attribute name to sort the listed data items by (“id”, “title” or any configured content database attribute name).
sort_dir	“ASC” or “DESC” to sort the listed data items in ascending or descending order. As default data items will be sorted in ascending order.
Response	
<pre>[{ "database": "Events", "id": "123", "Title": "ABC", } , , { "database": "Events", "id": "789", "Title": "XYZ", }]</pre>	

10.3.3.1.1 Livegrid (DEPRECATED)

This duplicates the general list functionality as described above but with output in Rico Livegrid XML format as used by current and older versions of the web content management system administration pages.

Note: DEPRECATED – use GET /webadmin/rest/data/ (Please see section 10.3.3.1 List).

GET /webadmin/rest/data/livegrid/?database=DATABASE&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
database	Content database id or name (database or databasename required).
databasename	Content database name (database or databasename required).
columns	Content database attribute names to be returned. As default all content database attributes will be returned.
pagesize	Maximum number of data items to list. As default all matching data items will be listed.
offset	First data item number of listed data items to return. For example “offset=0&pagesize=10” for first 10 data items



	(1-10) and “offset=10&pagesize=10” for next 10 data items (11-20) and “offset=20&pagesize=10” for next 10 data items (21-30) etc. As default all matching data items will be listed.
sort_col	Content database attribute name to sort the listed data items by (“id”, “title” or any configured content database attribute name).
sort_dir	“ASC” or “DESC” to sort the listed data items in ascending or descending order. As default data items will be sorted in ascending order.
Response	
XML code	

10.3.3.2 Search

If a “search” parameter is given then a search results list of content database data items which match the “search” parameter and all other given parameters will be returned.

GET /webadmin/rest/data/?database=DATABASE&search=SEARCH&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
database	Content database id or name (required).
search	Search word(s).
attribute	Content database attribute names to be searched separated by commas or “” for all the attributes.
columns	Content database attribute names to be returned. As default all content database attributes will be returned.
pagesize	Maximum number of data items to list. As default all matching data items will be listed.
offset	First data item number of listed data items to return. For example “offset=0&pagesize=10” for first 10 data items (1-10) and “offset=10&pagesize=10” for next 10 data items (11-20) and “offset=20&pagesize=10” for next 10 data items (21-30) etc. As default all matching data items will be listed.
sort_col	Content database attribute name to sort the listed data items by (“id”, “title” or any configured content database attribute name).
sort_dir	“ASC” or “DESC” to sort the listed data items in ascending or descending order. As default data items will be sorted in ascending order.
Response	
<pre>[{ "database": "Events", "id": "123", "Title": "ABC", } , , { "database": "Events",</pre>	



```
"id": "789",  
  "Title": "XYZ",  
}  
]
```

10.3.3.3 Read

If an “id” parameter and no “search” parameter is given then a GET request will return that content database data item’s data. The returned data includes all the content database data item attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the content database data item.

GET /webadmin/rest/data/?database=DATABASE&id=ID	
database	Content database id or name (required).
id	Content database data item id number.
Response	
<pre>{ "database": "Events", "id": "1", "title": "First Quarter Results", /* title column */ "Title": "First Quarter Results", "Category": "Announcement", "Location": "", "From": "2021-03-17", "To": "2021-03-17", "Description": "First quarter results are announced.", "Details": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam imperdiet erat pulvinar ante mollis non sodales orci rutrum. Nulla ac quam ut arcu rhoncus molestie. Donec in lectus eros. Sed sit amet sapien vitae purus consectetur interdum. Nulla ultrices arcu id magna ultrices cursus. Ut sed ante odio, et ullamcorper augue.", "Signup": "Not Required", "Signup_Expiry": "", "Created_By": "admin", "Created_Date": "2010-03-16 12:34:46", "Type": "Shareholder", "_adminpage": { "id": "123", "stylesheets": "", "scripts": "", "htmlheader": "", "body": "" }, "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>	

10.3.3.4 Create

A POST request will create a new content database data item with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the content database data item with that id will be copied and its attributes will be (partially) overwritten by the posted data.



The POST request will return the created data item's data. The returned data includes all the data item attributes as well as true/false flags for the currently logged in website administrator's "user", "creator", "editor", "publisher" and "administrator" access permissions for the data item.

POST /webadmin/rest/data/	
database	Content database id or name (required).
id	Content database data item id number (optional).
.....	All or any number and combination of content database attributes as configured for the content database.
Request	
<pre>{ "database": "Events", "id": "123", "Title": "First Quarter Results", "Category": "Announcement", "Location": "", "From": "2021-03-17", "To": "2021-03-17", "Description": "First quarter results are announced.", "Details": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam imperdiet erat pulvinar ante mollis non sodales orci rutrum. Nulla ac quam ut arcu rhoncus molestie. Donec in lectus eros. Sed sit amet sapien vitae purus consectetur interdum. Nulla ultrices arcu id magna ultrices cursus. Ut sed ante odio, et ullamcorper augue.", "Signup": "Not Required", "Signup_Expiry": "", "Created_By": "admin", "Created_Date": "2010-03-16 12:34:46", "Type": "Shareholder", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>	
Response	
<pre>{ "database": "Events", "id": "789", "Title": "First Quarter Results", "Category": "Announcement", "Location": "", "From": "2021-03-17", "To": "2021-03-17", "Description": "First quarter results are announced.", "Details": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam imperdiet erat pulvinar ante mollis non sodales orci rutrum. Nulla ac quam ut arcu rhoncus molestie. Donec in lectus eros. Sed sit amet sapien vitae purus consectetur interdum. Nulla ultrices arcu id magna ultrices cursus. Ut sed ante odio, et ullamcorper augue.", "Signup": "Not Required", "Signup_Expiry": "", "Created_By": "admin", "Created_Date": "2010-03-16 12:34:46", "Type": "Shareholder", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>	



}

10.3.3.5 Update

A PUT request will update the existing content database data item with the given “id” number with the given data attributes.

The PUT request will return the updated data item’s data. The returned data includes all the data item attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the data item.

PUT /webadmin/rest/data/	
database	Content database id or name (required).
id	Content database data item id number (required).
.....	All or any number and combination of content database attributes as configured for the content database.
Request	
{ "database": "Events", "id": "789", "title": "First Quarter Results", "category": "Announcement", "location": "", "from": "2021-03-17", "to": "2021-03-17", "description": "First quarter results are announced.", "details": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam imperdiet erat pulvinar ante mollis non sodales orci rutrum. Nulla ac quam ut arcu rhoncus molestie. Donec in lectus eros. Sed sit amet sapien vitae purus consectetur interdum. Nulla ultrices arcu id magna ultrices cursus. Ut sed ante odio, et ullamcorper augue.", "signup": "Not Required", "signup_expiry": "", "created_by": "admin", "created_date": "2010-03-16 12:34:46", "type": "Shareholder", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }	
Response	
{ "database": "Events", "id": "789", "title": "First Quarter Results", "category": "Announcement", "location": "", "from": "2021-03-17", "to": "2021-03-17", "description": "First quarter results are announced.", "details": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam imperdiet erat pulvinar ante mollis non sodales orci rutrum. Nulla ac quam ut arcu rhoncus molestie. Donec in lectus eros. Sed sit amet sapien vitae purus consectetur interdum. Nulla ultrices arcu id magna ultrices cursus. Ut sed ante odio, et ullamcorper augue.", "signup": "Not Required", }	



```
{
  "Signup_Expiry": "",
  "Created_By": "admin",
  "Created_Date": "2010-03-16 12:34:46",
  "Type": "Shareholder",
  "user": true,
  "creator": true,
  "editor": true,
  "publisher": true,
  "administrator": true
}
```

10.3.3.6 Delete

A DELETE request will delete the existing content database data item with the given “id” number.

The DELETE request will return the deleted/unpublished data item’s data. The returned data includes all the data item attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the data item.

DELETE /webadmin/rest/data/	
database	Content database id or name (required).
id	Content database data item id number (required).
Request	
{ "database": "Events", "id": "123" }	
Response	
{ "database": "Events", "id": "123", "Title": "First Quarter Results", "Category": "Announcement", "Location": "", "From": "2021-03-17", "To": "2021-03-17", "Description": "First quarter results are announced.", "Details": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam imperdiet erat pulvinar ante mollis non sodales orci rutrum. Nulla ac quam ut arcu rhoncus molestie. Donec in lectus eros. Sed sit amet sapien vitae purus consectetur interdum. Nulla ultrices arcu id magna ultrices cursus. Ut sed ante odio, et ullamcorper augue.", "Signup": "Not Required", "Signup_Expiry": "", "Created_By": "admin", "Created_Date": "2010-03-16 12:34:46", "Type": "Shareholder", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }	

10.3.3.1 Export

A list of content database data will be returned in tab-separated values (tsv/txt) format.



GET /webadmin/rest/data/export/txt/											
database				Content database id or name (required).							
Response											
id	Title	Category	Location	From	To	Description	Details	Signup	Signup	Expiry	Created By
1	First Quarter Results Announcement					2021-03-17	2021-03-17	First			
	quarter results are announced.	Not Required					admin	2010-03-16		
12:34:46	Shareholder										
2	Annual General Meeting	Conference	New York, US			2021-10-27	19:00	2021-10-			
27	21:00	AGM where annual results are presented and other matters are									
	discussed.	Required	2021-10-20	admin	2010-03-16	13:12:09				
	Shareholder										
3	My Corporation Exhibition	Exhibition	London, UK			2021-05-20	2021-05-20				
	Exhibition to showcase our latest products.	Required					2021-05-01			
	admin	2010-03-16	13:13:43	Product							
4	Childrens Charity Auction	Fundraising	Hong Kong, China			2021-01-12	12:00				
	2021-01-12	15:00	We are auctioning our products for charity.	Not						
	Required	admin	2010-03-16	13:15:46	Other						

10.3.3.2 Import

A tab-separated values (tsv/txt) format file will be imported, and the output text (if any) from the import will be returned; optionally as HTML code.

POST /webadmin/rest/data/import/	
database	Content database id or name (required).
file	HTML FORM “file” type input upload (required).
delete	“yes” to delete the current content database data.
Response	
{ "output": "" }	

POST /webadmin/rest/data/import/html/	
database	Content database id or name (required).
file	HTML FORM “file” type input upload (required).
delete	“yes” to delete the current content database data.
Response	
HTML code	

10.3.4 Users

The “/webadmin/rest/users/” REST API function returns all data for one or more user accounts and creates, updates and deletes user accounts.

10.3.4.1 List

If no “id” and no “search” parameter is given then a list of user accounts which match the other given parameters will be returned.

GET	
/webadmin/rest/users/?userclass=USERCLASS&usergroup=USERGROUP&usertype=USER TYPE&status=STATUS&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE& sort_dir=DESC	



userclass	User class name (“administrator”, “user” or “” for any user class).
usergroup	User group name or “-“ for none.
usertype	User type name or “-“ for none.
status	User status code (“pending”, “active”, “expiring”, “expired”).
pagesize	Maximum number of user accounts to list. As default all matching user accounts will be listed.
offset	First user account number of listed user accounts to return. For example “offset=0&pagesize=10” for first 10 user accounts (1-10) and “offset=10&pagesize=10” for next 10 user accounts (11-20) and “offset=20&pagesize=10” for next 10 user accounts (21-30) etc. As default all matching user accounts will be listed.
sort_col	User account attribute name to sort the listed user accounts by (“id”, “name”, “username”, “userclass”, “usergroup”, “usertype” etc).
sort_dir	“ASC” or “DESC” to sort the listed user accounts in ascending or descending order. As default user accounts will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "username": "ABC" } , , { "id": "789", "username": "XYZ" }]</pre>	

10.3.4.1.1 Livegrid (DEPRECATED)

This duplicates the general list functionality as described above but with output in Rico Livegrid XML format as used by current and older versions of the web content management system administration pages.

Note: DEPRECATED – use GET /webadmin/rest/users/ (Please see section 10.3.4.1 List).

GET /webadmin/rest/users/livegrid/?userclass=USERCLASS&usergroup=USERGROUP&usertype=USERTYPE&status=STATUS&pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
userclass	User class name (“administrator”, “user” or “” for any user class).
usergroup	User group name or “-“ for none.



usertype	User type name or “-“ for none.
status	User status code (“pending”, “active”, “expiring”, “expired”).
pagesize	Maximum number of user accounts to list. As default all matching user accounts will be listed.
offset	First user account number of listed user accounts to return. For example “offset=0&pagesize=10” for first 10 user accounts (1-10) and “offset=10&pagesize=10” for next 10 user accounts (11-20) and “offset=20&pagesize=10” for next 10 user accounts (21-30) etc. As default all matching user accounts will be listed.
sort_col	User account attribute name to sort the listed user accounts by (“id”, “name”, “username”, “userclass”, “usergroup”, “usertype” etc).
sort_dir	“ASC” or “DESC” to sort the listed user accounts in ascending or descending order. As default user accounts will be sorted in ascending order.
Response	
XML code	

10.3.4.2 Search

If a “search” parameter is given then a search results list of user accounts which match the “search” parameter and all other given parameters will be returned.

GET /webadmin/rest/users/?search=SEARCH& attribute=ATTRIBUTE &pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
search	Search word(s).
attribute	Content attribute names to be searched (“name”, “organisation”, “email”, “username”, “delivery_name”, “delivery_organisation”, “delivery_email”, “delivery_website”, “invoice_name”, “invoice_organisation”, “invoice_email”, “invoice_website”, “notes”, “userinfo”) separated by commas or “” for all the attributes.
pagesize	Maximum number of user accounts to list. As default all matching user accounts will be listed.
offset	First user account number of listed user accounts to return. For example “offset=0&pagesize=10” for first 10 user accounts (1-10) and “offset=10&pagesize=10” for next 10 user accounts (11-20) and “offset=20&pagesize=10” for next 10 user accounts (21-30) etc. As default all matching user accounts will be listed.
sort_col	User account attribute name to sort the listed user accounts by (“id”, “name”, “username”, “userclass”, “usergroup”, “usertype” etc).
sort_dir	“ASC” or “DESC” to sort the listed user accounts in



	ascending or descending order. As default user accounts will be sorted in ascending order.
Response	
<pre>[{ "id": "123", "username": "ABC" } , , { "id": "789", "username": "XYZ" }]</pre>	

10.3.4.2.1 Livegrid (DEPRECATED)

This duplicates the general search functionality as described above but with output in Rico Livegrid XML format as used by current and older versions of the web content management system administration pages.

Note: DEPRECATED – use GET /webadmin/rest/users/ (Please see section 10.3.4.2 Search).

GET /webadmin/rest/users/livegrid/?search=SEARCH& attribute=ATTRIBUTE &pagesize=PAGESIZE&offset=OFFSET&sort_col=ATTRIBUTE&sort_dir=DESC	
search	Search word(s).
attribute	Content attribute names to be searched (“name”, “organisation”, “email”, “username”, “delivery_name”, “delivery_organisation”, “delivery_email”, “delivery_website”, “invoice_name”, “invoice_organisation”, “invoice_email”, “invoice_website”, “notes”, “userinfo”) separated by commas or “” for all the attributes.
pagesize	Maximum number of user accounts to list. As default all matching user accounts will be listed.
offset	First user account number of listed user accounts to return. For example “offset=0&pagesize=10” for first 10 user accounts (1-10) and “offset=10&pagesize=10” for next 10 user accounts (11-20) and “offset=20&pagesize=10” for next 10 user accounts (21-30) etc. As default all matching user accounts will be listed.
sort_col	User account attribute name to sort the listed user accounts by (“id”, “name”, “username”, “userclass”, “usergroup”, “usertype” etc).
sort_dir	“ASC” or “DESC” to sort the listed user accounts in ascending or descending order. As default user accounts will be sorted in ascending order.
Response	



XML code

10.3.4.3 Read

If an “id” parameter and no “search” parameter is given then a GET request will return that user account’s data. The returned data includes all the user account attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user account.

GET /webadmin/rest/users/?id=ID	
id	Content item id number (required).
Response	
<pre>{ "id": "6", "created": "2010-03-24 13:59:08", "updated": "2020-02-28 11:04:32", "created_by": "admin", "updated_by": "admin", "password_updated": "2010-04-01 14:13:53", "username": "paulgreen", "password": "XYABQ0EB1LG7MCCUXTUJXRXBK1FJ0LXL4H86HEGVY8VK5QHRHGA71KB9AXGLVBYN", "onetimepassword": "", "onetimepassword_updated": "2020-02-28 11:04:32", "secret": "", "secret_updated": "2020-02-28 11:04:32", "title": "", "name": "Paul Green", "organisation": "", "email": "", "gender": "", "photo": "", "birthdate": "", "birthyear": "", "birthmonth": "", "birthday": "", "notes": "", "userinfo": "", "userinfos": { }, "keywords": "", "description": "", "userclass": "administrator", "usergroup": "Human Resources", "usertype": "", "usergroups": ["Website Administrators"], "usertypes": [], "users_group": "User Managers", "users_type": "", "users_users": "", "creators_group": "User Managers", "creators_type": "", "creators_users": "",</pre>	



```
"editors_group": "User Managers",
"editors_type": "",
"editors_users": "",
"publishers_group": "User Managers",
"publishers_type": "",
"publishers_users": "",
"administrators_group": "Website Administrators",
"administrators_type": "",
"administrators_users": "",

"scheduled_publish": "",
"scheduled_publish_email": "",
"scheduled_notify": "",
"scheduled_notify_email": "",
"scheduled_unpublish": "",
"scheduled_unpublish_email": "",
"scheduled_last": "",

"invoice_name": "",
"invoice_organisation": "",
"invoice_address": "",
"invoice_postalcode": "",
"invoice_city": "",
"invoice_state": "",
"invoice_country": "",
"invoice_phone": "",
"invoice_fax": "",
"invoice_email": "",
"invoice_website": "",

"delivery_name": "",
"delivery_organisation": "",
"delivery_address": "",
"delivery_postalcode": "",
"delivery_city": "",
"delivery_state": "",
"delivery_country": "",
"delivery_phone": "",
"delivery_fax": "",
"delivery_email": "",
"delivery_website": "",

"card_type": "",
"card_number": "",
"card_issuedmonth": "",
"card_issuedyear": "",
"card_expirymonth": "",
"card_expiryyear": "",
"card_name": "",
"card_cvc": "",
"card_issue": "",
"card_postalcode": "",

"content_editor": "",
"hardcore_upload": "",
"hardcore_format": "",
"hardcore_width": "",
"hardcore_height": "",
"hardcore_onenter": "",
"hardcore_onctrlenter": "",
"hardcore_onshiftenter": "",
"hardcore_onaltenter": "",
"hardcore_skin": "",
"hardcore_toolbar": "",
"hardcore_toolbar1": "",
```



```
"hardcore_toolbar2": "",
"hardcore_toolbar3": "",
"hardcore_toolbar4": "",
"hardcore_toolbar5": "",
"hardcore_toolbar6": "",
"hardcore_toolbar7": "",
"hardcore_toolbar8": "",
"hardcore_toolbar9": "",
"hardcore_toolbar10": "",
"hardcore_formatblock": "",
"hardcore_fontname": "",
"hardcore_fontsize": "",
"hardcore_customscript": "",
"hardcore_encoding": "",
"hardcore_language": "",

"startpage": "",

"workspace_sections": "",
"index_workspace": "",
"index_workspace_projects": "",
"index_workspace_comments": "",
"index_content": "",
"index_library": "",
"index_search": "",
"index_searchadvanced": "",
"index_searchreplace": "",
"index_product": "",
"index_stock": "",
"index_order": "",
"index_segments": "",
"index_usertests": "",
"index_heatmaps": "",
"index_user": "",
"index_websites": "",
"index_sales": "",
"index_databases": "",
"index_hosting": "",
"index_projects": "",

"menu_selection": "",
"statistics_reports": "",
"sales_reports": "",

"shopcart": "",
"wishlist": "",

"failed": 0,
"locked": 0,

"webadmin_ecommerce": "forbid",
"webadmin_library_packages": "forbid",
"webadmin_user": "forbid",

"userslog": [
],

"user": true,
"creator": true,
"editor": true,
"publisher": true,
"administrator": true
}
```



10.3.4.4 Create

A POST request will create a new user account with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the user account with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created user account’s data. The returned data includes all the user account attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user account.

POST /webadmin/rest/users/	
id	User account id number to be copied (optional).
.....	All or any number and combination of user account attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "userclass": "administrator", "usergroup": "Home", "usertype": "", "username": "johndoe", "password": "SECRET" }</pre>	
Response	
<pre>{ "id": "789", "created": "2020-01-31 23:59:59", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.4.5 Update

A PUT request will update the existing user account with the given “id” number with the given data attributes.

The PUT request will return the updated user account’s data. The returned data includes all the user account attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user account.

PUT /webadmin/rest/users/	
id	User account id number to be updated (required).
.....	All or any number and combination of user account attributes as used by the web content management



	system (see the GET request example response above).
Request	
<pre>{ "id": "789", "userclass": "administrator", "usergroup": "Home", "usertype": "", "username": "johndoe", "password": "SECRET" }</pre>	
Response	
<pre>{ "id": "789", "created": "2020-01-01 00:00:00", "updated": "2020-01-31 23:59:59", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": false }</pre>	

10.3.4.6 Delete

A DELETE request will delete the existing user account with the given “id” number.

The DELETE request will return the deleted user account’s data. The returned data includes all the user account attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user account.

DELETE /webadmin/rest/users/	
id	User account id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "created": "2020-01-01 00:00:00", "updated": "2020-01-31 23:59:59", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	



10.3.4.7 Export

A list of the users will be returned in comma-separated values (csv) format.

GET /webadmin/rest/users/export/csv/	
Response	
id,username,password,onetimepassword,secret,name,organisation,email,userclass,usertype,usergroup,usertypes,usergroups,users_type,users_group,users_users,creators_type,creators_group,creators_users,editors_type,editors_group,editors_users,publishers_type,publishers_group,publishers_users,administrators_type,administrators_group,administrators_users,content_editor,hardcore_language,hardcore_upload,hardcore_format,hardcore_width,hardcore_height,hardcore_onenter,hardcore_onctrlenter,hardcore_onshiftenter,hardcore_onaltenter,hardcore_skin,hardcore_toolbar,hardcore_toolbar1,hardcore_toolbar2,hardcore_toolbar3,hardcore_toolbar4,hardcore_toolbar5,hardcore_toolbar6,hardcore_toolbar7,hardcore_toolbar8,scheduled_publish,scheduled_notify,scheduled_unpublish,scheduled_publish_email,scheduled_notify_email,scheduled_unpublish_email,card_type,card_number,card_is_suedmonth,card_issuedyear,card_expiryyear,card_expiryyear,card_name,card_cvc,card_issue,card_postalcode,delivery_name,delivery_organisation,delivery_addresses,delivery_postalcode,delivery_city,delivery_state,delivery_country,delivery_phone,delivery_fax,delivery_email,delivery_website,invoice_name,invoice_organisation,invoice_address,invoice_postalcode,invoice_city,invoice_state,invoice_country,invoice_phone,invoice_fax,invoice_email,invoice_website,notes,keywords,description,created,updated,password_updated,onetimepassword_updated,secret_updated,userinfo,menu_selection,workspace_sections,statistics_reports,sales_reports,index_content,index_library,index_search,index_searchadvanced,index_searchreplace,index_product,index_stock,index_order,index_segments,index_usertests,index_heatmaps,index_user,index_websites,index_hosting,index_databases,index_workspace,title,gender,photo,birthdate,birthday,birthmonth,birthyear,formatblock,fontname,fontsize,customscript,startpage,webadmin_workspace,webadmin_workspace_checkedout,webadmin_workspace_updated,webadmin_workspace_created,webadmin_workspace_expired,webadmin_workspace_workflow,webadmin_browseedit,webadmin_structure,webadmin_content,webadmin_content_structure,webadmin_content_pages,webadmin_content_elements,webadmin_content_templates,webadmin_content_stylesheets,webadmin_content_scripts,webadmin_content_packages,webadmin_content_bundles,webadmin_library,webadmin_library_images,webadmin_library_files,webadmin_library_links,webadmin_library_packages,webadmin_library_bundles,webadmin_ecommerce,webadmin_ecommerce_products,webadmin_ecommerce_stock,webadmin_ecommerce_orders,webadmin_ecommerce_packages,webadmin_ecommerce_bundles,webadmin_databases,webadmin_databases_data,webadmin_databases_export,webadmin_databases_import,webadmin_experience,webadmin_experience_segments,webadmin_experience_usertests,webadmin_experience_heatmaps,webadmin_user,webadmin_user_administrators,webadmin_user_templates,webadmin_user_users,webadmin_statistics,webadmin_statistics_summary,webadmin_statistics_what,webadmin_statistics_what_websites,webadmin_statistics_what_contents,webadmin_statistics_what_library,webadmin_statistics_what_ecommerce,webadmin_statistics_what_databases,webadmin_statistics_when,webadmin_statistics_when_daily,webadmin_statistics_when_weekly,webadmin_statistics_when_monthly,webadmin_statistics_when_yearly,webadmin_statistics_when_hours,webadmin_statistics_when_weekdays,webadmin_statistics_when_days,webadmin_statistics_when_weeks,webadmin_statistics_when_months,webadmin_statistics_who,webadmin_statistics_who_countries,webadmin_statistics_who_clienthosts,webadmin_statistics_who_visitors,webadmin_statistics_who_robots,webadmin_statistics_who_operatingsystems,webadmin_statistics_who_webbrowsers,webadmin_statistics_who_devices,webadmin_statistics_who_users,webadmin_statistics_why,webadmin_statistics_why_referrals,webadmin_statistics_why_searchengines,webadmin_statistics_why_searchqueries,webadmin_statistics_why_searchwords,webadmin_statistics_how,webadmin_statistics_how_entry,webadmin_statistics_how_paths,webadmin_statistics_how_exit,webadmin_statistics_how_duration,webadmin_statistics_how_visits,webadmin_updates	
.....	



10.3.4.8 Import

A comma-separated values (csv) format file will be imported, and the output text (if any) from the import will be returned; optionally as HTML code.

POST /webadmin/rest/users/import/	
file	HTML FORM “file” type input upload (required).
Response	
{ "output": "" }	

POST /webadmin/rest/users/import/html/	
file	HTML FORM “file” type input upload (required).
Response	
HTML code	

10.3.4.9 Email

A POST request will compose or send an email to the given email address(es) and given user accounts.

POST /webadmin/rest/users/email/	
from	The email address to be used as “From” address; otherwise the configured “contact_form_recipient” email address will be used.
to	The email addresses to be used as “To” addresses; otherwise the configured “contact_form_recipient” email address will be used.
cc	The email addresses to be used as “Cc” addresses; otherwise, if no “action” parameter is given, the currently logged in user’s email address will be used.
bcc	The email addresses to be used as “Bcc” addresses; otherwise, if no “action” parameter is given, the email addresses for the given user account ids or user groups/types.
action	If blank (“”) then an email will only be composed and returned as the response without being sent. If not blank (fx. “action=send”) then an email will be sent.
id	If no “action” parameter is given, and one or more user account “id” parameters are given the email addresses (if any) for those user accounts will be retrieved and returned as “bcc” response data attributes.
usergroup	If no “action” and “id” parameters are given, and one or more “usergroup” name parameters are given, the email addresses (if any) for all user accounts of the given user group names will be retrieved and returned as “bcc” response data



	attributes. If both “usergroup” and “usertype” parameters are given then only users who are both of one the given user groups as well as one of the given user types will be considered.
usertype	If no “action” and “id” parameters are given, and one or more “usertype” name parameters are given, the email addresses (if any) for all user accounts of the given user type names will be retrieved and returned as “bcc” response data attributes. If both “usergroup” and “usertype” parameters are given then only users who are both of one the given user groups as well as one of the given user types will be considered.
subject	Text to be used as the email “Subject”.
content	Text to be used as the email content (HTML code).
content_plaintext	Text to be used as the plain text email content.
content_id	If no “action” parameter is given, and an “content_id” parameter is given, the content item with the given id will be read and set as the email subject (title) and content.
stylesheet	Style sheet URL address to be used for email content (optional).
style	CSS code to be used for email content (optional).
send	If “send=now” parameter is given then an email will be sent. (Note: use “action=” and “send=now” parameters to compose and immediately send email).
Request	
<pre>{ "from": "", "to": "", "cc": "", "bcc": "", "action": "", "id": "", "usergroup": "", "usertype": "", "subject": "", "content": "", "content_plaintext": "", "content_id": "", "stylesheet": "", "style": "", "send": "", }</pre>	
Response	
<pre>{ "error": "", "from": "", "to": "", "cc": "", "bcc": "", "subject": "", "content": "", "content_plaintext": "", "stylesheet": "",</pre>	



```
"style": ""  
}
```

10.3.4.10 Password Update

A POST request will update the login password for the currently logged in website administrator user account.

POST /webadmin/rest/users/password/	
old_password	The existing password for the currently logged in user account (required).
new_password	The new password to be used for the currently logged in user account (required).
new_password2	The new password (again) to be used for the currently logged in user account (required).
Request	
<pre>{ "old_password": "OLDSECRET", "new_password": "NEWSECRET", "new_password2": " NEWSECRET " }</pre>	
Response	
<pre>{ "error": "" }</pre>	

10.3.4.11 Update All User Passwords

A GET request will refresh (read and re-save) the passwords for all user accounts returning the old and new passwords for all user accounts; optionally as HTML code.

WARNING: This special functionality should only be used for very specific use cases such as changing plain text stored user account passwords to encrypted/hashed stored user account passwords. If this special functionality is used on already encrypted/hashed stored user account passwords then the user account passwords may become “corrupted” and require all website users and website administrators to reset their user account passwords.

GET /webadmin/rest/users/update_all/	
Response	
<pre>[{ "id": "123", "username": "abc", "old_password": "OLDSECRET", "password": "NEWSECRET" }, ... { "id": "789", "username": "xyz", "old_password": "OLDSECRET", "password": "NEWSECRET" }]</pre>	



GET /webadmin/rest/users/update all/html/	
Response	
HTML code	

10.3.5 Micro-Websites

The “/webadmin/rest/websites/” REST API function returns all data for one or more micro-websites and creates, updates and deletes micro-websites.

10.3.5.1 List

If no “id” parameter is given then a list of micro-websites will be returned.

GET /webadmin/rest/websites/	
Response	
<pre>[{ "id": "123", "domain": "ABC", }, , { "id": "789", "domain": "XYZ", }]</pre>	

10.3.5.2 Read

If an “id” parameter is given then a GET request will return micro-website’s data. The returned data includes all the micro-website attributes for the micro-website.

GET /webadmin/rest/websites/?id=ID	
id	Micro-website id number (required).
Response	
<pre>{ "id": "1", "domain": "amp.yourwebsite.com", "remote": "", "useragent": "", "language": "", "referrer": "", "keywords": "", "default_version": "", "default_template": "", "default_stylesheet": "", "default_doctype": "", "default_html": "", }</pre>	



```
"default_head": "",

"default_page": "123",
"default_page_nonexisting": "",
"default_page_unpublished": "",
"default_page_expired": "",
"default_login": "",
"default_login_code": "",
"default_login_code_email": "",
"default_searchresults_page": "",
"default_searchresults_entry": "",
"default_register_confirmation_page": "",
"default_register_notification_page": "",
"retrieve_password_page": "",
"retrieve_password_confirmation": "",
"retrieve_password_email": "",
"retrieve_password_error": "",

"default_price": "",
"default_country": "",
"default_state": "",

"default_list_entry": "",
"default_publish_ready": "",
"default_personal_admin_page": ""
}
```

10.3.5.3 Create

A POST request will create a new micro-website with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the micro-website with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created micro-website’s data. The returned data includes all the micro-website attributes for the micro-website.

POST /webadmin/rest/websites/	
id	Micro-website id number to be copied (optional).
.....	All or any number and combination of micro-website attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "domain": "XYZ" }	
Response	
{ "id": "789", "domain": "XYZ", }	



10.3.5.4 Update

A PUT request will update the existing micro-website with the given “id” number with the given data attributes.

The PUT request will return the updated micro-website’s data. The returned data includes all the micro-website attributes for the micro-website.

PUT /webadmin/rest/websites/	
id	Micro-website id number to be updated (required).
.....	All or any number and combination of micro-website attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "default_page": "789" }	
Response	
{ "id": "123", "domain": "ABC", "default_page": "789", }	

10.3.5.5 Delete

A DELETE request will delete the existing micro-website with the given “id” number.

The DELETE request will return the deleted micro-website’s data. The returned data includes all the micro-website attributes for the micro-website.

DELETE /webadmin/rest/websites/	
id	Micro-website id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "domain": "ABC", }	

10.3.6 Content Elements

The “/webadmin/rest/elements/” REST API function returns all data for one or more content elements and creates, updates and deletes content elements.

10.3.6.1 List

If no “id” parameter is given then a list of content elements will be returned.



GET /webadmin/rest/elements/	
Response	
<pre>[{ "id": "123", "element": "ABC", }, { "id": "789", "element": "XYZ", }]</pre>	

10.3.6.2 Read

If an “id” parameter is given then a GET request will return that content element’s data. The returned data includes all the content element attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content element.

GET /webadmin/rest/elements/?id=ID	
id	Content element id number (required).
Response	
<pre>{ "id": "1", "element": "banner", "users_group": "", "users_type": "", "users_users": "", "creators_group": "", "creators_type": "", "creators_users": "", "developers_group": "", "developers_type": "", "developers_users": "", "editors_group": "", "editors_type": "", "editors_users": "", "publishers_group": "", "publishers_type": "", "publishers_users": "", "administrators_group": "", "administrators_type": "", "administrators_users": "", "user": true, "creator": true, "developer": true, "editor": true, "publisher": true, "administrator": true }</pre>	



10.3.6.3 Create

A POST request will create a new content element with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the content element with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created content element’s data. The returned data includes all the content element attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content element.

POST /webadmin/rest/elements/	
id	Content element id number to be copied (optional).
.....	All or any number and combination of content element attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "element": "XYZ" }	
Response	
{ "id": "789", "element": "XYZ",, "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.6.4 Update

A PUT request will update the existing content element with the given “id” number with the given data attributes.

The PUT request will return the updated content element’s data. The returned data includes all the content element attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content element.

PUT /webadmin/rest/elements/	
id	Content element id number to be updated (required).
.....	All or any number and combination of content element attributes as used by the web content management system (see the GET request example response above).
Request	
{	



<pre>"id": "123", "administrators_group": "Website Administrators" }</pre>
Response
<pre>{ "id": "123", "element": "ABC", "administrators_group": "Website Administrators", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>

10.3.6.5 Delete

A DELETE request will delete the existing content element with the given “id” number.

The DELETE request will return the deleted content element’s data. The returned data includes all the content element attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content element.

DELETE /webadmin/rest/elements/	
id	Content element id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "element": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.7 Content Groups

The “/webadmin/rest/contentgroups/” REST API function returns all data for one or more content groups and creates, updates and deletes content groups.

10.3.7.1 List

If no “id” parameter is given then a list of content groups will be returned.

GET /webadmin/rest/contentgroups/	
Response	



```
[
  {
    "id": "123",
    .....
    "contentgroup": "ABC"
  },
  .....
  {
    "id": "789",
    .....
    "contentgroup": "XYZ"
  }
]
```

10.3.7.2 Read

If an “id” parameter is given then a GET request will return that content group’s data. The returned data includes all the content group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content group.

GET /webadmin/rest/contentgroups/?id=ID	
id	Content group id number (required).
Response	
<pre>{ "id": "1", "contentgroup": "About Us", "parentgroup": "", "_subgroups": 0, "template": "", "stylesheet": "", "title_prefix": "", "title_suffix": "", "doctype": "", "users_group": "", "users_type": "", "users_users": "", "creators_group": "Press Officers", "creators_type": "", "creators_users": "", "developers_group": "Website Developers", "developers_type": "", "developers_users": "", "editors_group": "Press Officers", "editors_type": "", "editors_users": "", "publishers_group": "Press Officers", "publishers_type": "", "publishers_users": "", "administrators_group": "Website Administrators", "administrators_type": "", "administrators_users": "", "user": true, "creator": true, "developer": true,</pre>	



```
"editor": true,  
"publisher": true,  
"administrator": true  
}
```

10.3.7.3 Create

A POST request will create a new content group with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the content group with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created content group’s data. The returned data includes all the content group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content group.

POST /webadmin/rest/contentgroups/	
id	Content group id number to be copied (optional).
.....	All or any number and combination of content group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "contentgroup": "XYZ" }	
Response	
{ "id": "789", "contentgroup": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.7.4 Update

A PUT request will update the existing content group with the given “id” number with the given data attributes.

The PUT request will return the updated content group’s data. The returned data includes all the content group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content group.

PUT /webadmin/rest/contentgroups/	
id	Content group id number to be updated (required).



.....	All or any number and combination of content group attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "template": "456", "stylesheet": "789" }</pre>	
Response	
<pre>{ "id": "123", "contentgroup": "ABC", "parentgroup": "", "template": "456", "stylesheet": "789" "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.7.5 Delete

A DELETE request will delete the existing content group with the given “id” number.

The DELETE request will return the deleted content group’s data. The returned data includes all the content group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content group.

DELETE /webadmin/rest/contentgroups/	
id	Content group id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "contentgroup": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	



10.3.7.6 Exists

If a “contentgroup” parameter is given then a GET request will return “YES” or “NO” depending on if a content group with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a content group with the given name and another id than the given id exists.

GET /webadmin/rest/contentgroups/exists/?contentgroup=CONTENTGROUP	
contentgroup	Content group name (required).
id	Content group id number (optional).
Response	
{ "exists": "YES" }	

10.3.8 Content Types

The “/webadmin/rest/contenttypes/” REST API function returns all data for one or more content types and creates, updates and deletes content types.

10.3.8.1 List

If no “id” parameter is given then a list of content types will be returned.

GET /webadmin/rest/contenttypes/	
Response	
[{ "id": "123", "contenttype": "ABC" } , , { "id": "789", "contenttype": "XYZ" }]	

10.3.8.2 Read

If an “id” parameter is given then a GET request will return that content type’s data. The returned data includes all the content type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content type.

GET /webadmin/rest/contenttypes/?id=ID	
id	Content type id number (required).
Response	
{ "id": "1", "contenttype": "ABC" }	



```
"contenttype": "News",
"parenttype": "",
"_subtypes": 0,

"template": "",
"stylesheet": "",
"title_prefix": "",
"title_suffix": "",
"doctype": "",

"users_group": "",
"users_type": "",
"users_users": "",
"creators_group": "Press Officers",
"creators_type": "",
"creators_users": "",
"developers_group": "Website Developers",
"developers_type": "",
"developers_users": "",
"editors_group": "Press Officers",
"editors_type": "",
"editors_users": "",
"publishers_group": "Press Officers",
"publishers_type": "",
"publishers_users": "",
"administrators_group": "Website Administrators",
"administrators_type": "",
"administrators_users": "",

"user": true,
"creator": true,
"developer": true,
"editor": true,
"publisher": true,
"administrator": true
}
```

10.3.8.3 Create

A POST request will create a new content type with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the content type with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created content type’s data. The returned data includes all the content type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content type.

POST /webadmin/rest/contenttypes/	
id	Content type id number to be copied (optional).
.....	All or any number and combination of content type attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123",	



<pre>"contenttype": "XYZ" }</pre>
Response
<pre>{ "id": "789", "contenttype": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>

10.3.8.4 Update

A PUT request will update the existing content type with the given “id” number with the given data attributes.

The PUT request will return the updated content type’s data. The returned data includes all the content type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content type.

PUT /webadmin/rest/contenttypes/	
id	Content type id number to be updated (required).
.....	All or any number and combination of content type attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "template": "456", "stylesheet": "789" }</pre>	
Response	
<pre>{ "id": "123", "contenttype": "ABC", "parenttype": "", "template": "456", "stylesheet": "789" "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	



10.3.8.5 Delete

A DELETE request will delete the existing content type with the given “id” number.

The DELETE request will return the deleted content type’s data. The returned data includes all the content type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the content type.

DELETE /webadmin/rest/contenttypes/	
id	Content type id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "contenttype": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.8.6 Exists

If a “contenttype” parameter is given then a GET request will return “YES” or “NO” depending on if a content type with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a content type with the given name and another id than the given id exists.

GET /webadmin/rest/contenttypes/exists/?contenttype=CONTENTTYPE	
contenttype	Content type name (required).
id	Content type id number (optional).
Response	
{ "exists": "YES" }	

10.3.9 Image Formats

The “/webadmin/rest/imageformats/” REST API function returns all data for one or more image formats and creates, updates and deletes image formats.

10.3.9.1 List

If no “id” parameter is given then a list of image formats will be returned.

GET /webadmin/rest/imageformats/	



Response
<pre>[{ "id": "123", "filenameextension": "ABC" }, { "id": "789", "filenameextension": "XYZ" }]</pre>

10.3.9.2 Read

If an “id” parameter is given then a GET request will return that image format’s data. The returned data includes all the image format attributes for the image format.

GET /webadmin/rest/imageformats/?id=ID	
id	Image format id number (required).
Response	
<pre>{ "id": "1", "filenameextension": "gif" }</pre>	

10.3.9.3 Create

A POST request will create a new image format with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the image format with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created image format’s data. The returned data includes all the image format attributes for the image format.

POST /webadmin/rest/imageformats/	
id	Image format id number to be copied (optional).
.	All or any number and combination of image format attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "filenameextension": "XYZ" }</pre>	
Response	
<pre>{ "id": "789", "filenameextension": "XYZ" }</pre>	



10.3.9.4 Update

A PUT request will update the existing image format with the given “id” number with the given data attributes.

The PUT request will return the updated image format’s data. The returned data includes all the image format attributes for the image format.

PUT /webadmin/rest/imageformats/	
id	Image format id number to be updated (required).
.....	All or any number and combination of image format attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "filenameextension": "XYZ" }	
Response	
{ "id": "123", "filenameextension": "XYZ" }	

10.3.9.5 Delete

A DELETE request will delete the existing image format with the given “id” number.

The DELETE request will return the deleted image format’s data. The returned data includes all the image format attributes as well for the image format.

DELETE /webadmin/rest/imageformats/	
id	Image format id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "filenameextension": "ABC" }	

10.3.10 Image Groups

The “/webadmin/rest/imagegroups/” REST API function returns all data for one or more image groups and creates, updates and deletes image groups.

10.3.10.1 List

If no “id” parameter is given then a list of image groups will be returned.

GET /webadmin/rest/imagegroups/	



Response
<pre>[{ "id": "123", "imagegroup": "ABC" }, { "id": "789", "imagegroup": "XYZ" }]</pre>

10.3.10.2 Read

If an “id” parameter is given then a GET request will return that image group’s data. The returned data includes all the image group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the image group.

GET /webadmin/rest/imagegroups/?id=ID	
id	Image group id number (required).
Response	
<pre>{ "id": "1", "imagegroup": "Template", "parentgroup": "", "_subgroups": 0, "template": "", "stylesheet": "", "title_prefix": "", "title_suffix": "", "doctype": "", "users_group": "", "users_type": "", "users_users": "", "creators_group": "Website Developers", "creators_type": "", "creators_users": "", "developers_group": "Website Developers", "developers_type": "", "developers_users": "", "editors_group": "Website Developers", "editors_type": "", "editors_users": "", "publishers_group": "Website Developers", "publishers_type": "", "publishers_users": "", "administrators_group": "Website Administrators", "administrators_type": "", "administrators_users": "", "user": true, "creator": true,</pre>	



```
"developer": true,  
"editor": true,  
"publisher": true,  
"administrator": true  
}
```

10.3.10.3 Create

A POST request will create a new image group with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the image group with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created image group’s data. The returned data includes all the image group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the image group.

POST /webadmin/rest/imagegroups/	
id	Image group id number to be copied (optional).
.....	All or any number and combination of image group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "imagegroup": "XYZ" }	
Response	
{ "id": "789", "imagegroup": "XYZ",, "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.10.4 Update

A PUT request will update the existing image group with the given “id” number with the given data attributes.

The PUT request will return the updated image group’s data. The returned data includes all the image group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the image group.

PUT /webadmin/rest/imagegroups/



id	Image group id number to be updated (required).
.....	All or any number and combination of image group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "parentgroup": "XYZ" }	
Response	
{ "id": "123", "imagegroup": "ABC", "parentgroup": "XYZ",, "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.10.5 Delete

A DELETE request will delete the existing image group with the given “id” number.

The DELETE request will return the deleted image group’s data. The returned data includes all the image group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the image group.

DELETE /webadmin/rest/imagegroups/	
id	Image group id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "imagegroup": "ABC",, "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	



10.3.10.6 Exists

If an “imagegroup” parameter is given then a GET request will return “YES” or “NO” depending on if an image group with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if an image group with the given name and another id than the given id exists.

GET /webadmin/rest/imagegroups/exists/?imagegroup=IMAGEGROUP	
imagegroup	Image group name (required).
id	Image group id number (optional).
Response	
{ "exists": "YES" }	

10.3.11 Image Types

The “/webadmin/rest/imagetypes/” REST API function returns all data for one or more image types and creates, updates and deletes image types.

10.3.11.1 List

If no “id” parameter is given then a list of image types will be returned.

GET /webadmin/rest/imagetypes/	
Response	
[{ "id": "123", "imagetype": "ABC" } , , { "id": "789", "imagetype": "XYZ" }]	

10.3.11.2 Read

If an “id” parameter is given then a GET request will return that image type’s data. The returned data includes all the image type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the image type.

GET /webadmin/rest/imagetypes/?id=ID	
id	Image type id number (required).
Response	
{ "id": "1", "imagetype": "ABC" }	



```
"imagetype": "Thumbnails",
"parenttype": "",
"_subtypes": 0,

"template": "",
"stylesheet": "",
"title_prefix": "",
"title_suffix": "",
"doctype": "",

"users_group": "",
"users_type": "",
"users_users": "",
"creators_group": "Website Editors",
"creators_type": "",
"creators_users": "",
"developers_group": "Website Developers",
"developers_type": "",
"developers_users": "",
"editors_group": "Website Editors",
"editors_type": "",
"editors_users": "",
"publishers_group": "Website Editors",
"publishers_type": "",
"publishers_users": "",
"administrators_group": "Website Administrators",
"administrators_type": "",
"administrators_users": "",

"user": true,
"creator": true,
"developer": true,
"editor": true,
"publisher": true,
"administrator": true
}
```

10.3.11.3 Create

A POST request will create a new image type with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the image type with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created image type’s data. The returned data includes all the image type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the image type.

POST /webadmin/rest/imagetypes/	
id	Image type id number to be copied (optional).
.....	All or any number and combination of image type attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123",	



<pre>"imagetype": "XYZ" }</pre>
Response
<pre>{ "id": "789", "imagetype": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>

10.3.11.4 Update

A PUT request will update the existing image type with the given “id” number with the given data attributes.

The PUT request will return the updated image type’s data. The returned data includes all the image type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the image type.

PUT /webadmin/rest/imagetypes/	
id	Image type id number to be updated (required).
.....	All or any number and combination of image type attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "parenttype": "XYZ" }</pre>	
Response	
<pre>{ "id": "123", "imagetype": "ABC", "parenttype": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.11.5 Delete

A DELETE request will delete the existing image type with the given “id” number.



The DELETE request will return the deleted image type's data. The returned data includes all the image type attributes as well as true/false flags for the currently logged in website administrator's "user", "creator", "editor", "publisher", "developer" and "administrator" access permissions for the image type.

DELETE /webadmin/rest/imagetypes/	
id	Image type id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "imagetype": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.11.6 Exists

If an "imagetype" parameter is given then a GET request will return "YES" or "NO" depending on if an image type with the given name exists.

Optionally, if an "id" parameter is also given then a GET request will return "YES" or "NO" depending on if an image type with the given name and another id than the given id exists.

GET /webadmin/rest/imagetypes/exists/?imagetype=IMAGETYPE	
imagetype	Image type name (required).
id	Image type id number (optional).
Response	
{ "exists": "YES" }	

10.3.12 File Formats

The "/webadmin/rest/fileformats/" REST API function returns all data for one or more file formats and creates, updates and deletes file formats.

10.3.12.1 List

If no "id" parameter is given then a list of file formats will be returned.

GET /webadmin/rest/fileformats/	
Response	
[{ "id": "123",	



```
    "filenameextension": "ABC"
  },
  ,
  .....
  ,
  {
    "id": "789",
    "filenameextension": "XYZ"
  }
]
```

10.3.12.2 Read

If an “id” parameter is given then a GET request will return that file format’s data. The returned data includes all the file format attributes for the file format.

GET /webadmin/rest/fileformats/?id=ID	
id	File format id number (required).
Response	
{ "id": "1", "filenameextension": "pdf" }	

10.3.12.3 Create

A POST request will create a new file format with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the file format with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created file format’s data. The returned data includes all the file format attributes for the file format.

POST /webadmin/rest/fileformats/	
id	File format id number to be copied (optional).
.....	All or any number and combination of file format attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "filenameextension": "XYZ" }	
Response	
{ "id": "789", "filenameextension": "XYZ" }	



10.3.12.4 Update

A PUT request will update the existing file format with the given “id” number with the given data attributes.

The PUT request will return the updated file format’s data. The returned data includes all the file format attributes for the file format.

PUT /webadmin/rest/fileformats/	
id	File format id number to be updated (required).
.....	All or any number and combination of file format attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "filenameextension": "XYZ" }	
Response	
{ "id": "123", "filenameextension": "XYZ" }	

10.3.12.5 Delete

A DELETE request will delete the existing file format with the given “id” number.

The DELETE request will return the deleted file format’s data. The returned data includes all the file format attributes as well for the file format.

DELETE /webadmin/rest/fileformats/	
id	File format id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "filenameextension": "ABC" }	

10.3.13 File Groups

The “/webadmin/rest/filegroups/” REST API function returns all data for one or more file groups and creates, updates and deletes file groups.

10.3.13.1 List

If no “id” parameter is given then a list of file groups will be returned.

GET /webadmin/rest/filegroups/	



Response
<pre>[{ "id": "123", "filegroup": "ABC" }, { "id": "789", "filegroup": "XYZ" }]</pre>

10.3.13.2 Read

If an “id” parameter is given then a GET request will return that file group’s data. The returned data includes all the file group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the file group.

GET /webadmin/rest/filegroups/?id=ID	
id	File group id number (required).
Response	
<pre>{ "id": "1", "filegroup": "Careers", "parentgroup": "", "_subgroups": 0, "template": "", "stylesheet": "", "title_prefix": "", "title_suffix": "", "doctype": "", "users_group": "", "users_type": "", "users_users": "", "creators_group": "", "creators_type": "", "creators_users": "", "developers_group": "Human Resources", "developers_type": "", "developers_users": "", "editors_group": "Human Resources", "editors_type": "", "editors_users": "", "publishers_group": "Human Resources", "publishers_type": "", "publishers_users": "", "administrators_group": "Website Administrators", "administrators_type": "", "administrators_users": "", "user": true, "creator": true,</pre>	



```
"developer": true,  
"editor": true,  
"publisher": true,  
"administrator": true  
}
```

10.3.13.3 Create

A POST request will create a new file group with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the file group with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created file group’s data. The returned data includes all the file group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the file group.

POST /webadmin/rest/filegroups/	
id	File group id number to be copied (optional).
.....	All or any number and combination of file group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "filegroup": "XYZ" }	
Response	
{ "id": "789", "filegroup": "XYZ",, "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.13.4 Update

A PUT request will update the existing file group with the given “id” number with the given data attributes.

The PUT request will return the updated file group’s data. The returned data includes all the file group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the file group.

PUT /webadmin/rest/filegroups/



id	File group id number to be updated (required).
.....	All or any number and combination of file group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "parentgroup": "XYZ" }	
Response	
{ "id": "123", "filegroup": "ABC", "parentgroup": "XYZ",, "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.13.5 Delete

A DELETE request will delete the existing file group with the given “id” number.

The DELETE request will return the deleted file group’s data. The returned data includes all the file group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the file group.

DELETE /webadmin/rest/filegroups/	
id	File group id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "filegroup": "ABC",, "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	



10.3.13.6 Exists

If a “filegroup” parameter is given then a GET request will return “YES” or “NO” depending on if a file group with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a file group with the given name and another id than the given id exists.

GET /webadmin/rest/filegroups/exists/?filegroup=FILEGROUP	
filegroup	File group name (required).
id	File group id number (optional).
Response	
{ "exists": "YES" }	

10.3.14 File Types

The “/webadmin/rest/filetypes/” REST API function returns all data for one or more file types and creates, updates and deletes file types.

10.3.14.1 List

If no “id” parameter is given then a list of file types will be returned.

GET /webadmin/rest/filetypes/	
Response	
[{ "id": "123", "filetype": "ABC" } , , { "id": "789", "filetype": "XYZ" }]	

10.3.14.2 Read

If an “id” parameter is given then a GET request will return that file type’s data. The returned data includes all the file type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the file type.

GET /webadmin/rest/filetypes/?id=ID	
id	File type id number (required).
Response	
{ "id": "1", "filetype": "ABC" }	



```
"filetype": "Instruction Manuals",
"parenttype": "",
"_subtypes": 0,

"template": "",
"stylesheet": "",
"title_prefix": "",
"title_suffix": "",
"doctype": "",

"users_group": "",
"users_type": "",
"users_users": "",
"creators_group": "Website Editors",
"creators_type": "",
"creators_users": "",
"developers_group": "Website Developers",
"developers_type": "",
"developers_users": "",
"editors_group": "Website Editors",
"editors_type": "",
"editors_users": "",
"publishers_group": "Website Editors",
"publishers_type": "",
"publishers_users": "",
"administrators_group": "Website Administrators",
"administrators_type": "",
"administrators_users": "",

"user": true,
"creator": true,
"developer": true,
"editor": true,
"publisher": true,
"administrator": true
}
```

10.3.14.3 Create

A POST request will create a new file type with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the file type with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created file type’s data. The returned data includes all the file type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the file type.

POST /webadmin/rest/filetypes/	
id	File type id number to be copied (optional).
.....	All or any number and combination of file type attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123",	



<pre>"filetype": "XYZ" }</pre>
Response
<pre>{ "id": "789", "filetype": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>

10.3.14.4 Update

A PUT request will update the existing file type with the given “id” number with the given data attributes.

The PUT request will return the updated file type’s data. The returned data includes all the file type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the file type.

PUT /webadmin/rest/filetypes/	
id	File type id number to be updated (required).
.....	All or any number and combination of file type attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "parenttype": "XYZ" }</pre>	
Response	
<pre>{ "id": "123", "filetype": "ABC", "parenttype": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.14.5 Delete

A DELETE request will delete the existing file type with the given “id” number.



The DELETE request will return the deleted file type's data. The returned data includes all the file type attributes as well as true/false flags for the currently logged in website administrator's "user", "creator", "editor", "publisher", "developer" and "administrator" access permissions for the file type.

DELETE /webadmin/rest/filetypes/	
id	File type id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "filetype": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.14.6 Exists

If a "filetype" parameter is given then a GET request will return "YES" or "NO" depending on if a file type with the given name exists.

Optionally, if an "id" parameter is also given then a GET request will return "YES" or "NO" depending on if a file type with the given name and another id than the given id exists.

GET /webadmin/rest/filetypes/exists/?filetype=FILETYPE	
filetype	File type name (required).
id	File type id number (optional).
Response	
{ "exists": "YES" }	

10.3.15 Link Groups

The "/webadmin/rest/linkgroups/" REST API function returns all data for one or more link groups and creates, updates and deletes link groups.

10.3.15.1 List

If no "id" parameter is given then a list of link groups will be returned.

GET /webadmin/rest/linkgroups/	
Response	
[{ "id": "123",	



```
.....
  "linkgroup": "ABC"
}
,
.....
,
{
  "id": "789",
  .....
  "linkgroup": "XYZ"
}
]
```

10.3.15.2 Read

If an “id” parameter is given then a GET request will return that link group’s data. The returned data includes all the link group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the link group.

GET /webadmin/rest/linkgroups/?id=ID	
id	Link group id number (required).
Response	
<pre>{ "id": "1", "linkgroup": "Partners", "parentgroup": "", "_subgroups": 0, "template": "", "stylesheet": "", "title_prefix": "", "title_suffix": "", "doctype": "", "users_group": "", "users_type": "", "users_users": "", "creators_group": "", "creators_type": "", "creators_users": "", "developers_group": "Website Developers", "developers_type": "", "developers_users": "", "editors_group": "Website Editors", "editors_type": "", "editors_users": "", "publishers_group": "Website Publishers", "publishers_type": "", "publishers_users": "", "administrators_group": "Website Administrators", "administrators_type": "", "administrators_users": "", "user": true, "creator": true, "developer": true, "editor": true, "publisher": true, "administrator": true }</pre>	



```
}
```

10.3.15.3 Create

A POST request will create a new link group with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the link group with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created link group’s data. The returned data includes all the link group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the link group.

POST /webadmin/rest/linkgroups/	
id	Link group id number to be copied (optional).
.....	All or any number and combination of link group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "linkgroup": "XYZ" }	
Response	
{ "id": "789", "linkgroup": "XYZ",, "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.15.4 Update

A PUT request will update the existing link group with the given “id” number with the given data attributes.

The PUT request will return the updated link group’s data. The returned data includes all the link group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the link group.

PUT /webadmin/rest/linkgroups/	
id	Link group id number to be updated (required).
.....	All or any number and combination of link group attributes as used by the web content management



	system (see the GET request example response above).
Request	
<pre>{ "id": "123", "parentgroup": "XYZ" }</pre>	
Response	
<pre>{ "id": "123", "linkgroup": "ABC", "parentgroup": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.15.5 Delete

A DELETE request will delete the existing link group with the given “id” number.

The DELETE request will return the deleted link group’s data. The returned data includes all the link group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the link group.

DELETE /webadmin/rest/linkgroups/	
id	Link group id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "linkgroup": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.15.6 Exists

If a “linkgroup” parameter is given then a GET request will return “YES” or “NO” depending on if a link group with the given name exists.



Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a link group with the given name and another id than the given id exists.

GET /webadmin/rest/linkgroups/exists/?linkgroup=LINKGROUP	
linkgroup	Link group name (required).
id	Link group id number (optional).
Response	
<pre>{ "exists": "YES" }</pre>	

10.3.16 Link Types

The “/webadmin/rest/linktypes/” REST API function returns all data for one or more link types and creates, updates and deletes link types.

10.3.16.1 List

If no “id” parameter is given then a list of link types will be returned.

GET /webadmin/rest/linktypes/	
Response	
<pre>[{ "id": "123", "linktype": "ABC" } , , { "id": "789", "linktype": "XYZ" }]</pre>	

10.3.16.2 Read

If an “id” parameter is given then a GET request will return that link type’s data. The returned data includes all the link type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the link type.

GET /webadmin/rest/linktypes/?id=ID	
id	Link type id number (required).
Response	
<pre>{ "id": "1", "linktype": "Product Websites", "parenttype": "", "_subtypes": 0, "template": "",</pre>	



```
"stylesheet": "",
"title_prefix": "",
"title_suffix": "",
"doctype": "",

"users_group": "",
"users_type": "",
"users_users": "",
"creators_group": "Website Editors",
"creators_type": "",
"creators_users": "",
"developers_group": "Website Developers",
"developers_type": "",
"developers_users": "",
"editors_group": "Website Editors",
"editors_type": "",
"editors_users": "",
"publishers_group": "Website Editors",
"publishers_type": "",
"publishers_users": "",
"administrators_group": "Website Administrators",
"administrators_type": "",
"administrators_users": "",

"user": true,
"creator": true,
"developer": true,
"editor": true,
"publisher": true,
"administrator": true
}
```

10.3.16.3 Create
A POST request will create a new link type with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the link type with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created link type’s data. The returned data includes all the link type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the link type.

POST /webadmin/rest/linktypes/	
id	Link type id number to be copied (optional).
.....	All or any number and combination of link type attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "linktype": "XYZ" }	
Response	
{	



```
    "id": "789",  
  
    "linktype": "XYZ",  
    .....  
    "user": true,  
    "creator": true,  
    "editor": true,  
    "publisher": true,  
    "developer": true,  
    "administrator": true  
}
```

10.3.16.4 Update

A PUT request will update the existing link type with the given “id” number with the given data attributes.

The PUT request will return the updated link type’s data. The returned data includes all the link type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the link type.

PUT /webadmin/rest/linktypes/	
id	Link type id number to be updated (required).
.....	All or any number and combination of link type attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "parenttype": "XYZ" }	
Response	
{ "id": "123", "linktype": "ABC", "parenttype": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.16.5 Delete

A DELETE request will delete the existing link type with the given “id” number.

The DELETE request will return the deleted link type’s data. The returned data includes all the link type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the link type.



DELETE /webadmin/rest/linktypes/	
id	Link type id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "linktype": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.16.6 Exists

If a “linktype” parameter is given then a GET request will return “YES” or “NO” depending on if a link type with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a link type with the given name and another id than the given id exists.

GET /webadmin/rest/linktypes/exists/?linktype=LINKTYPE	
linktype	Link group name (required).
id	Link group id number (optional).
Response	
{ "exists": "YES" }	

10.3.17 Content Versions

The “/webadmin/rest/versions/” REST API function returns all data for one or more content versions and creates, updates and deletes content versions.

10.3.17.1 List

If no “id” parameter is given then a list of content versions will be returned.

GET /webadmin/rest/versions/	
Response	
[{ "id": "123", "version": "ABC", } , ,]	



```
{
  "id": "789",
  "version": "XYZ",
  .....
}
```

10.3.17.2 Read

If an “id” parameter is given then a GET request will return that content version’s data. The returned data includes all the content version attributes for the content version.

GET /webadmin/rest/versions/?id=ID	
id	Content version id number (required).
Response	
{ "id": "1", "version": "Danish", "currencies": "2" }	

10.3.17.3 Create

A POST request will create a new content version with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the content version with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created content version’s data. The returned data includes all the content version attributes for the content version.

POST /webadmin/rest/versions/	
id	Content version id number to be copied (optional).
.....	All or any number and combination of content version attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "version": "XYZ" }	
Response	
{ "id": "789", "version": "XYZ", }	

10.3.17.4 Update

A PUT request will update the existing content version with the given “id” number with the given data attributes.



The PUT request will return the updated content version's data. The returned data includes all the content version attributes for the content version.

PUT /webadmin/rest/versions/	
id	Content version id number to be updated (required).
.....	All or any number and combination of content version attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "currencies": "1" }	
Response	
{ "id": "123", "version": "ABC", "currencies": "1" }	

10.3.17.5 Delete

A DELETE request will delete the existing content version with the given "id" number.

The DELETE request will return the deleted content version's data. The returned data includes all the content version attributes for the content version.

DELETE /webadmin/rest/versions/	
id	Content version id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "version": "ABC", }	

10.3.18 Content Bundles

The "/webadmin/rest/contentbundles/" REST API function returns all data for one or more content bundles and updates and deletes content bundles.

10.3.18.1 List

If no "id" parameter is given then a list of content bundles will be returned.

GET /webadmin/rest/contentbundles/	
Response	
[{ 	



```
"contentbundle": "ABC"
},
.....
,
{
  "contentbundle": "XYZ",
}
]
```

10.3.18.2 Read

If an “id” parameter is given then a GET request will return that content bundle’s data. The returned data includes all the content bundle attributes for the content bundle.

GET /webadmin/rest/contentbundles/?id=ID	
id	Content bundle id name (required).
Response	
<pre>{ "contentbundle": "ABC", "pages": [{ "id": "359", "title": "Definition Probability" }, { "id": "360", "title": "Definition Risk" }, { "id": "353", "title": "Definition Severity" }, { "id": "355", "title": "Definition Status" }, { "id": "354", "title": "Definition Urgency" }, { "id": "346", "title": "My Tickets - Active Ticket Summary Entry" }, { "id": "370", "title": "My Tickets - Active Tickets - NAB Customer (1)" }, { "id": "371", "title": "My Tickets - Active Tickets - NAB Customer (2)" }, { "id": "368", "title": "My Tickets - Active Tickets - NAB Provider (1)" }, { "id": "369", "title": "My Tickets - Active Tickets - NAB Provider (2)" }, { "id": "342", "title": "My Tickets - Archived Ticket Entry" }, { "id": "345", "title": "My Tickets - Archived Ticket Summary Entry" }, { "id": "344", "title": "My Tickets - Archived Tickets" }, { "id": "367", "title": "My Tickets - Comment Entry" }, { "id": "375", "title": "My Tickets - Confirmation" }, { "id": "339", "title": "My Tickets - New Ticket" }, { "id": "374", "title": "My Tickets - New Ticket - Validation Error" }, { "id": "358", "title": "My Tickets - Toolbar" }, { "id": "337", "title": "My Tickets - View Ticket" }, { "id": "343", "title": "Support Tickets" }, { "id": "348", "title": "Support Tickets Admin" }, { "id": "351", "title": "Ticket Administration - Advanced Search" }, { "id": "350", "title": "Ticket Administration - Advanced Search Results" }, { "id": "352", "title": "Ticket Administration - Awaiting Customer Response" }, { "id": "380", "title": "Ticket Administration - Comment Entry" }, { "id": "338", "title": "Ticket Administration - Keyword Search Results" }, { "id": "377", "title": "Ticket Administration - NAB Customer (1)" }, { "id": "378", "title": "Ticket Administration - NAB Customer (2)" }, { "id": "372", "title": "Ticket Administration - NAB Provider (1)" }, { "id": "373", "title": "Ticket Administration - NAB Provider (2)" }, { "id": "357", "title": "Ticket Administration - New Ticket" }, { "id": "379", "title": "Ticket Administration - New Ticket - Validation Error" }, { "id": "356", "title": "Ticket Administration - Search Bar" },] }</pre>	



```
{
  { "id": "365", "title": "Ticket Administration - Search Director" },
  { "id": "366", "title": "Ticket Administration - Search No Tickets" },
  { "id": "347", "title": "Ticket Administration - Summary Entry" },
  { "id": "361", "title": "Ticket Administration - Toolbar" },
  { "id": "349", "title": "Ticket Administration - Update Ticket" },
  { "id": "340", "title": "Tickets" }
},
"elements": [
],
"templates": [
],
"stylesheets": [
],
"scripts": [
  { "id": "362", "title": "Support Tickets" }
],
"images": [
],
"files": [
],
"links": [
],
"products": [
]
}
```

10.3.18.3 Update

A PUT request will update the existing content bundle with the given “id” name with the given data attributes.

The PUT request will return the updated content bundle’s data. The returned data includes all the content bundle attributes for the content bundle.

PUT /webadmin/rest/contentbundles/	
id	Content bundle id name to be updated (required).
contentbundle	New content bundle id name for the content bundle to be renamed to.
Request	
{ "id": "ABC", "contentbundle": "XYZ" }	
Response	
{ "contentbundle": "XYZ" }	

10.3.18.4 Delete

A DELETE request will delete the existing content bundle with the given “id” name.

The DELETE request will return the deleted content bundle’s data. The returned data includes all the content bundle attributes for the content bundle.

DELETE /webadmin/rest/contentbundles/	
id	Content bundle id name (required).



Request
<pre>{ "contentbundle": "ABC" }</pre>
Response
<pre>{ "contentbundle": "ABC" }</pre>

10.3.19 Content Packages

The “/webadmin/rest/contentpackages/” REST API function returns all data for one or more content packages and updates and deletes content packages.

10.3.19.1 List

If no “id” parameter is given then a list of content packages will be returned.

GET /webadmin/rest/contentpackages/	
Response	
<pre>[{ "contentpackage": "ABC" }, { "contentpackage": "XYZ", }]</pre>	

10.3.19.2 Read

If an “id” parameter is given then a GET request will return that content package data. The returned data includes all the content package attributes for the content package.

GET /webadmin/rest/contentpackages/?id=ID	
id	Content package id name (required).
Response	
<pre>{ "contentpackage": "ABC", "pages": [{ "id": "359", "title": "Definition Probability" }, { "id": "360", "title": "Definition Risk" }, { "id": "353", "title": "Definition Severity" }, { "id": "355", "title": "Definition Status" }, { "id": "354", "title": "Definition Urgency" }, { "id": "346", "title": "My Tickets - Active Ticket Summary Entry" }, { "id": "370", "title": "My Tickets - Active Tickets - NAB Customer (1)" }, { "id": "371", "title": "My Tickets - Active Tickets - NAB Customer (2)" }, { "id": "368", "title": "My Tickets - Active Tickets - NAB Provider (1)" }, { "id": "369", "title": "My Tickets - Active Tickets - NAB Provider (2)" }, { "id": "342", "title": "My Tickets - Archived Ticket Entry" },] }</pre>	



```
{ "id": "345", "title": "My Tickets - Archived Ticket Summary Entry" },
{ "id": "344", "title": "My Tickets - Archived Tickets" },
{ "id": "367", "title": "My Tickets - Comment Entry" },
{ "id": "375", "title": "My Tickets - Confirmation" },
{ "id": "339", "title": "My Tickets - New Ticket" },
{ "id": "374", "title": "My Tickets - New Ticket - Validation Error" },
{ "id": "358", "title": "My Tickets - Toolbar" },
{ "id": "337", "title": "My Tickets - View Ticket" },
{ "id": "343", "title": "Support Tickets" },
{ "id": "348", "title": "Support Tickets Admin" },
{ "id": "351", "title": "Ticket Administration - Advanced Search" },
{ "id": "350", "title": "Ticket Administration - Advanced Search
Results" },
{ "id": "352", "title": "Ticket Administration - Awaiting Customer
Response" },
{ "id": "380", "title": "Ticket Administration - Comment Entry" },
{ "id": "338", "title": "Ticket Administration - Keyword Search
Results" },
{ "id": "377", "title": "Ticket Administration - NAB Customer (1)" },
{ "id": "378", "title": "Ticket Administration - NAB Customer (2)" },
{ "id": "372", "title": "Ticket Administration - NAB Provider (1)" },
{ "id": "373", "title": "Ticket Administration - NAB Provider (2)" },
{ "id": "357", "title": "Ticket Administration - New Ticket" },
{ "id": "379", "title": "Ticket Administration - New Ticket -
Validation Error" },
{ "id": "356", "title": "Ticket Administration - Search Bar" },
{ "id": "365", "title": "Ticket Administration - Search Director" },
{ "id": "366", "title": "Ticket Administration - Search No Tickets" },
{ "id": "347", "title": "Ticket Administration - Summary Entry" },
{ "id": "361", "title": "Ticket Administration - Toolbar" },
{ "id": "349", "title": "Ticket Administration - Update Ticket" },
{ "id": "340", "title": "Tickets" }
},
"elements": [
],
"templates": [
],
"stylesheets": [
],
"scripts": [
{ "id": "362", "title": "Support Tickets" }
],
"images": [
],
"files": [
],
"links": [
],
"products": [
]
}
```

10.3.19.3 Update

A PUT request will update the existing content package with the given “id” name with the given data attributes.

The PUT request will return the updated content package’s data. The returned data includes all the content package attributes for the content package.

PUT /webadmin/rest/contentpackages/	
id	Content package id name to be updated (required).



contentpackage	New content package id name for the content package to be renamed to.
Request	
{ "id": "ABC", "contentpackage": "XYZ" }	
Response	
{ "contentpackage": "XYZ" }	

10.3.19.4 Delete

A DELETE request will delete the existing content package with the given “id” name.

The DELETE request will return the deleted content package’s data. The returned data includes all the content package attributes for the content package.

DELETE /webadmin/rest/contentpackages/	
id	Content package id name (required).
Request	
{ "contentpackage": "ABC" }	
Response	
{ "contentpackage": "ABC" }	

10.3.20 Content Metadata (DEPRECATED)

The “/webadmin/rest/metadata/” REST API functions returns all content metadata for auditing.

Note: DEPRECATED – use GET /webadmin/rest/content/metadata/ (Please see section 10.3.2.25 Metadata).

10.3.20.1 List (DEPRECATED)

If no “contentclass” parameter is given then metadata for all content will be returned.

Note: DEPRECATED – use GET /webadmin/rest/content/metadata/ (Please see section 10.3.2.25 Metadata).

GET /webadmin/rest/metadata/contentclass=CONTENTCLASS	
contentclass	“page” or “product”
Response	
[{ "id": "154", "contentclass": "page", "contentgroup": "Store Locator", "contenttype": "", "version": "", "title": "@@@include:database=Stores:id=###id###:Store Name@@@", }]	



```
    "author": "",
    "description": "",
    "keywords": "",
    "metainfo": ""
  },
  {
    "id": "657",
    "contentclass": "page",
    "contentgroup": "Store Locator",
    "contenttype": "",
    "version": "",
    "title": "@@include:database=Stores:id###id###:Store Name@@@",
    "author": "",
    "description": "",
    "keywords": "",
    "metainfo": ""
  },
  .....
]
```

10.3.21 User Groups

The “/webadmin/rest/usergroups/” REST API function returns all data for one or more user groups and creates, updates and deletes user groups.

10.3.21.1 List

If no “id” parameter is given then a list of user groups will be returned.

GET /webadmin/rest/usergroups/	
Response	
<pre>[{ "id": "123", "usergroup": "ABC" } , , { "id": "789", "usergroup": "XYZ" }]</pre>	

10.3.21.2 Read

If an “id” parameter is given then a GET request will return that user group’s data. The returned data includes all the user group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user group.

GET /webadmin/rest/usergroups/?id=ID	
id	User group id number (required).
Response	
{	



```
"id": "1",

"usergroup": "Event Managers",
"parentgroup": "",
"_subgroups": 0,

"supergroups": [
],

"subgroups": [
],

"login_page": "",

"users_group": "",
"users_type": "",
"users_users": "",
"creators_group": "",
"creators_type": "",
"creators_users": "",
"editors_group": "",
"editors_type": "",
"editors_users": "",
"publishers_group": "",
"publishers_type": "",
"publishers_users": "",
"administrators_group": "",
"administrators_type": "",
"administrators_users": "",
"subscribers_group": "",
"subscribers_type": "",

"user": true,
"creator": true,
"editor": true,
"publisher": true,
"administrator": true
}
```

10.3.21.3 Create

A POST request will create a new user group with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the user group with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created user group’s data. The returned data includes all the user group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user group.

POST /webadmin/rest/usergroups/	
id	User group id number to be copied (optional).
.....	All or any number and combination of user group attributes as used by the web content management system (see the GET request example response above).
Request	



<pre>{ "id": "123", "usergroup": "XYZ" }</pre>
Response
<pre>{ "id": "789", "usergroup": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>

10.3.21.4 Update

A PUT request will update the existing user group with the given “id” number with the given data attributes.

The PUT request will return the updated user group’s data. The returned data includes all the user group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user group.

PUT /webadmin/rest/usergroups/	
id	User group id number to be updated (required).
.....	All or any number and combination of user group attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "parentgroup": "XYZ" }</pre>	
Response	
<pre>{ "id": "123", "usergroup": "ABC", "parentgroup": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>	

10.3.21.5 Delete

A DELETE request will delete the existing user group with the given “id” number.



The DELETE request will return the deleted user group's data. The returned data includes all the user group attributes as well as true/false flags for the currently logged in website administrator's "user", "creator", "editor", "publisher" and "administrator" access permissions for the user group.

DELETE /webadmin/rest/usergroups/	
id	User group id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "usergroup": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }	

10.3.21.6 Exists

If a "usergroup" parameter is given then a GET request will return "YES" or "NO" depending on if a user group with the given name exists.

Optionally, if an "id" parameter is also given then a GET request will return "YES" or "NO" depending on if a user group with the given name and another id than the given id exists.

GET /webadmin/rest/usergroups/exists/?usergroup=USERGROUP	
usergroup	User group name (required).
id	User group id number (optional).
Response	
{ "exists": "YES" }	

10.3.22 User Types

The "/webadmin/rest/usertypes/" REST API function returns all data for one or more user types and creates, updates and deletes user types.

10.3.22.1 List

If no "id" parameter is given then a list of user types will be returned.

GET /webadmin/rest/usertypes/	
Response	
{ { "id": "123", }	



```
    "usertype": "ABC"
  },
  ,
  .....
  ,
  {
    "id": "789",
    .....
    "usertype": "XYZ"
  }
]
```

10.3.22.2 Read

If an “id” parameter is given then a GET request will return that user type’s data. The returned data includes all the user type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user type.

GET /webadmin/rest/usertypes/?id=ID	
id	User type id number (required).
Response	
<pre>{ "id": "1", "usertype": "Senior Management", "parenttype": "", "_subtypes": 0, "supertypes": [], "subtypes": [], "login_page": "", "users_group": "", "users_type": "", "users_users": "", "creators_group": "", "creators_type": "", "creators_users": "", "editors_group": "", "editors_type": "", "editors_users": "", "publishers_group": "", "publishers_type": "", "publishers_users": "", "administrators_group": "", "administrators_type": "", "administrators_users": "", "subscribers_group": "", "subscribers_type": "", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>	



10.3.22.3 Create

A POST request will create a new user type with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the user type with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created user type’s data. The returned data includes all the user type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user type.

POST /webadmin/rest/usertypes/	
id	User type id number to be copied (optional).
.....	All or any number and combination of user type attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "usertype": "XYZ" }	
Response	
{ "id": "789", "usertype": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }	

10.3.22.4 Update

A PUT request will update the existing user type with the given “id” number with the given data attributes.

The PUT request will return the updated user type’s data. The returned data includes all the user type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user type.

PUT /webadmin/rest/usertypes/	
id	User type id number to be updated (required).
.....	All or any number and combination of user type attributes as used by the web content management system (see the GET request example response above).



Request
<pre>{ "id": "123", "parenttype": "XYZ" }</pre>
Response
<pre>{ "id": "123", "usertype": "ABC", "parenttype": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>

10.3.22.5 Delete

A DELETE request will delete the existing user type with the given “id” number.

The DELETE request will return the deleted user type’s data. The returned data includes all the user type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the user type.

DELETE /webadmin/rest/usertypes/	
id	User type id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "usertype": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>	

10.3.22.6 Exists

If a “usertype” parameter is given then a GET request will return “YES” or “NO” depending on if a user type with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a user type with the given name and another id than the given id exists.

GET /webadmin/rest/usertypes/exists/?usertype=USERTYPE	
usertype	User type name (required).



id	User type id number (optional).
Response	
{ "exists": "YES" }	

10.3.23 User Segments

The “/webadmin/rest/segments/” REST API function returns all data for one or more user segments and creates, updates and deletes user segments.

10.3.23.1 List

If no “id” parameter is given then a list of user groups will be returned.

GET /webadmin/rest/segments/	
Response	
[{ "id": "123", "segmentid": "ABC", } , , { "id": "789", "segmentid": "XYZ", }]	

10.3.23.2 Read

If an “id” or “segmentid” parameter is given then a GET request will return that user segment’s data. The returned data includes all the user segment attributes for the user segment.

GET /webadmin/rest/segments/?id=ID	
id	User segment id number (id or segmentid required).
GET /webadmin/rest/segments/?segmentid=SEGMENTID	
segmentid	User segment id text (id or segmentid required).
Response	
{ " id": "1", "created": "", "updated": "", "created_by": "", "updated_by": "", "segmentid": "Christmas", "segment": "New Christmas period user", "weight": "100", }	



```
"active": "1",
"scheduled": "",
"unscheduled": "",

"datetimefull": "",
"datetimeyear": "",
"datetimemonth": "",
"datetimeday": "",
"datetimeweek": "",
"datetimeweekday": "",
"datetimehour": "",
"datetimemin": "",
"datetimesec": "",

"clientaddr": "",
"clienthost": "",
"clientuser": "",
"clientuseragent": "",
"clientbrowser": "",
"clientversion": "",
"clientes": "",
"clientesversion": "",
"clientdevice": "",
"clientdeviceid": "",
"clientdeviceversion": "",
"preferredlanguage": "",
"acceptedlanguages": "",

"referrerhost": "",
"referrerpath": "",
"referrerquery": "",
"referrersearchengine": "",
"referrersearchquery": "",
"referrerid": "",
"referrerclass": "",
"referrerdatabase": "",

"requestid": "",
"requestclass": "",
"requestdatabase": "",
"requesthost": "",
"requestpath": "",
"requestquery": "",
"requestmethod": "",
"requestport": "",
"requestprotocol": "",

"title": "",
"name": "",
"organisation": "",
"email": "",
"gender": "",
"age": "",
"birthdate": "",
"birthyear": "",
"birthmonth": "",
"birthday": "",
"notes": "",
"userinfo": "",

"username": "",
"password": "",

"userclass": "",
"usergroup": "",
```



```
"usertype": "",

"scheduled_publish": "",
"scheduled_notify": "",
"scheduled_unpublish": "",
"scheduled_publish_email": "",
"scheduled_notify_email": "",
"scheduled_unpublish_email": "",
"scheduled_last": "",

"invoice_name": "",
"invoice_organisation": "",
"invoice_address": "",
"invoice_postalcode": "",
"invoice_city": "",
"invoice_state": "",
"invoice_country": "",
"invoice_phone": "",
"invoice_fax": "",
"invoice_email": "",
"invoice_website": "",

"delivery_name": "",
"delivery_organisation": "",
"delivery_address": "",
"delivery_postalcode": "",
"delivery_city": "",
"delivery_state": "",
"delivery_country": "",
"delivery_phone": "",
"delivery_fax": "",
"delivery_email": "",
"delivery_website": "",

"card_type": "",
"card_number": "",
"card_issuedmonth": "",
"card_issuedyear": "",
"card_expirymonth": "",
"card_expiryyear": "",
"card_name": "",
"card_postalcode": "",

"geoipcountry": "",
"geoipregion": "",
"geoipcity": "",
"geoippostalcode": "",
"geoiplatitude": "",
"geoiplongitude": "",
"geoiptimezone": "",

"description": "",
"keywords": "",
"shopcart": "",
"wishlist": ""
}
```

10.3.23.3 Create

A POST request will create a new user segment with the given data attributes and an automatically assigned unique “id” number.



If an “id” parameter is given then the user segment with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created user segment’s data. The returned data includes all the user segment attributes for the user segment.

POST /webadmin/rest/segments/	
id	User segment id number to be copied (optional).
.....	All or any number and combination of user segment attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "segmentid": "XYZ", "segment": "Segment XYZ", }	
Response	
{ "id": "789", "segmentid": "XYZ", "segment": "Segment XYZ", }	

10.3.23.4 Update

A PUT request will update the existing user segment with the given “id” number with the given data attributes.

The PUT request will return the updated user segment’s data. The returned data includes all the user segment attributes for the user segment.

PUT /webadmin/rest/segments/	
id	User segment id number to be updated (required).
.....	All or any number and combination of user segment attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "weight": "100", "active": "1" }	
Response	
{ "id": "123", "segmentid": "ABC", "segment": "Segment ABC", "weight": "100", "active": "1", }	



```
.....  
}
```

10.3.23.5 Delete

A DELETE request will delete the existing user segment with the given “id” number.

The DELETE request will return the deleted user segment’s data. The returned data includes all the user segment attributes for the user segment.

DELETE /webadmin/rest/segments/	
id	User segment id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "segmentid": "ABC", "segment": "Segment ABC", }	

10.3.23.6 Clear User Segments Data

If a “segment” parameter is given then user test data older than given or current date/time will marked to be cleared when website users access the website.

POST /webadmin/rest/segments/clear/segment=SEGMENT& datetime=DATETIME	
segment	User segment name, or “*” for all user segments (required).
datetime	Date and time older than which user segments data should be cleared in the format “YYYY-MM-DD hh:mm:ss”; otherwise the current date and time will be used (optional).
Response	
{ "error": "" }	

10.3.24 User Tests

The “/webadmin/rest/usertests/” REST API function returns all data for one or more user tests and creates, updates and deletes user tests.

10.3.24.1 List

If no “id” parameter is given then a list of user tests will be returned.

GET /webadmin/rest/usertests/	
Response	



```
[
  {
    "id": "123",
    "usertest": "ABC",
    .....
  },
  .....
  ,
  {
    "id": "789",
    "usertest": "XYZ",
    .....
  }
]
```

10.3.24.2 Read

If an “id” or “usertest” parameter is given then a GET request will return that user test’s data. The returned data includes all the user test attributes for the user test.

GET /webadmin/rest/usertests/?id=ID	
id	User test id number (id or usertest required).
GET /webadmin/rest/usertests/?usertest=USERTEST	
usertest	User test id text (id or usertest required).
Response	
{ "id": "1", "usertest": "Logo", "description": "Test response to new logo.", "variants": "Old\r\nNew", "type": "ab", "confidence": "95", "probability": "100", "visitors": "all", "goals": "", "active": "1", "scheduled": "", "unscheduled": "" }	

10.3.24.3 Create

A POST request will create a new user test with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the user test with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created user test’s data. The returned data includes all the user test attributes for the user test.

POST /webadmin/rest/usertests/	
id	User test id number to be copied (optional).
.....	All or any number and combination of user test attributes as used by the web content management system (see the GET request example response



	above).
Request	
{ "id": "123", "usertest": "Featured", "description": "Home page feature story", }	
Response	
{ "id": "789", "usertest": "Featured", "description": "Home page feature story", }	

10.3.24.4 Update

A PUT request will update the existing user test with the given “id” number with the given data attributes.

The PUT request will return the updated user test’s data. The returned data includes all the user test attributes for the user test.

PUT /webadmin/rest/usertests/	
id	User test id number to be updated (required).
.....	All or any number and combination of user test attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "confidence": "95", "active": "1" }	
Response	
{ "id": "123", "confidence": "95", "active": "1", }	

10.3.24.5 Delete

A DELETE request will delete the existing user test with the given “id” number.

The DELETE request will return the deleted user test’s data. The returned data includes all the user test attributes for the user test.

DELETE /webadmin/rest/usertests/	
id	User test id number (required).
Request	



<pre>{ "id": "123" }</pre>
Response
<pre>{ "id": "123", "usertest": "Logo", "description": "Test response to new logo.", }</pre>

10.3.24.6 Exists

If a “usertest” parameter is given then a GET request will return “YES” or “NO” depending on if a user test with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a user test with the given name and another id than the given id exists.

GET /webadmin/rest/usertests/exists/?usertest=USERTEST	
usertest	User test name (required).
id	User test id number (optional).
Response	
<pre>{ "exists": "YES" }</pre>	

10.3.24.7 Variants

If no “id” parameter is given then a list of all user tests variants will be returned. If an “id” parameter is given then a list of the given user test’s variants will be returned.

GET /webadmin/rest/usertests/variants/	
GET /webadmin/rest/usertests/variants/?id=ID	
id	User test id number
Response	
<pre>[{ "id": "123", "usertest": "ABC", "variant": "Colour test 1", }, { "id": "123", "usertest": "ABC", "variant": "Colour test 2", },]</pre>	



```
[
  {
    "id": "789",
    "usertest": "XYZ",
    .....
    "variant": "Colour test 1",
    .....
  },
  {
    "id": "789",
    "usertest": "XYZ",
    .....
    "variant": "Colour test 2",
    .....
  }
]
```

10.3.24.8 Clear User Tests Data

If a “usertest” parameter is given then user test data older than given or current date/time will marked to be cleared when website users access the website.

POST /webadmin/rest/usertests/clear/usertest=USERTEST& datetime=DATETIME	
usertest	User test name, or “*” for all user tests (required).
datetime	Date and time older than which user tests data should be cleared in the format “YYYY-MM-DD hh:mm:ss”; otherwise the current date and time will be used (optional).
Response	
{ "error": "" }	

10.3.25 User Test Results

The “/webadmin/rest/experience/usertest/” REST API functions returns results data for a user test.

GET /webadmin/rest/usertest/id=ID GET /webadmin/rest/usertest/usertest=ID	
id	User test id number (id or usertest required).
usertest	User test name (id or usertest required).
Response	
{ "id": "", "usertest": "", "description": "", "variants": "", "type": "", "confidence": "", "probability": "", "visitors": "", "goals": "", "active": "", "scheduled": "", "unscheduled": "", "conversions": [] }	



⋮

}

10.3.26 Workflows

The “/webadmin/rest/workflows/” REST API function returns all data for one or more workflows and creates, updates and deletes workflows.

10.3.26.1 List

If no “id” parameter is given then a list of workflows will be returned.

GET /webadmin/rest/workflows/	
Response	
<pre>[{ "id": "123", "title": "ABC", "action": "abc", }, , { "id": "789", "title": "XYZ", "action": "xyz", }]</pre>	

10.3.26.2 Read

If an “id” parameter is given then a GET request will return that workflow action’s data. The returned data includes all the workflow action attributes for the workflow action.

GET /webadmin/rest/workflows/?id=ID	
id	Workflow action id number (required).
Response	
<pre>{ "id": "9", "title": "Two-step approval", "action": "Approve publishing", "fromstate": "Approved", "tostate": "", "usertype": "null", "usergroup": "null", "contentclass": "null", "contenttype": "null", "contentgroup": "null", "version": "null", "notify_email": "null", "contentchanges": "null", "workflow_program": "null", "userrestriction": "null" }</pre>	



10.3.26.3 Create

A POST request will create a new workflow action with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the workflow action with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created workflow action’s data. The returned data includes all the workflow action attributes for the workflow action.

POST /webadmin/rest/workflows/	
id	Workflow action id number to be copied (optional).
.....	All or any number and combination of workflow action attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ", "action": "xyz", }	
Response	
{ "id": "789", "title": "XYZ", "action": "xyz", }	

10.3.26.4 Update

A PUT request will update the existing workflow action with the given “id” number with the given data attributes.

The PUT request will return the updated workflow action’s data. The returned data includes all the workflow action attributes for the workflow action.

PUT /webadmin/rest/workflows/	
id	Workflow action id number to be updated (required).
.....	All or any number and combination of workflow action attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "fromstate": "Approved", "tostate": "" }	
Response	
{ "id": "123",	



```
"title": "ABC",  
"action": "abc",  
"fromstate": "Approved",  
"tostate": "",  
.....  
}
```

10.3.26.5 Delete

A DELETE request will delete the existing workflow action with the given “id” number.

The DELETE request will return the deleted workflow action’s data. The returned data includes all the workflow action attributes for the workflow action.

DELETE /webadmin/rest/workflows/	
id	Workflow action id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "title": "ABC", "action": "abc", }	

10.3.26.6 Workflow States

A list of all workflow states for the configured workflows will be returned.

GET /webadmin/rest/workflows/states/	
Response	
[{ "state": "Started" } , , { "state": "Ended" }]]	

10.3.27 Workflow Actions

The “/webadmin/rest/workflowactions/” REST API function returns all workflow actions options for a given content item workflow status and user.

10.3.27.1 List

GET
/webadmin/rest/workflowactions/?fromstate=STATE&usergroups=USERGROUPS&usertype s=USERTYPES&contentclass=CONTENTCLASS&contentgroup=CONTENTGROUP&cont



enttype=CONTENTTYPE&version=VERSION	
fromstate	Content item's current workflow state.
usergroups	User's assigned user groups.
usertypes	User's assigned user types.
contentclass	Content item's content class.
contentgroup	Content item's content group.
contenttype	Content item's content type.
Response	
<pre>{ "workflowactionsoptions": "<option value=\"9\">Two-step approval: Approve publishing [= &gt;]</option>\r\n<option value=\"5\">Two-step approval: Approve publishing [= &gt;Approved by manager]</option>\r\n<option value=\"6\">Two- step approval: Approve publishing [= &gt;Approved]</option>\r\n<option value=\"1\">Two-step approval: Keep private [= &gt;Private]</option>\r\n<option value=\"8\">Two-step approval: Reject publishing [= &gt;Approved by manager]</option>\r\n<option value=\"7\">Two- step approval: Reject publishing [= &gt;Pending]</option>\r\n<option value=\"4\">Two-step approval: Reject publishing [= &gt;Private]</option>\r\n<option value=\"2\">Two-step approval: Request approval and publishing [= &gt;Pending]</option>\r\n<option value=\"3\">Two- step approval: Request approval and publishing [= &gt;Pending]</option>\r\n", "workflowactions": [{ "id": "1", "title": "Two-step approval", "action": "Keep private", "fromstate": "", "tostate": "Private" }, { "id": "2", "title": "Two-step approval", "action": "Request approval and publishing", "fromstate": "", "tostate": "Pending" }, { "id": "3", "title": "Two-step approval", "action": "Request approval and publishing", "fromstate": "Private", "tostate": "Pending" }, { "id": "4", "title": "Two-step approval", "action": "Reject publishing", "fromstate": "Pending", "tostate": "Private" }, { "id": "5", "title": "Two-step approval", "action": "Approve publishing", "fromstate": "Pending", "tostate": "Approved by manager" }, { "id": "6", "title": "Two-step approval", "action": "Approve publishing", "fromstate": "Approved by manager",</pre>	



```
        "tostate": "Approved"
      },
      {
        "id": "7",
        "title": "Two-step approval",
        "action": "Reject publishing",
        "fromstate": "Approved by manager",
        "tostate": "Pending"
      },
      {
        "id": "8",
        "title": "Two-step approval",
        "action": "Reject publishing",
        "fromstate": "Approved",
        "tostate": "Approved by manager"
      },
      {
        "id": "9",
        "title": "Two-step approval",
        "action": "Approve publishing",
        "fromstate": "Approved",
        "tostate": ""
      }
    ]
  }
```

10.3.28 Workflow Programs

The “/webadmin/rest/workflowprograms/” REST API function returns all workflow programs available for the configuration of work actions.

10.3.28.1 List

GET /webadmin/rest/workflowprograms/	
Response	
<pre>[{ "workflow_program": "Test Workflow Action", "filename": "Test Workflow Action.jsp" }, "description": "" }, { "workflow_program": "Workflow Action Module Template", "filename": "Workflow Action Module Template.jsp", "description": "" }]</pre>	

10.3.29 Comments

The “/webadmin/rest/comments/” REST API function returns all data for one or more comments and creates, updates and deletes comments.

10.3.29.1 List

If no “id” parameter is given then a list of comments will be returned.

GET /webadmin/rest/comments/	



Response
<pre>[{ "id": "123", "title": "ABC", }, { "id": "789", "title": "XYZ", }]</pre>

10.3.29.2 Read

If an “id” parameter is given then a GET request will return that comment’s data. The returned data includes all the comment attributes for the comment as well as a number of utility attributes for administration usage.

GET /webadmin/rest/comments/?id=ID	
id	Comment id number (required).
Response	
<pre>{ "id": "15", "created": "2018-04-04 17:58:00", "created_by": "paulgreen", "title": "Job Applications", "content": "Job applications are automatically uploaded to this folder when users apply.", "section": "library", "cid": "", "class": "file", "group": "Careers", "type": "", "package": "", "bundle": "", "version": "", "status": "", "stock": "", "locked": "1", "_ctitle": "Job Applications (15)", "_ctitleurl": " Job Applications (15)", "class": "file", /* content class or database title */ "_user": true, "_creator": false, "_editor": true, "_administrator": false }</pre>	



10.3.29.3 Create

A POST request will create a new comment with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the comment with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created comment’s data. The returned data includes all the comment attributes for the comment.

POST /webadmin/rest/comments/	
id	Comment id number to be copied (optional).
.....	All or any number and combination of comment attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "title": "XYZ", "content": "xyz" }</pre>	
Response	
<pre>{ "id": "789", "title": "XYZ", "content": "xyz", }</pre>	

10.3.29.4 Update

A PUT request will update the existing comment with the given “id” number with the given data attributes.

The PUT request will return the updated comment’s data. The returned data includes all the comment attributes for the comment.

PUT /webadmin/rest/comments/	
id	Comment id number to be updated (required).
.....	All or any number and combination of comment attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "title": "XYZ", "content": "xyz" }</pre>	
Response	
<pre>{ "id": "123", }</pre>	



```
"title": "XYZ",
"content": "xyz",
.....
}
```

10.3.29.5 Delete

A DELETE request will delete the existing comment with the given “id” number.

The DELETE request will return the deleted comment’s data. The returned data includes all the comment attributes for the comment.

DELETE /webadmin/rest/comments/	
Id	Comment id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "title": "ABC", }</pre>	

10.3.30 Projects

The “/webadmin/rest/projects/” REST API function returns all data for one or more projects and creates, updates and deletes projects.

10.3.30.1 List

If no “id” parameter is given then a list of projects will be returned.

GET /webadmin/rest/projects/	
Response	
<pre>[{ "id": "123", "title": "ABC", }, { "id": "789", "title": "XYZ", }]</pre>	



10.3.30.1.1 Livegrid (DEPRECATED)

This duplicates the general list functionality as described above but with output in Rico Livegrid XML format as used by current and older versions of the web content management system administration pages.

Note: DEPRECATED – use GET /webadmin/rest/projects/ (Please see section 10.3.30.1 List).

GET /webadmin/rest/projects/livegrid/	
Response	
XML code	

10.3.30.2 Read

If an “id” parameter is given then a GET request will return that project’s data. The returned data includes all the project attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the project.

GET /webadmin/rest/projects/?id=ID	
id	Project id number (required).
Response	
{ "id": "6", "created": "2018-04-04 16:59:39", "updated": "2018-04-04 17:11:16", "closed": "", "deleted": "", "created_by": "admin", "updated_by": "admin", "closed_by": "", "deleted_by": "", "title": "Refresh Product A Presentation", "description": "Review, suggest and implement a refresh of the the presentation of Product A.", "category": "", "status": "", "status_by": "admin", "priority": "", "severity": "", "duration_amount": "2", "duration_period": "604800", "started": "", "ended": "", "assets_contents": "484,177,182,183,188", "assets_contentgroups": "", "assets_contenttypes": "", "assets_imagegroups": "", "assets_imagetypes": "", "assets_filegroups": "", "assets_filetypes": "", "assets_linkgroups": "", "assets_linktypes": "", "assets_productgroups": "", "assets_producttypes": "",	



```
"assets_databases": "",
"assets_users": "",
"assets_usergroups": "",
"assets_usertypes": "",
"assets_orders": "",
"assets_sales": "",
"assets_segments": "",
"assets_usertests": "",
"assets_heatmaps": "",
"assets_usage": "",
"assets_clients": "",
"assets_hostinggroups": "",
"assets_hostingtypes": "",
"assets_websites": "",
"assets_packages": "",
"assets_bundles": "",
"assets_other": "",

"users_group": "",
"users_type": "",
"users_users": "",
"creators_group": "",
"creators_type": "",
"creators_users": "",
"editors_group": "",
"editors_type": "",
"editors_users": "",
"publishers_group": "",
"publishers_type": "",
"publishers_users": "",
"administrators_group": "",
"administrators_type": "",
"administrators_users": "",

"categories":
":#EEEEEE:white\r\nBacklog:#E2445C:white\r\nReview:#FDAB3D:white\r\nImplement:
#00C875:white\r\nUpcoming:#0086C0:white\r\nSuggest:#579BFC:white\r\n:#A25DDC:w
hite\r\n:#037F4C:white\r\n:#CAB641:white\r\n:#FFCB00:white\r\n:#333333:white\r
\n:#BB3354:white\r\n:#FF158A:white\r\n:#FF5AC4:white\r\n:#784BD1:white\r\n:#9C
D326:white\r\n:#66CCFF:white\r\n:#808080:white\r\n:#7F5347:white\r\n:#FF642E:w
hite",
"statuses": "To Begin:#EEEEEE:white\r\nIn
Progress:#00C875:white\r\nAwaiting Review:#FDAB3D:white\r\nNot
Started:#E2445C:white\r\nAwaiting
Suggestions:#0086C0:white\r\n:#579BFC:white\r\nAwaiting
Implementation:#A25DDC:white\r\nCompleted:#037F4C:white\r\n:#CAB641:white\r\n:
#FFCB00:white\r\n:#333333:white\r\n:#BB3354:white\r\n:#FF158A:white\r\n:#FF5AC
4:white\r\n:#784BD1:white\r\n:#9CD326:white\r\n:#66CCFF:white\r\n:#808080:whit
e\r\n:#7F5347:white\r\n:#FF642E:white",
"priorities":
":#EEEEEE:white\r\nLow:#00C875:white\r\nMedium:#FDAB3D:white\r\nHigh:#E2445C:w
hite\r\n:#0086C0:white\r\n:#579BFC:white\r\nCritical:#A25DDC:white\r\nNone:#03
7F4C:white\r\n:#CAB641:white\r\n:#FFCB00:white\r\n:#333333:white\r\n:#BB3354:w
hite\r\n:#FF158A:white\r\n:#FF5AC4:white\r\n:#784BD1:white\r\n:#9CD326:white\r
\n:#66CCFF:white\r\n:#808080:white\r\n:#7F5347:white\r\n:#FF642E:white",
"severities":
":#EEEEEE:white\r\nMinor:#00C875:white\r\nModerate:#FDAB3D:white\r\nSignifican
t:#E2445C:white\r\n:#0086C0:white\r\n:#579BFC:white\r\nExtensive:#A25DDC:white
\r\nNone:#037F4C:white\r\n:#CAB641:white\r\n:#FFCB00:white\r\n:#333333:white\r
\n:#BB3354:white\r\n:#FF158A:white\r\n:#FF5AC4:white\r\n:#784BD1:white\r\n:#9C
D326:white\r\n:#66CCFF:white\r\n:#808080:white\r\n:#7F5347:white\r\n:#FF642E:w
hite",

"user": true,
"creator": true,
```



```
"editor": true,  
"publisher": true,  
"administrator": true  
}
```

10.3.30.3 Create

A POST request will create a new project with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the project with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created project’s data. The returned data includes all the project attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the project.

POST /webadmin/rest/projects/	
id	Project id number to be copied (optional).
.....	All or any number and combination of project attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "789", "title": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }	

10.3.30.4 Update

A PUT request will update the existing project with the given “id” number with the given data attributes.

The PUT request will return the updated project’s data. The returned data includes all the project attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the project.

PUT /webadmin/rest/projects/	
id	Project id number to be updated (required).
.....	All or any number and combination of project attributes as used by the web content management



	system (see the GET request example response above).
Request	
<pre>{ "id": "123", "title": "XYZ" }</pre>	
Response	
<pre>{ "id": "123", "title": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.30.5 Delete

A DELETE request will delete the existing project with the given “id” number.

The DELETE request will return the deleted project data. The returned data includes all the project attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the project.

DELETE /webadmin/rest/projects/	
id	Project id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "title": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.31 Project Tasks

The “/webadmin/rest/projecttasks/” REST API function returns all data for one or more project tasks and creates, updates and deletes project tasks.

10.3.31.1 List

If a “project_id” is given and no “id” parameter is given then a list of project tasks will be returned.



GET /webadmin/rest/projecttasks/	
project_id	Project id number (required).
Response	
<pre>[{ "id": "123", "project_id": "456", "title": "ABC", }, { "id": "789", "project_id": "456", "title": "XYZ", }]</pre>	

10.3.31.2 Read

If an “id” parameter is given then a GET request will return that project task’s data. The returned data includes all the project task attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the project task.

GET /webadmin/rest/projecttasks/?id=ID	
id	Project task id number (required).
Response	
<pre>{ "id": "15", "project_id": "6", "parent_id": "", "created": "2018-04-04 17:03:43", "updated": "2018-04-04 17:06:20", "closed": "", "deleted": "", "created_by": "admin", "updated_by": "admin", "closed_by": "", "deleted_by": "", "_created_by": false, /* created by the currently logged in website administrator */ "title": "Review Current Presentation", "description": "Review the current presentation and make notes about areas that need to be improved.", "category": "Review", "status": "In Progress", "status_by": "admin", "priority": "Medium", "severity": "Extensive", "duration_amount": "2", "duration_period": "86400", "duration": "2 days",</pre>	



```
"started": "",
"ended": "",

"after_tasks": "",
"before_tasks": "",

"assets_contents": "",
"assets_contentgroups": "",
"assets_contenttypes": "",
"assets_imagegroups": "",
"assets_imagetypes": "",
"assets_filegroups": "",
"assets_filetypes": "",
"assets_linkgroups": "",
"assets_linktypes": "",
"assets_productgroups": "",
"assets_producttypes": "",
"assets_databases": "",
"assets_users": "",
"assets_usergroups": "",
"assets_usertypes": "",
"assets_orders": "",
"assets_sales": "",
"assets_segments": "",
"assets_usertests": "",
"assets_heatmaps": "",
"assets_usage": "",
"assets_clients": "",
"assets_hostinggroups": "",
"assets_hostingtypes": "",
"assets_websites": "",
"assets_packages": "",
"assets_bundles": "",
"assets_other": "",

"users_group": "",
"users_type": "",
"users_users": "",
"creators_group": "",
"creators_type": "",
"creators_users": "",
"editors_group": "",
"editors_type": "",
"editors_users": "",
"publishers_group": "",
"publishers_type": "",
"publishers_users": "",
"administrators_group": "",
"administrators_type": "",
"administrators_users": "",

"user": true,
"creator": true,
"editor": true,
"publisher": true,
"administrator": true
}
```

10.3.31.3 Create

A POST request will create a new project task with the given data attributes and an automatically assigned unique “id” number.



If an “id” parameter is given then the project task with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created project task’s data. The returned data includes all the project task attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the project task.

POST /webadmin/rest/projecttasks/	
id	Project task id number to be copied (optional).
.....	All or any number and combination of project task attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "project_id": "456", "id": "123", "title": "XYZ" }</pre>	
Response	
<pre>{ "id": "789", "project_id": "456", "title": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>	

10.3.31.4 Update

A PUT request will update the existing project task with the given “id” number with the given data attributes.

The PUT request will return the updated project task’s data. The returned data includes all the project task attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the project task.

PUT /webadmin/rest/projecttasks/	
id	Project task id number to be updated (required).
.....	All or any number and combination of project task attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "title": "XYZ" }</pre>	



Response
<pre>{ "id": "123", "project_id": "456", "title": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>

10.3.31.5 Delete

A DELETE request will delete the existing project task with the given “id” number.

The DELETE request will return the deleted project task data. The returned data includes all the project task attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the project task.

DELETE /webadmin/rest/projecttasks/	
id	Project task id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "project_id": "456", "title": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>	

10.3.32 Currencies

The “/webadmin/rest/currency/” REST API function returns all data for one or more currencies and creates, updates and deletes currencies.

10.3.32.1 List

If no “id” parameter is given then a list of currencies will be returned.

GET /webadmin/rest/currency/	
Response	
<pre>[{ "id": "123",</pre>	



```
        "title": "ABC",
        .....
    }
    ,
    .....
    ,
    {
        "id": "789",
        "title": "XYZ",
        .....
    }
}
```

10.3.32.2 Read

If an “id” parameter is given then a GET request will return that currency’s data. The returned data includes all the currency attributes for the currency.

GET /webadmin/rest/currency/?id=ID	
id	Currency id number (required).
Response	
{ "id": "1", "title": "GBP", "symbol": "£", "rate": "100" }	

10.3.32.3 Create

A POST request will create a new currency with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the currency with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created currency’s data. The returned data includes all the currency attributes for the currency.

POST /webadmin/rest/currency/	
id	Currency id number to be copied (optional).
.....	All or any number and combination of currency attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "789", "title": "XYZ", }	



10.3.32.4 Update

A PUT request will update the existing currency with the given “id” number with the given data attributes.

The PUT request will return the updated currency’s data. The returned data includes all the currency attributes for the currency.

PUT /webadmin/rest/currency/	
id	Currency id number to be updated (required).
.....	All or any number and combination of currency attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "123", "title": "XYZ", }	

10.3.32.5 Delete

A DELETE request will delete the existing currency with the given “id” number.

The DELETE request will return the deleted currency’s data. The returned data includes all the currency attributes for the currency.

DELETE /webadmin/rest/currency/	
id	Currency id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "title": "ABC", }	

10.3.32.6 Export

A list of the currencies will be returned in comma-separated values (csv) format.

GET /webadmin/rest/currency/export/csv/	
Response	



```
id,title,symbol,rate
123,GBP,£,100
456,EUR,€,90
789,USD,$,80
```

10.3.32.7 Import

A comma-separated values (csv) format file will be imported, and the output text (if any) from the import will be returned; optionally as HTML code.

POST /webadmin/rest/currency/import/	
file	HTML FORM “file” type input upload (required).
Response	
{ "output": "" }	

POST /webadmin/rest/currency/import/html/	
file	HTML FORM “file” type input upload (required).
Response	
HTML code	

10.3.33 Products

10.3.33.1 Export

A list of the products will be returned in comma-separated values (csv) format.

GET /webadmin/rest/products/export/csv/	
Response	
id,group,type,version,title,brand,colour,size,weight,width,height,depth,volume,info,options,code,location,currency,price,period,cost,stock,stocktext,lowstock,lowstocktext,restocked,nostocktext,prebackorder,author,description,keywords,metainfo	

10.3.33.2 Import

A comma-separated values (csv) format file will be imported, and the output text (if any) from the import will be returned; optionally as HTML code.

POST /webadmin/rest/products/import/	
file	HTML FORM “file” type input upload (required).
Response	
{ "output": "" }	

POST /webadmin/rest/products/import/html/	
file	HTML FORM “file” type input upload (required).



Response
HTML code

10.3.34 Product Groups

The “/webadmin/rest/productgroups/” REST API function returns all data for one or more product groups and creates, updates and deletes product groups.

10.3.34.1 List

If no “id” parameter is given then a list of product groups will be returned.

GET /webadmin/rest/productgroups/	
Response	
<pre>[{ "id": "123", "productgroup": "ABC" } , , { "id": "789", "productgroup": "XYZ" }]</pre>	

10.3.34.2 Read

If an “id” parameter is given then a GET request will return that product group’s data. The returned data includes all the product group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the product group.

GET /webadmin/rest/productgroups/?id=ID	
id	Product group id number (required).
Response	
<pre>{ "id": "1", "productgroup": "Subscription Services", "parentgroup": "", "_subgroups": 0, "template": "", "stylesheet": "", "title_prefix": "", "title_suffix": "", "doctype": "", "users_group": "", "users_type": "", "users_users": "", "creators_group": "Shop Managers",</pre>	



```
"creators_type": "",
"creators_users": "",
"developers_group": "Website Developers",
"developers_type": "",
"developers_users": "",
"editors_group": "Shop Managers",
"editors_type": "",
"editors_users": "",
"publishers_group": "Shop Managers",
"publishers_type": "",
"publishers_users": "",
"administrators_group": "Website Administrators",
"administrators_type": "",
"administrators_users": "",

"user": true,
"creator": true,
"developer": true,
"editor": true,
"publisher": true,
"administrator": true
}
```

10.3.34.3 Create

A POST request will create a new product group with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the product group with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created product group’s data. The returned data includes all the product group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the product group.

POST /webadmin/rest/productgroups/	
id	Product group id number to be copied (optional).
.....	All or any number and combination of product group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "productgroup": "XYZ" }	
Response	
{ "id": "789", "productgroup": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, }	



```
"administrator": true  
}
```

10.3.34.4 Update

A PUT request will update the existing product group with the given “id” number with the given data attributes.

The PUT request will return the updated product group’s data. The returned data includes all the product group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the product group.

PUT /webadmin/rest/productgroups/	
id	Product group id number to be updated (required).
.....	All or any number and combination of product group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "template": "456", "stylesheet": "789" }	
Response	
{ "id": "123", "productgroup": "ABC", "parentgroup": "", "template": "456", "stylesheet": "789" "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.34.5 Delete

A DELETE request will delete the existing product group with the given “id” number.

The DELETE request will return the deleted product group’s data. The returned data includes all the product group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the product group.

DELETE /webadmin/rest/productgroups/	
id	Product group id number (required).
Request	



<pre>{ "id": "123" }</pre>
Response
<pre>{ "id": "123", "productgroup": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>

10.3.34.6 Exists

If a “productgroup” parameter is given then a GET request will return “YES” or “NO” depending on if a product group with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a product group with the given name and another id than the given id exists.

GET /webadmin/rest/productgroups/exists/?productgroup=PRODUCTGROUP	
productgroup	Product group name (required).
id	Product group id number (optional).
Response	
<pre>{ "exists": "YES" }</pre>	

10.3.35 Product Types

The “/webadmin/rest/producttypes/” REST API function returns all data for one or more product types and creates, updates and deletes product types.

10.3.35.1 List

If no “id” parameter is given then a list of product types will be returned.

GET /webadmin/rest/producttypes/	
Response	
<pre>[{ "id": "123", "producttype": "ABC" }, { "id": "789", "producttype": "XYZ" }</pre>	



```
}  
]
```

10.3.35.2 Read

If an “id” parameter is given then a GET request will return that product type’s data. The returned data includes all the product type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the product type.

GET /webadmin/rest/producttypes/?id=ID	
id	Product type id number (required).
Response	
<pre>{ "id": "1", "producttype": "Digital Services", "parenttype": "", "_subtypes": 0, "template": "", "stylesheet": "", "title_prefix": "", "title_suffix": "", "doctype": "", "users_group": "", "users_type": "", "users_users": "", "creators_group": "Shop Managers", "creators_type": "", "creators_users": "", "developers_group": "Website Developers", "developers_type": "", "developers_users": "", "editors_group": "Shop Managers", "editors_type": "", "editors_users": "", "publishers_group": "Shop Managers", "publishers_type": "", "publishers_users": "", "administrators_group": "Website Administrators", "administrators_type": "", "administrators_users": "", "user": true, "creator": true, "developer": true, "editor": true, "publisher": true, "administrator": true }</pre>	

10.3.35.3 Create

A POST request will create a new product type with the given data attributes and an automatically assigned unique “id” number.



If an “id” parameter is given then the product type with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created product type’s data. The returned data includes all the product type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the product type.

POST /webadmin/rest/producttypes/	
id	Product type id number to be copied (optional).
.....	All or any number and combination of product type attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "producttype": "XYZ" }	
Response	
{ "id": "789", "producttype": "XYZ", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }	

10.3.35.4 Update

A PUT request will update the existing product type with the given “id” number with the given data attributes.

The PUT request will return the updated product type’s data. The returned data includes all the product type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the product type.

PUT /webadmin/rest/producttypes/	
id	Product type id number to be updated (required).
.....	All or any number and combination of product type attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "template": "456", "stylesheet": "789" }	



Response
<pre>{ "id": "123", "producttype": "ABC", "parenttype": "", "template": "456", "stylesheet": "789" "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>

10.3.35.5 Delete

A DELETE request will delete the existing product type with the given “id” number.

The DELETE request will return the deleted product type’s data. The returned data includes all the product type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher”, “developer” and “administrator” access permissions for the product type.

DELETE /webadmin/rest/producttypes/	
id	Product type id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "producttype": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "developer": true, "administrator": true }</pre>	

10.3.35.6 Exists

If a “producttype” parameter is given then a GET request will return “YES” or “NO” depending on if a product type with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a product type with the given name and another id than the given id exists.

GET /webadmin/rest/producttypes/exists/?producttype=PRODUCTTYPE	
producttype	Product type name (required).



.....

id	Product type id number (optional).
Response	
{ "exists": "YES" }	

10.3.36 Product Availability Programs

The “/webadmin/rest/productavailability/” REST API function returns all product availability API programs available for product delivery for products.

10.3.36.1 List

GET /webadmin/rest/productavailability/	
Response	
[{ "filename": "domainnameregistration.jsp" }, "description": "" }, { "filename": "test-available.jsp" }, "description": "" }, { "filename": "test-instock.jsp" }, "description": "" }, { "filename": "test-unavailable.jsp", "description": "" }]	

10.3.37 Product Delivery Programs

The “/webadmin/rest/productdelivery/” REST API function returns all product delivery API programs available for product delivery for products.

10.3.37.1 List

GET /webadmin/rest/productdelivery/	
Response	
[{ "filename": "domainnameregistration.jsp" }, "description": "" }, { "filename": "hello.jsp" }, "description": "" }]	



10.3.38 Discounts

The “/webadmin/rest/discounts/” REST API function returns all data for one or more discounts and creates, updates and deletes discounts.

10.3.38.1 List

If no “id” parameter is given then a list of discounts will be returned.

GET /webadmin/rest/discounts/	
Response	
<pre>[{ "id": "123", "title": "ABC", }, , { "id": "789", "title": "XYZ", }]</pre>	

10.3.38.2 Read

If an “id” parameter is given then a GET request will return that discount’s data. The returned data includes all the discount attributes for the discount.

GET /webadmin/rest/discounts/?id=ID	
id	Discount id number (required).
Response	
<pre>{ "id": "1", "title": "10% Off Everything", "country": "", "state": "", "product_id": "", "product_group": "", "product_type": "", "product_weight_from": "0.000", "product_weight_to": "0.000", "product_volume_from": "0.00", "product_volume_to": "0.00", "product_width_from": "0.00", "product_width_to": "0.00", "product_height_from": "0.00", "product_height_to": "0.00", "product_depth_from": "0.00", "product_depth_to": "0.00", "quantity_from": "0", "quantity_to": "0", "total_currency": "", "total_from": "0.00", "total_to": "0.00", "total_weight_from": "0.000", "total_weight_to": "0.000", </pre>	



```
{
  "total_volume_from": "0.00",
  "total_volume_to": "0.00",
  "total_width_from": "0.00",
  "total_width_to": "0.00",
  "total_height_from": "0.00",
  "total_height_to": "0.00",
  "total_depth_from": "0.00",
  "total_depth_to": "0.00",
  "discount_description": "",
  "discount_type": "general",
  "discount_quantity": "0",
  "discount_quantity2": "0",
  "discount_products": "",
  "discount_currency": "€",
  "discount_amount": "10.00",
  "discount_orderitems": "total",
  "user_username": "",
  "user_group": "",
  "user_type": "",
  "user_code": "",
  "period_start": "",
  "period_end": ""
}
```

10.3.38.3 Create

A POST request will create a new discount with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the discount with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created discount’s data. The returned data includes all the discount attributes for the discount.

POST /webadmin/rest/discounts/	
id	Discount id number to be copied (optional).
.....	All or any number and combination of discount attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "789", "title": "XYZ", }	

10.3.38.4 Update

A PUT request will update the existing discount with the given “id” number with the given data attributes.



The PUT request will return the updated discount's data. The returned data includes all the discount attributes for the currency.

PUT /webadmin/rest/discounts/	
id	Discount id number to be updated (required).
.....	All or any number and combination of discount attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "123", "title": "ABC", }	

10.3.38.5 Delete

A DELETE request will delete the existing discount with the given "id" number.

The DELETE request will return the deleted discount's data. The returned data includes all the discount attributes for the discount.

DELETE /webadmin/rest/discounts/	
id	Discount id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "title": "ABC", }	

10.3.39 Shipping

The "/webadmin/rest/shipping/" REST API function returns all data for one or more shipping rates and creates, updates and deletes shipping rates.

10.3.39.1 List

If no "id" parameter is given then a list of shipping rates will be returned.

GET /webadmin/rest/shipping/	
Response	
[{ 	



```
    "id": "123",  
    "title": "ABC",  
    .....  
  },  
  ,  
  .....  
  ,  
  {  
    "id": "789",  
    "title": "XYZ",  
    .....  
  }  
]
```

10.3.39.2 Read

If an “id” parameter is given then a GET request will return that shipping rate’s data. The returned data includes all the shipping rate attributes for the shipping rate.

GET /webadmin/rest/shipping/?id=ID	
id	Shipping rate id number (required).
Response	
<pre>{ "id": "1", "title": "Royal Mail 2nd Class Small Parcel", "country": "", "state": "", "product_id": "", "product_group": "Physical Products", "product_type": "", "product_weight_from": "0.000", "product_weight_to": "0.000", "product_volume_from": "0.00", "product_volume_to": "0.00", "product_width_from": "0.00", "product_width_to": "0.00", "product_height_from": "0.00", "product_height_to": "0.00", "product_depth_from": "0.00", "product_depth_to": "0.00", "quantity_from": "0", "quantity_to": "0", "total_currency": "", "total_from": "0.00", "total_to": "0.00", "total_weight_from": "0.000", "total_weight_to": "2.000", "total_volume_from": "0.00", "total_volume_to": "0.00", "total_width_from": "0.00", "total_width_to": "0.00", "total_height_from": "0.00", "total_height_to": "0.00", "total_depth_from": "0.00", "total_depth_to": "0.00", "ship_description": "Basic Shipping", "ship_currency": "", "ship_order": "3.00", "ship_item": "0.00", "ship_percent": "0.00", "ship_total": "0.00" }</pre>	



10.3.39.3 Create

A POST request will create a new shipping rate with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the shipping rate with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created shipping rate’s data. The returned data includes all the shipping rate attributes for the shipping rate.

POST /webadmin/rest/shipping/	
id	Shipping rate id number to be copied (optional).
.....	All or any number and combination of shipping rate attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "789", "title": "XYZ", }	

10.3.39.4 Update

A PUT request will update the existing shipping rate with the given “id” number with the given data attributes.

The PUT request will return the updated shipping rate’s data. The returned data includes all the shipping rate attributes for the shipping rate.

PUT /webadmin/rest/shipping/	
id	Shipping rate id number to be updated (required).
.....	All or any number and combination of shipping rate attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "123", "title": "ABC", }	



.....

```
}
```

10.3.39.5 Delete

A DELETE request will delete the existing shipping rate with the given “id” number.

The DELETE request will return the deleted shipping rate’s data. The returned data includes all the shipping rate attributes for the shipping rate.

DELETE /webadmin/rest/shipping/	
id	Shipping rate id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "title": "ABC", }	

10.3.40 Tax

The “/webadmin/rest/tax/” REST API function returns all data for one or more tax rates and creates, updates and deletes tax rates.

10.3.40.1 List

If no “id” parameter is given then a list of tax rates will be returned.

GET /webadmin/rest/tax/	
Response	
[{ "id": "123", "title": "ABC", } , , { "id": "789", "title": "XYZ", }]	

10.3.40.2 Read

If an “id” parameter is given then a GET request will return that tax rate’s data. The returned data includes all the tax rate attributes for the tax rate.

GET /webadmin/rest/tax/?id=ID



.....

id	Tax rate id number (required).
Response	
<pre>{ "id": "1", "title": "United Kingdom VAT", "country": "United Kingdom", "state": "", "product_id": "", "product_group": "", "product_type": "", "product_weight_from": "0.000", "product_weight_to": "0.000", "product_volume_from": "0.00", "product_volume_to": "0.00", "product_width_from": "0.00", "product_width_to": "0.00", "product_height_from": "0.00", "product_height_to": "0.00", "product_depth_from": "0.00", "product_depth_to": "0.00", "quantity_from": "0", "quantity_to": "0", "total_currency": "", "total_from": "0.00", "total_to": "0.00", "total_weight_from": "0.000", "total_weight_to": "0.000", "total_volume_from": "0.00", "total_volume_to": "0.00", "total_width_from": "0.00", "total_width_to": "0.00", "total_height_from": "0.00", "total_height_to": "0.00", "total_depth_from": "0.00", "total_depth_to": "0.00", "tax_description": "United Kingdom VAT 20%", "tax_currency": "", "tax_order": "0.00", "tax_item": "0.00", "tax_percent": "20.00", "tax_total": "0.00" }</pre>	

10.3.40.3 Create

A POST request will create a new tax rate with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the tax rate with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created tax rate’s data. The returned data includes all the tax rate attributes for the tax rate.

POST /webadmin/rest/tax/	
id	Tax rate id number to be copied (optional).
.....	All or any number and combination of tax rate attributes as used by the web content management system (see the GET request example response



	above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "789", "title": "XYZ", }	

10.3.40.4 Update

A PUT request will update the existing tax rate with the given “id” number with the given data attributes.

The PUT request will return the updated tax rate’s data. The returned data includes all the tax rate attributes for the tax rate.

PUT /webadmin/rest/tax/	
id	Tax rate id number to be updated (required).
.....	All or any number and combination of tax rate attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "123", "title": "ABC", }	

10.3.40.5 Delete

A DELETE request will delete the existing tax rate with the given “id” number.

The DELETE request will return the deleted tax rate’s data. The returned data includes all the tax rate attributes for the tax rate.

DELETE /webadmin/rest/tax/	
id	Tax rate id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123",	



```
    "title": "ABC",  
    .....  
}
```

10.3.41 Orders

The “/webadmin/rest/orders/” REST API function returns all data for one or more orders and creates, updates and deletes orders.

10.3.41.1 List

If no “id” parameter is given then a list of orders will be returned.

GET /webadmin/rest/orders/	
Response	
<pre>[{ "id": "123", }, , { "id": "789", }]</pre>	

10.3.41.1.1 Livegrid (DEPRECATED)

This duplicates the general list functionality as described above but with output in Rico Livegrid XML format as used by current and older versions of the web content management system administration pages.

Note: DEPRECATED – use GET /webadmin/rest/orders/ (Please see section 10.3.41.1 List).

GET /webadmin/rest/orders/livegrid/	
Response	
XML code	

10.3.41.1.2 Delete Confirmation

If a “index=delete” parameter is given then a list of orders which match the “id” parameters will be returned.

GET /webadmin/rest/orders/?index=delete&id=ID&id=ID&id=ID	
Index	“delete”.
id	Order id.
Response	
<pre>[{ "id": "123",</pre>	



```
.....
}
,
.....
,
{
  "id": "789",
  .....
}
]
```

10.3.41.2 Read

If an “id” parameter is given then a GET request will return that order’s data. The returned data includes all the order attributes for the order.

GET /webadmin/rest/orders/?id=ID	
id	Order id number (required).
Response	
{ "id": "1", "user_id": "5", "created": "2013-08-20 12:08:48", "updated": "2013-08-20 12:08:48", "closed": "", "created_by": "johnsmith", "updated_by": "johnsmith", "closed_by": "", "paid": "", "status": "", "_status": "", "status_by": "", "checkedout": "", "revision": "", "card_type": "VISA", "card_number": "4213451248543248", "card_issuedmonth": "", "card_issuedyear": "", "card_expirymonth": "04", "card_expiryyear": "2014", "card_name": "Mr. John Smith", "card_cvc": "564", "card_issue": "", "card_postalcode": "", "delivery_name": "John Smith", "delivery_organisation": "", "delivery_address": "10 Melbourne Street", "delivery_postalcode": "E17 2SW", "delivery_city": "London", "delivery_state": "", "delivery_country": "United Kingdom", "delivery_phone": "", "delivery_fax": "", "delivery_email": "admin@asbrusoft.com", "delivery_website": "", "invoice_name": "", "invoice_organisation": "", "invoice_address": "", }	



```
"invoice_postalcode": "",
"invoice_city": "",
"invoice_state": "",
"invoice_country": "",
"invoice_phone": "",
"invoice_fax": "",
"invoice_email": "",
"invoice_website": "",

"order_quantity": "1",
"order_currency": "&#163;",
"order_currencytitle": "GBP",
"discount_description": "",
"discount_description": "",
"discount_total": "0.00",
"discount_total_base": "0.00",
"order_subtotal": "100.00",
"order_subtotal_base": "100.00",
"tax_description": "<span class=\"tax\">United Kingdom VAT 17.5%</span>",
"tax_description": "United Kingdom VAT 17.5%",
"tax_total": "17.50",
"tax_total_base": "17.50",
"shipping_description": "<span class=\"shipping\">1 x Basic Shipping</span>",
"shipping_description": "1 x Basic Shipping",
"shipping_total": "3.50",
"shipping_total_base": "3.50",
"order_total": "121.00",
"order_total_base": "121.00",
"order_availability": "",

"usersegments": "",
"usertests": ""

"administrator": true,
"creator": true,
"editor": true,
"user": true,
"workflowoptions": "",
"workflowpermissions": "",
"workflow_action_select_options": "",
"checkedout_assign_select_options": ""
}
```

10.3.41.3 Create

A POST request will create a new order with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the order with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created order’s data. The returned data includes all the order attributes for the order.

POST /webadmin/rest/orders/	
id	Order id number to be copied (optional).
.....	All or any number and combination of order attributes as used by the web content management system (see the GET request example response



	above).
Request	
{ "id": "123", "user_id": "456" }	
Response	
{ "id": "789", "user_id": "456" }	

10.3.41.4 Update

A PUT request will update the existing order with the given “id” number with the given data attributes.

The PUT request will return the updated order’s data. The returned data includes all the order attributes for the order.

PUT /webadmin/rest/orders/	
id	Order id number to be updated (required).
.....	All or any number and combination of order attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "paid": "2020-01-31 23:59:59" }	
Response	
{ "id": "123", "paid": "2020-01-31 23:59:59", }	

10.3.41.5 Delete

A DELETE request will delete the existing order with the given “id” number.

The DELETE request will return the deleted order data. The returned data includes all the order attributes for the order.

DELETE /webadmin/rest/orders/	
id	Order id number (required).
Request	
{ "id": "123" }	
Response	



```
{  
  "id": "123",  
  "user_id": "456"  
  .....  
}
```

DELETE /webadmin/rest/orders/	
id	Order id numbers (required).
Request	
<pre>{ "id": "123", "id": "456", "id": "789" }</pre>	
Response	
<pre>{ "error": "OK" }</pre>	

10.3.41.6 Checkin

The “/webadmin/rest/orders/checkin/” REST API function checkins the given orders.

The POST request returns an error response or “OK”.

POST /webadmin/rest/orders/checkin/	
id	Order id number to checkin (required).
Request	
<pre>{ "id": "123" }</pre>	
Request	
<pre>{ "id": ["123", "456", "789"] }</pre>	
Response	
<pre>{ "error": "XXXXXX" }</pre>	

10.3.41.7 Checkout

The “/webadmin/rest/orders/checkout/” REST API function checkouts the given orders.

The POST request returns an error response or “OK”.

POST /webadmin/rest/orders/checkout/	
id	Order id number to checkout (required).
Request	
<pre>{ "id": "123" }</pre>	
Request	
<pre>{ "id": ["123", "456", "789"] }</pre>	



}
Response
{ "error": "XXXXXX" }

10.3.41.8 Delete (DEPRECATED)

The “/webadmin/rest/orders/delete/” REST API function deletes the given orders.

The POST request returns an error response or “OK”.

Note: DEPRECATED – use DELETE /webadmin/rest/orders/ (Please see section 10.3.41.5 Delete).

POST /webadmin/rest/orders/delete/	
id	Order id number to be deleted (required).
Request	
{ "id": "123" }	
Request	
{ "id": ["123", "456", "789"] }	
Response	
{ "error": "XXXXXX" }	

10.3.41.9 Move (workflow)

The “/webadmin/rest/orders/move/” REST API function moves the given orders.

The POST request returns an error response or “OK”.

POST /webadmin/rest/orders/move/	
id	Order id number to be moved (required).
status	Workflow action id number (optional).
Request	
{ "id": "123", "status": "pending" }	
Request	
{ "id": ["123", "456", "789"], "status": "pending" }	
Response	
{ "error": "XXXXXX" }	



10.3.41.10 Open

The “/webadmin/rest/orders/open/” REST API function marks the given order items as “open”.

The POST request returns an error response or “OK”.

POST /webadmin/rest/orders/open/	
id	Order item id number to mark as “open” (required).
Request	
{ "id": "123" }	
Request	
{ "id": ["123", "456", "789"] }	
Response	
{ "error": "XXXXX" }	

10.3.41.11 Close

The “/webadmin/rest/orders/close/” REST API function marks the given order items as “closed”.

The POST request returns an error response or “OK”.

POST /webadmin/rest/orders/open/	
id	Order item id number to mark as “closed” (required).
Request	
{ "id": "123" }	
Request	
{ "id": ["123", "456", "789"] }	
Response	
{ "error": "XXXXX" }	

10.3.41.12 Email

A POST request will compose or send an email to the given order ids’ email address(es).

POST /webadmin/rest/orders/email/	
from	The email address to be used as “From” address; otherwise the configured “contact_form_recitipient” email address will be used.
to	The email addresses to be used as “To” addresses; otherwise the configured “contact_form_recitipient” email address will be used.



cc	The email addresses to be used as “Cc” addresses; otherwise, if no “action” parameter is given, the currently logged in user’s email address will be used.
bcc	The email addresses to be used as “Bcc” addresses; otherwise, if no “action” parameter is given, the email addresses for the given user account ids or user groups/types.
action	If blank (“”) then an email will only be composed and returned as the response without being sent. If not blank (fx. “action=send”) then an email will be sent.
id	If no “action” parameter is given, and one or more order “id” parameters are given the email addresses (if any) for those orders will be retrieved and returned as “bcc” response data attributes.
subject	Text to be used as the email “Subject”.
content	Text to be used as the email content (HTML code).
content_plaintext	Text to be used as the plain text email content.
content_id	If no “action” parameter is given, and an “content_id” parameter is given, the content item with the given id will be read and set as the email subject (title) and content.
stylesheet	Style sheet URL address to be used for email content (optional).
style	CSS code to be used for email content (optional).
send	If “send=now” parameter is given then an email will be sent. (Note: use “action=” and “send=now” parameters to compose and immediately send email).
Request	
<pre>{ "from": "", "to": "", "cc": "", "bcc": "", "action": "", "id": "", "subject": "", "content": "", "content_plaintext": "", "content_id": "", "stylesheet": "", "style": "", "send": "" }</pre>	
Response	
<pre>{ "error": "", "from": "", "to": "", "cc": "", "bcc": "", "subject": "", "content": "", </pre>	



•
•
•
•
•
•
•

```
"content_plaintext": "",  
"stylesheet": "",  
"style": ""  
}
```

10.3.41.13 Export

A list of the orders data (excluding order items data) will be returned in comma-separated values (csv) format.

GET /webadmin/rest/orders/export/csv/	
Response	
order_id, user_id, created, updated, published, paid, created_by, updated_by, published_by, status, order_quantity, order_currency, order_subtotal, tax_description, tax_total, shipping_description, shipping_total, order_total, card_type, card_number, card_issuedmonth, card_issuedyear, card_expiryyear, card_expiryyear, card_name, card_cvc, card_issue, card_postalcode, delivery_name, delivery_organisation, delivery_address, delivery_postalcode, delivery_city, delivery_state, delivery_country, delivery_phone, delivery_fax, delivery_email, delivery_website, invoice_name, invoice_organisation, invoice_address, invoice_postalcode, invoice_city, invoice_state, invoice_country, invoice_phone, invoice_fax, invoice_email, invoice_website, discount_description, discount_total, affiliate 1,5,"2013-08-20 12:08:48","2013-08-20 12:08:48",,,johnsmith,johnsmith,,,1,£,100.00,"United Kingdom VAT 17.5%",17.50,"1 x Basic Shipping",3.50,121.00,VISA,4213451248543248,,,04,2014,"Mr. John Smith",564,,, "John Smith",,"10 Melbourne Street", "E17 2SW",London,, "United Kingdom",,,admin@asbrusoft.com,,,,,,,,,,,,,0.00,	

10.3.41.14 Workflow Username Options

The “/webadmin/rest/orders/workflow_username_options/html/” REST API function returns workflow username options in HTML format for a given order id and workflow action.

GET /webadmin/rest/orders/workflow_username_options/html/?id=ID&workflow=WORKFLOW	
id	Order id number.
workflow	Workflow action id number.
Response	
HTML code	

10.3.42 Order Items

The “/webadmin/rest/orderitems/” REST API function returns all data for one or more order items and creates, updates and deletes order items.

10.3.42.1 List

If a “order_id” is given and no “id” parameter is given then a list of order items will be returned.

GET /webadmin/rest/orderitems/



order_id	Order id number (required).
Response	
<pre>[{ "id": "123", "order_id": "456", }, { "id": "789", "order_id": "456", }]</pre>	

10.3.42.2 Read

If an “id” parameter is given then a GET request will return that order item’s data. The returned data includes all the order item attributes for the order item.

GET /webadmin/rest/orderitems/?id=ID	
id	Order item id number (required).
Response	
<pre>{ "id": "1", "order_id": "1", "item_quantity": "1", "discount_description": "", "discount_description": "", "discount_total": "0.00", "discount_total_base": "", "item_subtotal": "100.00", "item_subtotal_base": "", "tax_description": "United Kingdom VAT 17.5%", "tax_description": "United Kingdom VAT 17.5%", "tax_total": "17.50", "tax_total_base": "", "shipping_description": "1 x Basic Shipping", "shipping_description": "1 x Basic Shipping", "shipping_total": "3.50", "shipping_total_base": "", "item_total": "100.00", "item_total_base": "", "product_id": "169", "product_code": "Item Id: #54PRD001", "product_currencytitle": "GBP", "product_currency": "£", "product_price": "100", "product_period": "", "product_weight": "1", "product_volume": "3", "product_width": "100", "product_height": "200", "product_depth": "300", "product_brand": "",</pre>	



```
    "product_colour": "",
    "product_size": "",
    "product_stock": "400",
    "product_comment": "In Stock",
    "product_status": "",
    "product_email": "",
    "product_url": "",
    "product_delivery": "",
    "product_user": "",
    "product_page": "",
    "product_program": "",
    "product_hosting": "",
    "product_options": "",
    "product_content": "",
    "product_contentgroup": "",
    "product_contenttype": "",
    "product_imagegroup": "",
    "product_imagetype": "",
    "product_filegroup": "",
    "product_filetype": "",
    "product_linkgroup": "",
    "product_linktype": "",
    "product_productgroup": "",
    "product_producttype": "",
    "product_usergroup": "",
    "product_usertype": "",
    "version": "",
    "device": "",
    "usersegment": "",
    "usertest": "",
    "title": "Product 1",
    "author": "",
    "keywords": "",
    "description": "Product 1 #54PRD001.",
    "summary": "Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
    "server_filename": "",
    "image1": "422",
    "image2": "414",
    "image3": "415",
    "file1": "",
    "file2": "",
    "file3": "",
    "link1": "",
    "link2": "",
    "link3": ""
}
```

10.3.42.3 Create

A POST request will create a new order item with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the order item with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created order item’s data. The returned data includes all the order item attributes for the order item.

POST /webadmin/rest/orderitems/	
id	Order item id number to be copied (optional).
.....	All or any number and combination of order item



	attributes as used by the web content management system (see the GET request example response above).
Request	
{ "order_id": "456", "id": "123", "title": "XYZ" }	
Response	
{ "id": "789", "order_id": "456", "title": "XYZ", }	

10.3.42.4 Update

A PUT request will update the existing order item with the given “id” number with the given data attributes.

The PUT request will return the updated order item’s data. The returned data includes all the order item attributes for the order item.

PUT /webadmin/rest/orderitems/	
id	Order item id number to be updated (required).
.....	All or any number and combination of order item attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "123", "order_id": "456", "title": "XYZ", }	

10.3.42.5 Delete

A DELETE request will delete the existing order item with the given “id” number.

The DELETE request will return the deleted order item data. The returned data includes all the order item attributes for the project task.

DELETE /webadmin/rest/orderitems/	
id	Order item id number (required).



Request
<pre>{ "id": "123" }</pre>
Response
<pre>{ "id": "123", "order_id": "456", }</pre>

10.3.42.6 Export

A list of the order items data will be returned in comma-separated values (csv) format.

GET /webadmin/rest/orderitems/export/csv/	
Response	
<pre>order_id, user_id, created, updated, published, paid, created_by, updated_by, published_by, status, order_quantity, order_currency, order_subtotal, tax_description, tax_total, shipping_description, shipping_total, order_total, card_type, card_number, card_issuedmonth, card_issuedyear, card_expirymonth, card_expiryyear, card_name, card_cvc, card_issue, card_postalcode, delivery_name, delivery_organisation, delivery_address, delivery_postalcode, delivery_city, delivery_state, delivery_country, delivery_phone, delivery_fax, delivery_email, delivery_website, invoice_name, invoice_organisation, invoice_address, invoice_postalcode, invoice_city, invoice_state, invoice_country, invoice_phone, invoice_fax, invoice_email, invoice_website, orderitem_id, product_id, version, title, summary, image1, image2, image3, file1, file2, file3, link1, link2, link3, author, keywords, description, product_code, product_currency, product_price, product_period, product_stock, product_comment, product_email, product_url, product_delivery, product_user, product_page, product_program, product_hosting, product_brand, product_colour, product_size, product_options, item_quantity, item_subtotal, item_total, item_shipping_total, item_shipping_description, item_tax_total, item_tax_description, item_discount_total, item_discount_description, discount_total, discount_description, affiliate 1,5,"2013-08-20 12:08:48","2013-08-20 12:08:48",,,johnsmith,johnsmith,,,1,&#163;,100.00,"United Kingdom VAT 17.5%",17.50,"1 x Basic Shipping",3.50,121.00,VISA,4213451248543248,,,04,2014,"Mr. John Smith",564,,,,"John Smith",,"10 Melbourne Street",,"E17 2SW",London,,"United Kingdom",,,admin@asbrusoft.com,,,,,,,,,,,,,1,169,,"Product 1",,"Lorem ipsum dolor sit amet, consectetur adipiscing elit.",422,414,415,,,,,,,,,"Product 1 #54PRD001.",,"Item Id: #54PRD001",,&#163;,100,,400,"In Stock",,,,,,,,,,1,100.00,100.00,3.50,"1 x Basic Shipping",17.50,"United Kingdom VAT 17.5%",0.00,,0.00,,</pre>	

10.3.43 Sales

The “/webadmin/rest/sales/REPORT/” REST API functions return all data for various sales reports.

10.3.43.1 Sales Reports

GET /webadmin/rest/sales/REPORT/	
summary	Summary
websites	What / Websites/Domains



products	What / Products ?id=ID for single product with given id. ?contentgroup=PRODUCTGROUP for all products of given product group. ?contenttype=PRODUCTTYPE for all products of a given product type.
productgroups	What / Products / Groups
producttypes	What / Products / Types
daily	When / Daily
weekly	When / Weekly
monthly	When / Monthly
yearly	When / Yearly
hours	When / Hours
weekdays	When / Weekdays
days	When / Days (of month)
weeks	When / Weeks
months	When / Months
countries	Who / Countries
operatingsystems	Who / Operating Systems
webbrowsers	Who / Web Browsers
devices	Who / Devices
users	Who / Users ?usergroup=USERGROUP for all orders of given user group. ?usertype=USERTYPE for all orders of given user type.
usergroups	Who / Users / Groups
usertypes	Who / Users / Types
affiliates	Why / Affiliates
referers	Why / Referrers
searchengines	Why / Search Engines
searchqueries	Why / Search Queries
searchwords	Why / Search Words
entry	How / Entry

10.3.43.2 Request Parameters

Most of the sales reports support the same request parameters to specify which sales data to report.

GET /webadmin/rest/sales/REPORT/?period=PERIOD&limit=LIMIT¤cy=CURRENCY &details=DETAILS	
GET /webadmin/rest/sales/REPORT/?period start=PERIODSTART&period end=PERIODEND& limit=LIMIT¤cy=CURRENCY&details=DETAILS	
period	The period for which to report sales data where period is "now" (default) (30 minutes) / "today" / "last24hours" / "yesterday" / "thisweek" / "last7days" / "lastweek" / "last14days" /



	“thismonth” / “last30days” / “lastmonth” / “thisquarter” / “last3months” / “lastquarter” / “thishalfyear” / “last6months” / “lasthalfyear” / “this year” / “last12months” / “last year” / “all”.
period_start	The start of the period which to report sales data in “YYYY-MM-DD hh:mm:ss” format.
period_end	The end of the period which to report sales data in “YYYY-MM-DD hh:mm:ss” format.
limit	Max number of sales data to report. Only report the top LIMIT number of sales data.
currency	The order currency id number to report sales data for. “*” for all currencies.
details	Optional additional details to report sales data for. "+countusersegments" for sales data per user segment. "+countusertests" for sales data per user test. "+countusersegments+countusertests" for sales data per user segment and user test.

10.3.43.1 Report Data

Most of the sales reports report the same types of data for the given requested data.

countcustomers	Number of customers.
countorders	Number of orders.
countorderitems	Number of order items.
sumitemquantity	Quantity of ordered items.
sumitemtotal	Sum of ordered items.
maxcustomers	Max reported number of customers.
maxorders	Max reported number of orders.
maxorderitems	Max reported number of order items.
maxitemquantity	Max reported quantity of ordered items.
maxitemtotal	Max reported sum of ordered items.
totalcustomers	Total reported number of customers.
totalorders	Total reported number of orders.
totalorderitems	Total reported number of order items.
totalitemquantity	Total reported quantity of ordered items.
totalitemtotal	Total reported sum of ordered items.

10.3.44 Databases

The “/webadmin/rest/databases/” REST API function returns all (configuration) data for one or more content databases and creates, updates and deletes content databases.

10.3.44.1 List

If no “id” parameter is given then a list of content databases will be returned.

GET /webadmin/rest/databases/



Response	
<pre>[{ "id": "123", "title": "ABC", }, { "id": "789", "title": "XYZ", }]</pre>	

10.3.44.2 Read

If an “id” parameter is given then a GET request will return that content database’s (configuration) data. The returned data includes all the content database attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the content database.

GET /webadmin/rest/databases/?id=ID	
id	Content database id (required).
Response	
<pre>{ "id": "1", "title": "Job Directory", "content": "1 1 Name text 80 1\r\n2 2 Summary html 800 600\r\n4 4 Parent datum Job Directory Name\r\n5 5 Created By createdby \r\n6 6 Created Date created \r\n8 8 Sub-Category text 80 1", "searchresults": "", "searchresult": "", "viewpage": "", "adminpage": "", "adminindex": "", "users_group": "", "users_type": "", "users_users": "", "creators_group": "Human Resources", "creators_type": "", "creators_users": "", "editors_group": "Human Resources", "editors_type": "", "editors_users": "", "publishers_group": "Human Resources", "publishers_type": "", "publishers_users": "", "administrators_group": "Website Administrators", "administrators_type": "", "administrators_users": "", "titleid": "1", "titlename": "Name", "titlecolumn": "coll",</pre>	



```
"indexcolumns": "",
"columns": [
  {
    "id": "1",
    "order": "1",
    "name": "Name",
    "index": "",
    "type": "text",
    "param1": "80",
    "param2": "1",
    "options": "",
    "description": "",
    "cols": "80",
    "rows": "1",
    "width": "80",
    "height": "1",
    "digits": "80",
    "decimals": "1",
    "contentclass": "1",
    "size": "80",
    "size2": "1",
    "databasename": "1",
    "databasecontent": "",
    "column": "col1"
  },
  {
    "id": "2",
    "order": "2",
    "name": "Summary",
    "index": "",
    "type": "html",
    "param1": "800",
    "param2": "600",
    "options": "",
    "description": "",
    "cols": "800",
    "rows": "600",
    "width": "800",
    "height": "600",
    "digits": "800",
    "decimals": "600",
    "contentclass": "600",
    "size": "800",
    "size2": "600",
    "databasename": "600",
    "databasecontent": "",
    "column": "col2"
  },
  {
    "id": "4",
    "order": "4",
    "name": "Parent",
    "index": "",
    "type": "datum",
    "param1": "",
    "param2": "Job Directory",
    "options": "Name",
    "description": "",
    "cols": "",
    "rows": "Job Directory",
    "width": "",
    "height": "Job Directory",
    "digits": "",
    "decimals": "Job Directory",
    "contentclass": "Job Directory",
```



```
        "size": "",
        "size2": "Job Directory",
        "databasename": "Job Directory",
        "databasecontent": "Name",
        "column": "col4"
    },
    {
        "id": "5",
        "order": "5",
        "name": "Created By",
        "index": "",
        "type": "createdby",
        "param1": "",
        "param2": "",
        "options": "",
        "description": "",
        "cols": "",
        "rows": "",
        "width": "",
        "height": "",
        "digits": "",
        "decimals": "",
        "contentclass": "",
        "size": "",
        "size2": "",
        "databasename": "",
        "databasecontent": "",
        "column": "col5"
    },
    {
        "id": "6",
        "order": "6",
        "name": "Created Date",
        "index": "",
        "type": "created",
        "param1": "",
        "param2": "",
        "options": "",
        "description": "",
        "cols": "",
        "rows": "",
        "width": "",
        "height": "",
        "digits": "",
        "decimals": "",
        "contentclass": "",
        "size": "",
        "size2": "",
        "databasename": "",
        "databasecontent": "",
        "column": "col6"
    },
    {
        "id": "8",
        "order": "8",
        "name": "Sub-Category",
        "index": "",
        "type": "text",
        "param1": "80",
        "param2": "1",
        "options": "",
        "description": "",
        "cols": "80",
        "rows": "1",
        "width": "80",
```



```
        "height": "1",
        "digits": "80",
        "decimals": "1",
        "contentclass": "1",
        "size": "80",
        "size2": "1",
        "databasename": "1",
        "databasecontent": "",
        "column": "col8"
    }
],
"namedcolumns": {
    "Parent": {
        "id": "4",
        "order": "4",
        "name": "Parent",
        "index": "",
        "type": "datum",
        "param1": "",
        "param2": "Job Directory",
        "options": "Name",
        "description": "",
        "cols": "",
        "rows": "Job Directory",
        "width": "",
        "height": "Job Directory",
        "digits": "",
        "decimals": "Job Directory",
        "contentclass": "Job Directory",
        "size": "",
        "size2": "Job Directory",
        "databasename": "Job Directory",
        "databasecontent": "Name",
        "column": "col4"
    },
    "Created Date": {
        "id": "6",
        "order": "6",
        "name": "Created Date",
        "index": "",
        "type": "created",
        "param1": "",
        "param2": "",
        "options": "",
        "description": "",
        "cols": "",
        "rows": "",
        "width": "",
        "height": "",
        "digits": "",
        "decimals": "",
        "contentclass": "",
        "size": "",
        "size2": "",
        "databasename": "",
        "databasecontent": "",
        "column": "col6"
    },
    "Summary": {
        "id": "2",
        "order": "2",
        "name": "Summary",
        "index": "",
        "type": "html",
        "param1": "800",
```



```
        "param2": "600",
        "options": "",
        "description": "",
        "cols": "800",
        "rows": "600",
        "width": "800",
        "height": "600",
        "digits": "800",
        "decimals": "600",
        "contentclass": "600",
        "size": "800",
        "size2": "600",
        "databasename": "600",
        "databasecontent": "",
        "column": "col2"
    },
    "Created By": {
        "id": "5",
        "order": "5",
        "name": "Created By",
        "index": "",
        "type": "createdby",
        "param1": "",
        "param2": "",
        "options": "",
        "description": "",
        "cols": "",
        "rows": "",
        "width": "",
        "height": "",
        "digits": "",
        "decimals": "",
        "contentclass": "",
        "size": "",
        "size2": "",
        "databasename": "",
        "databasecontent": "",
        "column": "col5"
    },
    "Sub-Category": {
        "id": "8",
        "order": "8",
        "name": "Sub-Category",
        "index": "",
        "type": "text",
        "param1": "80",
        "param2": "1",
        "options": "",
        "description": "",
        "cols": "80",
        "rows": "1",
        "width": "80",
        "height": "1",
        "digits": "80",
        "decimals": "1",
        "contentclass": "1",
        "size": "80",
        "size2": "1",
        "databasename": "1",
        "databasecontent": "",
        "column": "col8"
    },
    "Name": {
        "id": "1",
        "order": "1",
```



```
        "name": "Name",
        "index": "",
        "type": "text",
        "param1": "80",
        "param2": "1",
        "options": "",
        "description": "",
        "cols": "80",
        "rows": "1",
        "width": "80",
        "height": "1",
        "digits": "80",
        "decimals": "1",
        "contentclass": "1",
        "size": "80",
        "size2": "1",
        "databasename": "1",
        "databasecontent": "",
        "column": "coll"
    },
    "user": true,
    "creator": true,
    "editor": true,
    "publisher": true,
    "administrator": true
}
```

10.3.44.3 Create
A POST request will create a new content database with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the content database with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created content database’s (configuration) data. The returned data includes all the content database attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the content database.

POST /webadmin/rest/databases/	
id	Content database id number to be copied (optional).
.....	All or any number and combination of content database attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "789", "title": "XYZ",	



```
{
  "user": true,
  "creator": true,
  "editor": true,
  "publisher": true,
  "administrator": true
}
```

10.3.44.4 Update

A PUT request will update the existing content database with the given “id” number with the given data attributes.

The PUT request will return the updated content database’s (configuration) data. The returned data includes all the content database attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the content database.

PUT /webadmin/rest/databases/	
id	Content database id number to be updated (required).
.....	All or any number and combination of content database attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "title": "XYZ" }	
Response	
{ "id": "123", "title": "XYZ",, "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }	

10.3.44.5 Delete

A DELETE request will delete the existing content database with the given “id” number.

The DELETE request will return the deleted content database’s (configuration) data. The returned data includes all the content database attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the content database.

DELETE /webadmin/rest/databases/	
id	Content database id number (required).
Request	
{ "id": "123" }	



}
Response
<pre>{ "id": "123", "title": "ABC", "user": true, "creator": true, "editor": true, "publisher": true, "administrator": true }</pre>

10.3.44.6 Exists

If a “title” parameter is given then a GET request will return “YES” or “NO” depending on if a content database with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a content database with the given name and another id than the given id exists.

GET /webadmin/rest/databases/exists/?title=TITLE	
title	Content database name (required).
id	Content database id number (optional).
Response	
<pre>{ "exists": "YES" }</pre>	

10.3.45 Usage Analytics

The “/webadmin/rest/usage/REPORT/” REST API functions return all data for various usage analytics reports.

10.3.45.1 Usage Reports

GET /webadmin/rest/usage/	
	General usage statistics information (count, oldest detailed, oldest summarised, newest summarised)

GET /webadmin/rest/usage/REPORT/	
summary	Summary
websites	What / Websites/Domains
contents	What / Website Content
contentitem	What / Website Content / Content Item
pages	What / Website Content / Pages ?contentgroup=CONTENTGROUP ?contenttype=CONTENTTYPE
pagegroups	What / Website Content / Pages / Groups
pagetypes	What / Website Content / Pages / Types
contacts	What / Website Content / Contacts
posts	What / Website Content / Posts



logins	What / Website Content / Logins
logouts	What / Website Content / Logouts
register	What / Website Content / Register
unregister	What / Website Content / Unregister
stylesheets	What / Website Content / Style Sheets
scripts	What / Website Content / Scripts
library	What / Media Library
images	What / Media Library / Images ?contentgroup=IMAGEGROUP ?contenttype=IMAGETYPE
imagegroups	What / Media Library / Images / Groups
imagetypes	What / Media Library / Images / Types
files	What / Media Library / Files ?contentgroup=FILEGROUP ?contenttype=FILETYPE
filegroups	What / Media Library / Files / Groups
filetypes	What / Media Library / Files / Types
links	What / Media Library / Links ?contentgroup=LINKGROUP ?contenttype=LINKTYPE
linkgroups	What / Media Library / Links / Groups
linktypes	What / Media Library / Links / Types
products	What / Ecommerce / Products ?contentgroup=PRODUCTGROUP ?contenttype=PRODUCTTYPE
productgroups	What / Ecommerce / Products / Groups
producttypes	What / Ecommerce / Products / Types
shopcart	What / Ecommerce / Shopcart Add ?contentgroup=PRODUCTGROUP ?contenttype=PRODUCTTYPE
shopcartgroups	What / Ecommerce / Shopcart Add / Groups
shopcarttypes	What / Ecommerce / Shopcart Add / Types
shopcart.checkout	What / Ecommerce / Shopcart Checkout ?contentgroup=PRODUCTGROUP ?contenttype=PRODUCTTYPE
shopcartgroups.checkout	What / Ecommerce / Shopcart Checkout / Groups
shopcarttypes.checkout	What / Ecommerce / Shopcart Checkout / Types
shopcart.confirm	What / Ecommerce / Shopcart Confirm ?contentgroup=PRODUCTGROUP ?contenttype=PRODUCTTYPE
shopcartgroups.confirm	What / Ecommerce / Shopcart Confirm / Groups
shopcarttypes.confirm	What / Ecommerce / Shopcart Confirm / Types
shopcart.complete	What / Ecommerce / Shopcart Complete ?contentgroup=PRODUCTGROUP ?contenttype=PRODUCTTYPE
shopcartgroups.complete	What / Ecommerce / Shopcart Complete / Groups
shopcarttypes.complete	What / Ecommerce / Shopcart Complete / Types
shopcart.conversion	What / Ecommerce / Shopcart Conversion ?contentgroup=PRODUCTGROUP ?contenttype=PRODUCTTYPE



databases	What / Content Databases
database	What / Content Databases / Database ?database=DATABASE
daily	When / Daily
weekly	When / Weekly
monthly	When / Monthly
yearly	When / Yearly
hours	When / Hours
weekdays	When / Weekdays
days	When / Days (of month)
weeks	When / Weeks
months	When / Months
countries	Who / Countries
clienthosts	Who / Hosts
clienthost	Who / Hosts / Host
visitors	Who / Visitors
robots	Who / Robots
operatingsystems	Who / Operating Systems
webbrowsers	Who / Web Browsers
devices	Who / Devices
users	Who / Users ?usergroup=USERGROUP ?usertype=USERTYPE
usergroups	Who / Users / Groups
usertypes	Who / Users / Types
usersegments	Who / Users / Segments
usertests	Who / Users / Tests
referrers	Why / Referers
searchengines	Why / Search Engines
searchqueries	Why / Search Queries
searchwords	Why / Search Words
entry	How / Entry (page)
paths	How / Paths (pages)
exit	How / Exit (page)
duration	How / Duration
visits	How / Visits
visit	How / Visits / Visit
username	How / Visits / Username
visitor	How / Visits / Visitor

10.3.45.2 Request Parameters

Most of the usage reports support the same request parameters to specify which usage analytics data to report.

GET /webadmin/rest/usage/REPORT/?period=PERIOD&limit=LIMIT&details=DETAILS
GET /webadmin/rest/usage/REPORT/?period_start=PERIODSTART&period_end=PERIODEND &limit=LIMIT&details=DETAILS



period	The period for which to report usage data where period is “now” (default) (30 minutes) / ”today” / “last24hours” / “yesterday” / “thisweek” / “last7days” / “lastweek” / “last14days” / “thismonth” / “last30days” / “lastmonth” / “thisquarter” / “last3months” / “lastquarter” / “thishalfyear” / “last6months” / “lasthalfyear” / “this year” / “last12months” / “last year” / “all”.
period_start	The start of the period which to report usage data in “YYYY-MM-DD hh:mm:ss” format.
period_end	The end of the period which to report usage data in “YYYY-MM-DD hh:mm:ss” format.
limit	Max number of usage data to report. Only report the top LIMIT number of usage data.
details	Optional additional details to report usage data for. "+countusersegments" for usage data per user segment. "+countusertests" for usage data per user test. "+countusersegments+countusertests" for usage data per user segment and user test.

10.3.45.3 Report Data

Most of the usage reports report the same types of data for the given requested data.

countclienthosts	Number of unique website visitor Internet address domain names or IP-numbers.
countvisitors	Number of unique website visitors as identified by the web content management system using “cookies”.
countvisits	Number of separate visits as identified by the web content management system using “session ids”.
countpages	Number of requested web pages.
counthits	Number of requested web pages, images, files, style sheets and scripts.
maxclienthosts	Max reported number of unique website visitor Internet address domain names or IP-numbers.
maxvisitors	Max reported number of unique website visitors as identified by the web content management system using “cookies”.
maxvisits	Max reported number of separate visits as identified by the web content management system using “session ids”.
maxpages	Max reported number of requested web pages.
maxhits	Max reported number of requested web pages, images, files, style sheets and scripts.
totalclienthosts	Total reported number of unique website



	visitor Internet address domain names or IP-numbers.
totalvisitors	Total reported number of unique website visitors as identified by the web content management system using “cookies”.
totalvisits	Total reported number of separate visits as identified by the web content management system using “session ids”.
totalpages	Total reported number of requested web pages.
totalhits	Total reported number of requested web pages, images, files, style sheets and scripts.

10.3.45.4 Summarise

A GET request will summarise usagelog data, optionally, for testing and review, or committed to reduce storage and processing of large amounts of usagelog data details.

GET /webadmin/rest/usage/summarise/html/?test=*&summarised_period=PERIOD&summarised_details=DETAILS	
GET /webadmin/rest/usage/summarise/html/?commit=*&summarised_period=PERIOD&summarised_details=DETAILS	
GET /webadmin/rest/usage/summarise/html/?all=*&summarised_period=PERIOD&summarised_details=DETAILS	
GET /webadmin/rest/usage/summarise/html/?force=*&commit=*&summarised_period=PERIOD&summarised_details=DETAILS	
GET /webadmin/rest/usage/summarise/html/?force=*&all=*&summarised_period=PERIOD&summarised_details=DETAILS	
summarised_period	“hourly” (default), “daily” or “monthly” (optional).
summarised_details	“pageviews,hits” (default), “visitors,pageviews,hits”, “visits,pageviews,hits” or “visitors,visits,pageviews,hits” (optional).
test	“*” – If no “test” and no “commit” and no “all” parameter is given, the usagelog data will not be summarised (“test,” “commit” or “all” required).
commit	“*” - If no “commit” parameter is given, the summarised usagelog data will not be saved and the summarised usagelog data details will not be deleted – the summarised usagelog data will only be returned in the response for testing and review (“test,” “commit” or “all” required).
all	“*” - If an “all=*” parameter is given, the



	usagelog data will be summarised for all summarised periods. If no “all” parameter is given, the usagelog data will only be summarised for the first summarised period; and the summarise functionality will have to be requested multiple times – once for each period to be summarised (to avoid application server timeouts for long-running processing to summarise the usagelog data) (“test,” “commit” or “all” required).
force	If not blank (“”), the usagelog data will be summarised even if the summarise functionality has been started (without ending) previously (optional).
Response	
HTML code	

10.3.46 Usagelog Configuration Options

The “/webadmin/rest/usagelog/” REST API functions returns usage log options available for the configuration of user experience management configuration etc.

10.3.46.1 List – Web Browsers

GET /webadmin/rest/clientbrowser/	
Response	
<pre>[{ "id": "Safari", "name": "Apple Safari" }, { "id": "Chrome", "name": "Google Chrome" }, { "id": "MSIE", "name": "Microsoft Internet Explorer" }, ]</pre>	

10.3.46.2 List – Web Browser Devices

GET /webadmin/rest/clientdevice/	
Response	
<pre>[{ "id": "AndroidPhone" }, { "id": "AndroidTablet" }, { "id": "BlackBerry" }, { "id": "iPad" }, { "id": "iPhone" }, { "id": "iPod" }, { "id": "JavaPhone" }, { "id": "MicrosoftMobile" }, { "id": "MicrosoftPhone" }, { "id": "MicrosoftTablet" }]</pre>	

10.3.46.3 List – Web Browser Device Types

GET /webadmin/rest/clientdevicetype/



Response	
<pre>[{ "id": "Device" }, { "id": "Phone" }, { "id": "Tablet" }]</pre>	

10.3.46.4 List – Web Browser Device Versions

GET /webadmin/rest/clientdeviceversion/	
Response	
<pre>[{ "id": "iPad", "name": "Apple iPad" }, { "id": "iPhone", "name": "Apple iPhone" }, { "id": "iPod", "name": "Apple iPod" }, { "id": "BlackBerry", "name": "BlackBerry Phone" }, { "id": "AndroidPhone", "name": "Google Android Phone" }, { "id": "AndroidTablet", "name": "Google Android Tablet" }, { "id": "JavaPhone", "name": "Java Mobile (J2ME)" }, { "id": "JavaPhone", "name": "Java Mobile (JavaFX)" }, { "id": "JavaPhone", "name": "Java Mobile (JME)" }, { "id": "MicrosoftMobile", "name": "Microsoft Windows Mobile" }, { "id": "MicrosoftPhone", "name": "Microsoft Windows Phone" }, { "id": "MicrosoftTablet", "name": "Microsoft Windows Tablet" }]</pre>	

10.3.46.5 List – Web Browser Operating Systems

GET /webadmin/rest/clientos/	
Response	
<pre>[{ "id": "Android", "name": "Android" }, { "id": "Android", "name": "Google Android" }, { "id": "iOS", "name": "iOS" }, { "id": "iOS", "name": "Apple iOS" }, { "id": "Mac OS", "name": "Mac OS" }, { "id": "Mac OS X", "name": "Mac OS X" }, { "id": "Mac OS", "name": "Apple Mac OS" }, { "id": "Mac OS X", "name": "Apple Mac OS X" }, { "id": "Windows", "name": "Windows" }, { "id": "Windows 7", "name": "Windows 7" }, { "id": "Windows 8", "name": "Windows 8" }, { "id": "Windows 8.1", "name": "Windows 8.1" }, { "id": "Windows 10", "name": "Windows 10" }, ]</pre>	

10.3.46.6 List – Search Engines

GET /webadmin/rest/searchengines/	
Response	
<pre>[{ "id": "1klik", "name": "1Klik.dk (1klik.dk)" }, { "id": "3721", "name": "3721 (www.3721.com)" },]</pre>	



.....

```
{ "id": "google4counter", "name": "4-counter (Google) (www.4-counter.com)"
},
.....
]
```

10.3.47 Hosting Clients

The “/webadmin/rest/hosting/” REST API function returns all data for one or more hosting clients and creates, updates and deletes hosting clients.

10.3.47.1 List

If no “id” parameter is given then a list of hosting clients will be returned.

GET /webadmin/rest/hosting/	
Response	
<pre>[{ "id": "123", "user_id": "0", "clientaddress": "www.abc.com", } , , { "id": "789", "user_id": "0", "clientaddress": "www.xyz.com", }]</pre>	

10.3.47.1.1 Livegrid (DEPRECATED)

This duplicates the general list functionality as described above but with output in Rico Livegrid XML format as used by current and older versions of the web content management system administration pages.

Note: DEPRECATED – use GET /webadmin/rest/hosting/ (Please see section 10.3.47.1 List).

GET /webadmin/rest/hosting/livegrid/	
Response	
XML code	

10.3.47.1.2 Delete Confirmation

If a “index=delete” parameter is given then a list of hosting clients which match the “id” parameters will be returned.

GET /webadmin/rest/hosting/?index=delete&id=ID&id=ID&id=ID	
index	“delete”.
id	Hosting client id.



Response
<pre>[{ "id": "123", "user_id": "0", "clientaddress": "www.abc.com", }, { "id": "789", "user_id": "0", "clientaddress": "www.xyz.com", }]</pre>

10.3.47.2 Read

If an “id” parameter is given then a GET request will return that hosting client’s data. The returned data includes all the hosting client attributes for the hosting client.

GET /webadmin/rest/hosting/?id=ID	
id	Hosting client id number (required).
Response	
<pre>{ "id": "1", "user_id": "0", "client_address": "www.abc.com", "client_URLrootpath": "\/SITEPERS\/", "client_database": "mysql:mysql:\\\\/username:password@localhost\\/foocom", "personal_license": "HOSTINGTRIAL:personal:8T4EXQERBGX9A7C4", "professional_license": "", "enterprise_license": "", "hosting_license": "", "ecommerce_license": "", "community_license": "", "databases_license": "", "statistics_license": "", "experience_license": "", "collaboration_license": "", "superadmin": "admin", "superadmin_password": "", "superadmin_email": "", "hostinggroup": "", "hostingtype": "", "scheduled_publish": "", "scheduled_notify": "", "scheduled_unpublish": "", "scheduled_publish_email": "", "scheduled_notify_email": "", "scheduled_unpublish_email": "", "scheduled_last": "", "client": "", "scheduled_last": "", "status": "", "client_config_personal_license": "HOSTINGTRIAL:personal:8T4EXQERBGX9A7C4", "client_config_professional_license": "", "client_config_enterprise_license": "", "client_config_hosting_license": "" }</pre>	



```
}
  "client_config_ecommerce_license": "",
  "client_config_community_license": "",
  "client_config_databases_license": "",
  "client_config_statistics_license": "",
  "client_config_experience_license": "",
  "client_config_collaboration_license": "",
  "client_config_superadmin": "admin",
  "client_config_superadmin_password": "",
  "client_config_superadmin_email": ""
}
```

10.3.47.3 Create

A POST request will create a new hosting client with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the hosting client with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created hosting client’s data. The returned data includes all the hosting client attributes for the hosting client.

POST /webadmin/rest/hosting/	
id	Hosting client id number to be copied (optional).
.....	All or any number and combination of hosting client attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "clientaddress": "www.xyz.com" }	
Response	
{ "id": "789", "clientaddress": "www.xyz.com", }	

10.3.47.4 Update

A PUT request will update the existing hosting client with the given “id” number with the given data attributes.

The PUT request will return the updated hosting client’s data. The returned data includes all the hosting client attributes for the hosting client.

PUT /webadmin/rest/hosting/	
id	Hosting client id number to be updated (required).
.....	All or any number and combination of hosting client attributes as used by the web content management system (see the GET request example response above).



Request
<pre>{ "id": "123", "personal_license": "www.xyz.com:personal:3TH6XBEUJGX4A9C2" }</pre>
Response
<pre>{ "id": "123", "clientaddress": "www.xyz.com", "personal_license": "www.xyz.com:personal:3TH6XBEUJGX4A9C2" }</pre>

10.3.47.5 Delete

A DELETE request will delete the existing hosting client with the given “id” number(s).

The DELETE request will return the deleted hosting client’s data for a single given id. The returned data includes all the hosting client attributes for the hosting client.

The DELETE request will simply return an error message or “OK” for multiple given ids.

DELETE /webadmin/rest/hosting/	
id	Hosting client id number (required).
Request	
<pre>{ "id": "123" }</pre>	
Response	
<pre>{ "id": "123", "clientaddress": "www.abc.com", }</pre>	

DELETE /webadmin/rest/hosting/	
id	Hosting client id numbers (required).
Request	
<pre>{ "id": "123", "id": "456", "id": "789" }</pre>	
Response	
<pre>{ "error": "OK" }</pre>	

10.3.47.6 Delete (DEPRECATED)

The “/webadmin/rest/hosting/delete/” REST API function deletes the given hosting clients.

The POST request returns an error response or “OK”.



Note: DEPRECATED – use DELETE /webadmin/rest/hosting/ (Please see section 10.3.47.5 Delete).

POST /webadmin/rest/hosting/delete/	
id	Hosting client id number to be deleted (required).
Request	
{ "id": "123" }	
Request	
{ "id": ["123", "456", "789"] }	
Response	
{ "error": "XXXXX" }	

10.3.47.7 Move

The “/webadmin/rest/hosting/move/” REST API function moves the given hosting clients.

The POST request returns an error response or “OK”.

POST /webadmin/rest/hosting/move/	
id	Hosting client id number to be moved (required).
hostinggroup	Hosting group name for moved hosting clients (optional).
hostingtype	Hosting type name for moved hosting clients (optional).
Request	
{ "id": "123", "hostinggroup": "Customers", "hostingtype": "Production" }	
Request	
{ "id": ["123", "456", "789"], "hostinggroup": "Customers", "hostingtype": "Production" }	
Response	
{ "error": "XXXXX" }	

10.3.48 Hosting Licenses

The “/webadmin/rest/license/” REST API function generates web content management system software license keys for website hosting clients (for the Hosting Edition).



The POST request returns an error response or “OK”.

GET /webadmin/rest/license/	
owner	Hosting client id email address or domain name (required).
product	“personal”, “professional”, “enterprise”, “hosting”, “community”, “databases”, “ecommerce”, “statistics”, “experience”, “collaboration” (required).
Response	
{ "key": "....." }	

10.3.49 Hosting Groups

The “/webadmin/rest/hostinggroups/” REST API function returns all data for one or more hosting groups and creates, updates and deletes hosting groups.

10.3.49.1 List

If no “id” parameter is given then a list of hosting groups will be returned.

GET /webadmin/rest/hostinggroups/	
Response	
{ { "id": "123", "hostinggroup": "ABC" } , , { "id": "789", "hostinggroup": "XYZ" } }	

10.3.49.2 Read

If an “id” parameter is given then a GET request will return that hosting group’s data. The returned data includes all the hosting group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the hosting group.

GET /webadmin/rest/hostinggroups/?id=ID	
id	Hosting group id number (required).
Response	
{ "id": "1", "hostinggroup": "Enterprise", }	



```
    "users_group": "",
    "users_type": "",
    "users_users": "",
    "creators_group": "",
    "creators_type": "",
    "creators_users": "",
    "editors_group": "",
    "editors_type": "",
    "editors_users": "",
    "publishers_group": "",
    "publishers_type": "",
    "publishers_users": "",
    "administrators_group": "",
    "administrators_type": "",
    "administrators_users": "",

    "user": true,
    "creator": true,
    "developer": true,
    "editor": true,
    "publisher": true,
    "administrator": true
}
```

10.3.49.3 Create

A POST request will create a new hosting group with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the hosting group with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created hosting group’s data. The returned data includes all the hosting group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the hosting group.

POST /webadmin/rest/hostinggroups/	
id	Hosting group id number to be copied (optional).
.....	All or any number and combination of hosting group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "hostinggroup": "XYZ" }	
Response	
{ "id": "789", "hostinggroup": "XYZ", "user": true, "creator": true, "developer": true, "editor": true,	



```
"publisher": true,  
"administrator": true  
}
```

10.3.49.4 Update

A PUT request will update the existing hosting group with the given “id” number with the given data attributes.

The PUT request will return the updated hosting group’s data. The returned data includes all the hosting group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the hosting group.

PUT /webadmin/rest/hostinggroups/	
id	Hosting group id number to be updated (required).
.....	All or any number and combination of hosting group attributes as used by the web content management system (see the GET request example response above).
Request	
{ "id": "123", "administrators_group": "XYZ" }	
Response	
{ "id": "123", "hostinggroup": "ABC",, "administrators_group": "Hosting Administrators",, "user": true, "creator": true, "developer": true, "editor": true, "publisher": true, "administrator": true }	

10.3.49.5 Delete

A DELETE request will delete the existing hosting group with the given “id” number.

The DELETE request will return the deleted hosting group’s data. The returned data includes all the hosting group attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the hosting group.

DELETE /webadmin/rest/hostinggroups/	
id	Hosting group id number (required).
Request	
{ "id": "123" }	



}
Response
{ "id": "123", "hostinggroup": "ABC", "user": true, "creator": true, "developer": true, "editor": true, "publisher": true, "administrator": true }

10.3.49.6 Exists

If a “hostinggroup” parameter is given then a GET request will return “YES” or “NO” depending on if a hosting group with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a hosting group with the given name and another id than the given id exists.

GET /webadmin/rest/hostinggroups/exists/?hostinggroup=FILEGROUP	
hostinggroup	Hosting group name (required).
id	Hosting group id number (optional).
Response	
{ "exists": "YES" }	

10.3.50 Hosting Types

The “/webadmin/rest/hostingtypes/” REST API function returns all data for one or more hosting types and creates, updates and deletes hosting types.

10.3.50.1 List

If no “id” parameter is given then a list of hosting types will be returned.

GET /webadmin/rest/hostingtypes/	
Response	
[{ "id": "123", "hostingtype": "ABC" }, { "id": "789", "hostingtype": "XYZ" }]	



10.3.50.2 Read

If an “id” parameter is given then a GET request will return that hosting type’s data. The returned data includes all the hosting type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the hosting type.

GET /webadmin/rest/hostingtypes/?id=ID	
id	Hosting type id number (required).
Response	
<pre>{ "id": "1", "hostingtype": "Trial Users", "users_group": "", "users_type": "", "users_users": "", "creators_group": "", "creators_type": "", "creators_users": "", "editors_group": "", "editors_type": "", "editors_users": "", "publishers_group": "", "publishers_type": "", "publishers_users": "", "administrators_group": "", "administrators_type": "", "administrators_users": "", "user": true, "creator": true, "developer": true, "editor": true, "publisher": true, "administrator": true }</pre>	

10.3.50.3 Create

A POST request will create a new hosting type with the given data attributes and an automatically assigned unique “id” number.

If an “id” parameter is given then the hosting type with that id will be copied and its attributes will be (partially) overwritten by the posted data.

The POST request will return the created hosting type’s data. The returned data includes all the hosting type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the hosting type.

POST /webadmin/rest/hostingtypes/	
id	Hosting type id number to be copied (optional).
.....	All or any number and combination of hosting type attributes as used by the web content management



	system (see the GET request example response above).
Request	
<pre>{ "id": "123", "hostingtype": "XYZ" }</pre>	
Response	
<pre>{ "id": "789", "hostingtype": "XYZ", "user": true, "creator": true, "developer": true, "editor": true, "publisher": true, "administrator": true }</pre>	

10.3.50.4 Update

A PUT request will update the existing hosting type with the given “id” number with the given data attributes.

The PUT request will return the updated hosting type’s data. The returned data includes all the hosting type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the hosting type.

PUT /webadmin/rest/hostingtypes/	
id	Hosting type id number to be updated (required).
.....	All or any number and combination of hosting type attributes as used by the web content management system (see the GET request example response above).
Request	
<pre>{ "id": "123", "administrators_group": "XYZ" }</pre>	
Response	
<pre>{ "id": "123", "hostingtype": "ABC", "administrators_group": "Hosting Administrators", "user": true, "creator": true, "developer": true, "editor": true, "publisher": true, "administrator": true }</pre>	



10.3.50.5 Delete

A DELETE request will delete the existing hosting type with the given “id” number.

The DELETE request will return the deleted hosting type’s data. The returned data includes all the hosting type attributes as well as true/false flags for the currently logged in website administrator’s “user”, “creator”, “editor”, “publisher” and “administrator” access permissions for the hosting type.

DELETE /webadmin/rest/hostingtypes/	
id	Hosting type id number (required).
Request	
{ "id": "123" }	
Response	
{ "id": "123", "hostingtype": "ABC", "user": true, "creator": true, "developer": true, "editor": true, "publisher": true, "administrator": true }	

10.3.50.6 Exists

If a “hostingtype” parameter is given then a GET request will return “YES” or “NO” depending on if a hosting type with the given name exists.

Optionally, if an “id” parameter is also given then a GET request will return “YES” or “NO” depending on if a hosting type with the given name and another id than the given id exists.

GET /webadmin/rest/hostingtypes/exists/?hostingtype=HOSTINGTYPE	
hostingtype	Hosting type name (required).
id	Hosting type id number (optional).
Response	
{ "exists": "YES" }	

10.3.51 Configuration Settings

The “/webadmin/rest/config/” REST API function returns all data for one or more configuration settings.

10.3.51.1 Read

If an “id” parameter is given then a GET request will return those configuration settings’ data.

GET /webadmin/rest/config/?id=ID,ID,ID	
id	Configuration setting id name (required).



	One or more configuration setting id names separated by commas.
id	“_platform” Returns “asp”, “jsp” or “php” depending on the web content management system programming language version (optional).
id	“_administrator_license_restriction” Returns the max. number of license key permitted website administrator accounts or “-1” for unlimited (optional).
id	“_administrator_count” Returns the current number of website administrator accounts (optional).
id	“_personal_license_restriction” Returns the Hosting Edition/Suite max. number of license key permitted personal websites or “-1” for unlimited (optional).
id	“_smallbusiness_license_restriction” Returns the Hosting Edition/Suite max. number of license key permitted personal websites or “-1” for unlimited (optional).
id	“_professional_license_restriction” Returns the Hosting Edition/Suite max. number of license key permitted personal websites or “-1” for unlimited (optional).
id	“_smallenterprise_license_restriction” Returns the Hosting Edition/Suite max. number of license key permitted personal websites or “-1” for unlimited (optional).
id	“_enterprise_license_restriction” Returns the Hosting Edition/Suite max. number of license key permitted personal websites or “-1” for unlimited (optional).
id	“_hosting_license_restriction” Returns the Hosting Edition/Suite max. number of license key permitted personal websites or “-1” for unlimited (optional).
id	“_request_localname” Returns the web content management system server’s local name (optional).
id	“_request_servername” Returns the web content management system server’s external name (optional).
id	“_content_metainfo” Returns all content metainfo names used for any content items (optional).
id	“_content_productinfo” Returns all content productinfo names used for any content items (optional).
id	“_wizard”



	Returns the Quickstart Configuration stage (optional).
id	“_database” Returns the database connection strings and errors (optional).
id	“_database_import” Returns the current/latest datatabase import log (optional).
Response	
{ "default_template": "35", "default_stylesheet": "1", "default_version": "" }	

10.3.51.2 Update

A PUT request will update the configuration settings with the given data attributes.

If an “id” parameter is given then the PUT request will return those configuration settings’ (updated) data.

PUT /webadmin/rest/config/?id=ID,ID,ID	
id	Configuration setting id name (optional). One or more configuration setting id names separated by commas.
.....	All or any number and combination of website administrator user account attributes as used by the web content management system (see the GET request example response above).
Request	
{ "default_template": "35", "default_stylesheet": "1", "default_version": "" }	
Response	
{ "default_template": "35", "default_stylesheet": "1", "default_version": "" }	

10.3.51.3 Systemcheck

A GET request will check and report on various system environment settings such as file system locations and permissions, application server environment, caching and system environment settings.

GET /webadmin/rest/config/systemcheck/html/	
Request	
{ }	
Response	



HTML code

10.3.51.4 Database Connection(s)

A POST request will configure the database connection setting(s) for the Asbru Web Content Management System. A POST request will return the current and configured database connection string configuration settings; or optionally a response in HTML format.

Note: Database Connection(s) and Database Initialise and Database Import and Database Website Import parameters can be combined as a single request.

POST /webadmin/rest/config/database/	
database	Database connection string in the format required by your application server platform and database server for read/write access (required).
database_reader	Database connection string in the format required by your application server platform and database server for read-only access (optional).
logdatabase	Database connection string in the format required by your application server platform and database server for usagelog data only read/write access (optional).
logdatabase_reader	Database connection string in the format required by your application server platform and database server for usagelog data only read-only access (optional).
Request	
<pre>{ "database": ".....", "database_reader": ".....", "logdatabase": ".....", "logdatabase_reader": "....." }</pre>	
Response	
<pre>{ "error": "", "_database": ".....", "database": ".....", "database_reader": ".....", "logdatabase": ".....", "logdatabase_reader": ".....", "output": "" }</pre>	

POST /webadmin/rest/config/database/html/	
database	Database connection string in the format required by your application server platform and database server for read/write access (required).
database_reader	Database connection string in the format required by your application server platform and database server for read-only access (optional).
logdatabase	Database connection string in the format required by your application server platform and database server for usagelog data only read/write access (optional).
logdatabase_reader	Database connection string in the format required by your application server platform and database server



	for usagelog data only read-only access (optional).
Request	
<pre>{ "database": ".....", "database_reader": ".....", "logdatabase": ".....", "logdatabase_reader": "....." }</pre>	
Response	
HTML code	

10.3.51.5 Database Initialise

A POST request will initialise the web content management system database schema. A POST request will return the current and configured database connection string configuration settings as well as the output from the database initialisation; or optionally a response in HTML format.

Note: Database Connection(s) and Database Initialise and Database Import and Database Website Import parameters can be combined as a single request.

POST /webadmin/rest/config/database/	
create	“yes” will create all the database schema tables and indexes used by the web content management system.
drop	“yes” will drop all the database schema tables and indexes used by the web content management system. This should only be used to completely delete a web content management system instance; and eventually to recreate the database schema. WARNING: This will delete all web content management system database data.
Request	
<pre>{ "create": "yes", "drop": "yes" }</pre>	
Response	
<pre>{ "error": "", "_database": ".....", "database": ".....", "database_reader": ".....", "logdatabase": ".....", "logdatabase_reader": ".....", "output": "" }</pre>	

POST /webadmin/rest/config/database/html/	
create	“yes” will create all the database schema tables and indexes used by the web content management system.
drop	“yes” will drop all the database schema tables and



	indexes used by the web content management system. This should only be used to completely delete a web content management system instance; and eventually to recreate the database schema. WARNING: This will delete all web content management system database data.
Request	
{ "create": "yes", "drop": "yes" }	
Response	
HTML code	

10.3.51.6 Database Import

A POST request will import data into (and optionally delete the current data from) the web content management system database. A POST request will return the current and configured database connection string configuration settings as well as the output from the database initialisation; or optionally a response in HTML format.

Note: Database Connection(s) and Database Initialise and Database Import and Database Website Import parameters can be combined as a single request.

POST /webadmin/rest/config/database/	
insert	“yes” will import all database from the given database import file. Existing web content management system database data will be retained. (“importfile” or “file” required for “insert”).
delete	“yes” will delete all current data from the database used by the web content management system. WARNING: This will delete all web content management system database data.
importfile	The server-side database import file name. (“importfile” or “file” required for “insert”).
file	HTML FORM “file” type input upload. (“importfile” or “file” required for “insert”).
Request	
{ "insert": "yes", "delete": "yes", "importfile": "database.Business_Website.xml" }	
Response	
{ "error": "", " database": ".....", "database": ".....", "database_reader": ".....", "logdatabase": ".....", "logdatabase_reader": ".....", "output": "....." }	



POST /webadmin/rest/config/database/html/	
insert	“yes” will import all database from the given database import file. Existing web content management system database data will be retained. (“importfile” or “file” required for “insert”).
delete	“yes” will delete all current data from the database used by the web content management system. WARNING: This will delete all web content management system database data.
importfile	The server-side database import file name. (“importfile” or “file” required for “insert”).
file	HTML FORM “file” type input upload. (“importfile” or “file” required for “insert”).
Request	
<pre>{ "insert": "yes", "delete": "yes", "importfile": "database.Business_Website.xml" }</pre>	
Response	
HTML code	

10.3.51.7 Database Website Import

A POST request will import existing website pages, images and files into the web content management system database. A POST request will return the current and configured database connection string configuration settings as well as the output from the database initialisation; or optionally a response in HTML format.

Note: Database Connection(s) and Database Initialise and Database Import and Database Website Import parameters can be combined as a single request.

POST /webadmin/rest/config/database/	
insertwebsite	“yes” will import existing website pages
titleelement	Name of a defined editable region in the existing website pages to be used as the Title for the imported website pages; otherwise the website page filename will be used.
contentelement	Name of a defined editable region in the existing website pages to be used as the Content for the imported website pages; otherwise the website page HTML BODY will be used.
insertimages	“yes” will import existing website images from the “/image/” folder.
insertfiles	“yes” will import existing website files from the “/file/” folder.
Request	
<pre>{ "insertwebsite": "yes", "titleelement": "Title", "contentelement": "Main", "insertimages": "yes", "insertfiles": "yes" }</pre>	



}	
Response	
{ "error": "", "_database": ".....", "database": ".....", "database_reader": ".....", "logdatabase": ".....", "logdatabase_reader": ".....", "output": "....." }	

POST /webadmin/rest/config/database/html/	
insertwebsite	“yes” will import existing website pages
titleelement	Name of a defined editable region in the existing website pages to be used as the Title for the imported website pages; otherwise the website page filename will be used.
contentelement	Name of a defined editable region in the existing website pages to be used as the Content for the imported website pages; otherwise the website page HTML BODY will be used.
insertimages	“yes” will import existing website images from the “/image/” folder.
insertfiles	“yes” will import existing website files from the “/file/” folder.
Request	
{ "insertwebsite": "yes", "titleelement": "Title", "contentelement": "Main", "insertimages": "yes", "insertfiles": "yes" }	
Response	
HTML code	

10.3.51.8 Cache

A DELETE request will clear all memory cached configuration settings and other data.

DELETE /webadmin/rest/config/cache/	
Request	
{ }	
Response	
{ "error": "OK" }	

10.3.51.8.1 Clear (DEPRECATED)

A GET request will clear all memory cached configuration settings and other data.



Note: DEPRECATED – use DELETE /webadmin/rest/config/cache/ (Please see 10.3.51.8 Cache).

GET /webadmin/rest/config/cache/clear/	
Request	
{ }	
Response	
"OK"	

10.3.1 Database

The “/webadmin/rest/database/” REST API functions give access to upgrade, export and manage the database export files.

10.3.1.1 Upgrade

The “/webadmin/rest/database/upgrade/” REST API function upgrades the database schema and contents as may be required for a new release of the web content management system and returns the output from the database upgrade; optionally as HTML code.

GET /webadmin/rest/database/upgrade/	
database_version	Optionally, sets the old database version to upgrade from (to re-run old database version upgrades which may have failed for some reason); otherwise upgrade from the current database version to the current web content management system version (optional).
Response	
{ "output": "....." }	

GET /webadmin/rest/database/upgrade/html/	
database_version	Optionally, sets the old database version to upgrade from (to re-run old database version upgrades which may have failed for some reason); otherwise upgrade from the current database version to the current web content management system version (optional).
Response	
HTML code	

10.3.1.2 Content Dependencies

The “/webadmin/rest/database/contentdependencies/” REST API function upgrades how the content dependencies are stored in the database after changing the “use_contentdependencies_tables” configuration setting.

GET /webadmin/rest/database/contentdependencies/html/	
Response	
HTML code	



10.3.1.3 Export

The “/webadmin/rest/database/export/” REST API function exports the database data and optionally images and other files to a database export/import format XML file).

POST /webadmin/rest/database/export/	
export_content	“yes” to export the “content”, “content_elements”, “content_public”, “content_public_elements”, “content_archive”, “content_archive_elements” database tables; and optionally also any of the following database tables listed below.
export_contentbundles	“*” or comma-separated contentbundle names for content items to be exported.
export_contentpackages	“*” or comma-separated contentbundle names for content items to be exported.
export_contentclasses	“*” or comma-separated contentclass names for content items and “elements” database table data to be exported.
export_contentgroups	“*” or comma-separated contentgroup names for content items and “contentgroups” database table data to be exported.
export_contenttypes	“*” or comma-separated contenttype names for content items and “contenttypes” database table data to be exported.
export_filegroups	“*” or comma-separated filegroup names for content items and “filegroups” database table data to be exported.
export_filetypes	“*” or comma-separated filetype names for content items and “filetypes” database table data to be exported.
export_imagegroups	“*” or comma-separated imagegroup names for content items and “imagegroups” database table data to be exported.
export_imagetypes	“*” or comma-separated imagetype names for content items and “imagetypes” database table data to be exported.
export_linkgroups	“*” or comma-separated linkgroup names for content items and “linkgroups” database table data to be exported.
export_linktypes	“*” or comma-separated linktype names for content items and “linktypes” database table data to be exported.
export_productgroups	“*” or comma-separated productgroup names for content items and “productgroups” database table data to be exported.
export_producttypes	“*” or comma-separated producttype names for content items and “producttypes” database table data to be exported.
export_versions	“*” or comma-separated version names for content



	items and “versions” database table data to be exported.
export_fileformats	“*” to export the “fileformats” database table (requires “export_content=yes”).
export_imageformats	“*” to export the “imageformats” database table (requires “export_content=yes”).
export_segments	“*” to export the “segments” database table (requires “export_content=yes”).
export_usertests	“*” to export the “usertests” database table (requires “export_content=yes”).
export_images	“yes” to include exported “image” content items files’ contents in the database export file (requires “export_content=yes”).
export_files	“yes” to include exported “file” content items files’ contents in the database export file (requires “export_content=yes”).
export_allfolderfiles	“yes” to include all “/image/” and “/file/” folder files’ contents in the database export file.
export_ecommerce	“yes” to export any of the following database tables listed below
export_currencies	“yes” to export the “currencies” database table (requires “export_ecommerce=yes”).
export_discounts	“yes” to export the “discounts” database table (requires “export_ecommerce=yes”).
export_shipping	“yes” to export the “shipping” database table (requires “export_ecommerce=yes”).
export_tax	“yes” to export the “tax” database table (requires “export_ecommerce=yes”).
export_orders	“yes” to export the “orders” database table (requires “export_ecommerce=yes”).
export_orderitems	“yes” to export the “orderitems” database table (requires “export_ecommerce=yes”).
export_users	“yes” to export the “users” database table.
export_usergroups	“*” or comma-separated usergroup names for users and “usergroups”, “usergroups2”, “user2groups”, “users_usergroups”, “permissions”, “userslog” database table data to be exported (requires “users=yes”).
export_usertypes	“*” or comma-separated usertype names for users and “usertypes”, “usertypes2”, “user2types”, “users_usertypes”, “permissions”, “userslog” database table data to be exported (requires “users=yes”).
export_hosting	“yes” to export the “hosting” database table.
export_hostinggroups	“*” or comma-separated hostinggroup names for website hosting clients and “hostinggroups” database table data to be exported (requires “hosting=yes”).
export_hostingtypes	“*” or comma-separated hostingtype names for website hosting clients and “hostingtypes” database table data to be exported (requires “hosting=yes”).



export_config	“yes” to export the “config” database table.
export_comments	“yes” to export the “comments” database table.
export_databases	“yes” to export the “data” and “data*” database tables.
export_projects	“yes” to export the “projects” database table.
export_projecttasks	“yes” to export the “projecttasks” database table.
export_websites	“yes” to export the “websites” database table.
export_workflow	“yes” to export the “workflow” database table.
export_other	“yes” to export the “fonts”, “guestbook” database tables.
export_blanks	“yes” to export blank (“”) data attributes; otherwise blank data attributes will not be exported (reducing the database export file size).
description	Description of the database export (to be saved as a separate database.....html file) (optional).
Response	
<pre>{ "filename": "database.....xml", "bytesize": ".....", "filesize": ".....", "output": "....." }</pre>	

POST /webadmin/rest/database/export/html/	
...	Please see above for parameter details.
Response	
HTML code	

10.3.1.4 Backup

The “/webadmin/rest/database/backup/” REST API function gives access to list, download and delete database backup files (as generated through the database export functionality).

10.3.1.4.1 List

This lists the database backup files available for download.

GET /webadmin/rest/database/backup/	
Response	
<pre>{ "filename": "database.....xml", "description": "", "bytesize": ".....", "filesize": "....." }, { "filename": "database.....xml", "description": "", "bytesize": ".....", "filesize": "....." }</pre>	



```
{
  "filename": "database.....xml",
  "description": "",
  "bytesize": ".....",
  "filesize": "....."
}
```

10.3.1.4.2 Download

This downloads a given database export/backup file.

GET /webadmin/rest/database/backup/?backup=BACKUP	
backup	Database export/backup file name (required).
Response	
File contents	

10.3.1.4.3 Delete

This deletes a given database export/backup file.

DELETE /webadmin/rest/database/backup/?backup=BACKUP	
backup	Database export/backup file name (required).
Response	
{ "error": "" }	

10.3.1.5 Encrypt (DEPRECATED)

The “/webadmin/rest/database/encrypt/” REST API function updates all user accounts’ passwords to be stored using encryption (as configured for the web content management system).

WARNING: This functionality is only for use with the web content management system’s basic cipher encryption functionality. If used with the hashed user passwords functionality, the user account passwords may become corrupted, requiring users to reset their passwords to login to the website and/or web content management system.

GET /webadmin/rest/database/encrypt/html/	
Response	
HTML code	

10.3.1.6 Decrypt (DEPRECATED)

The “/webadmin/rest/database/decrypt/” REST API function updates all user accounts’ passwords to be stored without using encryption (as configured for the web content management system).

WARNING: This functionality is only for use with the web content management system’s basic cipher encryption functionality. If used with the hashed user passwords functionality,



the user account passwords may become corrupted, requiring users to reset their passwords to login to the website and/or web content management system.

GET /webadmin/rest/database/decrypt/html/	
Response	
HTML code	

10.3.2 Website Administrator Settings

The “/webadmin/rest/user/” REST API function returns and updates all website administrator user account settings for the logged in website administrator.

10.3.2.1 Read

The logged in website administrator’s user account settings are returned.

GET /webadmin/rest/user/	
Response	
<pre>{ "id": "6", "username": "paulgreen", "password": "XYABQ0EB1LG7MCCUXTUJXRXBK1FJ0LXL4H86HEGVY8VK5QHRHGA71KB9AXGLVBYN", "onetimepassword": "", "onetimepassword_updated": "2018-04-04 17:42:29", "title": "", "name": "Paul Green", "organisation": "", "email": "", "gender": "", "photo": "", "birthdate": "", "birthyear": "", "birthmonth": "", "birthday": "", "userclass": "administrator", "invoice_name": "", "invoice_organisation": "", "invoice_address": "", "invoice_postalcode": "", "invoice_city": "", "invoice_state": "", "invoice_country": "", "invoice_phone": "", "invoice_fax": "", "invoice_email": "", "invoice_website": "", "delivery_name": "", "delivery_organisation": "", "delivery_address": "", "delivery_postalcode": "", "delivery_city": "", "delivery_state": "",</pre>	



```
"delivery_country": "",
"delivery_phone": "",
"delivery_fax": "",
"delivery_email": "",
"delivery_website": "",

"card_type": "",
"card_number": "",
"card_issuedmonth": "",
"card_issuedyear": "",
"card_expirymonth": "",
"card_expiryear": "",
"card_name": "",
"card_cvc": "",
"card_issue": "",
"card_postalcode": "",

"content_editor": "",
"hardcore_upload": "",
"hardcore_format": "",
"hardcore_width": "",
"hardcore_height": "",
"hardcore_onenter": "",
"hardcore_onctrlenter": "",
"hardcore_onshiftenter": "",
"hardcore_onaltenter": "",
"hardcore_skin": "",
"hardcore_toolbar": "",
"hardcore_toolbar1": "",
"hardcore_toolbar2": "",
"hardcore_toolbar3": "",
"hardcore_toolbar4": "",
"hardcore_toolbar5": "",
"hardcore_toolbar6": "",
"hardcore_toolbar7": "",
"hardcore_toolbar8": "",
"hardcore_toolbar9": "",
"hardcore_toolbar10": "",
"hardcore_formatblock": "",
"hardcore_fontname": "",
"hardcore_fontsize": "",
"hardcore_customscript": "",
"hardcore_encoding": "",
"hardcore_language": "",

"startpage": "",

"webadmin_user": "forbid",
"webadmin_ecommerce": "forbid",
"webadmin_library_packages": "forbid",

"workspace_sections": "",
"index_workspace": "",
"index_workspace_projects": "",
"index_workspace_comments": "",
"index_content": "",
"index_library": "",
"index_search": "",
"index_searchadvanced": "",
"index_searchreplace": "",
"index_product": "",
"index_stock": "",
"index_order": "",
"index_segments": "",
"index_usertests": "",
```



```
"index_heatmaps": "",
"index_user": "",
"index_websites": "",
"index_sales": "",
"index_databases": "",
"index_hosting": "",
"index_projects": "",

"menu_selection": "",
"statistics_reports": "",
"sales_reports": "",

"shopcart": "",
"wishlist": "",

"failed": 0,
"locked": 0,

"userslog": [
],

"user": true,
"creator": true,
"editor": true,
"publisher": true,
"administrator": true,

"orderadministrator": false,
"salesadministrator": true,
"databasesadministrator": true,
"experienceadministrator": true,
"useradministrator": true,
"emailadministrator": true,
"usageadministrator": true,
"configadministrator": false,
"superadmin": false
}
```

10.3.2.2 Update

A PUT request will update the logged in website administrator’s user account settings with the given data attributes.

The PUT request will return the updated logged in website administrator’s user account settings.

PUT /webadmin/rest/user/	
.....	All or any number and combination of website administrator user account attributes as used by the web content management system (see the GET request example response above).
Request	
{ "password": "secret", "birthdate": "1999-12-31", "birthyear": "1999", "birthmonth": "12", "birthday": "31" }	
Response	



```
{
  "id": "6",
  "username": "paulgreen",
  "password":
"XYABQ0EB1LG7MCCUXTUJXRXBK1FJ0LXL4H86HEGVY8VK5QHRHGA71KB9AXGLVBYN",
  .....
  "birthdate": "1999-12-31",
  "birthyear": "1999",
  "birthmonth": "12",
  "birthday": "31",
  .....
}
```

10.3.3 Superadmin Password

The “/webadmin/rest/password/” REST API function will email the superadmin login details to the configured superadmin email address.

10.3.3.1 Read

An empty response is returned.

GET /webadmin/rest/password/	
Response	
{	
}	

10.3.4 Logout

The “/webadmin/rest/logout/” REST API function logs the logged in website administrator account out.

The POST request returns an empty response.

POST /webadmin/rest/logout/	
Request	
{	
}	
Response	
{	
}	

10.3.5 Custom Modules

The “/webadmin/rest/modules/” REST API functions returns custom module data available for the web content management system.

GET /webadmin/rest/modules/	
Response	
[
{	
"modulePaymentTitle": "Manual Payment Handling",	



```
"modulePaymentOptions": "Handle order and payment details manually.",
    "id": "0"
},
{
    "modulePaymentTitle": "Payment Module Template",
    "modulePaymentOptions": "Template for programming your own payment
module.<br>Your payment service provider account<br><input type=\"text\"
size=\"40\" name=\"payment_account\" value=\"\"><br>",
    "id": "1"
},
{
    "modulePaymentTitle": "PayPal",
    "modulePaymentOptions": "Your PayPal account\\email<br><input type=\"text\"
size=\"60\" name=\"paypal_email\" value=\"\"><br>Payment
instructions<br><input type=\"text\" size=\"60\" name=\"paypal_text\"
value=\"\"><br><br><input type=\"radio\" name=\"paypal_method\" value=\"get\"
checked> Use GET method for PayPal payment button forms for Mozilla
Thunderbird and Firefox compatibility.<br><input type=\"radio\"
name=\"paypal_method\" value=\"post\"> Use POST method for PayPal payment
button forms.<br><br><input type=\"radio\" name=\"paypal_cart\"
value=\"details\" checked> Order item details on PayPal payment
page.<br><input type=\"radio\" name=\"paypal_cart\" value=\"simple\"> No order
item details on PayPal payment page.<br><br><input type=\"radio\"
name=\"paypal_echeck\" value=\"Completed\" checked> Accept eCheck payments
when cleared.<br><input type=\"radio\" name=\"paypal_echeck\"
value=\"Pending\"> Accept eCheck payments immediately.<br><!--Payments
description (depreciated - for backwards compatibility only)<br><input
type=\"text\" size=\"60\" name=\"paypal_item_name\" value=\"\"><br>--><p
style=\"font-size: small; font-weight: normal;\"><a
href=\"https://www.paypal.com/uk/mrb/pal=SRD5TKTJUGFKJ\"
target=\"_blank\">Click here to sign up for a free PayPal Merchant account and
start accepting credit card payments instantly.</a></font><br><a
href=\"https://www.paypal.com/uk/mrb/pal=SRD5TKTJUGFKJ\"
target=\"_blank\"><img
src=\"https://www.paypalobjects.com/webstatic/mktg/Logo/pp-logo-
200px.png\" border=\"0\" alt=\"Sign up for PayPal and start accepting credit
card payments instantly.\"></a></p>",
    "id": "2"
},
{
    "moduleToolbarLink": "\\webadmin\\module\\example\\index.jsp?menu=example",
    "moduleToolbarImage": "\\webadmin\\module\\example\\module.gif",
    "moduleToolbarTitle": "Example Module",

    "moduleHomeIntroTitle": "Example Module Home",
    "moduleHomeIntroText": "This is the Example Module Home",
    "moduleHomeIntroImage": "\\webadmin\\module\\example\\module.gif",
    "moduleHomeIntroLink":
    "\\webadmin\\module\\example\\index.jsp?menu=example",
    "moduleHomeMenuTitle": "Example Module Home Menu",
    "moduleHomeMenuLink":
    "\\webadmin\\module\\example\\index.jsp?menu=example",

    "moduleContentIntroTitle": "Example Module Content",
    "moduleContentIntroText": "This is the Example Module Content",
    "moduleContentMenuTitle": "Example Module Content Menu",
    "moduleContentMenuLink":
    "\\webadmin\\module\\example\\index.jsp?menu=example",
```



```
"moduleContentTab": "exampletab",
"moduleContentTabTitle": "Example Tab",
"moduleContentTabContent": "Please wait while loading",
"moduleContentTabURL": "\\webadmin\\module\\example\\content.html",
"moduleContentTabScript": "alert('Example tab Javascript function');",

"moduleLibraryIntroTitle": "Example Module Library",
"moduleLibraryIntroText": "This is the Example Module Library",
"moduleLibraryMenuTitle": "Example Module Library Menu",
"moduleLibraryMenuLink":
"\\/webadmin\\module\\example\\/index.jsp?menu=example",

"moduleEcommerceIntroTitle": "Example Module E-Commerce",
"moduleEcommerceIntroText": "This is the Example Module E-Commerce",
"moduleEcommerceMenuTitle": "Example Module E-Commerce Menu",
"moduleEcommerceMenuLink":
"\\/webadmin\\module\\example\\/index.jsp?menu=example",

"moduleCommunityIntroTitle": "Example Module Community",
"moduleCommunityIntroText": "This is the Example Module Community",
"moduleCommunityMenuTitle": "Example Module Community Menu",
"moduleCommunityMenuLink":
"\\/webadmin\\module\\example\\/index.jsp?menu=example",

"moduleDatabasesIntroTitle": "Example Module Databases",
"moduleDatabasesIntroText": "This is the Example Module Databases",
"moduleDatabasesMenuTitle": "Example Module Databases Menu",
"moduleDatabasesMenuLink":
"\\/webadmin\\module\\example\\/index.jsp?menu=example",

"moduleDatabasesIntroTitle": "Example Module Databases",
"moduleDatabasesIntroText": "This is the Example Module Databases",
"moduleDatabasesMenuTitle": "Example Module Databases Menu",
"moduleDatabasesMenuLink":
"\\/webadmin\\module\\example\\/index.jsp?menu=example",

"moduleExperienceIntroTitle": "Example Module Experience Management",
"moduleExperienceIntroText": "This is the Example Module Experience
Management",
"moduleExperienceMenuTitle": "Example Module Experience Management Menu",
"moduleExperienceMenuLink":
"\\/webadmin\\module\\example\\/index.jsp?menu=example",

"moduleStatisticsIntroTitle": "Example Module Statistics",
"moduleStatisticsIntroText": "This is the Example Module Statistics",
"moduleStatisticsMenuTitle": "Example Module Statistics Menu",
"moduleStatisticsMenuLink":
"\\/webadmin\\module\\example\\/index.jsp?menu=example",

"moduleUsersIntroTitle": "Example Module Users",
"moduleUsersIntroText": "This is the Example Module Users",
"moduleUsersMenuTitle": "Example Module Users Menu",
"moduleUsersMenuLink":
"\\/webadmin\\module\\example\\/index.jsp?menu=example",

"moduleConfigIntroTitle": "Example Module Config",
"moduleConfigIntroText": "This is the Example Module Config",
"moduleConfigMenuTitle": "Example Module Config Menu",
"moduleConfigMenuLink":
"\\/webadmin\\module\\example\\/index.jsp?menu=example",

"moduleHostingIntroTitle": "Example Module Hosting",
"moduleHostingIntroText": "This is the Example Module Hosting",
"moduleHostingMenuTitle": "Example Module Hosting Menu",
"moduleHostingMenuLink":
```



```
"\"/webadmin\"/module\"/example\"/index.jsp?menu=example",  
  
  "id": "3"  
},  
  
{  
  "id": "4"  
}  
}
```

10.3.6 Custom Extension Programs

The “/webadmin/rest/extension/” REST API function returns all custom extension programs (special codes) available for the web content management system.

10.3.6.1 List

GET /webadmin/rest/extension/	
Response	
<pre>[{ "filename": "breadcrumbs.jsp" }, "description": "" }, { "filename": "user.jsp" }, "description": "" }]</pre>	

10.3.7 API Programs

The “/webadmin/rest/api/” REST API function returns all custom API programs available for the web content management system.

10.3.7.1 List

GET /webadmin/rest/api/	
Response	
<pre>[{ "filename": "copy.jsp" }, "description": "" }, { "filename": "upload.jsp" }, "description": "" }]</pre>	

10.3.7.2 Media (external)

The “/webadmin/rest/api/media/” REST API function returns a list of external media library folders and files available to the web content management system.



GET /webadmin/rest/api/media/	
.....	Custom media program script parameters (Please see “/webadmin/api/EXAMPLE.media.aspx jsp php” for details).
Response	
FOLDER1 FOLDER2 FOLDER3	
Response	
FILE1 URL1 FILE2 URL2 FILE3 URL3	

10.3.8 Publish Scheduled

The “/webadmin/rest/publishscheduled/txt/” REST API function publishes/unpublishes scheduled content and activates/notifies/expires user accounts (and website hosting clients for the Hosting Edition) for example to be called periodically through “cron” or another scheduler.

GET /webadmin/rest/publishscheduled/txt/	
Response	
{ "error": "" }	