



User Guide

© 2014 - 2016 Sequentum



Content Grabber

© 2014 - 2016 Sequentum

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: November 2017

Table of Contents

Foreword	0
Part I Introduction	8
1 Editions & Licensing.....	8
2 What is Web Scraping?.....	11
3 Web Scraping with Content Grabber.....	12
4 Web Scraping Limitations.....	13
5 Web Scraping Techniques.....	16
HTML Content	17
Dynamic Websites	18
XPath and Selection Techniques	19
Regular Expressions	20
6 Converting Visual Web Ripper Projects.....	20
7 Upgrading to Content Grabber 2.....	23
Part II Quick Start with Content Grabber	28
1 Installing Content Grabber.....	28
2 Content Grabber Basics.....	30
3 Exploring the Main Window.....	34
4 Building Your First Agent.....	35
Choosing a Start URL	37
Select the Content to Capture	38
Refine Your Data (optional)	45
Output Data Format	48
Test Your Agent	50
Scheduling	53
Run Your Agent	53
Part III Content Grabber Editor	55
1 Customizing the Editor Layout.....	55
2 Working With the Web Browser.....	57
3 Command Configuration.....	60
4 Copying or Moving Agent Commands.....	66
5 Selection Tools and Short-cut Keys.....	67
6 Navigating an Agent.....	73
7 Testing/Debugging an Agent.....	76
Using the Debugger	76
Using Logging	81
8 Scheduling.....	83
Cron Expressions	86
Windows Task Scheduler	89

9	Copying & Moving Agents.....	92
10	Exporting & Importing Agents.....	93
11	Automated Agent Backups.....	95
12	Template Libraries.....	96
13	Default Folders.....	96

Part IV Selection Techniques 98

1	Selection XPath.....	99
2	Optimizing Selections.....	108
3	List Selections.....	109
4	Selection Anchors.....	112
5	Editing XPath Manually.....	114

Part V Agent Commands 116

1	Container Commands.....	117
2	Capture Commands.....	118
	Web Element Content	120
	Download Image	123
	Download Video	126
	Download Document	128
	Download Screenshot	132
	Download Page	134
	Calculated Value	136
	Page Attribute	136
	Data Value	137
3	Action Commands.....	138
	Agent	140
	Navigate Link	141
	Navigate URL	142
	Navigate Pagination	145
	Crawl Website	148
	Set Form Field	149
	Manual Navigation	151
	Action Configuration	152
	Action.....	153
	Browser.....	155
	Events.....	157
	Wait.....	167
	Wait for Content.....	171
4	List Commands.....	174
	Data List	175
	Web Element List	175
5	Group Commands.....	176
	Group	177
	Group in Page Area	178
6	Branch Commands.....	178
	If..Elseif..Else	179

Exit or Retry	180
7 Other Commands.....	181
Execute Script	181
Remove Duplicate	188
Transform Page	190
Refresh Document	192
Close Browser	192
Screenshot Log	193
8 Composite Commands.....	194
Resolve CAPTCHA	194
9 Web Forms.....	195
10 Command Library.....	198
 Part VI Error Handling	 204
1 Agent Error Handling.....	205
2 Error Logs and Notifications.....	207
3 Post Status.....	214
 Part VII Extracting Data From Non-HTML Documents	 216
 Part VIII Crawling a Website	 221
 Part IX Data	 223
1 Database Connections.....	223
2 The Internal Database.....	224
3 Using Data Input.....	230
Data Providers	230
Input Parameters	237
Agent Data	241
4 Exporting Data.....	243
Selecting an Export Target	243
Exporting Downloaded Images and Files	246
Character Encoding	247
Changing the Default Data Structures	248
Exporting Data with Scripts	256
Export From Multiple Agents	256
5 Distributing Data.....	257
Email Distribution	257
FTP Distribution	260
Google Drive Distribution	261
Dropbox Distribution	268
Scripting Data Distribution	274
6 Removing Duplicate Data.....	274
7 Extracting New Data Only.....	276
8 Reusing Existing Data.....	279
9 Change Tracking.....	282

10	Data Counting.....	290
Part X Anonymous Web Scraping		293
Part XI Obeying Robots Rules		294
Part XII CAPTCHA & IP Blocking		297
1	CAPTCHA Blocking.....	297
2	IP Blocking & Proxy Servers.....	304
Part XIII Improving Agent Performance and Reliability		310
1	Using MS SQL as Internal Database.....	310
2	Using the HTML Parser.....	311
3	Optimizing Agent Commands.....	313
4	Performance Sessions.....	314
5	Multithreading.....	319
Part XIV Building Self-Contained Agents		321
1	Customizing the User Interface.....	324
2	Using Input Data.....	327
3	Using a Self-Contained Agent.....	328
Part XV Running Agents from the Command-Line		330
1	Using the Content Grabber runtime.....	334
Part XVI Running Multiple Instances of the Same Agent		336
Part XVII Scripting		337
1	Script Languages.....	338
2	Script Library.....	343
3	Assembly References.....	345
4	Script Utilities.....	346
5	Script Template Library.....	364
6	Agent Initialization Scripts.....	367
7	Data Export Scripts.....	371
8	Data Distribution Scripts.....	380
9	Content Transformation Scripts.....	383
10	Data Input Scripts.....	388
11	Image OCR Scripts.....	393
12	Convert Document to HTML Scripts.....	398
13	Custom Scripts.....	404

14	Condition Scripts.....	410
15	Command Transformation.....	414
Part XVIII Programming Interface		420
1	Building a Desktop Application.....	420
	Visual Studio Configuration	434
	Distributing Your Application	437
2	Building a Web Application.....	437
	Using the Content Grabber Agent Service	438
	Using Simple Web Requests	450
3	Sessions.....	485
4	API Access Keys.....	487
Part XIX Website Automation		489
1	Website Testing & Documentation.....	489
	Index	497

1 Introduction

Congratulations on your purchase of Content Grabber. We value you as a customer and we want to help you increase your web-scraping productivity. We have put years of effort into designing and developing Content Grabber. Our software can be a great help in simplifying the web-scraping process, increasing your efficiency, and optimizing the performance. We have the best web-scraping solution on the market and strive to exceed your expectations.

You can learn more in these sections:

- Editions and Licensing
- What is Web Scraping?
- Web Scraping with Content Grabber
- Web Scraping Limitations
- Web Scraping Techniques



We have designed this comprehensive user guide to help you get up and running quickly with Content Grabber, and then learn the advanced features so that you can get your job done more efficiently. After Installing Content Grabber, we recommend that you get familiar with Understanding the Concept, Exploring the Main Window and Building Your First Agent.

We welcome your input and feedback, so please send us an email at: info@ContentGrabber.com.au. We are keen to know how you are using Content Grabber, and welcome any inquiry from you or your colleagues.

1.1 Editions & Licensing

The Content Grabber software comes in three editions, each of which requires a license. Please see the license agreement in the Content Grabber installation folder for the full license agreement. In this article we highlight the most important differences among the Content Grabber editions.

These are the current license editions of Content Grabber:

- **Professional Edition**
- **Premium Edition**
- **Server Edition**

This user guide describes all features for all editions of the Content Grabber software. Some sections of this guide present features that are only available in specific versions, and these features may not be available in the software edition that you are using. We explain these differences below.

Professional Edition

If a current-license copy of the **Professional Edition** of the software is running on a computer, then a single user may create, edit, and run web-scraping agents without restrictions.

The **Professional Edition** can create self-contained agents that you can distribute and users can run royalty-free. These self-contained agents use a standard user interface that display a promotional message for **Content Grabber**, and it isn't possible to edit the display templates to remove this message.

It is not possible to open any agents in the **Professional Edition** that were built in the **Premium Edition**.

The **Content Grabber Runtime** cannot run or edit web-scraping agents built with the **Professional Edition**. This also means that the entire API and the Script Library is unavailable to agents built with the **Professional Edition**.

The **runagent.exe** command-line program will only run agents built with the **Premium Edition** (so long as that edition of the software is on the computer).

Premium Edition

The Professional Edition is the most feature-rich version of the Content Grabber software suite.

As with the **Professional Edition**, if a current-license copy of the **Premium Edition** of the software is running on a computer, then a single user may create, edit, and run web-scraping agents without restrictions.

With the **Premium Edition**, you can use the API and the Script Library.

The **Premium Edition** can create self-contained agents that you can distribute and users can run royalty-free. These self-contained agents use a standard user interface that displays a promotional message for Content Grabber, but the **Premium Edition** gives you the ability to change the display templates to remove this message.

You can open agents in the **Premium Edition** that were built in the **Professional Edition**, but the only features that will function are those that are available in the **Professional Edition**.

Server Edition

The Server Edition of Content Grabber is intended for use in production environments for running your agents and carrying out simple maintenance. It is not a developer license and is priced accordingly.

The **Server Edition** cannot create new agents, but it can run and edit agents built with either the **Premium Edition** or the **Professional Edition**.

Restrictions for the Server Edition:

- With this edition, you may not change the agent command structure, or alter an agent in a way that would change the output data structure.

- It is not possible to add, delete, move, or copy commands in this edition, nor can it modify the **Disabled** and **Export** command properties.
- This edition cannot export agents or create self-contained agents.
- This edition cannot edit script libraries, though it can run agents that use script libraries.

Content Grabber Runtime

The **Content Grabber Runtime** cannot change the agent command structure, or modify the agent in a way that would change the output data structure. Also the **Runtime** cannot add, delete, move or copy commands, nor can it change the **Disabled** and **Export** command properties.

With the **Premium Edition**, you can create and distribute agents royalty-free with the **Content Grabber Runtime**. Please be aware that the **Content Grabber Runtime** has its own license and restrictions.

The **Content Grabber Runtime** cannot run or edit web-scraping agents built with the **Professional Edition**. This also means that the entire API and the Script Library is unavailable to agents built with the **Professional Edition**.

An agent built with the **Professional Edition** that you later edit with the **Premium Edition** cannot be run with the **Content Grabber Runtime**.

1.2 What is Web Scraping?

We provide this brief introduction for those who are new to web-scraping. If you are familiar with web-scraping, then you may want to begin by reading the article [Installing Content Grabber](#).

Web-scraping is the process of extracting data from websites and storing that data in a structured, easy-to-use format. The value of a web-scraping tool like Content

Grabber is that you can easily specify and collect large amounts of source data that may be very dynamic (data that changes very frequently).

Usually, data available on the Internet has little or no structure and is only viewable with a web browser. Elements such as text, images, video, and sound are built into a web page so that they are presentable in a web browser. It can be very tedious to manually capture and separate this data, and can require many hours of effort to complete. With Content Grabber, you can automate this process and capture website data in a fraction of the time that it would take using other methods.

Web-scraping software interacts with websites in the same way as you do when using your web browser. However, in addition to displaying the data in a browser on your screen, web-scraping software saves the data from the web page to a local file or database.



You can configure web-scraping agents to run on multiple websites, and you can schedule each agent to run automatically. It's easy to configure your agent to run as frequently as you like (hourly, daily, weekly, monthly) to ensure that you are capturing the very latest data.

1.3 Web Scraping with Content Grabber

With Content Grabber, you can automatically harvest data from a website and deliver the content as structured data in multiple database formats (Oracle, SQLServer, My SQL, OLE DBE), or in other formats such as Excel spreadsheets, CSV or XML files.

Content Grabber can also extract data from highly dynamic websites where most other extraction tools are incapable. It can process AJAX-enabled websites, submit forms repeatedly to cover all possible input values, and manage website logins.

Web-scraping technology is transforming the Internet into a structured data source, and Content Grabber is opening up numerous business opportunities for both corporations and individuals. The following is just a small sample of how web-scraping technology is optimizing and enabling new businesses:

- Price Comparison Portals / Mobile Apps
- Collaborative lists (home foreclosures, job boards, & tourist attractions)
- News & Content Aggregation
- Competitive price monitoring
- Monitor dealers for price compliance
- Track inventory on retailer websites
- Social media and brand monitoring
- Locate the highest-ranking keywords of your competitors on all major search engines
- Background Checking
- Confirm the integrity of business partners
- Monitor online sources for copyright infringement
- Sales Lead Generation
- Content migration (CMS & CRM).

Content Grabber is a powerful, visual web-scraping tool that can do all of this and much more. We provide a comprehensive user guide to help you get up and running quickly. After Installing Content Grabber, we recommend that you look at Content Grabber Basics, and then get familiar with Exploring the Main Window and then Building Your First Agent.

1.4 Web Scraping Limitations

Web-scraping can be challenging if you want to mine data from complex, dynamic websites. If you're new to web-scraping, then we recommend that you begin with an easy website: one that is mostly static and has little, if any, AJAX or JavaScript.

After you get familiar with the navigation paths for your target website, you need to identify a good start URL. Sometimes this is simply the start URL of the website, but often the best URL is the one for a sub-page—such as a product listing. Once you have this URL, you'll need to copy it and then paste it into the address bar of Content Grabber.

NOTE: Some websites allow navigation without any corresponding change in the visible URL. In such cases, you may not have a start URL that points directly to your start webpage, and so you'll need to add preliminary steps to your agent to navigate to that webpage.

Web-scraping can be also challenging if you don't have the proper tools. Largely, you're completely at the mercy of the target website, and that website can change at anytime - without notice. Or, it may contain faulty JavaScript that causes it to crash and exhibit surprising behavior. The server that hosts the website may crash, or the website may undergo maintenance. Many potential problems can occur during a lengthy web-scraping session, and you have very little influence on any of them. Content Grabber offers an array of advanced error-handling and stability features that can help you manage many of the problems that a web-scraping agent is likely to encounter.

In addition to the unreliable websites, another challenge is that some web-scraping tasks are especially difficult to complete - including the following:

- Extracting data from complex websites
- Extracting data from websites that use deterrents
- Extracting huge amounts of data
- Extracting data from non-HTML content.

Extracting Data From Complex Websites

If you are developing web-scraping agents for a large number of different websites, you will probably find that around 50% of the websites are very easy, 30% are modest in difficulty, and 20% are very challenging. For a small percentage, it will be effectively impossible to extract meaningful data. It may take two weeks or more for a web-scraping expert to develop an agent for such a website, so the cost of developing the agent is likely to outweigh the value of the data you might be able to extract.

Extracting Data From Websites Using Deterrents

Web-scraping will always be challenging for any website with active deterrents in place. If it is necessary to login to access the content that you want to extract, then the website can always cancel your account and make it impractical to create new accounts.

Some websites use browser fingerprinting to identify and block your access to the website. Fingerprinting uses JavaScript to make a positive identification by examining your browser and computer specifications, and thereby making it impossible to circumvent.

Another method for websites that are wary of crawlers or scrapers is the use of CAPTCHA. Content Grabber includes tools you can use to overcome CAPTCHA protection, but you'll incur additional costs to get a 3rd-party to do automatic CAPTCHA processing. See CAPTCHA Blocking for more information.

The most common protection technique is using your IP address to identify and block your access to a website. You can usually circumvent this technique by using a proxy rotation service, which hides your actual IP address and uses a new IP address every time you request a web page from a website. See IP Blocking & Proxy Servers for more information.

NOTE: Ethically and legally, we recommend that you avoid websites that are actively taking measures to block your access, even if you are able to circumvent the protection.

Extracting Huge Amounts of Data

A web-scraping tool must actually visit a web page to extract data from it. Downloading a web page takes time, and it could take weeks and months to load and extract data from millions of web pages. For example, it's virtually impossible to extract all product data from Amazon.com, since there are too many web pages.

Extracting Data From Non-HTML Content

Some websites are built entirely in Flash, which is a small-footprint software application that runs in the web browser. Content Grabber can only work with HTML content, so it can only extract the Flash file. However, it can't interact with the Flash application or extract data from within the Flash application.

Many websites provide data in the form of PDF files and other file formats. Though it cannot directly extract data from such files, Content Grabber can easily download those files and convert the files into an HTML document using 3rd-party converters to extract data from the conversion output. The document conversion happens very quickly in real-time, so it will seem as though you are performing a direct extraction. It's important to realize that PDF documents and most file formats don't contain content that is easily convertible into structured HTML. To do that, you can use the Regular Expressions feature of Content Grabber to resolve the conversion output.

1.5 Web Scraping Techniques

Content Grabber makes it easy to extract data from most websites without requiring much prior knowledge about web-scraping techniques. However, you'll be able to build better web-scraping agents if you know some basic techniques. Some very difficult

websites will require in-depth knowledge, but this user guide can help you gain more understanding and direct you to additional resources.

The following topics are important if you want to become proficient at web-scraping, but they are not necessary a prerequisite for successful use of Content Grabber on all websites. Click the links to learn more about each topic:

- **HTML Content** - Web pages are driven by HTML, which is the basic language for building websites.
- **Dynamic Websites** - It can be challenging to perform data extraction on dynamic websites. So, it's good to have a general understanding of how JavaScript works, since it is found on most dynamic websites.
- **XPath and Selection Techniques** - Most web scraping tools extract data from a website by selecting web elements on the web page. **XPath** is a language that manages the web selection.
- **Regular Expressions** - **XPath** can select a web element such as a paragraph of text, but you may have interest only in a small part of the web element content. **Regular Expressions** is a language for extracting small bits of text from a larger text element.

1.5.1 HTML Content

HTML stands for **HyperText Markup Language** - the standard markup language for creating web pages. It consists of content that is defined in an HTML document by *tags* that appear in brackets, such as `<html>`. Typically, these tags are seen in pairs, with one on each end of the content that they represent (such as `<h1>` and `</h1>`). The first tag in a pair is the *start* tag, and the second tag is the *end* tag (also known as *opening* tags and *closing* tags). Some tags that represent empty elements don't come in pairs, such as ``.

The purpose of a web browser is to read HTML documents and compose them into visible web pages. The browser does not display the HTML tags, but rather interprets

the tags and displays content on the page that corresponds to that tag. HTML describes the structure of a web page *semantically*, with cues for presentation. This distinguishes it as a *markup* language rather than a programming language.

HTML elements are the building blocks of any website, including embedded images and objects, and also interactive forms. It provides the structure for a page by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. It can also contain scripts written in languages such as JavaScript - which controls the behavior of HTML web pages.

Content Grabber uses **XPath** to select specific HTML tags and then extracts content from those tags. An HTML tag can contain both text and attributes. For example, a HTML tag that displays an image will contain a **src** attribute that specifies the URL of the image to display. Content Grabber can extract both tag text and tag attributes, and may perform certain actions on the content it extracts. For example, it may extract the **src** attribute from an `<image>` HTML tag and then use the URL to download the image.

There are many websites that have HTML tutorials. Here is one example:

http://www.w3schools.com/html/html_intro.asp

1.5.2 Dynamic Websites

Using HTML script, a client-side dynamic web page will continue to load more content after the initial content loads and the page elements are available to the user. The most common language for client-side scripting is JavaScript, and it may use AJAX (Asynchronous JavaScript and XML) to load additional content onto a web page asynchronously. It may also modify existing content on a web page, such as enabling or disabling content when you click on particular web elements.

To extract data correctly, Content Grabber needs to detect any dynamic changes on a web page. For example, if you want to extract any additional data that AJAX loads

onto a web page, then you'll want to configure Content Grabber to wait on AJAX to finish processing the new content before it can start extracting it.

Content Grabber is excellent at the automatic detection of dynamic changes. However, sometimes JavaScript behaves unusually, and you may need to make adjustments to properly extract dynamic content. For example, Content Grabber can detect when JavaScript completes an AJAX load of dynamic content. But it cannot detect exactly when the JavaScript is done and so it will simply wait for a few milliseconds. If the JavaScript takes an unusually long time to display the dynamic content, you may need to use the timeout feature of Content Grabber to insert a short interval for the JavaScript to display the dynamic content (typically a few additional milliseconds).

Familiarity with JavaScript can make it much easier to configure a web-scraping agent to extract data from dynamic websites when Content Grabber is unable configure the agent automatically. You can learn more from various JavaScript tutorials available on the web, such as:

<http://www.w3schools.com/js/>

1.5.3 XPath and Selection Techniques

Proper selection technique is a critical aspect of web-scraping. The most basic selection technique is to point-and-click on elements in the web browser panel, which is the easiest way to add commands to an agent. **XPath** is a common syntax for selecting elements in HTML and XML documents. Each time you click on an element in the web browser, Content Grabber works in the background to calculate the selection **XPath**.

Content Grabber has a variety of tools that help you create precise **XPaths** without needing to know **XPath** syntax at all. If you want finer control over element selection, it's worthwhile to learn some specifics about **XPath** syntax. Although this user guide

doesn't include an **XPath** tutorial, there are many tutorials available on the web. We recommend this as a good place to learn more about **XPath**:

www.w3schools.com/XPath/xpath_syntax.asp

1.5.4 Regular Expressions

With **Regular Expressions**, you can write expressions that look for specific character sequences within strings and then extract small text strings out of larger ones.

Content Grabber uses **XPath** to select web elements on a web page, and then extracts content from those web elements. You may only want some parts of the content extraction, or you may want to transform it. For example, a single web element may contain the entire address of a company, but you may want to extract the content into separate elements such as street address, city, zip code and state. You can use **Regular Expressions** to split the address text into separate text strings.

There are many tutorial websites that teach **Regular Expressions**. Here is one example:

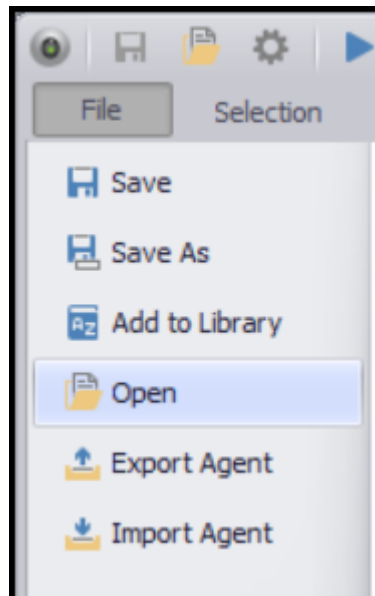
<http://www.regular-expressions.info/reference.html>

1.6 Converting Visual Web Ripper Projects

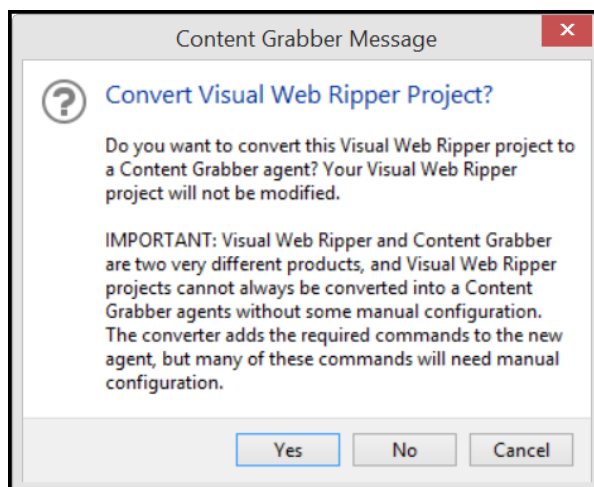
Visual Web Ripper is another web scraping tool published by Sequentum. Content Grabber can open Visual Web Ripper projects and convert them into Content Grabber agents. This is **NOT** a fully automated conversion, and many agents will need manual adjustments after conversion.

To convert a Visual Web Ripper project, simply open the project file in Content Grabber as if it was a normal Content Grabber agent. Content Grabber will then ask if you want to convert the

project file into an agent. The Visual Web Ripper project will be left intact, so you don't need to make a copy of the Visual Web Ripper project.



First open the Visual Web Ripper project as if it was a normal agent.



You will be asked to convert the Visual Web Ripper project to an agent.

Content Grabber will attempt to add all required commands to the converted agent, but it will not be able to set all command

properties correctly, so we recommend you test all commands in a converted agent to make sure they work correctly.

The following list includes features that will NOT be converted from a Visual Web Ripper project. This is not a complete list, and there may be other aspects of a project that will not be converted correctly.

- Visual Web Ripper scripts will not be converted, except for very simple Content Transformation scripts.
- OleDb and Oracle database connections will not be converted. All other database connections will be converted into shared connections.
- The Private Proxy Switch option is not available in Content Grabber.
- PAC Proxy configuration will not be converted.
- Most Page Transformation elements will require post configuration.
- The Tag Name attribute is not available in Content Grabber.
- The property **SaveDataMethod** will require post configuration when set to anything other than **Default**.
- Many Back templates will require post configuration.
- Multiple Page Navigation templates working together will require post configuration.
- Page Navigation templates using Dynamic Links or List of Links will sometimes require post configuration
- All actions will be converted into actions with the property **Detect Action** set to true. All other action configuration, including wait elements, will be ignored.
- Form Field elements using the **Start Index** and **Count** properties will often require post configuration.
- Form Field Lookup Data Sources will not be converted.

- Project user agent settings will not be converted.
- CAPTCHA configuration will not be converted.
- Document conversion will not be converted.
- Document download on form submit will not be converted.
- Scheduling configuration will not be converted.
- Notification configuration will not be converted.
- Duplicate checks will sometimes require post configuration
- Many rarely used options in projects, templates and contents don't have corresponding options in Content Grabber and will therefore not be converted.

1.7 Upgrading to Content Grabber 2

Content Grabber 2 can run alongside Content Grabber 1 on the same computer using the same license. This makes it easier to upgrade from Content Grabber 1 to Content Grabber 2, since new agents can be built in and run in Content Grabber 2, while existing agents can keep running in Content Grabber 1 until they have been converted to Content Grabber 2.

Content Grabber 2 is based on the Chrome browser engine, while Content Grabber 1 is based on Internet Explorer, so some websites will display differently in the two versions, and an agent created in Content Grabber 1 may need to be modified to work properly in Content Grabber 2. However, most Content Grabber 1 agents will work in Content Grabber 2 without any modifications required.

Some Content Grabber 1 agents using specialized action configurations will need to be modified to work properly in Content Grabber 2, since Activities have been removed and replaced with simpler options.

Some Content Grabber 1 agents using specialized export configurations will need to be modified to work properly in Content Grabber 2, since export options have changed so they now work on the commands where the options are configured, rather than sub-commands. Content Grabber 2 will automatically make most required configuration changes to a Content Grabber 1 agent, but may not be able to get the configuration completely right.

When upgrading a Content Grabber 1 agent to Content Grabber2, it's strongly recommended to save the converted agent to a new location, so the old agent is not overwritten. The default agent location for Content Grabber 2 is **My Documents\Content Grabber 2\Agents**.

Content Grabber 2 includes the following new features and changes.

Web Browser

- **Chrome based web browser**

- The embedded web browser is now based on Chrome and is completely self-contained, so no existing web browser is required on the computer where Content Grabber is running, and agent behavior no longer depends on existing browser configuration on the computer.

- **Edge Selections**

- Click on the edge of a web element to select its parent. This is particular convenient when selecting web elements such as table rows, which cannot be selected directly in the web browser because they are completely covered by table cells.

Agents

- **Data retention**

- New data retention options. Old data can now be deleted, kept and exported, or kept for duplicate checks only. New default duplicate scripts can be used to copy old data to the current data set, so you end up with a complete current data set. Previously, using duplicate scripts would always make it impossible to know when data was deleted from the target website.

- **Change tracking**

- An agent can now keep track of the latest changes that have been made to extracted data. The agent will mark extracted data as deleted, modified or added. If data is deleted but later returned, the data will be marked as "returned", or "returned modified" if the data returned in a modified state. An agent can be configured to only export data changes since last successful run, or since a specified number of days.
- Data will only be marked as deleted if an agent run successfully completes. This prevents data from being incorrectly marked as deleted if an agent fails halfway through a run.

- **Success criteria**

- Success criteria can now be defined to control when an agent run is considered successfully completed. Success criteria can be used to control notifications and change tracking.

- **Export to single table**

- Multiple agents can now export to a single database table.

- **Export script templates**

- Script templates make it easier to build export scripts for agents. A script template will contain the C# code required

to read all data from an agent, so all that needs added is the code required to write the data.

- **Export settings**

- Export Settings have been simplified. Data exported from a container can now be separated by setting a single option on the container.
- Exported data from list containers is now never separated by default, which should make the export data format more predictable.
- The export option to convert data rows into data columns is now only available for CSV and Excel export. The conversion is done when the CSV or Excel files are generated, so the internal data structures never reflect this conversion. If a container command has been configured to convert data rows into data columns, the internal data structures will contain a separate data table for the data extracted by that container command.

- **File downloads**

- More reliable file downloads when using "Click to Download".

- **Screenshots**

- Agents running in a service are now able to take screenshots.

- **Retry errors**

- After an agent stops, the agent can now continue data extraction, continue data extraction and retry errors, or just retry errors.
- The current version of an agent can now be used when continuing data extraction or retrying errors. The current version of the agent can differ from the version at the time the data extraction started. An error occurs if the agent has been modified in a way that requires a new internal data

structure. It's still possible to use a version of the agent that was current when the data extraction started.

User Interface

- **Simplified action configuration**

- Activities have been replaced with simple options to configure an action command to wait for page loads and/or AJAX. An agent can also wait for a specific URL or a specific selection XPath.

- **Multiple command selections**

- Multiple commands can be selected in the Agent Explorer to make it easier to delete, copy or move multiple commands.

- **Group commands**

- A new Group Commands feature allows all selected commands to be placed inside a Group Command with a single click.

- **Improved XPath editor**

- The XPath editor now has color highlighting and has been moved from the Ribbon menu to a docking window.

Self-Contained Agents

- **New self-contained agents**

- The user interface for self-contained agents is now pure HTML, and can be completely customized with the Premium edition of Content Grabber.

2 Quick Start with Content Grabber

We want to help you get up and running, so this chapter outlines the basics to help you quickly install the software, build a simple agent, and then test that agent. Further along in the guide, we explain more details and explore the advanced features. For now, let's see how easy it is to build an agent.

In this chapter, you'll learn:

- Installing Content Grabber
- Content Grabber Basics
- Exploring the Main Window
- Building Your First Agent



2.1 Installing Content Grabber

Content Grabber is a Windows-based software application. Windows 7, Windows 8, Windows 10, Windows 2008 R2, Windows 2012 and Windows 2012R2 are the supported versions of Windows. While a Macintosh version is not available, a number of our users utilize Parallel's windows emulation software to successfully run Content Grabber.

Installing Content Grabber is easy. Simply follow these steps:

1. Download the setup file from the Content Grabber website.
2. Find the **setup.zip** file on your local disk (likely to be your **Downloads** folder).
3. Open the zip file and then double-click the **setup.exe** file to launch it.
4. After a moment, you'll see the first panel of the Content Grabber setup wizard. Click the **Next** button to continue with the installation.



5. Click the option to **Accept the Agreement**, and then click the **Next** button.
6. In the next panel, you can change the installation location. If you accept the default location, simply click **Next**. Otherwise, click the **Browse** button to choose a different installation folder, and then choose the folder and click the **Open** button to close the pop-up, and then click the **Next** button.
7. In the next panel you can change the name of the folder that will appear in the **Windows Start Menu**. Click **Next** and, in the next panel, click the **Install** button to complete the installation.
8. Locate the Content Grabber item in the **Windows Start Menu** and then launch it.

Installing Flash

Flash is disabled in Content Grabber by default, and we recommend you keep it that way unless you have a specific need to run Flash. If a website interacts with a Flash component

and is unable to display correctly without Flash, then you may need to install Flash, but that would be a very rare scenario.

To install Flash you must download the NAPI Flash player from the Adobe website.

<https://get.adobe.com/flashplayer/otherversions/>

After installing the NAPI Flash player, you must enable Flash in Content Grabber. You can do that from the Application menu by clicking **Advanced Settings** and then set the option **Enable Flash** to true. You will need to restart Content Grabber before this change takes effect.

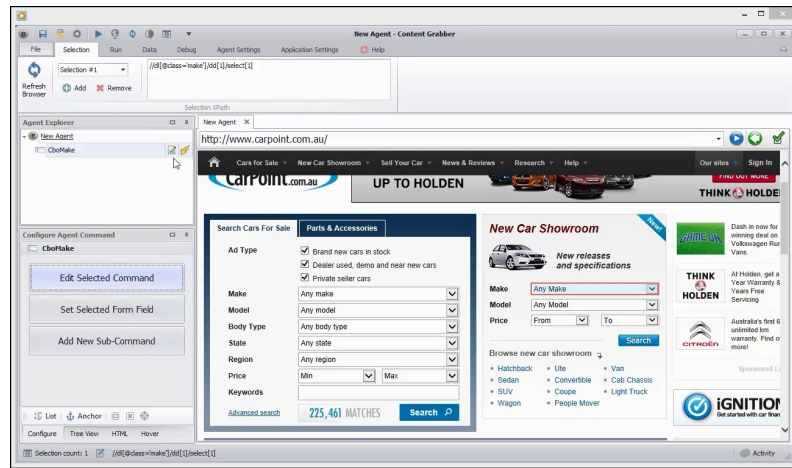
2.2 Content Grabber Basics

Web-scraping tools generally use macros or configuration methods, and follow a sequential list of commands. The macro approach is more user friendly and automatically records the actions of a user in a browser. However, there are typically restrictions on accessing the code behind the agent. The configuration approach allows the user to directly configure each part of the agent. They can introduce more code structure, controls, data refinements, or add their own naming conventions.

Content Grabber gives you the option to either follow the simple macro automation methods, or to take direct control over the treatment of each element and command within your agent.

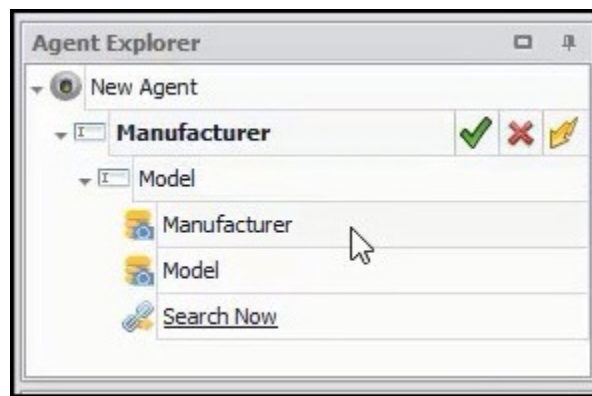
Content Grabber Agent Development

With Content Grabber, you can visually browse the website and click on the data elements in the order that you want to collect them. Based on the content elements selected, Content Grabber will automatically determine the corresponding action type and provide default names for each command as it builds the agent for you.



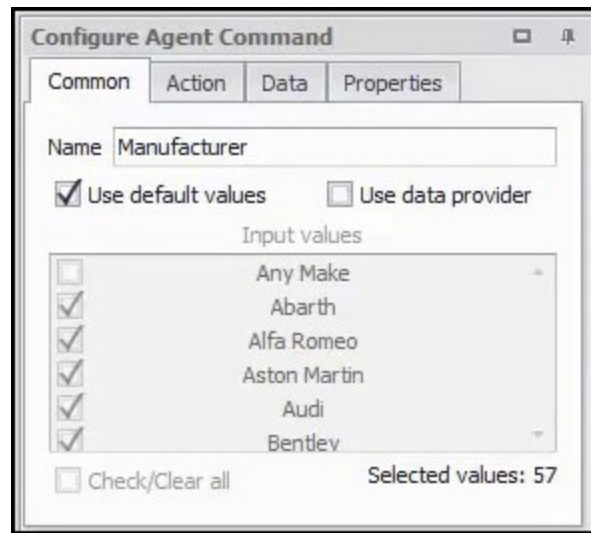
Content Grabber main screen - building CarPoint Agent

A Content Grabber agent is a collection of commands which are executed in serial until completed. The commands can either be actions (such as a jump to a URL) or data capture commands (e.g. capture text). These commands are recorded in order of execution in the **Agent Explorer** panel of the Content Grabber screen.



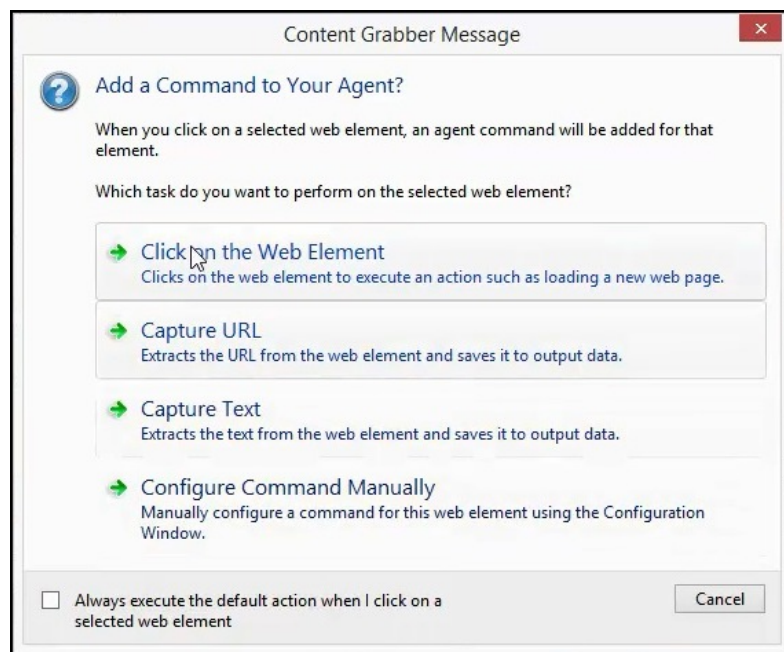
Agent Explorer panel with New Agent commands

If you want to make other adjustments or gain more control of your commands, you can make changes in the **Configure Agent Command** panel.



Configure Agent Command panel

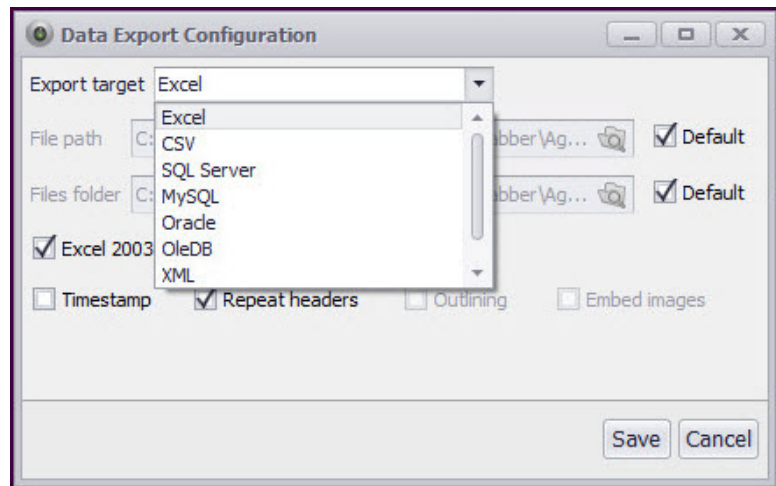
You can also add new commands to your agent, or configure existing ones. To do this, you simply click twice on any web element (content item) and the **Content Grabber Message** window will appear. From here you can select the command type you want and add it to the **Agent Explorer**.



Content Grabber Message window pop-up

Content Grabber Data Outputs

After you have finished building your Agent and run it for the first time, Content Grabber saves the data locally in a structured database format. Content Grabber can export extracted web data as a report or to numerous different database types. Data output options include CSV, Excel, XML, SQL Server, MySQL, Oracle and OLEDB.

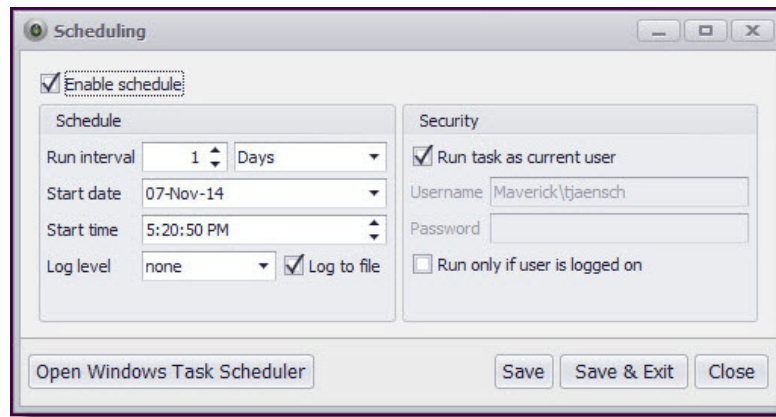


Content Grabber's Data Configuration window

You can also use a Content Grabber export script to completely customize the data export to your own database structures.

Scheduling

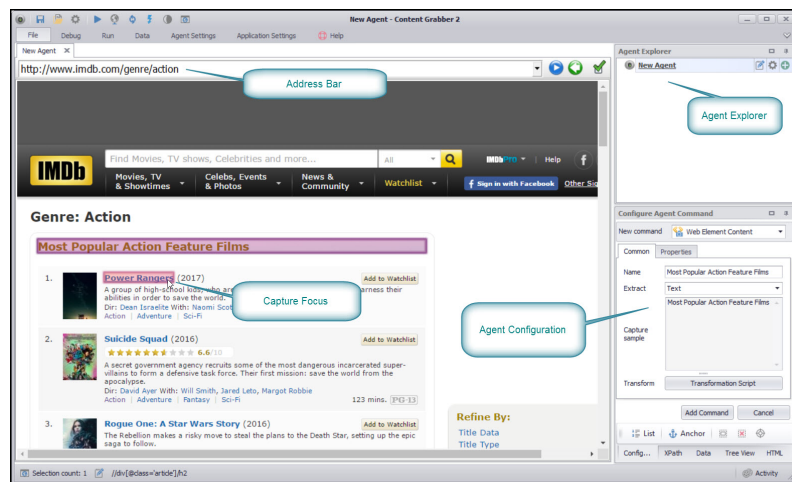
Content Grabber provides an Agent scheduling facility that enables you to automatically run your agent at predetermined time slots whenever you need it to run. This can be done every hour, every day, month, year and so on.



Content Grabber's Scheduling Window

2.3 Exploring the Main Window

In this section, we give a brief overview of the Content Grabber window.



Address Bar

The **Address Bar** is where you enter the URL of the page that is the start of your web-scraping agent. This is what we call the **Start URL**.

Web browser Panel with Data-Capture Box

The Content Grabber web browser appears in the main panel. The special feature of this integral web browser is this: as you move the mouse and pass over a web element, you'll see a yellow rectangle envelop that element. When you stop over one

element and click on it *twice*, the **Command** pop-up window will appear—prompting you to choose the command that you want to insert into the agent for this data element.

Agent Explorer Panel

As you select data elements in the browser panel, the corresponding **Command** item will appear in the **Agent Explorer** panel. The Agent Explorer is used to view agent commands for the type of web page loaded in the selected browser tab. When you select a new browser tab with another web page, a new set of commands will be displayed in the Agent Explorer.

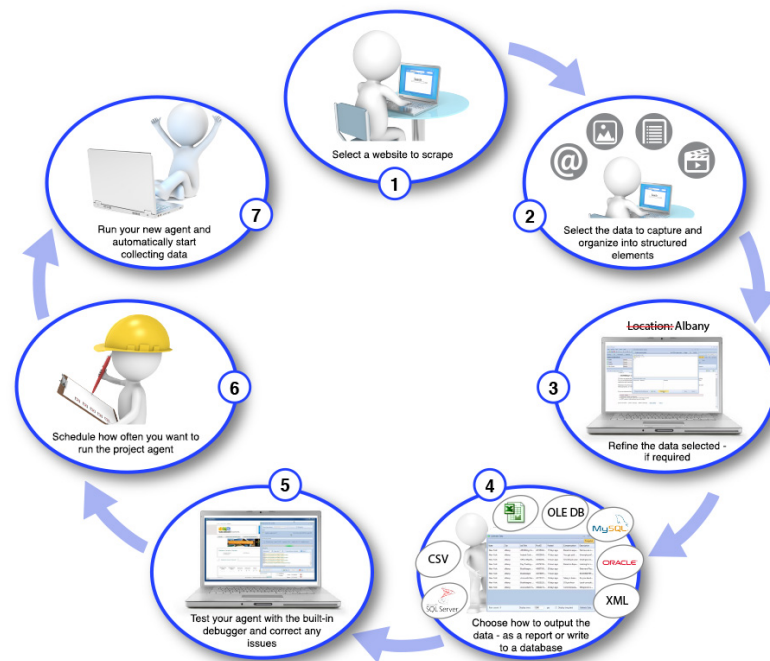
You can use the Agent Explorer to view, edit or execute commands.

Agent Configuration Panel

In this panel, you can edit an agent command, execute a command, or add a sub-command. Further along in this guide, we'll explain how to explore and edit each command.

2.4 Building Your First Agent

The diagram shows the key steps for building a web scraping agent. Below we provide links to the other topics which explain each of these steps in more detail.



Basic web scraping Agent creation process

Here in this section, we cover the identification of data elements on your target website and create a web-scraping agent. We'll work through each of the above steps with examples that match common web-scraping usage, so you can get comfortable building your own agents.

We cover these topics in this section:

- Choosing a Start URL
- Select the Data to Capture
- Refine your data (optional)
- Output Data Format
- Test Your Agent
- Scheduling
- Run your Agent

After you get comfortable creating an agent, you can learn more about the Content Grabber Editor.

2.4.1 Choosing a Start URL

The **Start URL** is the place where you begin data collection and corresponds to the starting point for your web-scraping agent.

In the following sections, we'll use the Cruise Direct website for our example.

<http://www.cruisedirect.com>

Note: In this example we start from the Cruise Direct home page, however, if the data you require is not located on a website home page, you can start the agent from a website sub-page. This approach will make the agent more efficient, so it's worth taking the time to be more specific.

1. We start by pasting the start web page url from the target website (<http://www.cruisedirect.com>) into **Content Grabber's Address Bar**.
2. Next click the Blue Play button to load the Cruise Direct home page.

Note: you can also press the Enter key to load the Cruise Direct home page.



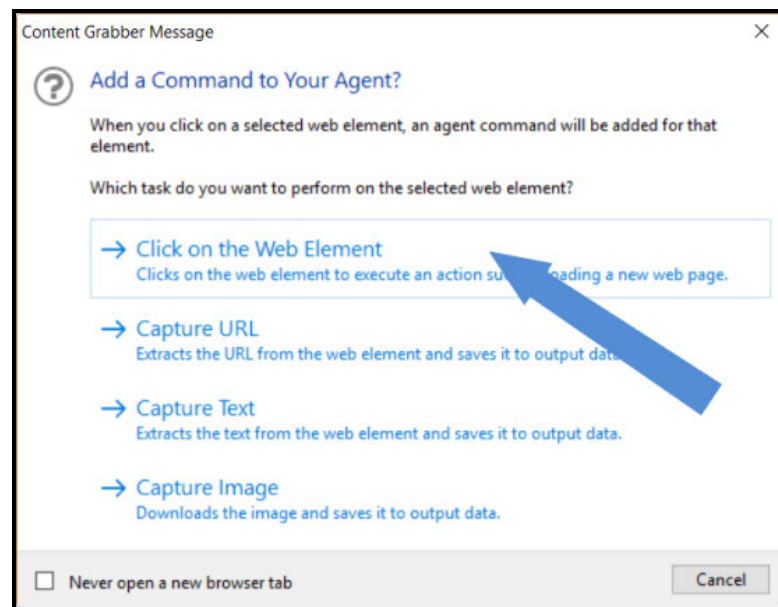
Content Grabber with Cruise Direct home page loaded

In the next section, Select the Content to Capture, we will continue to use the Cruise Direct website data for our example.

2.4.2 Select the Content to Capture

In the previous section, we selected our **Start URL** and loaded the web page into Content Grabber. Next, you can select the data you want to capture and start building your web scraping agent. In our Cruise Direct example, we plan to search for available cruise vacations and then extract details about each cruise.

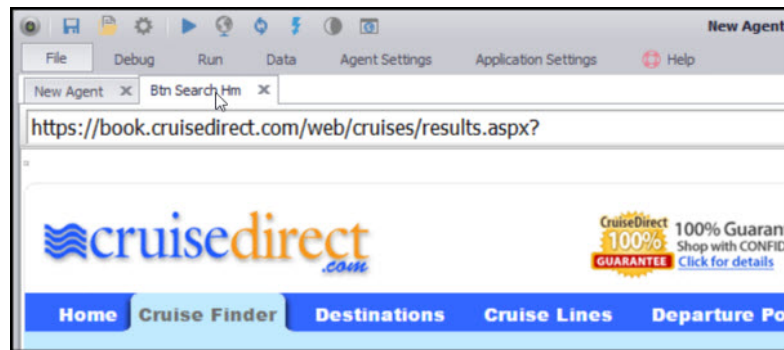
1. Firstly we need to perform a search to retrieve the data for the available cruises. To do this, we select the orange **Search** button element with the mouse, then click one more time to display the **Content Grabber Message** window.



Content Grabber Message Window

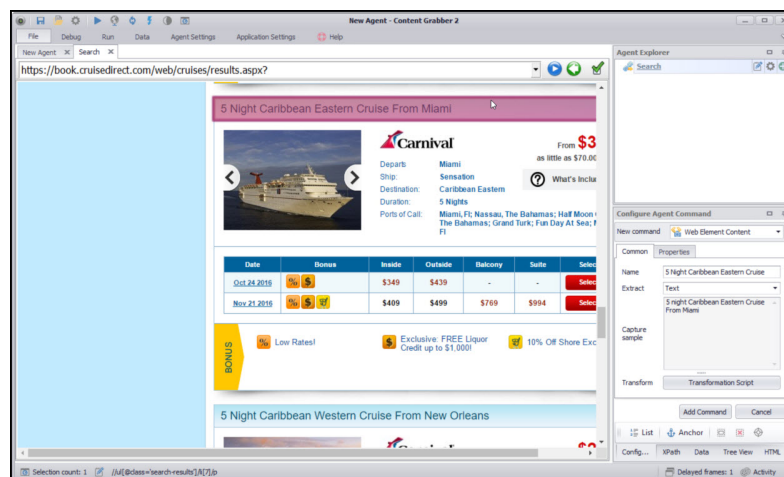
2. From the message window, we choose the **Click on the Web Element** option to add a new command to the agent that will execute the search and display the search results on a new web page. Notice that Content Grabber has added our

first command to **Agent Explorer** - in this case to execute the search and display the search results.



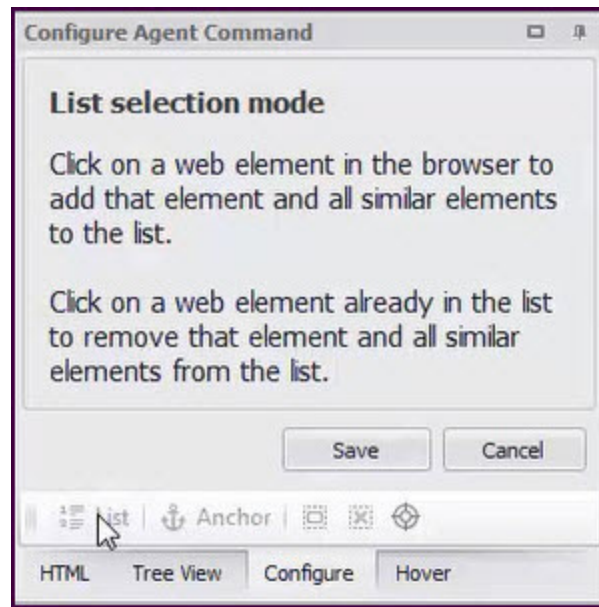
Agent Explorer with new Search command linked to new Search page

3. We are now ready to add commands to our agent to extract the cruise data. As we have a number of data elements in tables, we will use a list to simplify the extraction for us. To capture a data element, move your mouse precisely over the data element you want, until you see the data-capture box around it. We start by selecting the first cruise name.



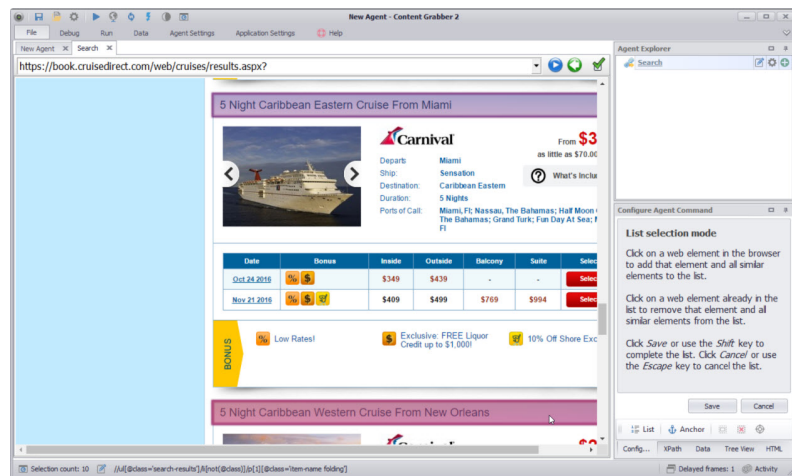
First Cruise Line data element selected within Content Grabber

4. Then click **List** in the **Configure Agent Command** panel to activate the **list selection mode**.



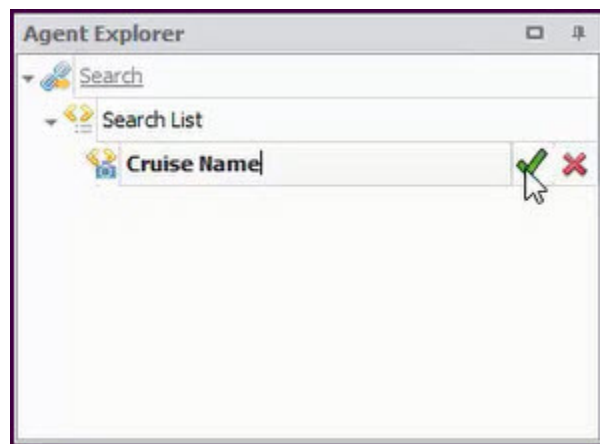
Activating List selection mode from the Configure Agent Command panel

5. In **list selection mode**, we can add web data elements into the list by clicking similar data elements. Now we'll click on the second cruise name and you will see Content Grabber has selected the remaining data elements on the page. Note: if any cruise data elements remain unselected, simply click on these to add to the list.



Second Cruise Line data element selected while in List selection mode

6. We now click **Save** to save the list and exit **list selection mode**. The Web Element list command defines the list area, so any elements within this area are now included within the list.
7. To capture the cruise name text, we click on any selected element to display the **Content Grabber Message** window. From the **Content Grabber Message** window, choose the **Capture Text** option to add the web element command to capture the cruise names. We have now added new web element list and web element commands to the **Agent Explorer** and Content Grabber has set default names for these commands.
8. To edit the names for the commands, we click the respective **Edit** icons and set the names of the commands to 'Search List' and 'Cruise Name'. Then click the **Green Tick** to save.



Agent Explorer with new Search List and Cruise Name commands

9. Now we plan to extract the individual cruise web elements from each table. So firstly click on the **Departs** web element. Content Grabber now automatically selects the **Departs** web element for all cruises because it is already defined as a list.
10. Next, click on the **Departs** web element one more time to display the **Content Grabber Message** window. Now choose the **Capture Text** option from the

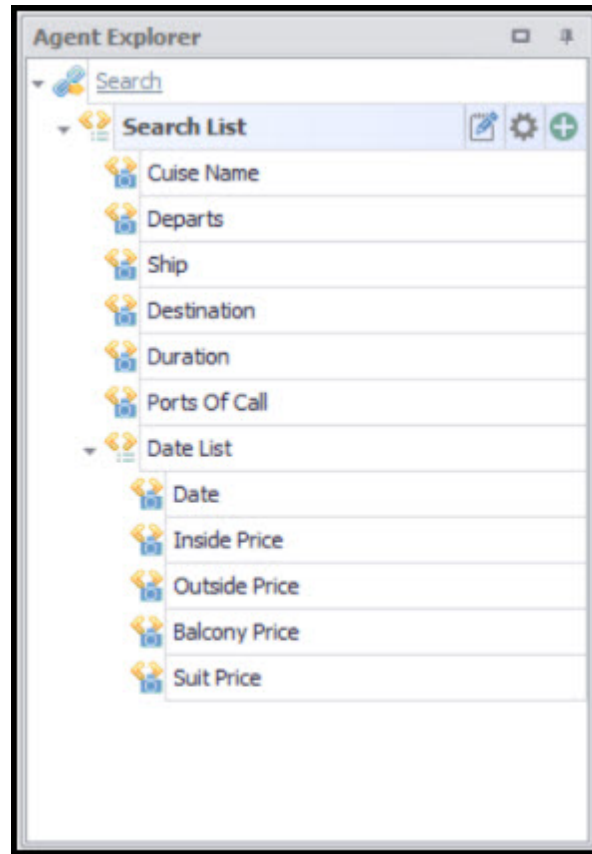
Content Grabber Message window to add the command to the Agent so we can capture the individual **Departs** web elements.

11. After that, we click the **Edit** icon to change the name of the command to "Departs" then save it.
12. Now we do the same for the Ship, Destination, Duration and Ports of Call web elements, then set the respective names of the commands and save them.
13. We also want to capture all the price information in the pricing tables, so as before, we select the first web element (Date) in the pricing table. We then click **List** in the **Configure Agent Command** panel to activate the list selection mode, and click on more **Date** web elements to generate the list.

Date	Bonus	Inside	Outside	Balcony	Suite	Select
Oct 24 2016	% \$ 🍷	\$129	\$149	\$309	-	Select
Oct 31 2016	% \$ 🍷	\$159	\$219	\$349	\$1,199	Select
Nov 21 2016	% \$ 🍷 🍷	\$329	\$379	\$789	\$1,399	Select
Nov 28 2016	% \$ 🍷 🍷	\$249	\$299	\$449	\$1,149	Select
Dec 05 2016	% \$ 🍷 🍷	\$259	\$299	\$549	\$1,829	Select
Dec 12 2016	% \$ 🍷 🍷	\$629	\$649	\$899	\$1,449	Select

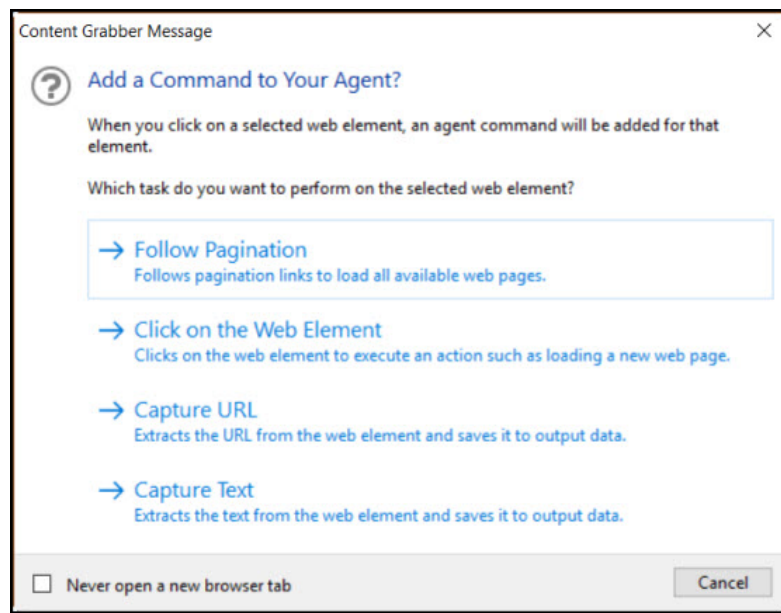
14. Click one more time on one of the Date web elements to display the **Content Grabber Message** Window. Then choose the **Capture Text** option to add the command to the Agent.
15. Add commands for the 'inside', 'outside', 'balcony' and 'suite' web elements by clicking twice on each of the web elements.

16. Change the names of the new commands so your agent looks like the image below.



Agent Explorer showing all Capture Text commands

17. Thus far we have created the Agent to extract all the cruise information on the first page. We need to set it up to iterate through all the search result pages. To do this, we need to use the **Follow Pagination command** to follow each of the pages. Scroll down the page and select the **Next** link. Then click one more time on the selected element to display the **Content Grabber Message** window.

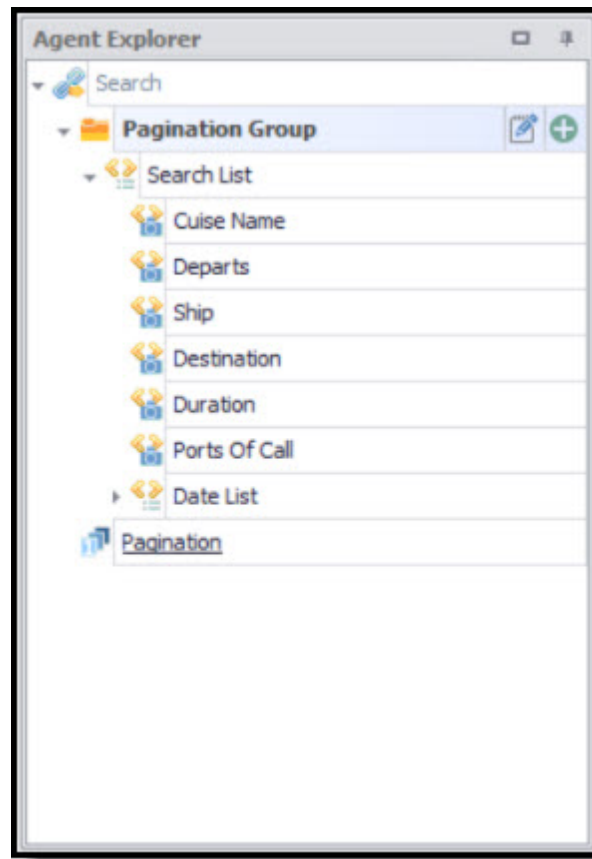


Content Grabber Message windows with Follow Pagination option selected

18. Now we choose the **Follow Pagination** option to add the pagination command to the Agent.

Content Grabber has added the pagination command to the Agent and loads the next page on the second browser tab.

19. When we click on the pagination command we can see all the search list commands inside of the pagination command. This means our agent will now iterate through all the search result pages to extract this information.



Agent Explorer showing the contents of the Pagination command

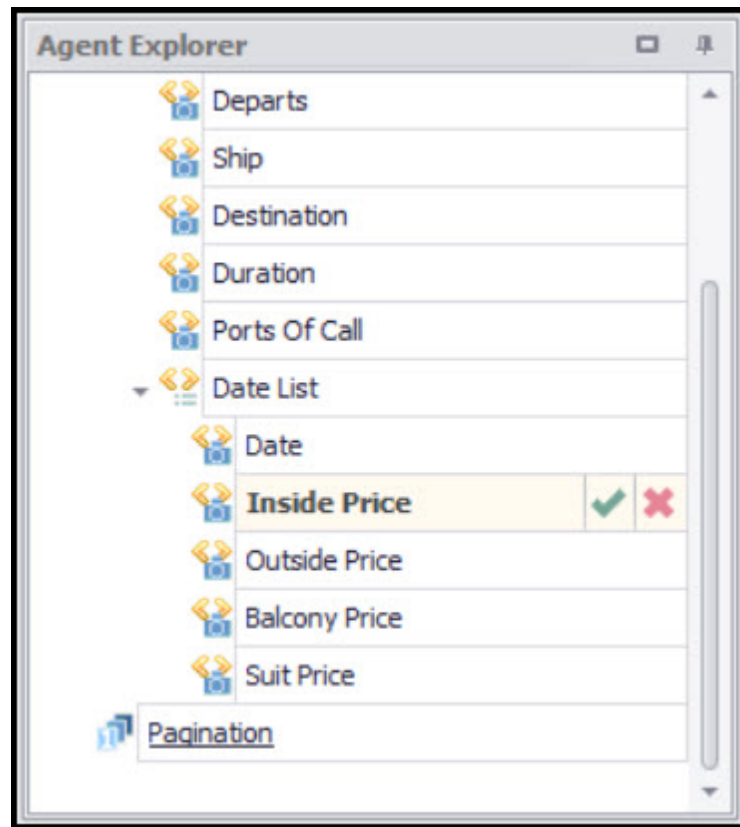
20. We have now finished building the Agent so we should save it. To save the Agent, choose **File > Save** in the **Content Grabber menu**, and then enter the Agent Name "cruisedirect". Then click the **Save** button to commit your changes.

In the next section, Refine Your Data we use Content Grabber's **Content Transformation** method to change the extracted price data.

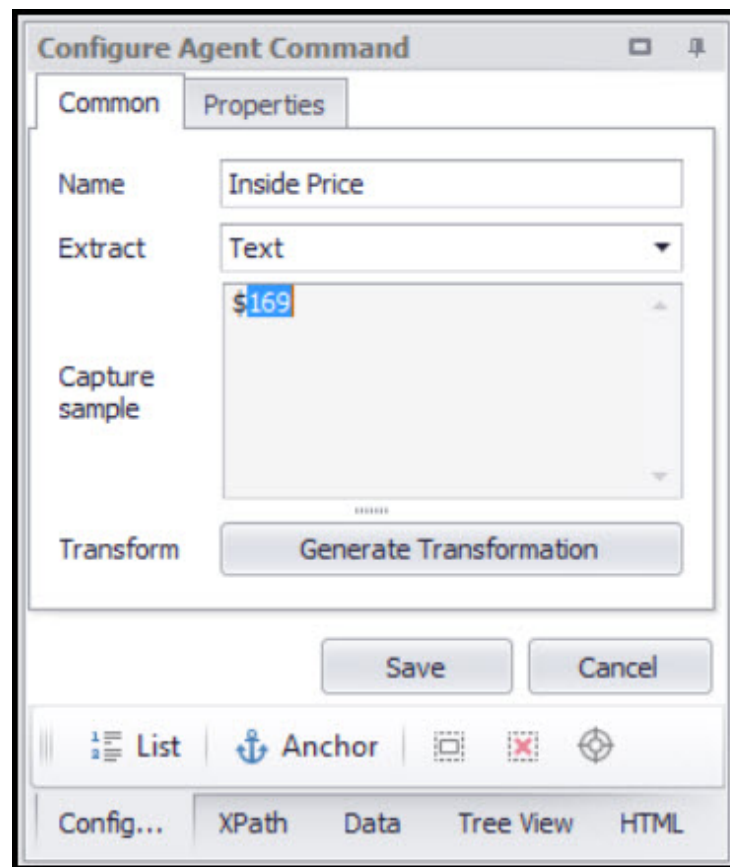
2.4.3 Refine Your Data (optional)

In the previous section, we added commands to our agent to capture all the Cruise price content we require. The prices include a \$ sign, but we want to get rid of that \$ sign, so we're left with a clean number.

1. Firstly edit the "Inside Price" command in the **Agent Explorer**.

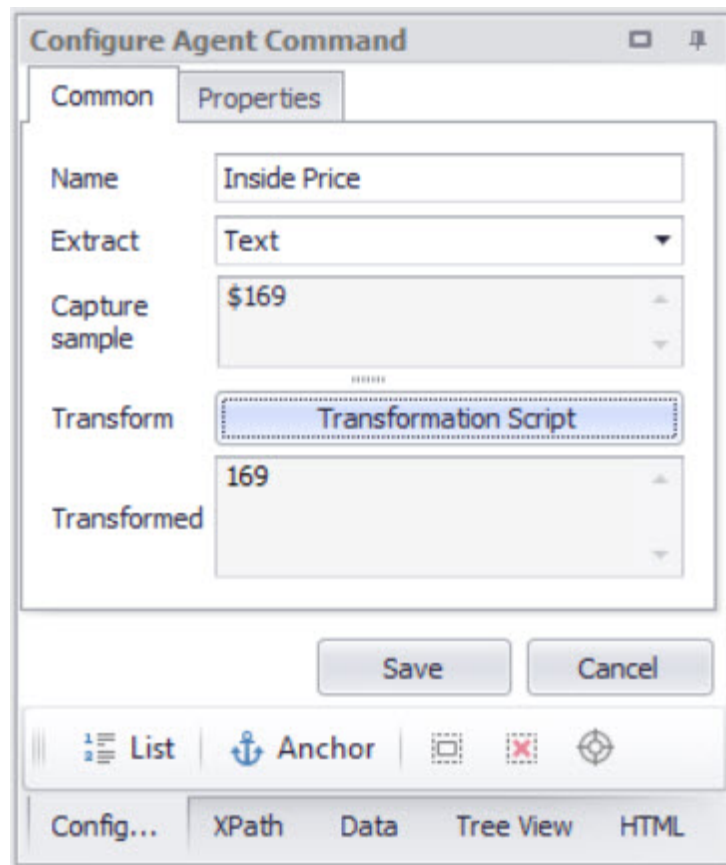


2. Next, we scroll down to the capture sample window, and select only the price number. You should notice that the **Transformation Script** button has now changed to a **Generate Transformation** button.



Selecting text to Transform in the Configure Agent Command panel

3. Now click on the **Generate Transformation** button, and we can now see only the price number in the **Transformed** window.



Transformed text in the Configure Agent Command panel

4. Click **Save** to save the transformation.
5. Repeat the steps above for Outside Price, Balcony Price and Suite Price.

In the next section, Output Data Format we look at the data output formats available for the extracted web data and show how to change and configure a new export target.

2.4.4 Output Data Format

Content Grabber can export extracted web data as a report or to numerous different database types. Data output options include CSV, Excel, XML, SQL Server, MySQL, Oracle and OleDb.

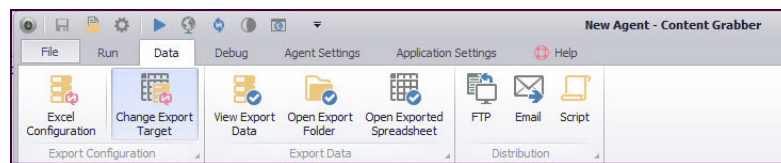
You can also use a Content Grabber export script to completely customize the data export to your own database structures. This is useful when you want the data updates to be dynamically reflected such as an online website / portal.

Content Grabber can export data to Excel 2003+ and take advantage of features in Excel 2007+ such as outlining and embedded images.

Data is exported automatically to your chosen export destination when a data extraction project completes, so you don't have to export data manually. However, you can always export extracted data manually at any time to any export destination.

The following steps show how easy it is to choose the data export type you want to use.

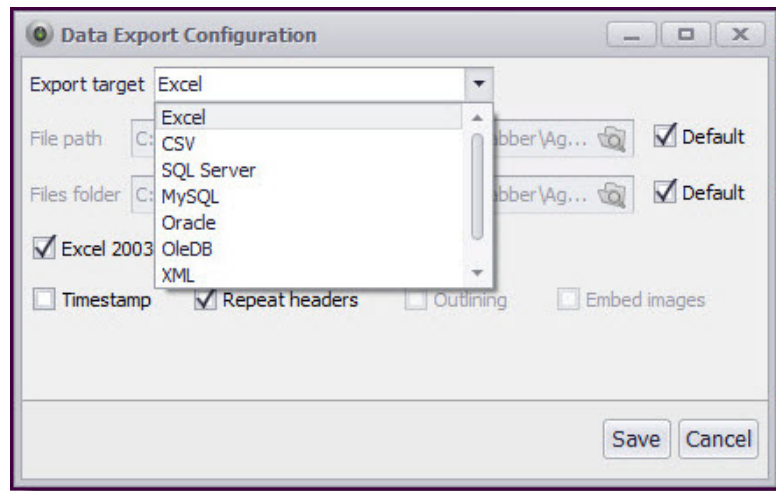
1. We start by clicking on Content Grabber's **Data menu** and then clicking the **Change Export Target** button.



Content Grabber's Data menu

2. After clicking the **Change Export Target** button the **Data Configuration Window** displays. This window allows you to change and configure a new export target. The default option is Excel 2003.

Click the **Export target** drop-down list box to display the different report and database export options available. Then choose the format you want to use. You can also change the default destination folder location for where the data file(s) are output.



Content Grabber's Data Configuration window

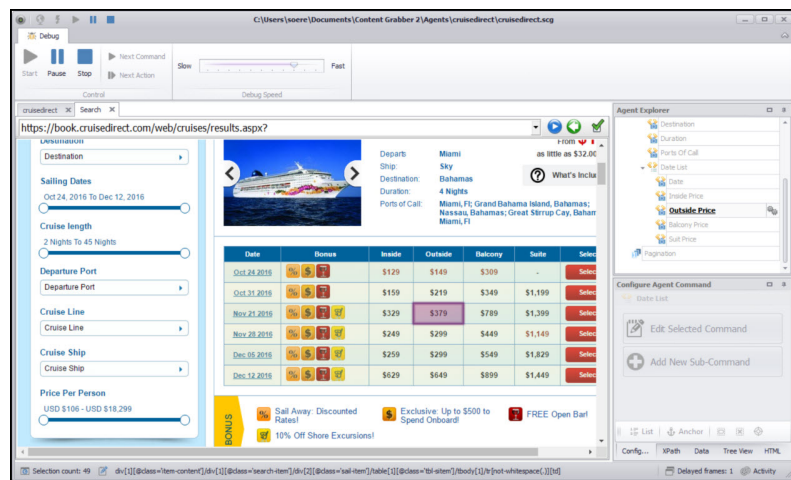
In the next section, Test Your Agent we use Content Grabber's Debugging features to test the "cruisedata" Agent is functioning as expected.

2.4.5 Test Your Agent

Once you have finished developing your agent, it is important to test it to ensure the correct data is being extracted and in the format you require. Content Grabber has a sophisticated debugging engine which enables you to carefully analyze all aspects of your agent and the data being extracted. It can also help you to pin-point any trouble spots in your agent code so you can quickly resolve them. For more detail on the Debugger features available in Content Grabber refer to Testing/Debugging an Agent.

Now let's run the Cruise Direct agent to check the commands are working properly.

To do the test run, [click the 'Debug' menu option at the top left of the screen and then the click 'Start' arrow button to start the debugging.](#)



Content Grabber Debugger running the Cruise Direct Agent

During the debugging, we can observe and check that Content Grabber goes through each of the commands in sequence and processes each of the web pages to extract the required data.

Part way through running the agent, we will click the 'Stop' button to stop the debugging. Then click 'View Export Data' to check the output results are correct.

Cruise Name	Departs	Ship	Destination	Duration	Ports Of Call	Date	Inside Price	Outside Price	Balcony Price	Suit Price
4 night Bahama...	Miami	Sky	Bahamas	4 Nights	Miami, FL Gran...	Oct 24 2016	\$129	\$149	\$309	-
4 night Bahama...	Miami	Sky	Bahamas	4 Nights	Miami, FL Gran...	Oct 31 2016	\$159	\$219	\$349	\$1,199
4 night Bahama...	Miami	Sky	Bahamas	4 Nights	Miami, FL Gran...	Nov 21 2016	\$329	\$379	\$789	\$1,399
4 night Bahama...	Miami	Sky	Bahamas	4 Nights	Miami, FL Gran...	Nov 28 2016	\$249	\$299	\$449	\$1,149
4 night Bahama...	Miami	Sky	Bahamas	4 Nights	Miami, FL Gran...	Dec 05 2016	\$259	\$299	\$549	\$1,829
4 night Bahama...	Miami	Sky	Bahamas	4 Nights	Miami, FL Gran...	Dec 12 2016	\$629	\$649	\$899	\$1,449
4 night Bahama...	Miami	Enchantment o...	Bahamas	4 Nights	Miami - Florida...	Oct 24 2016	\$199	\$219	-	-
4 night Bahama...	Miami	Enchantment o...	Bahamas	4 Nights	Miami - Florida...	Oct 31 2016	\$199	\$229	\$369	\$498
4 night Bahama...	Miami	Enchantment o...	Bahamas	4 Nights	Miami - Florida...	Nov 07 2016	\$329	\$379	\$578	-
4 night Bahama...	Miami	Enchantment o...	Bahamas	4 Nights	Miami - Florida...	Nov 14 2016	\$219	\$249	\$428	\$499
4 night Bahama...	Miami	Enchantment o...	Bahamas	4 Nights	Miami - Florida...	Nov 28 2016	\$201	\$258	\$441	\$598
4 night Bahama...	Miami	Enchantment o...	Bahamas	4 Nights	Miami - Florida...	Dec 05 2016	\$187	\$237	\$394	\$575
4 night Bahama...	Miami	Enchantment o...	Bahamas	4 Nights	Miami - Florida...	Dec 12 2016	\$192	\$245	\$457	\$595
4 night Bahama...	Port Canaveral	Dream	Bahamas	4 Nights	PORT CANAVER...	Oct 24 2016	\$660	\$904	\$740	\$1,820
4 night Bahama...	Port Canaveral	Majesty of the ...	Bahamas	4 Nights	Orlando (Port C...	Oct 24 2016	\$208	\$236	-	-
4 night Bahama...	Port Canaveral	Majesty of the ...	Bahamas	4 Nights	Orlando (Port C...	Oct 31 2016	\$169	\$169	-	-
4 night Bahama...	Port Canaveral	Majesty of the ...	Bahamas	4 Nights	Orlando (Port C...	Nov 14 2016	\$199	\$209	-	\$642
4 night Bahama...	Port Canaveral	Majesty of the ...	Bahamas	4 Nights	Orlando (Port C...	Nov 21 2016	\$286	\$337	-	-
4 night Bahama...	Port Canaveral	Majesty of the ...	Bahamas	4 Nights	Orlando (Port C...	Nov 28 2016	\$136	\$181	-	\$544
4 night Bahama...	Port Canaveral	Majesty of the ...	Bahamas	4 Nights	Orlando (Port C...	Dec 05 2016	\$176	\$187	-	\$1,089

Content Grabber's default view of the export data

To see the export data in Excel, we can simply click on the 'Open Exported Spreadsheet' button to open the Excel spreadsheet.

	A	B	C	D	E	F	G	H
	Cruise Name	Departs	Ship	Destination	Duration	Ports Of Call	Date	Inside Price
1	4 night Bahamas Cruise From Miami	Miami	Sky	Bahamas	4 Nights	Miami, FL; Grand Bahama Island, Bahamas; Nassau, Bahamas; Great Stirrup Cay, Bahamas; Miami, FL	Oct 24 2016	\$129
2	4 night Bahamas Cruise From Miami	Miami	Sky	Bahamas	4 Nights	Miami, FL; Grand Bahama Island, Bahamas; Nassau, Bahamas; Great Stirrup Cay, Bahamas; Miami, FL	Oct 31 2016	\$159
3	4 night Bahamas Cruise From Miami	Miami	Sky	Bahamas	4 Nights	Miami, FL; Grand Bahama Island, Bahamas; Nassau, Bahamas; Great Stirrup Cay, Bahamas; Miami, FL	Nov 21 2016	\$329
4	4 night Bahamas Cruise From Miami	Miami	Sky	Bahamas	4 Nights	Miami, FL; Grand Bahama Island, Bahamas; Nassau, Bahamas; Great Stirrup Cay, Bahamas; Miami, FL	Nov 28 2016	\$249

Viewing the Cruise Direct export results in an Excel Spreadsheet

The extracted data contains one row for each departure date, but you could also chose to save the date and price information in a separate data table. The data would then look like the image below when exported to Excel.

	A	B	C	D	E	F	G	H	I
	Cruise Name	Departs	Ship	Destination	Duration	Ports Of Call			
1	4 night Bahamas Cruise From Miami	Miami	Sky	Bahamas	4 Nights	Miami, FL; Grand Bahama Island, Bahamas; Nassau, Bahamas; Great Stirrup Cay, Bahamas; Miami, FL			
2									
3	cruisedirect Date List	Date	Inside Price	Outside Price	Balcony Price	Suit Price			
4		Oct 24 2016	\$129	\$149	\$309	-			
5		Oct 31 2016	\$159	\$219	\$349	\$1,199			
6		Nov 21 2016	\$329	\$379	\$789	\$1,399			
7		Nov 28 2016	\$249	\$299	\$449	\$1,149			
8		Dec 05 2016	\$259	\$299	\$549	\$1,829			
9		Dec 12 2016	\$629	\$649	\$899	\$1,449			
10	Cruise Name	Departs	Ship	Destination	Duration	Ports Of Call			
11	4 night Bahamas Cruise From Miami	Miami	Enchantment of the Seas	Bahamas	4 Nights	Miami - Florida; Nassau - Bahamas; Cococay - Bahamas; Key West - Florida; Miami - Florida			
12	cruisedirect Date List	Date	Inside Price	Outside Price	Balcony Price	Suit Price			

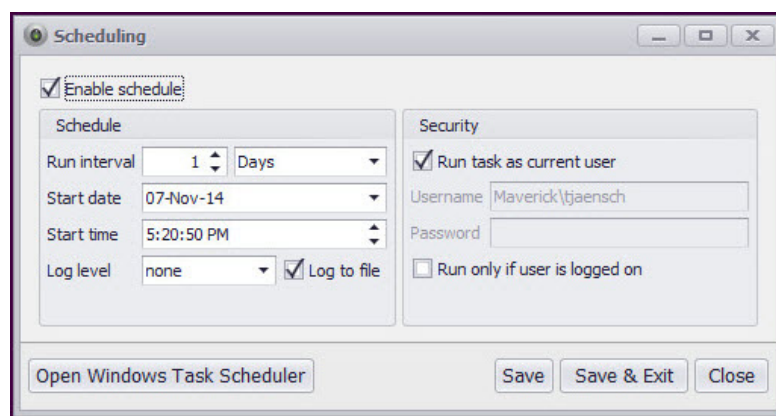
Viewing the Cruise Direct export results in an Excel Spreadsheet

In the next section, Scheduling you will learn how to set up an agent so it can be automatically run at intervals of your choosing.

2.4.6 Scheduling

Content Grabber provides an Agent scheduling facility that enables you to automatically run your agent at predetermined time slots whenever you need it to run. This can be done every hour, every day, month, year and so on.

Once you have completed your agent, you will be able to access this facility. From the **Agent Settings** menu at the top of the Content Grabber application. Simply Click on the **Schedule** menu option to display the Scheduling window.



Content Grabber's Scheduling Window

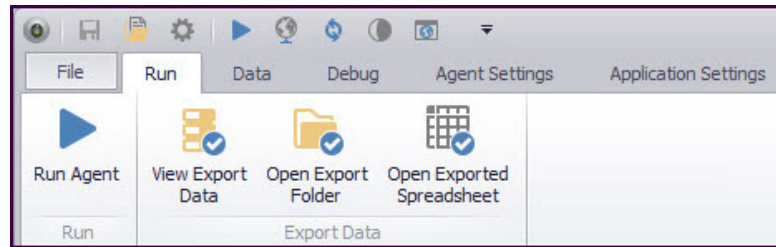
For details on how to configure Content Grabber's Scheduling feature, or if you'd like to learn how to use **Windows Task Scheduler** with your Agents, refer to Scheduling in the Content Grabber Editor section.

In the next section, Run Your Agent we will show you how you run your finished web-scraping Agent.

2.4.7 Run Your Agent

Now that we have finished developing and testing the Cruise Direct Agent it is ready to use.

To run your agent, you simply click on the **Run** menu at the top left of the Content Grabber application then click the **Run Agent** arrow selection.



Content Grabber's Run menu selections

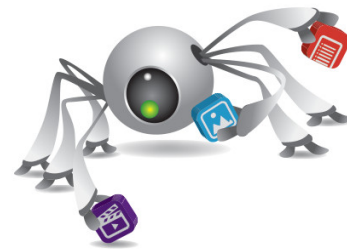
Note: if you have already scheduled your Agent to run at a later time or date, you can just leave it on your Internet enabled PC or Server and it will run automatically. Refer to Scheduling for more detail.

3 Content Grabber Editor

The Content Grabber Editor enables you to easily see and manipulate your web scraping agent as you build it. This section of the manual looks at various aspects of the Content Grabber interface to help ensure you understand and get the most out of the Content Grabber application.

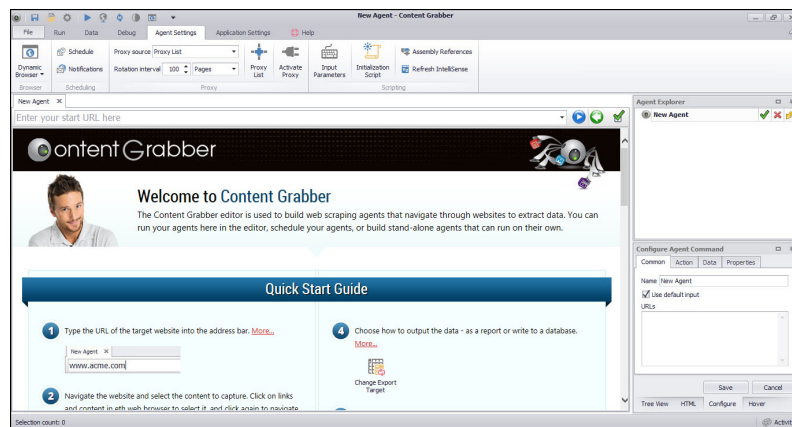
In this chapter, you'll learn:

- Customizing the Editor Layout
- Working With the Web Browser
- Command Configuration
- Selection Tools and Short-cut Keys
- Navigating an Agent
- Testing/Debugging an Agent
- Scheduling
- Copying & Moving Agents
- Exporting & Importing Agents
- Automated Agent Backups
- Template Libraries



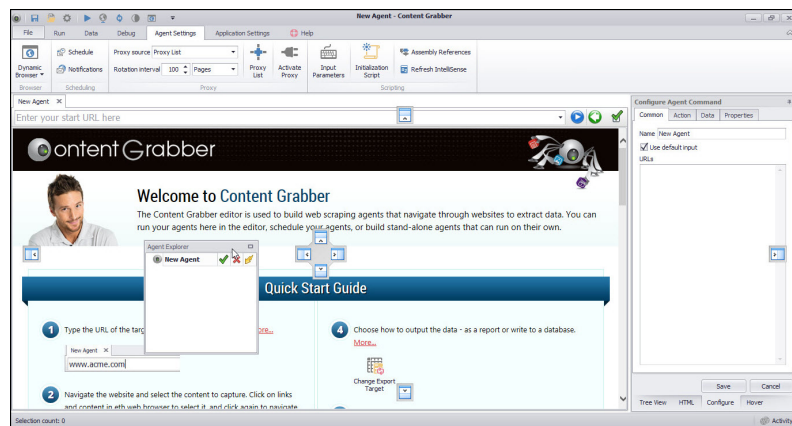
3.1 Customizing the Editor Layout

Content Grabber's editor layout includes a number of workspace windows which can be moved and re-sized to suit your development layout preferences. To help highlight this feature and demonstrate how you can configure the Editor Layout, we will reposition the Agent Explorer panel on the left of the Editor Layout screen.



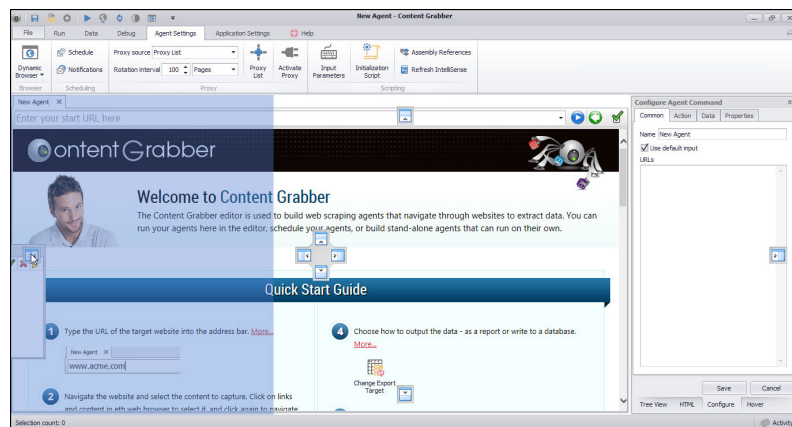
Content Grabber's default start screen with Agent Explorer on the upper right side of the screen

Now by simply positioning the cursor at the top label of the Agent Explorer panel and then clicking and holding the mouse button down, you can drag the panel to a new position on the screen. When you are dragging the Agent Explorer panel, you will notice a number of small docking objects appear with small arrows on them. These are the available docking areas to where you can drag and lock the panel into its new position.



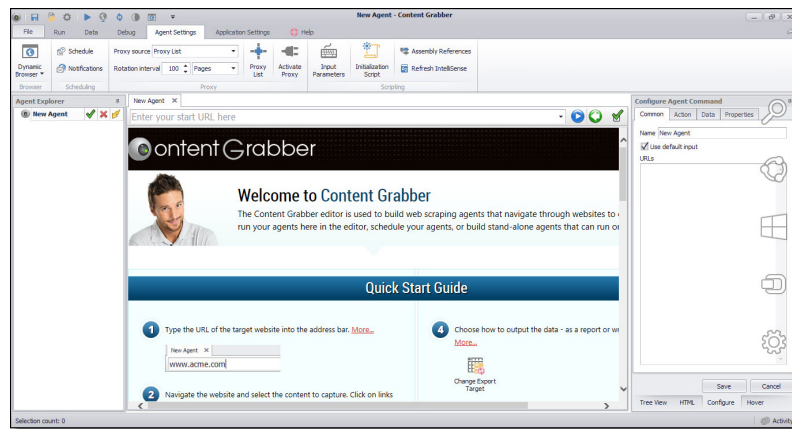
Moving the Agent Explorer panel with the mouse - docking positions are displayed

When you position the cursor over a docking position, you will notice the area changes color to blue. This means the panel is in position and you can let go of the mouse button to dock the Agent Explorer panel into this new position.



Docking position selected at the left of the screen - indicated by blue highlight

Once the mouse button is released, the Agent Explorer panel is relocated to the left of the Editor Layout screen.



Agent Explorer panel repositioned to the left of the Editor Layout screen

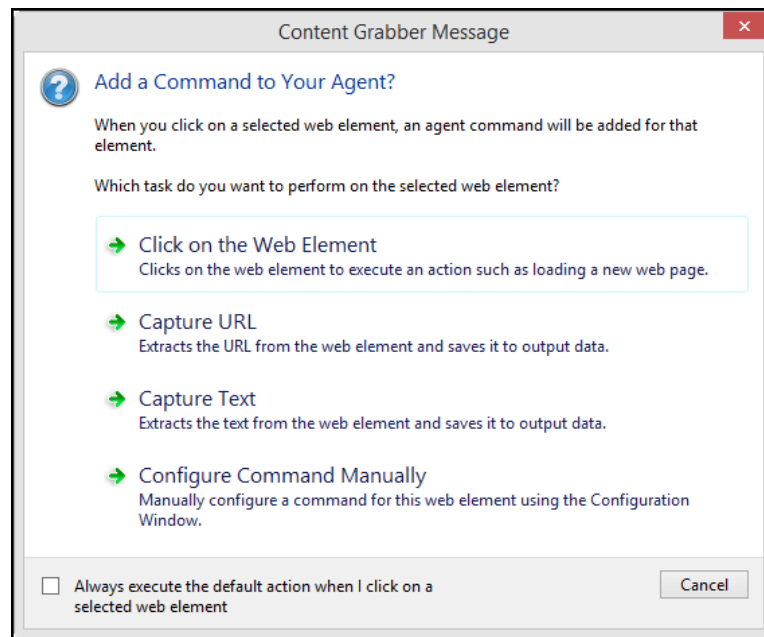
3.2 Working With the Web Browser

Content Grabber uses an embedded version of Internet Explorer as its web browser. The web browser has been greatly modified to suit the purpose of web scraping, but it essentially works the same way as the standard version of Internet Explorer you have installed on your computer. If a website does not work properly in Internet Explorer, it will also not work properly in Content Grabber.

Web Browser Selection Mode

When you click on links and buttons in a normal web browser, you will normally perform some sort of action, such as loading a new web page. Content Grabber intercepts all actions in its web browser, and when you click on a web element, it marks the element as selected instead of performing the default web browser action.

When you click on a web element that has already been selected, Content Grabber will give you an option to add an agent command that performs an action on the selected web element.



The available options, and the order of the options available, depends on what type of web element you have selected. For example, if you have clicked on a selected link element, the first option will be to navigate the link and open a new web page.

Web Browser Navigation Mode

Sometimes it's useful to navigate in the Content Grabber web browser the same way as a normal web browser. For example, you may load a URL, but then want to navigate to a particular web page and start the data extraction from that page.

You can switch the web browser from selection mode to navigation mode by clicking the **Navigate in Web Browser** button in the application menu.



Once you have reached the web page where you want to start extracting data, you can set the current URL as the start URL by clicking the check icon next to the URL address bar.



Important: Content Grabber must be able to load the start web page directly from the start URL. If the start web page cannot be loaded directly by using the start URL, then you must choose another start web page, and then use agent commands to navigate to the web page where you want data extraction to start.

Disabling Web Browser Events

Some websites display or hide web elements when certain events occur in the web browser, such as when the mouse moves over a web element, or when an input field loses focus. Sometimes it can be difficult to select such dynamic web elements, because they may show and hide as you move the mouse around.

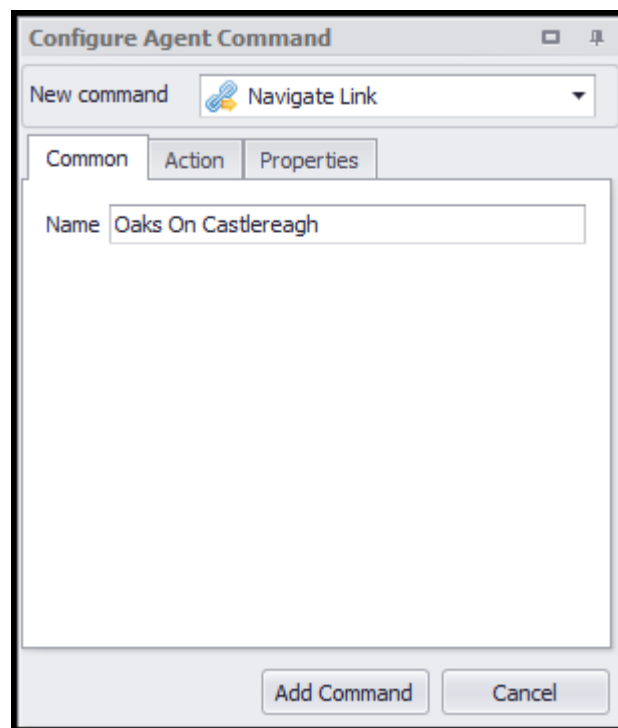
You can click the **Disable Web Browser Events** button in the application menu to block web browser events, so that dynamic web elements do not show and hide as you move your mouse for example. If you want to "freeze" the web page when your mouse is over a certain web element, then you can use the CTRL+D shortcut key to

disable events, so you don't have to move the mouse away from the web element to click the **Disable Web Browser Events** button.

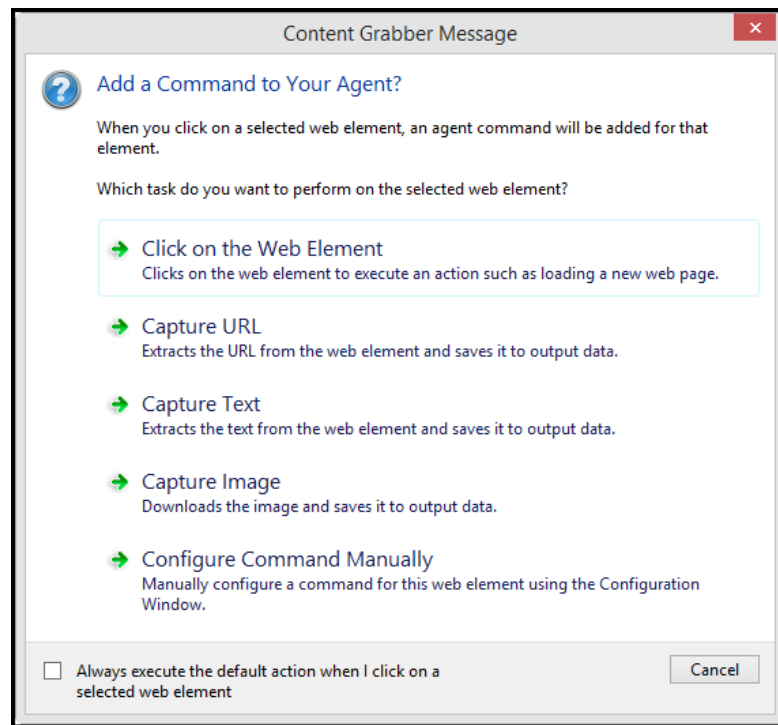
3.3 Command Configuration

Content Grabber agents consist of commands that are executed to navigate a website and extract content.

You can configure a command manually by selecting a web element in the Content Grabber web browser and then use the configuration window to specify and configure the action that should be performed on the selected web element.



You can also let Content Grabber configure a command automatically by clicking on a selected web element, and then choose from a list of appropriate actions for the selected web element type.

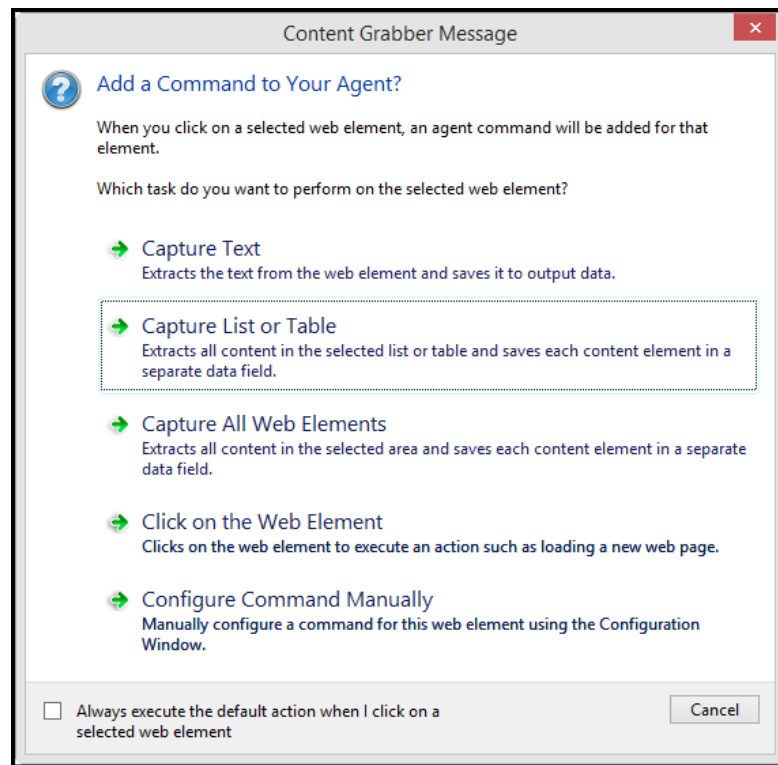


Please see Agent Commands for more information about configuring commands.

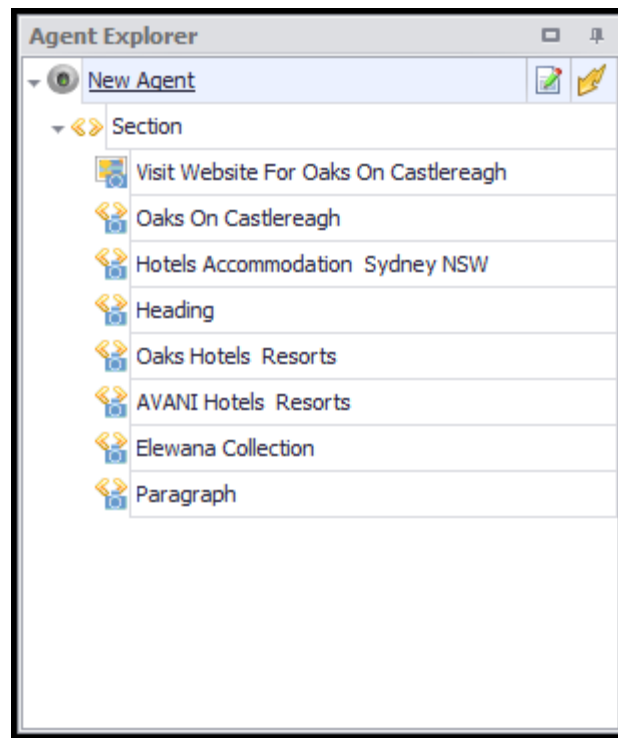
Configuring Multiple Commands Automatically

Content Grabber can automatically configure multiple commands to extract content from whole areas of a web page, such as complete HTML tables or search results.

To configure multiple commands automatically, select the entire page area you want to extract data from, such as a HTML table, and then click on the selected area again to open the Add Command dialog window.



If you have selected an area with more than one web element, you will get an option to **Capture All Web elements**. This option will add a default capture command for each web element in the selected area. The image below shows an example of commands added using the **Capture All Web Elements** option.



Content Grabber will often add more capture commands than you need, and you will have to manually delete the commands that are not needed.

Content Grabber will automatically try to assign names to the added commands, but in most cases you will have to change the names to make them more meaningful.

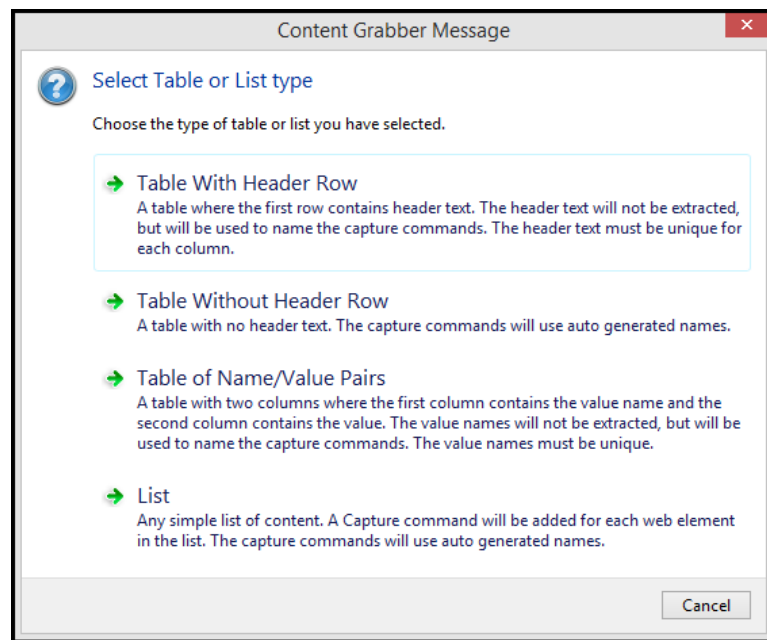
Capturing HTML Tables or Lists

If you have selected more than one web element in the web browser, you will get an option to **Capture List or Table** on the Add Command dialog window. This option will add a command for your selected list, and within the list it will add a default capture command for each web element.

If you have selected a single web element that contains a list or a HTML table, you will also get an option to **Capture List or Table** on the Add Command dialog window. This option will locate the first list or table within the selected web element,

and add a list command to your agent that iterates through each row in the table or list, and within the list it will add a default capture command for each web element.

After selecting the **Capture List or Table** option from the Add Command dialog window, you will get a new dialog window where you can select the type of table or list you have selected.

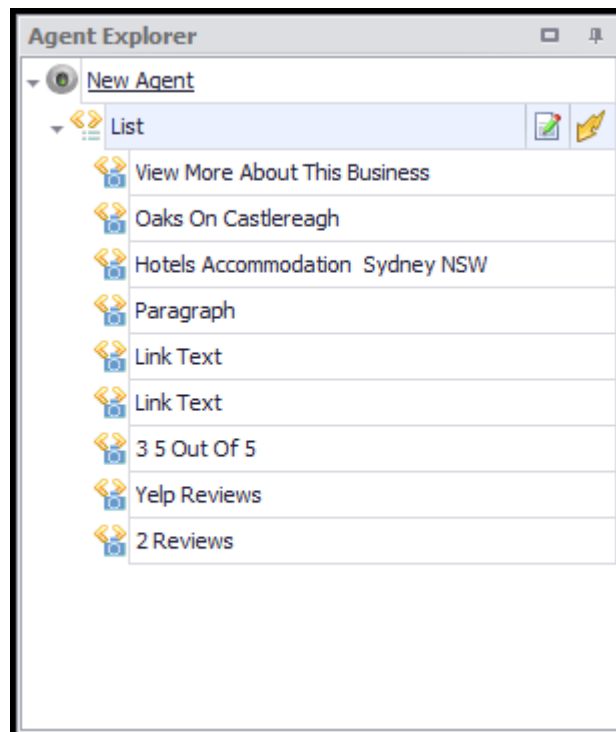


You can choose one of the following table or list types.

Table or List Type	Description
Table With Header Row	A table where the first row contains header text. The header text will not be extracted, but will be used to name the capture commands. The header text must be unique for each column.
Table Without	A table without header text. The capture commands will use auto generated names.

Header Text	
Table of Name/Value Pairs	A table with two columns where the first column contains the value name and the second column contains the value. The value names will not be extracted, but will be used to name the capture commands. The value names must be unique.
List	Any simple list of content. A capture command will be added for each web element in the list. The capture commands will use auto generated names.

After selecting the table or list type, Content Grabber will add a list command with capture sub-commands. The image below shows an example of commands added using the **Capture List or Table** option.



Content Grabber will often add more capture commands than you need, and you will have to manually delete the commands that are not needed.

Content Grabber will automatically try to assign names to the added commands, but if you have chosen the **List** or **Table Without Header Text** option, you may have to change the names to make them more meaningful.

Adding a List of Capture Commands with Specified Names

When creating an agent to match written requirements, you'll often have a long list of required field names, and will have to add a capture command for each of these field names. To make this process easy, it's possible to specify a list of command names when configuring a single capture command. Content Grabber will then add a command for each specified name.

You can specify multiple command names by separating the names with comma. You can paste command names separated by comma, tab or line break, and Content Grabber will automatically generate a comma separated list of names and also remove all white spaces from the command names. This makes it easy to copy and paste command names directly from a requirement document into Content Grabber.

3.4 Copying or Moving Agent Commands

You can move a single agent command from one location in the agent to another by simply dragging the command to the new location. Make sure you drag the command icon and not the

command name, since clicking on the command name will start a command edit.

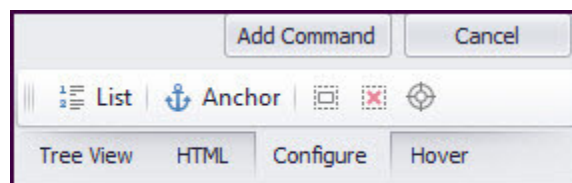
You can copy/cut/paste one or more commands by selecting the commands and using the short cut keys CTRL-c to copy, CTRL-x to cut, and CTRL-p to paste, or by right-clicking on the commands and use the context menus Copy, Cut and Paste.

You can copy commands between agents open in two different Content Grabber editors.

3.5 Selection Tools and Short-cut Keys

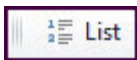
Content Grabber includes a handful of selection tools and short-cut keys that can significantly save you time when developing an agent. It is useful to understand how these work to improve your productivity. This section of the manual looks at these tools and explains how they are used. Also included are details on short-cut keys for further optimizing development time.

Most of the menu buttons / icons for the selection tools can be found at the bottom of the Configure Agent Command panel.



Bottom of Configure Agent Command panel showing selection tools

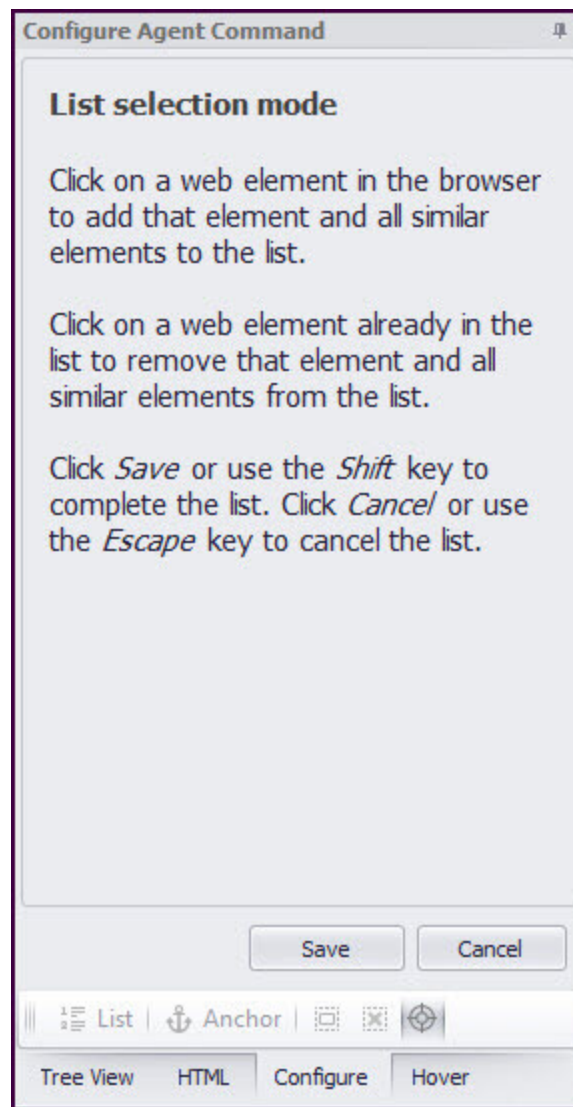
List Selection



The list selection is used to select common elements on a page and save them as a list. It can be text, titles, photos, captions, tabular data etc.

To use this function, simply select a web element in the browser then click the **List** button at the bottom of the **Configure Agent Command** panel. Clicking the List button enables **List Selection Mode**.

To collect multiple elements, select the next web element and Content Grabber will automatically add all the similar elements on the page to the list. While in List Selection Mode, you can continue to add items to your list by clicking on the elements if they weren't selected automatically. You can also click a selected item again to remove it from the list.



List Selection Mode enabled

Once you have selected all your content elements into the list, click the **Save** button to save the list and exit **List Selection Mode**.

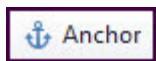
Shift Short-cut key

Instead of clicking the List button you can also enable List Selection Mode using the **Shift** key.

To do this, hold down the **Shift** key on your keyboard, click the first element (such as the first title in the list) and—continuing to hold down the **Shift** key—click the mouse on the very next element (e.g. the next title in the list).

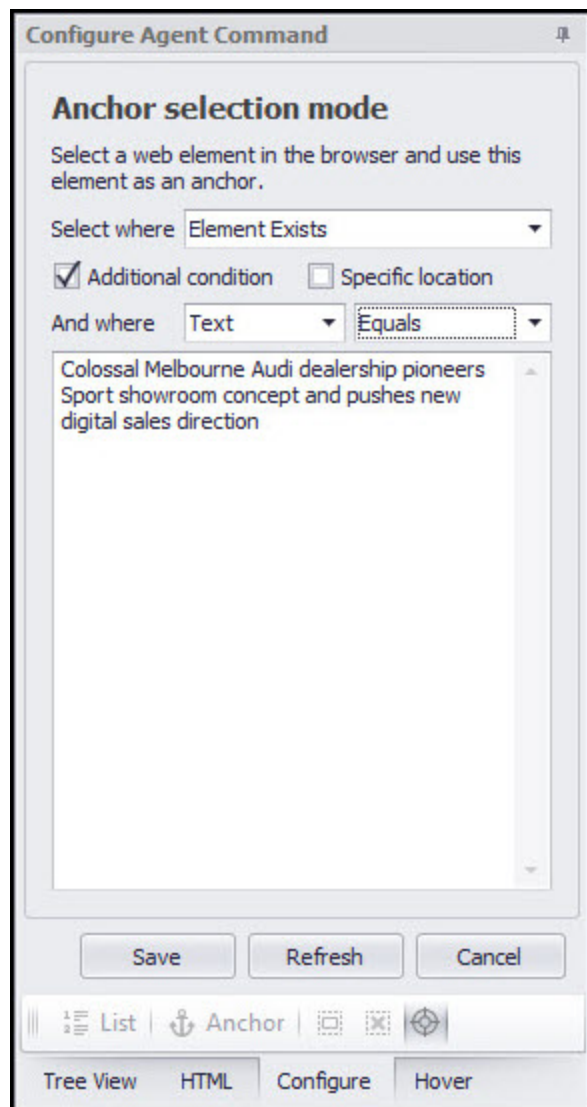
When you have finished selecting elements for you list, simply release the **Shift** key to save the list.

Anchor Selection



The Anchor Selection function is useful when you want to extract content that changes position on different pages. For example, you may extract data from a table that displays product properties, such as width, height and depth, but some properties may not apply to all products, so the location of each property may vary depending on how many properties are available for a certain product. You can use an anchor selection to tie the property value to the property name, so the property value you are extracting is always correct, no matter where it's located in the table.

To use this function, select one or more elements in the web browser then click the **Anchor** button at the bottom of the **Configure Agent Command** panel. This will enable **Anchor Selection Mode**. While in Anchor Selection Mode, click the web element you want to anchor your selection to.



Anchor Selection Mode enabled

Once you have made your selection, click the **Save** button to save the Anchor and exit **Anchor Selection Mode**.

Ctrl Short-cut key

Instead of clicking the Anchor button, you can also enable **Anchor Selection Mode** using use the **Control (Ctrl)** key.

To do this, hold down the **Ctrl** key on your keyboard, click the first element and continuing to hold down the **Ctrl** key. Next, click on the web element you want to Anchor this content element to. Once complete, release the **Ctrl** key to save the Anchor.

Command Naming Short-cut (Alt Key)

When creating a new Command in Agent Explorer, Content Grabber will automatically name the command for you. This is typically based on the naming conventions used in the HTML behind the web page.

If you hold down the **Alt** key when clicking on any text element in the browser, Content Grabber will name the new command with the text you click on.

Expand Selection



The Expand Selection tool is useful when only part of a web element has been selected (e.g. a multiple row and column table) and you want to expand the selection.

To use this function, simply click the **Expand Selection** icon at the bottom of the **Configure Agent Command** panel to expand the current selection in the web browser. Click again if you need to expand the selection further.

Clear Selection



The Clear Selection tool clears the current content element selection in the web browser.

To use this function, simply click the **Clear Selection** icon at the bottom of the **Configure Agent Command** panel to clear the current selection in the web browser.

Exact Selection



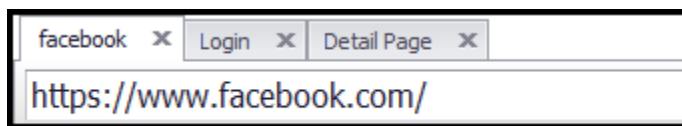
The Exact Selection tool is generally only used by advanced users. Use this option to generate selection XPathS with as much detail as possible. Exact selections are likely to fail if a website changes slightly, so only use this option if you understand the consequences.

To use this function, **select the content element on the browser page you want positional information for and then click the **Expand Selection** icon at the bottom of the **Configure Agent Command** panel.** The detailed positional data is then displayed in the **Selection XPath** window at the top of the Content Grabber screen.

This option is best used to create selection XPathS that can be used as a starting point when manually creating XPathS. Refer to Editing XPathS Manually for more detail.

3.6 Navigating an Agent

An agent loads and processes a number of different types of web pages, and each type of web page has a separate web browser tab in the agent editor.



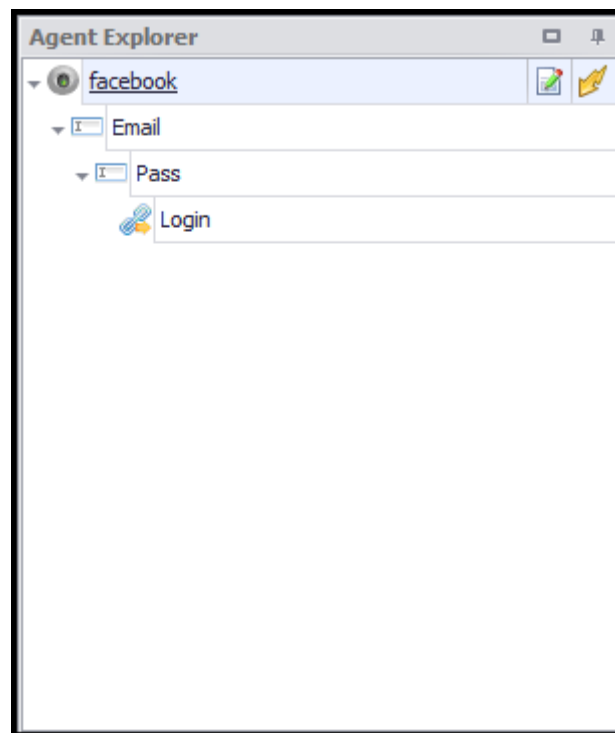
The above image shows the browser tabs for an agent that processes three different types of web pages.

An agent may process thousands of web pages, but you only need to define commands for each different type of web page. All web pages with the same layout

are considered the same type of web page. For example, if you are extracting data from a product catalog, all product detail pages are considered a single type of web page, and you therefore only need to define commands for a single product detail page.

Agent Explorer

The Agent Explorer is used to view agent commands for the type of web page loaded in the selected browser tab. When you select a new browser tab with another web page, a new set of commands will be displayed in the Agent Explorer.



You can use the Agent Explorer to view, edit or execute commands. If you execute a command that opens a new web page in a new web browser tab, Content Grabber will automatically switch web browser tab and load the new web page.

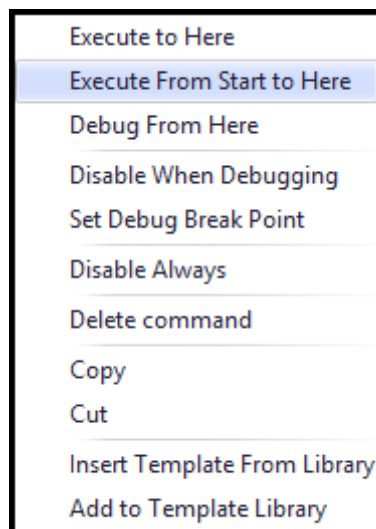
When you open a new agent, Content Grabber will automatically execute the Agent command which loads the first web page into the first web browser tab, but it will not execute all the other commands, so it will not load web pages into any other web

browser tabs. You can still switch browser tabs to view and edit the associated commands, but you will not be able to view the associated web page until you execute the command that loads a web page into the browser tab.

You generally need to execute all agent commands if you want to make sure web pages are loaded into all web browser tabs. It can be a very tedious task to manually execute all commands in an agent, so you can click the **Execute All Commands** button in the application menu to automatically execute all commands in the agent.



You can also execute all commands from start up until a specific command. Right-click on a command in the Agent Explorer and select **Execute From Start to Here** from the context menu.



If you just want to execute a set of commands on the web page in the selected web browser tab, then you can right-click on a command and select **Execute to Here** from the context menu. This is often used when you want to submit a web form, but don't want to manually execute all the Form Field commands required to set the form field

values. Since you just want to set form field values on the current web page, there's no need to execute commands in other web browser tabs.

3.7 Testing/Debugging an Agent

The agent debugger is an essential tool when trying to locate and correct issues in an agent. Running an agent through the debugger is normally the first thing you do after you have built your agent.

The debugger will warn you of any potential issues while it's running an agent, and allow you to pause and correct any issues.

An agent runs significantly slower through the debugger, and the debugger does not automatically manage JavaScript memory leaks and hanging/crashing web browsers, so you should only use the debugger for testing, and not for full agent runs.

Refer to [Using the Debugger](#) for more detail.

Logging

Logging is used to collect detailed information about the web scraping process and can be used both when debugging and running an agent. Logging includes links to the web pages that are being processed, so it's easy to pinpoint pages that may be causing problems.

Refer to [Using Logging](#) for more detail.

3.7.1 Using the Debugger

The agent debugger is used to test your agent, and to find and correct issues such as missing content elements.

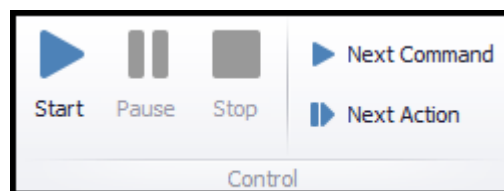
When building an agent, you often base the design of the agent on just a few web pages, and then rely on Content Grabber to execute your commands on all similar web pages it encounters.

For example, if you are extracting data from a product catalog, you may select the product name and price on a single product detail page and add agent commands to extract those two web elements. Content Grabber will then use these two agent commands to extract product name and price for all product detail pages in the product catalog. This works fine in most cases, but some products maybe on special, and the product detail pages for those products may display the price differently. Content Grabber may be unable to pick up the discount prices and the price for some products may therefore be missing. Even if you know some products have a discount and need special attention, it may be difficult to find such a product page in a large catalog.

The agent debugger can help you correct issues where an agent is unable to locate content, because it stops and warns you when content is missing and allows you to correct the agent command that caused the error. In the product catalog example, you would be able to correct the agent command that extracts the price, so it includes a selection for the discounted price.

Controlling the Debugger

Use the **Start** button in the Debug ribbon tab to start the debugger.



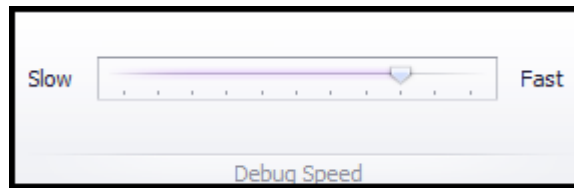
The agent will run directly in the agent editor, so you will be able to see the web pages being loaded into the web browser. The debugger will mark the content that is

being processed in the web browser, and it will also highlight the command that is being executed, in the Agent Explorer panel.

Once the agent is running in the debugger, you can pause or stop the debugger at any time. If you pause the debugger, you can use the **Next Command** button to execute one command at a time, and the **Next Action** button to execute commands until the debugger reaches an action command.

Setting the Debug Speed

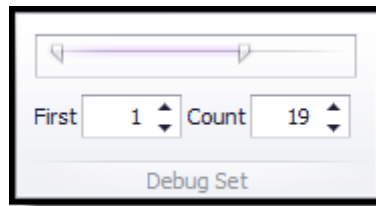
The debugger will mark the content that is being processed in the web browser, which is helpful when trying to work out how the agent is processing the website, but the debugger may run so fast that it's impossible to see what's going on. You can slow down the debugger to make it much easier to follow the process.



You can change the debug speed before you start the debugger or while the debugger is running.

Debugging a Data Subset

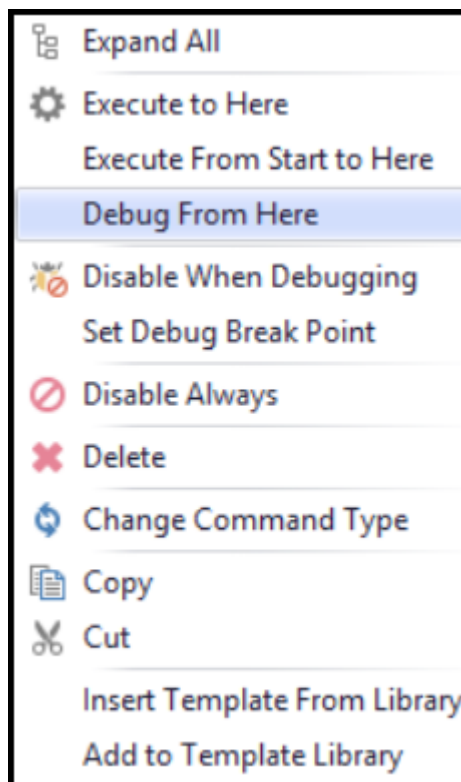
Agent list commands may process large data sets, such as a long list of Start URLs or a long list of form field input data. If you want to test an agent with data from the end of a list, it may take hours for the agent to reach the data you want to test. Instead of waiting for the agent to reach the data you want to test, you can specify the subset of data you want to include in the debugging session.



The **Debug Set** option is only available when you select a list command in the Agent Explorer. This setting is automatically saved to the agent command and will apply to the command every time you debug the agent. This option has no effect when you run an agent, only when you debug the agent.

Debug From a Specific Command

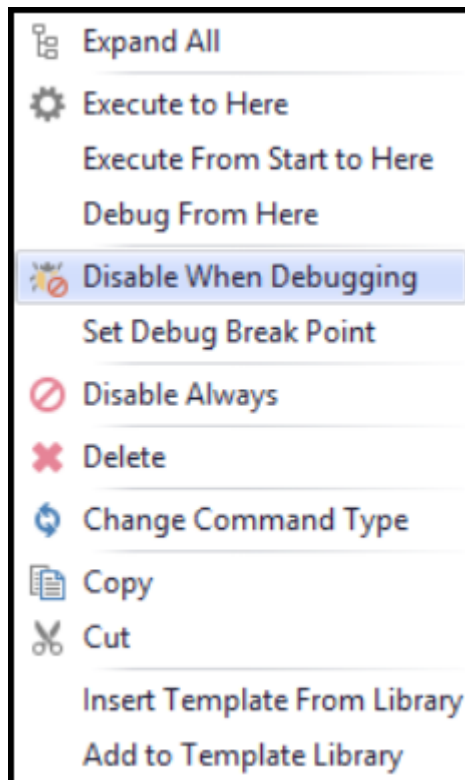
Sometimes you may want to only debug a small part of your agent. If you have a complex agent that processes a large website, it may take a long time before the debugger reaches certain part of your agent, so instead of starting from the beginning, you can select the agent command where the debugger should start.



When you start debugging from a specific command, the web browser tab that is associated with the command must have a web page loaded. You cannot start debugging from a blank web page.

Disable a Command While Debugging

When debugging a complex agent that processes a large website, it's often useful to exclude parts of the agent, so you only execute the part of the agent you want to test. you can do this by disabling commands while debugging.



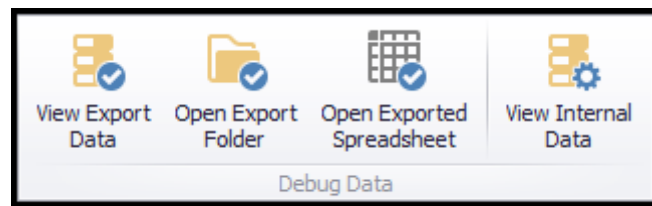
When you disable a container command, all sub-commands are also automatically disabled.

This setting is automatically saved to the agent command, so the command will be disabled every time you debug the agent until you enable the command again. This setting has no effect when you run an agent, only when you debug the agent.

Viewing Debug Data

Data collected while debugging an agent is saved to a separate data store, so it doesn't overwrite data that is collected when you run the agent. The debugger will export data to your chosen export target, but it will never distribute data. If you are exporting data to a database, the debugger will create separate data tables for the debug data. If you are exporting data to a file format, the files will be written to the agent's debug data folder.

Important: If you are using a script to export data, you are responsible for managing debug data if you want to separate debug data from normal data.



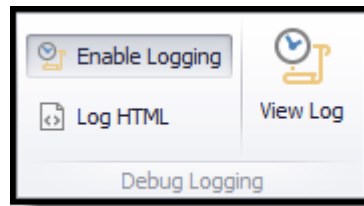
Please see the Data topic for more information about agent data.

3.7.2 Using Logging

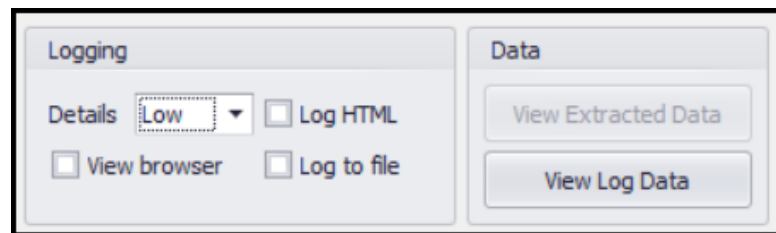
Logging can be used when debugging or running an agent to collect information about the web scraping process. Logging can be set to three different detail levels, low, medium and high. Low detail level only logs errors, medium detail level logs errors and warnings, and high detail level logs errors, warnings and general progress information. Debug logging is always set to high detail level and cannot be changed.

Debug log data is stored in a separate data store, so it doesn't overwrite or get mixed with normal log data.

Debug logging can be turned on in the **Debug** ribbon menu.



Normal logging can be turned on when running an agent on the **Run Agent** screen.



The Run Agent screen is not displayed when running an agent using the **Run** button in the application menu, so you will not be able to configure logging when running an agent this way.

Logging Raw HTML

Content Grabber automatically logs direct links to processed web pages when logging is turned on, so it's normally easy to view specific web pages that have been processed. For example, if an error or warning appears in the log, you can simply click on the associated URL to open the web page and see if there is anything special about the page that may cause an error.

Sometimes it's not possible to open a web page using a direct URL. For example, some websites implement CAPTCHA protection, which is web pages that appear randomly to ask the user to enter a verification code. If a CAPTCHA page is retrieved instead of a normal web page, your agent is likely to encounter errors, but if you click on the associated URL later, you may not get the CAPTCHA page because it appears randomly. In this case it may be difficult to determine what is causing the error, but you can use the **Log HTML** feature to log the raw HTML of all processed web pages,

and this will allow you to view the CAPTCHA page. Please see the CAPTCHA topic for more information about CAPTCHA.

Error Handling

Please see the Error Handling topic for more information about error handling, notifications and logging.

3.8 Scheduling

Content Grabber provides an Agent scheduling facility that enables you to automatically run your agents at predetermined time slots whenever you need it to run. This can be done every hour, every day, month, year and so on.

You can schedule agents using the Content Grabber Scheduler or the Windows Task Scheduler. We recommend you use the Content Grabber scheduler for new agents, since it provides more advanced features and integrates more tightly with the rest of the Content Grabber application. The Windows Task Scheduler integration is provided for backward compatibility. If read the topic Windows Task Scheduler if you prefer to use this scheduler.

Content Grabber Scheduler

If you have an Agent loaded in Content Grabber, you will be able to access the Scheduling feature. From the **Agent** menu at the top of the Content Grabber application. Simply Click on the **Schedules** menu option to display the Scheduling window

Add sequentum Schedule

Schedule

☒ Basic

Run every Days

☐ Advanced

Second

Minute

Hour

Day

Month

Weekday

Runs at 06:46:14 PM, every day.

First run will be 28/10/2017 6:46:14 PM.

Settings

Start date

Start time

Log level

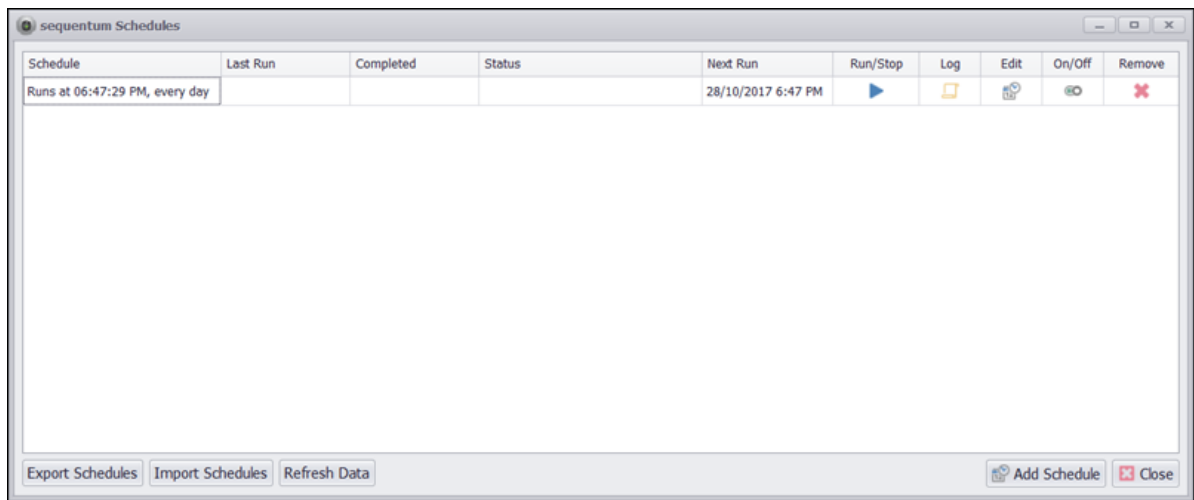
Session ID (optional)

Input parameters (optional)

Parameter Name	Value	Rem...
		X

☒ Add Schedule ☐ Cancel

If one or more schedules already exist for the current agent, a Window listing all schedules will open, and you can edit existing schedules or add more schedules from there.



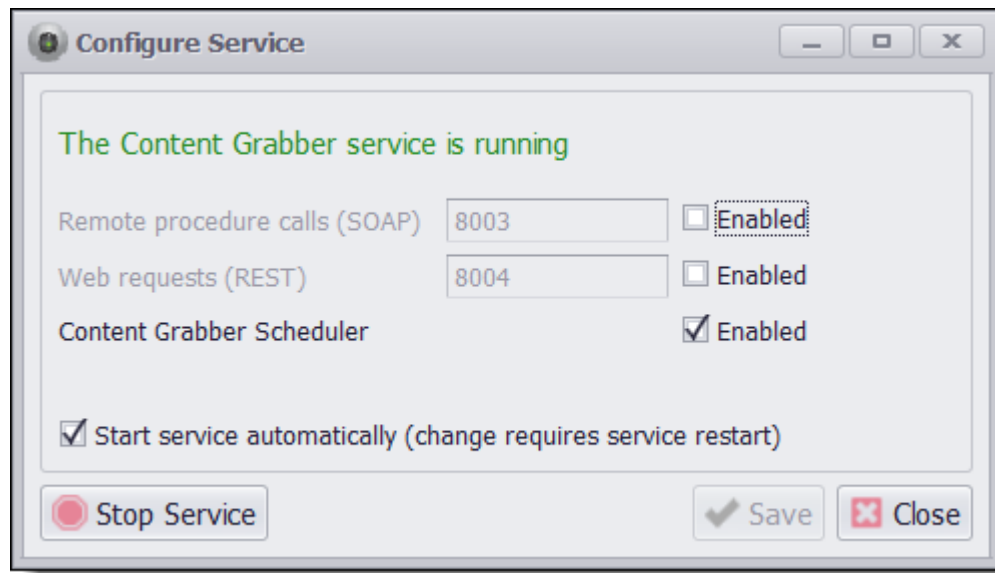
Cron Expressions

The Content Grabber Scheduler uses Cron expressions to define when the scheduler should run an agent. If you setup a basic schedule, Content Grabber will automatically convert the basic schedule into a Cron expression. When you edit an existing schedule, the Cron expression will always be shown, but you can simply switch back to the basic schedule if you don't want to work with Cron expressions.

Please read the topic Cron Expressions for more information about Cron syntax.

Scheduler Windows Service

The Scheduler depends on the Content Grabber Windows service, and the service must be running for the scheduler to be operational. If the service is not running, a message will be displayed at the top of the Scheduler Windows with a link allowing you to start the service. We recommend you configure the service to start automatically when your computer starts.



Make sure the Content Grabber scheduler is enabled before you start the service. **Remote procedure calls** and **Web request** are used by the API and are not required by the scheduler.

3.8.1 Cron Expressions

The Content Grabber Scheduler uses Cron expressions to define when the scheduler should run an agent. The scheduler supports standard Cron expressions with seconds, but without year, so the expressions consist of 6 parts.

Field Name	Mandatory	Allowed Values	Allowed Special Characters
Seconds	Yes	0-59	, - * /
Minutes	Yes	0-59	, - * /
Hours	Yes	0-23	, - * /
Day of Month	Yes	1-31	, - * ? / L W

Month	Yes	1-12 or JAN-DEC	, - * /
Day of Week	Yes	0-7 or SUN-SAT	, - * ? / L #

Cron Expression Examples

Cron Expression	Meaning
* * * * *	Execute a job every second
0 */5 * * * *	Execute a job every 5 minutes
0 0 * * * *	Execute a job every hour.
0 0 12 * * *	Executes a job at 12pm (noon) every day
0 0/5 14,18 * * *	Executes a job every 5 minutes starting at 2pm and ending at 2:55pm, AND fire every 5 minutes starting at 6pm and ending at 6:55pm, every day
0 0 0 * * 3	Executes a job at midnight every Wednesday.
0 15 10 L * *	Executes a job at 10:15am on the last day of every month
0 15 10 15 * *	Executes a job at 10:15am on the 15th day of every month
0 0 0 1,2 * * *	Executes a job at midnight of 1st, 2nd day of each month
0 15 10 * * 5#3	Executes a job at 10:15am on the third Friday of every month

Special characters

Asterisk (*) - The asterisk indicates that the cron expression matches for all values of the field. E.g., using an asterisk in the 2th field (minute) indicates every minute. * is a non-restricted character.

Slash (/) - Slashes describe increments of ranges. For example 3-59/15 in the 2st field (minutes) indicate the third minute of the hour and every 15 minutes thereafter. The form "*/..." is equivalent to the form "first-last/...", that is, an increment over the largest possible range of the field.

Comma (,) - Commas are used to separate items of a list. For example, using "1,2,5" in the 6th field (day of week) means Mondays, Wednesdays and Fridays.

Hyphen (-) - Hyphens define ranges. For example, using "MON-FRI" in the 6th field (day of week) means all weekdays, but not weekends.

L - 'L' stands for "last". When used in the day-of-week field, it allows you to specify constructs such as "the last Friday" ("5L") of a given month. In the day-of-month field, it specifies the last day of the month.

W - The 'W' character is allowed for the day-of-month field. This character is used to specify the weekday (Monday-Friday) nearest the given day. As an example, if you were to specify "15W" as the value for the day-of-month field, the meaning is: "the nearest weekday to the 15th of the month." So, if the 15th is a Saturday, the job executes on Friday the 14th. If the 15th is a Sunday, the job executes on Monday the 16th. If the 15th is a

Tuesday, then it executes on Tuesday the 15th. However if you specify "1W" as the value for day-of-month, and the 1st is a Saturday, the job executes on Monday the 3rd, as it does not 'jump' over the boundary of a month's days. The 'W' character can be specified only when the day-of-month is a single day, not a range or list of days.

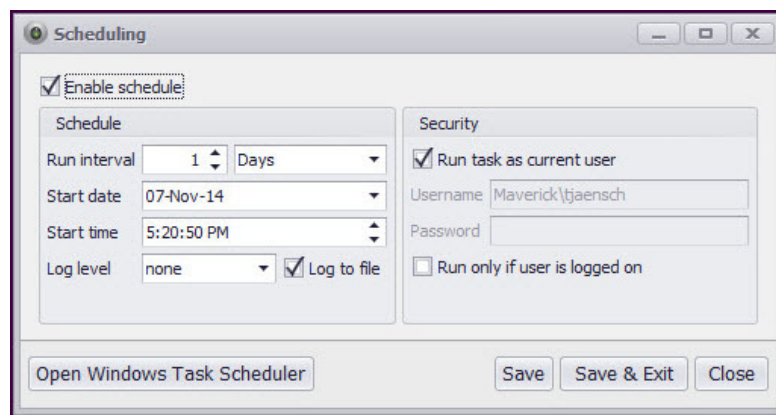
Hash (#) - '#' is allowed for the day-of-week field, and must be followed by a number between one and five. For example, 5#2 indicates "the second Friday" of a given month.

Question mark (?) - It is used instead of '*' for leaving either day-of-month or day-of-week blank. ? is a restricted character.

3.8.2 Windows Task Scheduler

Content Grabber allows you to add tasks to the Windows Task Scheduler, as an alternative to the Content Grabber scheduler.

If you have an Agent loaded in Content Grabber, you will be able to access the Scheduling feature. From the **Tools** menu at the top of the Content Grabber application. Simply Click on the **Add Schedule** menu option in the **Deprecated options** group to display the Scheduling window.



Content Grabber's Scheduling Window

The Enable schedule checkbox allows you to easily turn scheduling on or off for the Agent. Click the **Enable schedule** checkbox so you can begin configuring the schedule details.

The **Run interval** fields allow you to select how often your agent will run. This can be set in seconds, minutes, hours or days. **Start date** and **Start time** enable to precisely control from when the agent will start executing.

The log feature allows you to record when the agent runs. The **log level** setting you choose (i.e. none, low, medium, high) will determine the level of detail that is recorded about each run. For instance, you may want to track all error messages or issues encountered by the running agent so you might choose "high". However, be mindful that a high level of logging may slow the Agent performance.

Content Grabber also includes some scheduling security features. If you don't want your Agent to run when you aren't logged on to your computer, simply click the **Run only if user is logged on** check box.

If you don't set the option **Run only if user is logged on**, the agent will run in a special Windows desktop session that does not allow input focus. This is a Windows security feature and cannot be circumvented. This may cause JavaScript on some websites to work incorrectly and the agent may not be able to extract data correctly. This scenario is very rare, but if it occurs you will need to set the scheduling option **Run only if user is logged on** and make sure the user is always logged on to the computer when the agent is running.

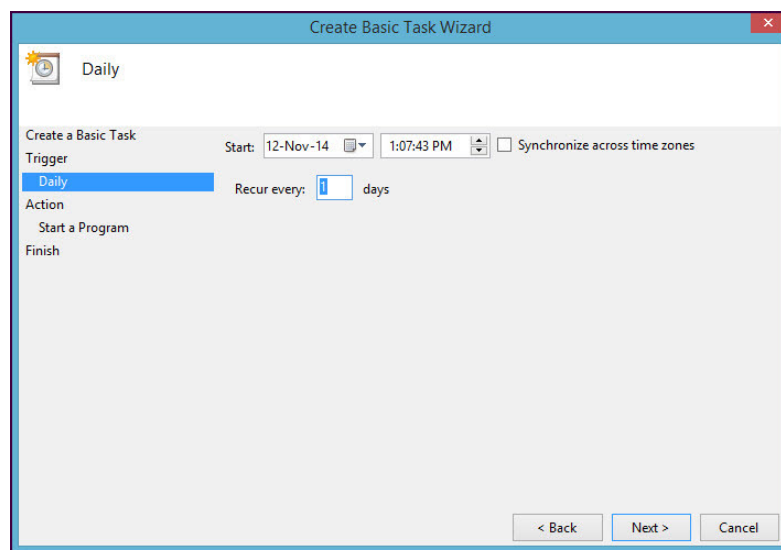
When using the option **Run task as current user** you may get an error **Access Denied**. To avoid this error you can run Content Grabber as an administrator or you can clear the option **Run task as current user** and enter your username and password instead. To run Content Grabber as an administrator, right click on the Content Grabber program file or shortcut and select **Run as administrator** from the context menu.

Using Windows Task Scheduler

Should you want to manage multiple agents from one place or are looking for additional controls, or calendar functionality, you can utilize the Windows Task Scheduler instead. To do this, [simply click on the **Open Windows Scheduler** button on the Scheduling window.](#)

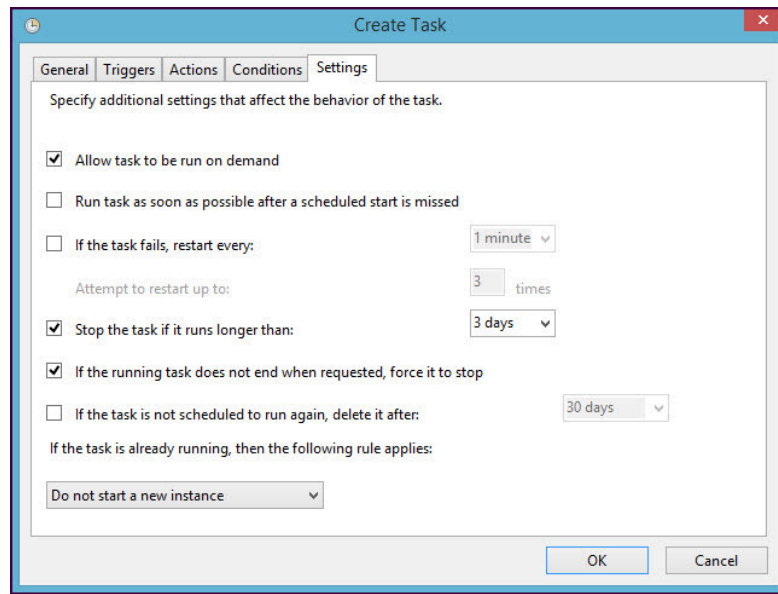
Windows Task Scheduler is used to create and manage common tasks that your computer will carry out automatically at the times you specify. In this instance the task will be to run a web-scraping Agent(s) at the scheduled time(s) you require.

To setup a basic Agent schedule task, simply go to the Action menu and select the Basic Task Wizard. This wizard will lead you through the required steps to setup your schedule task.



Windows Task Scheduler - Using the Basic Task Wizard

For more advanced scheduling options or settings such as multiple task actions or triggers, use the Create Task command in the Actions menu.



Windows Task Scheduler - Create Task Command (for advanced options)

Tasks are stored in folders in the Task Scheduler Library. To view or perform an operation on an individual task, select it in the Task Scheduler Library and click on a command in the Action menu.

3.9 Copying & Moving Agents

Moving an agent

You can move an agent by simply moving the agent folder to a new location. The agent will retain its unique identifier, so the agent will work as before although it is now in a new location.

Copying an agent

To copy an agent, simply choose **File > Save As** in the **Content Grabber** menu. The new agent will get a unique identifier - an identifier that is different from the original agent.

WARNING: We strongly recommend that you do not simply copy the agent from one folder to another on your disk. The result will be two agents having the same

identifier, since the copy will retain the original identifier.

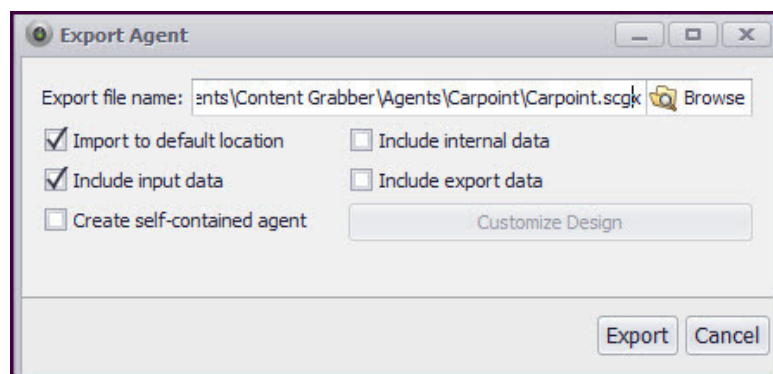
Content Grabber stores information about each agent in its database, and can only store one entry per identifier. If you do have two agents on your computer with the same identifier, the active agent will be the agent that you open most recently in the Content Grabber editor. In these circumstances, you risk a duplication of effort or a loss of work.

3.10 Exporting & Importing Agents

Exporting and Importing agents allows you to easily move your web scraping agents between machines. You can also export your agent for back-up to another device.

Exporting Agents

To export your agent, first make sure you save your agent and have it open. **Then from the File menu, select Export Agent.** The following Export Agent pop-up window will appear on your screen.

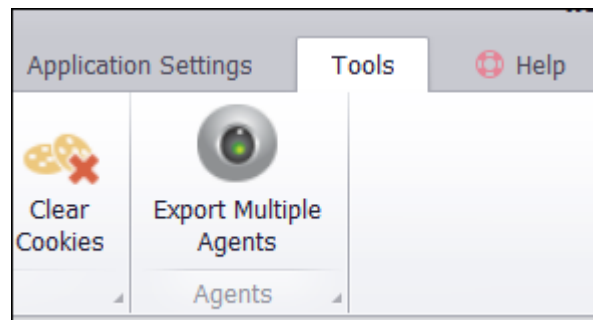


Export Agent window

Content Grabber's Export Agent facility enables you to package up all the associated files and folders into one file in a manner that is similar to a zip file. The default location for where agents are exported is the **Documents > Content Grabber > Agents** folder on your PC.

You can also elect to **Include internal data** and/or **Include export data** into the export file by selecting the corresponding check box. *Once you have made your selection, simply click the **Export** button.*

You can export multiple agents at once using the **Export Multiple Agents** button in the **Tools** menu.

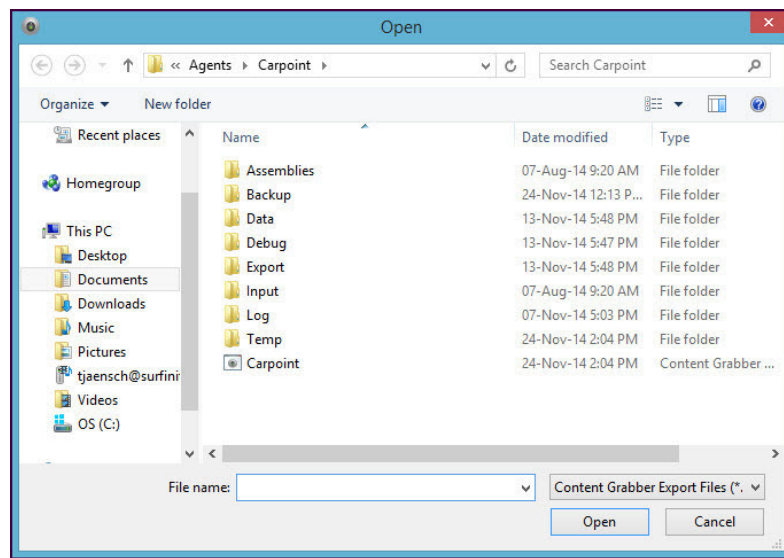


Export multiple agents at once

For detail on the **Create self-contained agent** check box, refer to Building a Self-contained Agent.

Importing Agents

You can only import agents that have previously been exported from Content Grabber. *To do this, select **Import Agent** from the **File** menu.* The following Export Agent pop-up window will appear on your screen.



Import Agent Open window - to select agent file for import

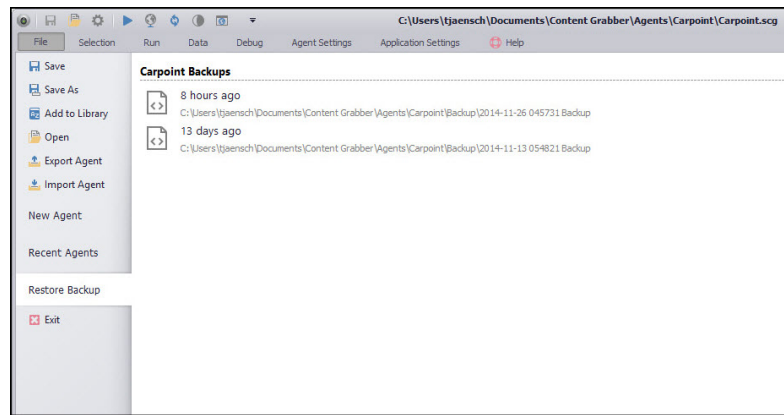
Simply select the Content Grabber agent file you wish to Import and then click the **Open** button.

All the folder structures and associated files for the agent will then be unpacked into the **Documents** folder on your machine.

3.11 Automated Agent Backups

Whenever you save an agent, Content Grabber automatically makes a backup for you which you can restore your agent from should you have any issues.

To see the different backups available to restore from, click on the **File** menu and you will see the agent backup listed under this menu.



Restore from Content Grabber's Automated Backups

Content Grabber also saves a backup restore file for you should the application close unexpectedly (e.g. should the application process terminate). This will be available for you to restore from so you do not lose your latest development work.

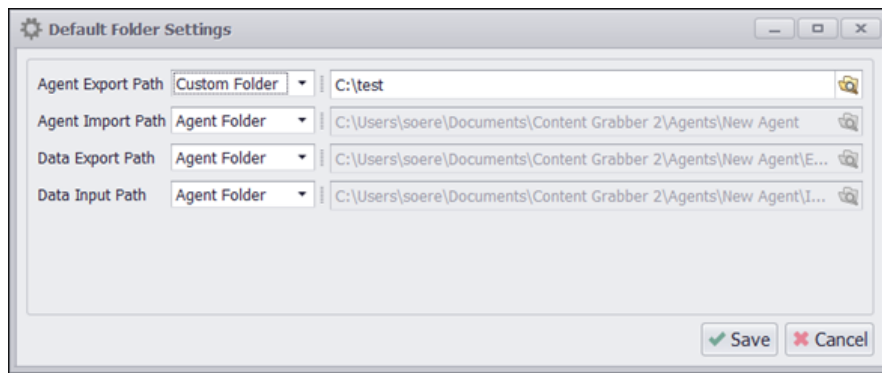
3.12 Template Libraries

A template library contains building blocks that you can use to build your agents. A building block can be an entire agent, a group of commands, or a script. For example, if you need to create many agents that have the same structure but will point to different websites, then you could add your own agent template and derive it from the first agent you create.

We encourage you to learn more about the Agent Template Library and the Command Template Library.

3.13 Default Folders

Use Default Folders to specify paths that apply to all agents on the computer.



Default folders

Content Grabber currently supports the follow default folders.

Folder	Description
Agent Export Path	Exported agents will be saved to this folder by default.
Agent Import Path	Imported agents will be loaded from this folder by default.
Data Export Path	All agents that export data to file formats will save the data to this folder by default. A sub-folder named Files will be created automatically if an agent extracts files, and another sub-folder named the same as the agent will contain the extracted files.
Data Input Path	All agents that read input data from this folder by default.

Each default folder can be set as **Agent folder**, **Agent Group Folder**, or **Custom Folder**.

- **Agent folder** is the default option and specifies a sub-folder inside the agent folder.
- **Agent Group Folder** specifies a sub-folder inside the agent's parent folder. The default agent parent folder is *My Documents\Content Grabber2\Agents*.
- **Custom Folder** can be any specified folder.

4 Selection Techniques

Mastering the essential selection techniques is a critical aspect of web-scraping. When you point-and-click on content in the web browser to create agent commands, you are using the most basic selection technique. In addition to the simple point-and-click feature, Content Grabber provides a range of tools to help you make precise selections, including:

- **XPath Editor** - gives you the ability to manipulate the selection **XPath**
- **Tree View Window** - gives a more precise view of a web page
- **List Tool** - helps you select a list of web page elements
- **Anchor Tool** - helps you apply conditions to the selection **XPath**.

It is important to realize that you have to be explicit with Content Grabber. You have to be deliberate and specific when selecting each of the HTML elements that you want to capture. Sometimes it can be confusing, as when you want to capture the elements of a search-result listing and each entry has a heading. In some cases, the heading will be a link and in other cases it will be plain text, so the HTML for the headings may look something like this:

```
<h1><a href="http://website.com">Heading as a link</a></h1>  
<h1><a href="http://website.com">Heading as a link</a></h1>  
<h1><span>Heading as plain text</span></h1>  
<h1><a href="http://website.com">Heading as a link</a></h1>
```

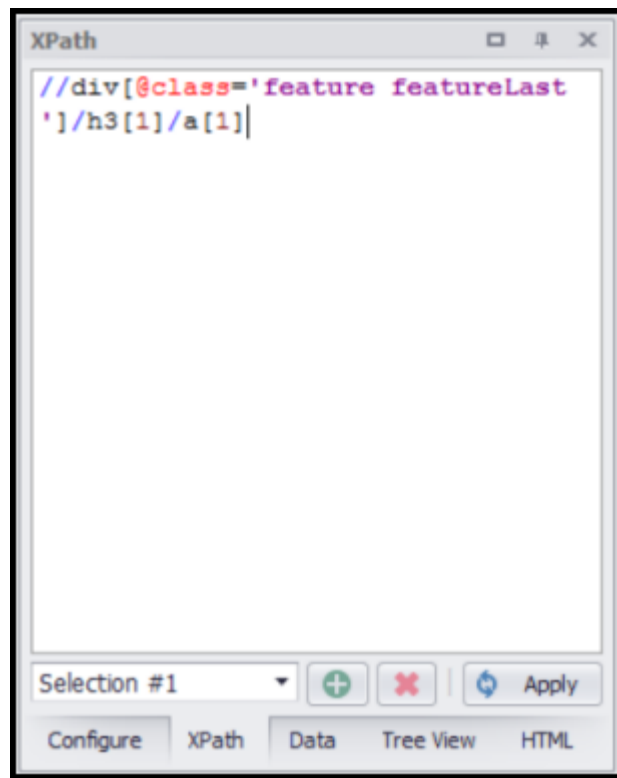
If the first two headings are links, then only the link headings will be chosen - not the headings that are plain text. To extract the text of all the headings in this example, you might use the first two headings to create a list selection. As with many software applications, you have to be explicit with Content Grabber. It does not somehow know

that you want to extract all the headings, and its default function will be to assume that you are trying to select only the links. In such a case, you would need to change the selection so that it selects the `<h1>` tag instead of the link tag - before you create the list.

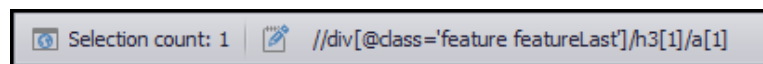
4.1 Selection XPath

Each time you click on content in the web browser panel, Content Grabber does some processing in the background to calculate the selection **XPath**. **XPath** is a common syntax for selecting in XML and HTML documents. Content Grabber uses a standard implementation that supports **XPath** v1.0 syntax, and also supports a range of new methods specifically designed to make web scraping easier. Content Grabber has a range of tools that helps you create a precise **XPath**, without the need to know the syntax. Eventually, you may find the need to fine-tune the **XPath** manually, and then you will need to learn the **XPath** syntax.

Each time you make a selection in the web browser, you can view the selection **XPath** either in the **XPath** panel or on the status bar (see the figures below).



The selection XPath panel



The selection XPath is shown on the status bar

The **XPath** in the figure above is:

```
//div[@class='feature featureLast']/h3[1]/a[1]
```

It contains these selection steps:

1. Selects all `<div>` tags within the webpage having the class attribute value `feature featureLast`.
2. Also selects all child `<h3>` tags from the selection in step 1 (`/h3[1]`).
3. Then selects all child `<a>` tags from the selection in step 2 (`/a[1]`).

Read more in the XPath and Selection Techniques article. To learn more about **XPath**, we recommend that you consult a good reference guide such as this one:

www.w3schools.com/XPath/xpath_syntax.asp

Multiple XPaths

A selection can consist of multiple XPaths. This is useful when trying to select web content that can appear in different locations on a web page.

To add an XPath to a selection, use the **Add** button in the **XPath** window. If you have multiple XPaths in a selection, you can select **View All** from the drop down box to view all the web content selected by all the XPaths.

Content Grabber XPath Functions

Content Grabber has a set of non-standard functions you can use in **XPath** selections:

Function	Description
bool equals(string source, string target)	Returns True if the two strings are equal. The comparison is case insensitive.
bool fuzzy-match(string source, string target, [double tolerance])	Uses the Jaro/Winkler distance algorithm to determine if two strings match. This function first splits each string into words and then compares words from each string. If one string contains

	<p>more words than the other string, the additional words are not considered when calculating the distance.</p> <p>Tolerance must be between 0 and 1. A tolerance of 1 means an exact match. A tolerance of 0.85 is used if tolerance is not specified.</p>
<p>bool fuzzy-match(string source, string target, double singleWordTolerance, int wordMismatchesAllowed, [bool isMatchAllSourceWords], [bool isMatchAllTargetWords])</p>	<p>Uses the Jaro/Winkler distance algorithm to determine if two strings match. This function first splits each string into words and then compares words from each string.</p> <p>If isMatchAllSourceWords is true, all words in the source string will be matches.</p> <p>If isMatchAllTargetWords is true, all words in the target string will be matches.</p> <p>if both isMatchAllSourceWords and isMatchAllTargetWords are true, all words in the string containing the most words will be matched.</p>

	<p>if both <code>isMatchAllSourceWords</code> and <code>isMatchAllTargetWords</code> are false, all words in the string containing the least words will be matched.</p> <p><code>SingleWordTolerance</code> must be between 0 and 1. A tolerance of 1 means an exact match. If a word match is less than the <code>SingleWordTolerance</code> value, a word mismatch is recorded.</p> <p><code>WordMismatchesAllowed</code> is the number of words that are allowed to mismatch before the function returns false.</p>
bool not-whitespace(string source)	Returns False if the string contains only white space characters.
add-bookmark()	Adds the current node to an internal list of bookmarks.
bool has-parent-bookmark()	Returns True if a parent of the current node is in the bookmark list.
NodeSet parent-bookmark()	Returns the first parent node that is in the bookmark list.
string html()	Returns the HTML of the current node.

string inner-html()	Returns the inner HTML of the current node.
string uniqueid()	Returns a unique ID.
string url()	Returns any URL property of the current node.
string image-url()	Returns any image URL of the current node.
string email()	Returns any email address of the current node.
string flash-url()	Returns any flash URL of the current node.
string tag-text()	Returns the text of the current node excluding text from any child nodes.
string find-data(string commandName)	Returns the extracted data for a command that has already been processed. If the command does not exist in the current container command, the function will keep searching in parent containers.
string get-data(string commandName)	Returns the extracted data for a command that has already been processed. The command must exist in the current container command.
string get-input-data(string commandName, string columnName)	Returns input data as a string value from the specified data column in the data provided by the specified command. The

	specified command must be a data provider.
string get-input-data(string columnName)	Returns input data as a string value from the specified data column in the data provided by the last data provider parent command.
string get-input-data()	Returns input data from the first data column that contains a string value in the data provided by the last data provider parent command.
string get-global-data(string name)	Returns the data entry with the specified name from the global data dictionary which includes input parameters. The data entry must be a string value or a simple value type that can be converted to a string value.
int node-position([nodeSet])	<p>Returns the position of a specific node among all nodes with the same parent node. If no node is given, then this is the position of the root node.</p> <p>When choosing elements inside a web element list, the current list element is the root node. So a call to node-position() would return the position of the current list element.</p>

	<p>The index is not zero based, so the first index is 1.</p>
NodeSet root([int nodeIndex])	<p>Returns the root node with a specific index. The index must be greater than 1. If no index is specified the current root node is returned.</p> <p>When selecting elements inside a web element list, the current list element is the root node. So, a call to root() would return the current list element, and a call to root(2) would return the second list element.</p>
int root-index()	<p>Returns the index of the current root node.</p> <p>When selecting elements inside a web element list, the current list element is the root node. So, a call to root-index() would return the index of the current list element.</p> <p>The index is not zero based, so the first index is 1.</p>
int last-root-index()	<p>Returns the index of the last root node.</p>

	<p>When selecting elements inside a web element list, the current list element is the root node. So, a call to last-root-index() would return the index of the last list element.</p> <p>The index is not zero based, so the first index is 1.</p>
int root-position()	<p>Returns the position of the current root node. This position is relative to all nodes with the same parent node.</p> <p>When selecting elements inside a web element list, the current list element is the root node. So, a call to root-position() would return the position of the current list element.</p> <p>The index is not zero based, so the first index is 1.</p>
NodeSet root-siblings()	<p>Returns following siblings of the current root node, but stops when it encounters another root node.</p> <p>This function is equivalent to the following selection:</p>

```
root()/following-sibling::*[root-
index()=last-root-index()]
```

or

```
position()<root-position(root-index()
+1)-root-position()]
```

When selecting elements inside a web element list, the current list element is the root node. So, a call to **root-siblings()** would return siblings of the current list element.

4.2 Optimizing Selections

An excessively long **XPath** is unlikely to work well if the target web page changes - even slightly. Every step in an **XPath** must match an HTML tag on the web page, and it will fail if any of these tags move to another location or go away entirely.

When you click on an HTML element in the web browser, Content Grabber will automatically attempt to create an optimal selection **XPath** by making it as short as possible. For example, the full selection **XPath** path to a `<div>` HTML element could be as follows:

```
DIV[1]/DIV[5]/TABLE[2]/TBODY[1]/TR[2]/TD[1]/DIV
```

If the DIV tag has an ID attribute with a unique value **listView**, the optimal **XPath** is:

```
//DIV[@id='listView']
```

Content Grabber will examine the entire web page for a DIV tag having the ID attribute value **listView**. This **XPath** example is very robust and not sensitive to future

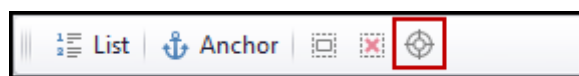
page changes, and it will work as long as this element exists on the web page - even if the rest of the page changes.

The design of Content Grabber is to prefer ID attributes for optimizing **XPaths**, because the expectation is that any given ID will be unique on a web page. However, sometimes websites use IDs that are unique to a specific content element, such as a product ID, and such IDs are not appropriate for use in an **XPath**. If you extract data from a product catalog by using an **XPath** to extract the product title from all product detail pages, then you do not want the **XPath** to depend on a specific product ID, such as we have in the case below:

```
//H1[@id='sku_245865']
```

Such an **XPath** would work for only one specific product and not for the others. Content Grabber will avoid using IDs that look like identifiers, such as a product identifier, but if it turns out wrong, then your only option is to optimize the **XPath** manually.

If you need to optimize a selection **XPath** manually, you can use the **Exact Selection** tool. This tool generates selection **XPaths** with as much detail as possible, and then you can manually remove the detail that is causing problems - such as an ID attribute containing a product identifier.



The Exact Selection tool

4.3 List Selections

List selections are important, since you will often want to extract a list of web content or follow a list of links.

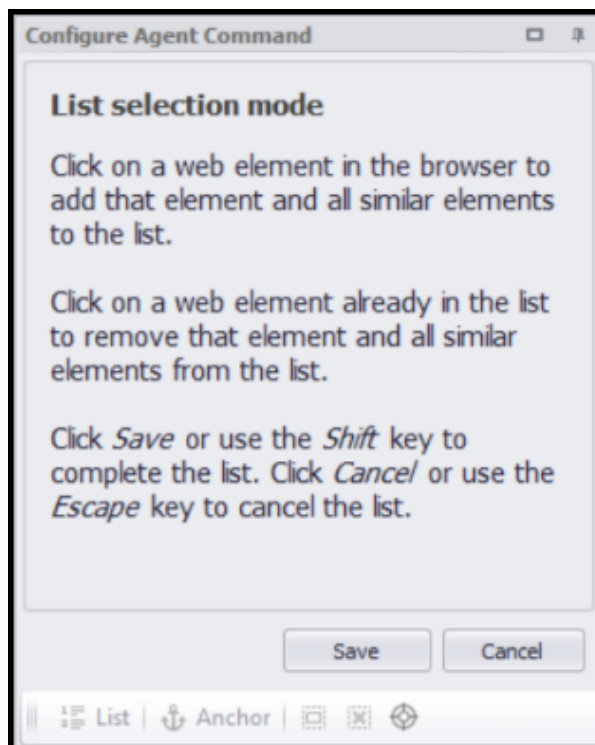
You can create list selections by editing the XPath manually, or by using the Content Grabber List Tool. You can start the List

Tool by selecting a web element and then holding down the SHIFT key while selecting more web elements that should be included in the list. Once you release the SHIFT key the list selection will be complete. You can also start the List Tool by selecting a web element and then click on List toolbar button.



The List Selection toolbar button

You can now select more web elements that should be included in your list. You can exclude web elements from the list by clicking on web elements that are already selected. You will need to click the Save button in the List mode window to complete the list.



The List Selection window

Content Grabber will not just add web elements you click on to the list, but also other web elements of the same type that are

naturally part of the same list on the page. For example, all rows in a HTML table are naturally part of the same list, so if you click on the first row in the table and then SHIFT-click on the second row, Content Grabber will create a list of all rows in the table, and not just the two rows you have clicked on.

You can only add web elements of the same type to a list, and all web elements in a list must naturally belong to the same list. For example, if you have a list of products in one area of a page, and another list of products in another area of the same page, you may not be able to add them to the same list.

Content Grabber will automatically attempt to create a natural list of the same type of web elements. If you want to create a list of table rows for example, you can select the first table row, and then SHIFT-click anywhere inside the second table row to create the list. Even if you select a table column, or any other child web element, in the second row, Content Grabber will know that it needs to create a list of table rows. It's important to notice that it DOES matter exactly which child web element you click on in the second table row, since Content grabber will use this information as a selection filter, and exclude all web elements from the list that don't contain this child web element. Content Grabber may also use the text or other properties of the child element to filter the list, but it will only do so if the filter does not filter away web elements you have specifically clicked on to add to the list.

Web Element Siblings

A list of web elements will automatically also include all siblings of the web elements. For example, you may have a list of table rows where the first row contains a product name and the next

couple of rows contain details about that product, and then another title row followed by more rows containing details about the second product, and so on. In this case you should create a list of the title rows only, and the siblings of the title rows containing product details will then be available to agent commands that references the list selection. For example, if a Web Element List command uses the list selection, then sub-commands can capture content within the sibling web elements.

Moving Through List Elements

When selecting a list command that highlights a list of web elements in the Content Grabber web browser, the web browser will scroll to the position of the first list element. If you are working on a long page, the selected list may contain web elements that are out of view. You can use the shortcut key CTRL+Right Arrow to move to the next web element in the list, and CTRL+Left Arrow to move to the previous element in the list.

4.4 Selection Anchors

Selection anchors are used to anchor a selected web element to another element, or to filter a list by a child element.

The most common use of selection anchors is when you have a list of name/value pairs on a page and want to extract the value for a specific name, but that specific name/value pair is not always located in the same position. For example, when extracting product specifications from a table, some products may have **size** and **weight** in the first two rows, but other products may have **model** and **memory** in the first two rows, so if you extract **size** based on position, you will extract **size** for

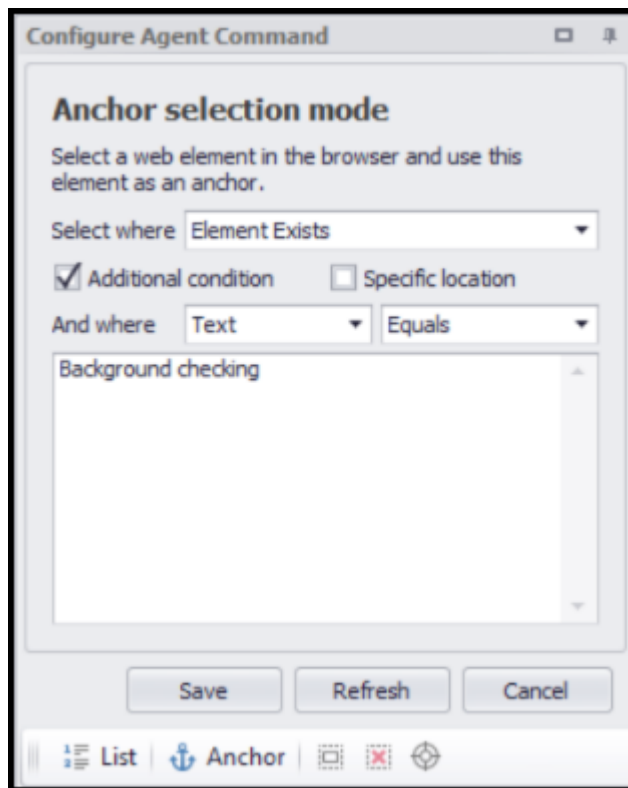
some products, but **model** for other products. To overcome this problem, you can anchor the selection of the **size** value to the text of the web element containing the **size** name.

You can anchor a selection by CTRL-clicking on the anchor element, or by clicking on the Anchor toolbar button and then selecting the anchor element. When using CTRL to select an anchor, the anchor will always be a text anchor, but when using the Anchor toolbar button, you can choose many different kinds of anchors.



The Anchor toolbar button

An anchor works as a filter when used on a list selection, and the anchor element must be selected inside the list. When using CTRL to select the anchor element, the anchor will filter the list so that only web elements that contain a child web element similar to the anchor element will be included in the list. When using the Anchor toolbar button, you can choose many different kinds of anchors to filter a list.

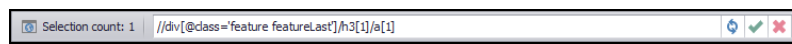


The Anchor window

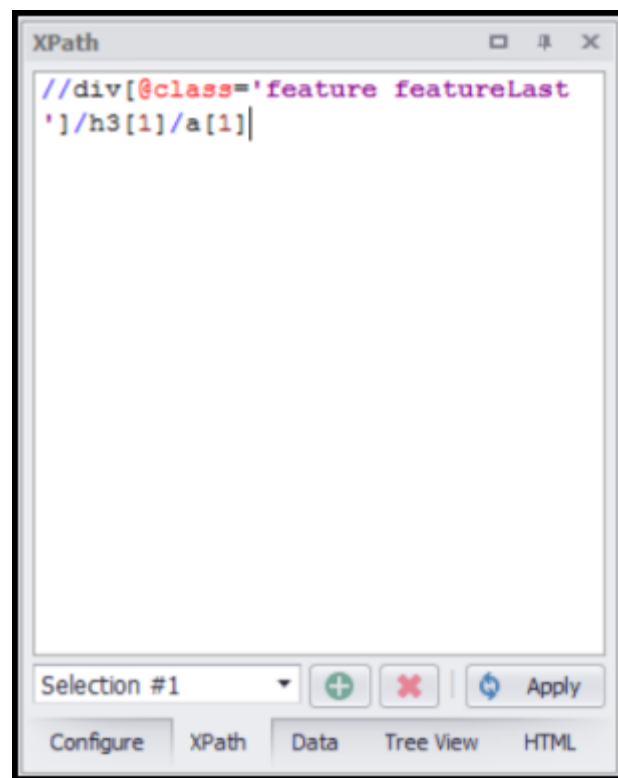
4.5 Editing XPath's Manually

Content Grabber will usually create an optimal selection **XPath**, but sometimes you may find it necessary to fine-tune the **XPath** manually. It can be difficult, however, to create an **XPath** from scratch. If you want to manually edit an **XPath**, we recommend that you first let Content Grabber generate the **XPath** and then edit it.

As we show in the figures below, you can use either the status bar **XPath** editor or the **XPath** panel to manually edit an **XPath**.



The status bar XPath editor

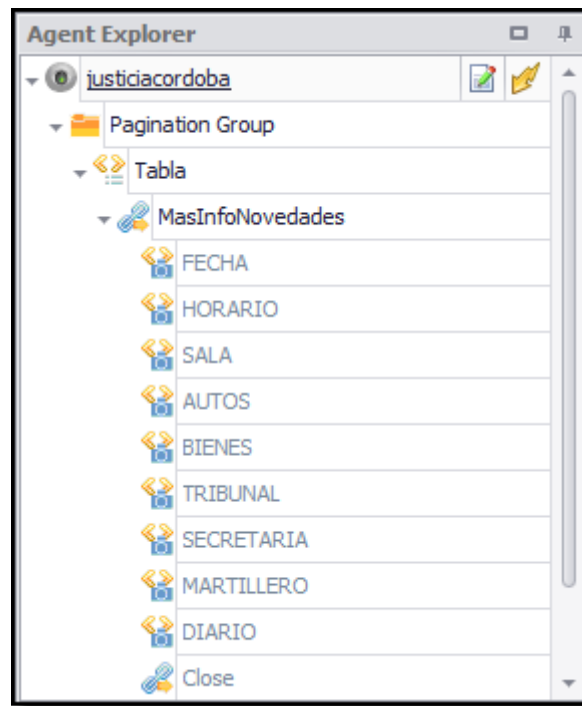


The XPath panel

5 Agent Commands

Content Grabber web-scraping agents usually contain a list of commands that execute sequentially. Sometimes, commands will execute in parallel (simultaneously). Each agent command performs an action, such as loading a new page or capturing content.

Some commands can contain sub-commands. These commands are called **container** commands and they execute all their sub-commands one or more times. If a container command executes its sub-commands more than one time, then the command is also called a list command. A list command iterates through a list of data elements or web elements and executes all its sub-commands once for each element in the list.



The Agent Explorer shows all commands in an agent

The most important command in a agent is the Agent command itself. This is the first command that executes. Since it contains all other commands for the agent, it is called a container command. The Agent Command loads the Start URL, the point at which data extraction starts. The **Agent Command** also controls many other important aspects of the agent, such as data export.

Some agent commands have a corresponding web selection that chooses an element from the current web page. One or more of these selections are put to use when the command executes. A Navigate Link command, for example, selects a link on the current web page, which will open a new web page.

See The Content Grabber Editor topic to learn how to use the **Agent Explorer** to add and configure agent commands.

Command Types

We classify agent commands into four types, according to function:

- Capture Commands
- List Commands
- Action Commands
- Container Commands

Capture commands do nothing but capture web content. **Container**, **List**, and **Action** commands may function as one or more types simultaneously. There are some special commands that don't fall into any of these 4 categories, such as the Execute Script command which simply runs a .NET script.

5.1 Container Commands

Container Commands contain one or more sub-commands. After the container command executes, all of the sub-commands execute at least once. If a container command executes its sub-commands more than once, then the command is also a List Command. A list command iterates through a list of data elements or web element, and then executes all its sub-commands - once for each element in the list.

In this chapter, we explain all of the container commands:

- Agent
- Navigate Link
- Navigate URL
- Scan Website
- Set Form Field
- Data List
- Web Element List
- Group Commands
- Group Commands in Page Area

5.2 Capture Commands

Capture commands capture data from a web page or from an input data provider, and saves the captured data to output. No other type of command can save data to output, so you need a capture command for each data field you want in your output data.

It is important to understand that some commands don't capture anything. For example, the Set Form Field command does not capture the input value that corresponds to a form field, but simply identifies the field. To capture data, you need to use one of the capture commands to acquire the input value and save it to data output.

These are all of the **Capture commands** available in Content Grabber:

- Web Element Content
- Download Image
- Download Document
- Download Screenshot

- Calculated Value
- Page Attribute
- Data Value

Data Types

Content Grabber supports the following data types for captured data.

Data Type	Description
Short Text	All content will be captured as Short Text by default. Short Text content can be up to 4000 characters long.
Long Text	Long Text content can be any length, but cannot always be used in comparisons, so you may not be able to include Long Text content in duplicate checks.
Integer	A whole number.
Float	A floating point number.
Date/Time	A date and/or time value.
Boolean	A value that can be true or false. Boolean values are stored as 1 or 0 integer values.

Extracted content will be converted to the selected data type at the time of extraction. If content cannot be converted to the chosen data type, a warning will be logged and the command will capture nothing.

When exporting data to Excel or PDF, data formatting will be set to General by default. This may cause issues when

exporting some data types, like Date/Time values. When exporting Date/Time values to Excel or PDF you should always set the `ExcelColumnFormat` property on the Capture command. The `ExcelColumnFormat` property takes the same kind of formatting string as the Custom formatting string in MS Excel. For example, the formatting string "yyyy-mm-dd" formats a Date/Time value as year, month and day.

5.2.1 Web Element Content

Typically, the **Web Element Content** command is the most common, since it is the command that captures text content from the target web page. The command allows you to choose which type of text to extract from a particular web element.

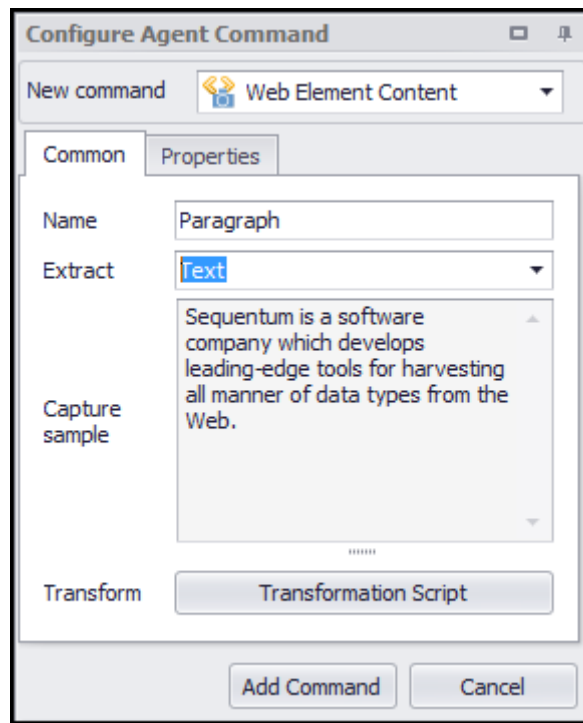
The properties shown in the following table are available to all **Web Element Content** commands:

Text Option	Description
Text	This is the text that displays in the web browser, and is the most common choice (it's also the default).
Formatted Text	This option will extract the entire HTML of the chosen web element and insert line breaks where appropriate.
HTML	This option will extract the entire HTML of the chosen web element, including the HTML of any child elements.
Clean HTML	This option will extract the HTML, but remove all attributes that are used to style the HTML.
Styled HTML	This option will extract the HTML, but remove all attributes that are used to style the HTML, except attributes used for inline styling.
Inner HTML	The entire HTML of all child elements of the selected web element, but not the tag HTML of the chosen element itself.

Tag Text	The text of the selected web element, excluding the text of any child elements.
Unique ID	A unique identifier. The web selection is ignored if this option is selected.
Default File Name	Returns any file name specified by a response from a web server. If no file name is specified, extracts a file name from a href attribute, or a unique identifier if the href does not exist. This attribute is mostly relevant to file capture commands.
Node Position	The position of the web elements among all siblings.
Position	The position of the web element among siblings of the same type.

In addition to the default options given above, some web elements may have other attributes available, such as Class, Name, ID, Value, URL, etc. If a web element has an attribute that is not shown in the default drop down box, then you can simply enter the name of the attribute you want to extract.

The figure below shows the **Command Properties** panel after choosing **Web Element Content** from the **New Command** drop-down:



Extracting URLs deserves a special mention. That's because it's more common to extract a **link URL** instead of navigating to the link, or to extract an **image URL** instead of downloading the image itself. If you want to extract a **URL** from a web element, simply select the element and extract the **URL** or **Image URL** attribute. You can also enter the actual HTML tag attributes: **src** for images and **href** for links. However, the **URL** and **Image URL** attributes automatically convert any relative URLs to absolute URLs-which is best in most cases.

Content Transformation Script

The **Web Element Content** command allows you to use regular expressions or a .NET script, to transform the extracted content. In most cases we recommend that you write expressions or use a script to clean the data that you extract. You can also separate data-such as the elements of a postal address-into separate fields.

Example: Consider a case in which you want to extract product data that includes a price of \$400. You could use a transformation script to strip off the "\$" character and leave only the numeric value.

Please see the Content Transformation Script topic for more information.

5.2.2 Download Image

The **Download Image** command extracts an image from a web page. The command will download an image, and then save it to the file system, send it to a database, or embed it into Excel - depending on your chosen export target. The web selection path for this command normally points to the image itself, but it could also point to a web element that contains a URL that links to the image.

The figure below shows the **Command Properties** panel after choosing **Download Image** from the **New Command** drop-down:

The screenshot shows the 'Configure Agent Command' dialog box. At the top, the title bar reads 'Configure Agent Command'. Below the title bar, there is a 'New command' dropdown menu with 'Download Image' selected. The dialog has two tabs: 'Common' and 'Properties', with 'Properties' currently active. Under the 'Properties' tab, there is a 'Name' field containing 'H3 Content'. Below this, there are three sub-tabs: 'File URL', 'File Name', and 'OCR', with 'File URL' selected. The 'File URL' sub-tab contains an 'Extract' dropdown menu with 'Image URL' selected. Below the 'Extract' dropdown is a large, empty text area. At the bottom of the dialog, there is a 'Transform' section with a 'Transformation Script' button.

Data Fields

If the agent is saving the image to a database, then by default this command will generate two data fields: one for the image data and another for the name of the image. If the agent is saving the image to the file system, the command will generate only one data field containing the full file path to the image. The command property **Export URL** can be used to also generate a data field that contains the image URL.

Command Configuration

The **Common** tab in the **Configure Agent Command** panel has three tabs:

- **File URL** - contains the URL for the image.
- **File Name** - contains the name of the downloaded image.
- **OCR** tab - check the box if you want to convert an image into text.

We explain the details of each below.

File URL

The entry in this tab determines the specific URL for the image, and the agent uses this URL to download the image at run time. You can choose the HTML attribute that the command should extract to get the image URL. The default value is **Image URL**, which extracts the **src** HTML attribute.

Click the **Transformation Script** button click to enter regular .NET expressions or write a .NET script that will transform the image URL to meet your requirements. This is often useful when you want to extract a large image, but it's easier to select a corresponding thumbnail image. You may then be able to transform the thumbnail image URL into the large image URL, and the agent will then use the transformed image URL to download the large image.

See the Content Transformation Script topic for more information about content transformation scripts.

You can choose the HTML attribute that the command should extract to get the image URL. The default value is **Image URL**, which extracts the **src** HTML attribute.

File Name

The entry in this tab contains the file name. From the drop-down menu you can choose the HTML attribute that you want to use as the name.

Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the image name to meet your requirements. See the Content Transformation Script topic for more information about content transformation scripts.

Use the **Data Value** option to specify that an agent data value will be used as file name. The agent data can come from a data provider, an input parameter or captured data.

Use the **Detect File Extension** option to specify if agent should try and detect the file type of the downloaded document, or if a transformation script or a data value will provide a file name that includes a file extension.

OCR

Using the **OCR** tab, you have the option to convert the image into text. For example, you might need to convert CAPTCHA images into text so that the agent can bypass CAPTCHA blocking websites. See the topic CAPTCHA Blocking for more information.

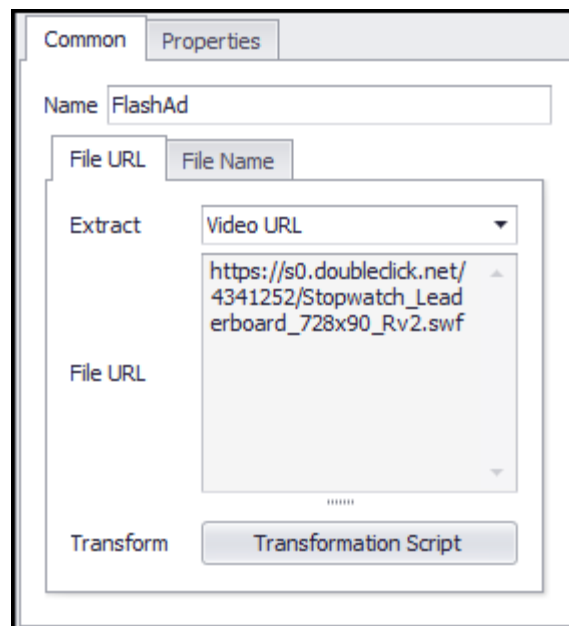
This tab gives you the option to export both the image file and the converted text, or just the converted text. To convert the image, you'll need to check the box and then enter a script to call an external OCR service. Content Grabber does not include an OCR feature, but allows you to integrate with 3rd party services by using this script feature. For more information, see the topic Image OCR Scripts.

5.2.3 Download Video

The **Download Video** command will download a video from a website and save it to your local disk or a database. The most common video formats are **mp4**, **ogg** and **Flash**. Content Grabber cannot extract content from within a video (such as Flash objects), but can only extract the video file as a whole.

The command has a corresponding web selection that will normally select the link to the video. Optionally, the command can extract data from a web element and then calculate the direct video URL.

The figure below shows the **Command Properties** panel after choosing **Download Video** from the **New Command** drop-down:



Data Fields

If the agent is saving the video to a database, then by default this command will generate two data fields: one data field will contain the video binary data and another will contain the name of the video. If you're saving the video to the file system, the command will generate only one data field containing the full file path to the video file.

You can also use the property **Export URL** to generate a data field that contains the video URL.

Command Configuration

The **Common** tab in the **Configure Agent Command** panel has two tabs:

- **File URL** - contains the URL for the video.
- **File Name** - contains the name of the downloaded video.

We explain the details of each below.

File URL

The entry in this tab determines the specific URL for the video, and the agent uses this URL to download the video at run time. You can choose the HTML attribute that the command should extract to get the video URL. The default value is **Video URL**, which will cause the command to look at the entire tag HTML and extract the first appropriate video URL.

Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the document URL to meet your requirements.

See the Content Transformation Script topic for more information about content transformation scripts.

File Name

The entry in this tab contains the file name. From the drop-down menu, you can choose the HTML attribute that you want to use as the name.

Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the document name to meet your requirements. See the

Content Transformation Script topic for more information about content transformation scripts.

Use the **Data Value** option to specify that an agent data value will be used as file name. The agent data can come from a data provider, an input parameter or captured data.

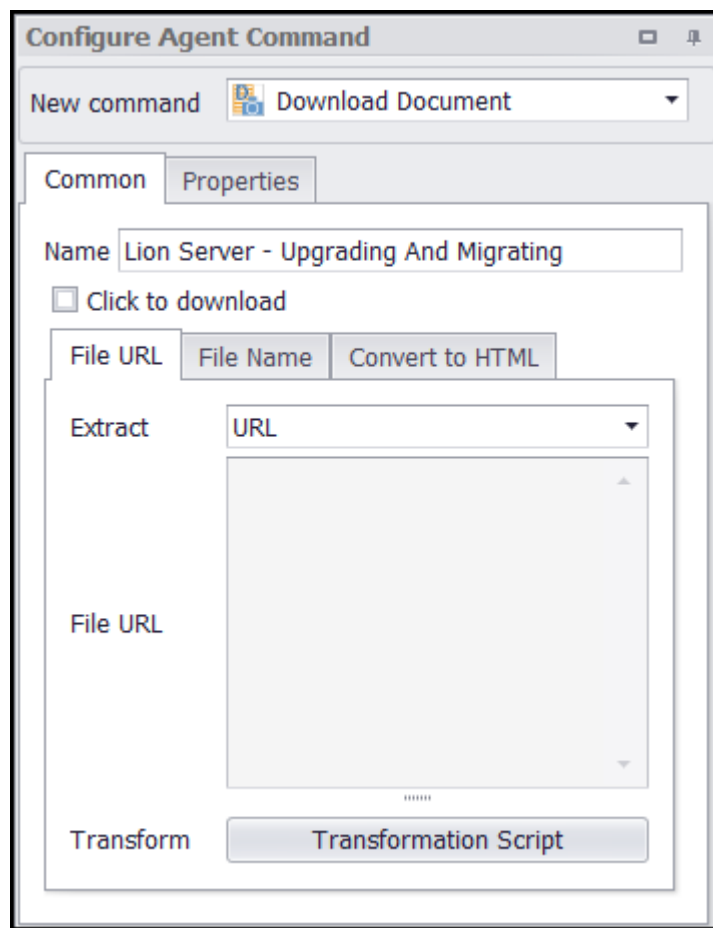
Use the **Detect File Extension** option to specify if agent should try and detect the file type of the downloaded document, or if a transformation script or a data value will provide a file name that includes a file extension.

5.2.4 Download Document

The **Download Document** command extracts a document from a web page. The command will download a document, and then save it to the file system or send it to a database - depending on your chosen export target.

The web selection path for this command normally points to the document link itself, but it could also point to a web element that contains information from which the document URL can be derived by using **Content Transformation**.

The figure below shows the **Command Properties** panel after choosing **Download Document** from the **New Command** drop-down:



Data Fields

If the agent is saving the document to a database, then by default this command will generate two data fields: one for the document binary data and another for the name of the document. If the agent is saving the document to the file system, then the command will generate only one data field containing the full file path to the document. The command property **Export URL** can be used to also generate a data field that contains the document URL.

Command Configuration

The **Common** tab in the **Configure Agent Command** panel has three tabs:

- **File URL** - contains the URL for the image.
- **File Name** - contains the name of the downloaded image.

- **Convert to HTML** - specifies if the downloaded document should be converted to HTML.

We explain the details of each below.

File URL

The entry in this tab determines the specific URL for the image, and the agent uses this URL to download the document at run time.

You can choose the HTML attribute that the command should extract to get the document URL. The default value is **URL**, which extracts the **href** HTML attribute (if the chosen web element is a link).

Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the document URL to meet your requirements. See the Content Transformation Script topic for more information about content transformation scripts.

Use the **Data Value** option to specify that an agent data value will be used as file URL. The agent data can come from a data provider, an input parameter or captured data.

File Name

The entry in this tab contains the file name. From the drop-down menu, you can choose the HTML attribute that you want to use as the name.

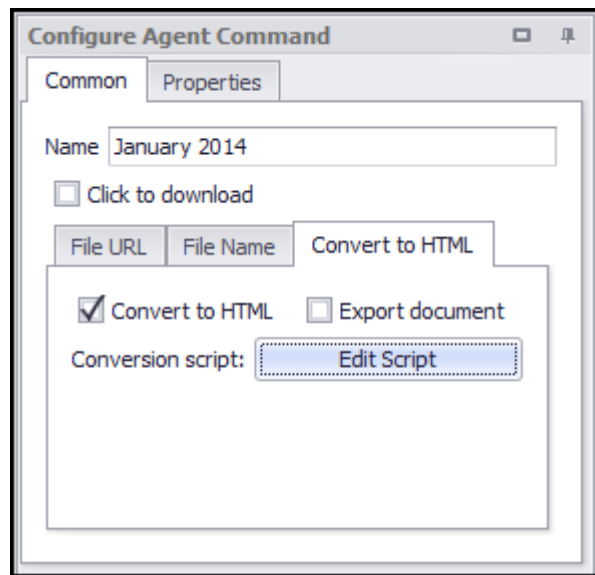
Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the document name to meet your requirements. See the Content Transformation Script topic for more information about content transformation scripts.

Use the **Data Value** option to specify that an agent data value will be used as file name. The agent data can come from a data provider, an input parameter or captured data.

Use the **Detect File Extension** option to specify if agent should try and detect the file type of the downloaded document, or if a transformation script or a data value will provide a file name that includes a file extension.

Convert To HTML

A downloaded document can be converted into a HTML page as it's being downloaded, and a URL command can later be used to open the HTML page. Capture commands can then be used to extract data from the HTML page.



Please see [Extracting Data From Non-HTML Documents](#) for more information.

Click To Download

Check this box when no direct URL is available for the document, but it is necessary to download the document by clicking on a web element - such as a button. When you enable this option:

- The **File URL** tab becomes unavailable
- Content Grabber will assign a unique identifier as the file name
- You will have access to the **Action** configuration tab where you can fine tune the behavior of the action.

See the topic Action Commands for more information.

5.2.5 Download Screenshot

The **Download Screenshot** command extracts a screenshot of the entire webpage or the chosen web element. The command will save the screenshot, save it to the file system or send it to a database - depending on your chosen export target. The web selection path for this command normally points to the document itself, but it could also point to a web element that contains a URL that links to the document.

The figure below shows the **Command Properties** panel after choosing **Download Screenshot** from the **New Command** drop-down:

The screenshot shows the 'Configure Agent Command' window. At the top, there's a 'New command' dropdown menu with 'Download Screenshot' selected. Below this are two tabs: 'Common' and 'Properties'. The 'Common' tab is active. Inside the 'Common' tab, there's a 'Name' field with the value 'H3 Content'. Below that, there's a 'File Name' sub-tab which is selected. Under 'File Name', there's an 'Extract' dropdown menu with 'Unique ID' selected. Below the dropdown is a list box containing two unique IDs: '44b0f76f664e43488068e61dc85846d5'. At the bottom of the 'Common' tab, there's a 'Transform' button and a 'Transformation Script' button.

Data Fields

If the agent is saving the screenshot to a database, then by default this command will generate two data fields: one for the screenshot data and another for the name of the screenshot. If the agent is saving the screenshot to the file system, the command will generate only one data field containing the full file path to the screenshot.

Command Configuration

The **Common** tab in the **Configure Agent Command** panel has two tabs:

- **File Name** tab - contains the name of the downloaded image.
- **OCR** tab - check the box if you want to convert an image into text.

We explain the details of each below.

File Name

The entry in this tab contains the file name. From the drop-down menu, you can choose the HTML attribute that you want to use as the name.

Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the screenshot name to meet your requirements. See the Content Transformation Script topic for more information about content transformation scripts.

Use the **Data Value** option to specify that an agent data value will be used as file name. The agent data can come from a data provider, an input parameter or captured data.

OCR

Using the **OCR** tab, you have the option to convert the screenshot into text. This tab gives you the option to export both the screenshot file and the converted text, or just the converted text. To convert the screenshot, you'll need to check the box and then enter a script to call an external OCR service. **Content Grabber** does not include an OCR feature, but allows you to integrate with 3rd party services by using this script feature. For more information, see Image OCR Scripts.

5.2.6 Download Page

The **Download Page** command extracts the entire web page as a PDF document or as HTML. Only the entire web page can be captured, not specific web elements on the web page.

The figure below shows the **Command Properties** panel after choosing **Download Page** from the **New Command** drop-down:

The screenshot shows a dialog box for the 'Page to PDF' command. It has two tabs: 'Common' and 'Properties'. The 'Properties' tab is selected. Inside the 'Properties' tab, there is a 'Name' field with the value 'New'. Below it is a 'Document type' dropdown menu set to 'PDF'. A section titled 'File Name' contains an 'Extract' dropdown menu set to 'Default File Name', a 'Data value' checkbox, and a text area with the text 'Web-Scraping-Solutions-Sequentum.pdf'. At the bottom of the 'File Name' section is a button labeled 'Transformation Script'. At the bottom of the dialog are 'Save' and 'Cancel' buttons.

Page to PDF command.

Data Fields

If the agent is saving the PDF or HTML to a database, then by default this command will generate two data fields: one for the PDF or HTML data and another for the name of the PDF or HTML file. If the agent is saving the PDF or HTML to the file system, the command will generate only one data field containing the full file path to the PDF or HTML file.

File Name

From the drop-down menu, you can choose the page attribute that you want to use as the name.

Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the PDF file name to meet your requirements. See the

Content Transformation Script topic for more information about content transformation scripts.

Use the **Data Value** option to specify that an agent data value will be used as file name. The agent data can come from a data provider, an input parameter or captured data.

5.2.7 Calculated Value

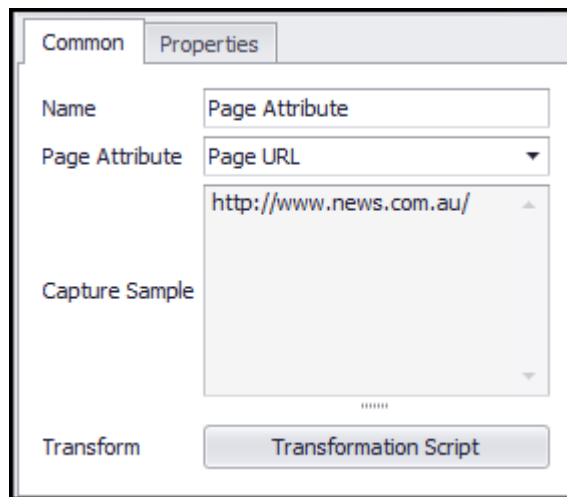
This command will save a static value to a data output. You can specify the value at design time, or as an Input Parameter value that will be set at run time.

The screenshot shows a dialog box titled 'Calculated Value' with two tabs: 'Common' and 'Properties'. The 'Common' tab is selected. Inside the dialog, there is a 'Name' field containing 'Calculated Value'. Below it is a checkbox labeled 'Use input parameter' which is unchecked. Underneath the checkbox is a text area labeled 'Fixed value' containing the number '12'. At the bottom of the dialog, there are two buttons: 'Transform' and 'Transformation Script'.

Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the value to meet your requirements. See the Content Transformation Script topic for more information about content transformation scripts.

5.2.8 Page Attribute

Use this command to extract general attributes from the current web page, such a page URL or page meta data.



The screenshot shows a configuration window with two tabs: 'Common' and 'Properties'. The 'Properties' tab is active. It contains the following fields and controls:

- Name:** A text box containing 'Page Attribute'.
- Page Attribute:** A dropdown menu currently showing 'Page URL'.
- Capture Sample:** A large text area containing the URL 'http://www.news.com.au/'.
- Transform:** A button located at the bottom left.
- Transformation Script:** A button located at the bottom right.

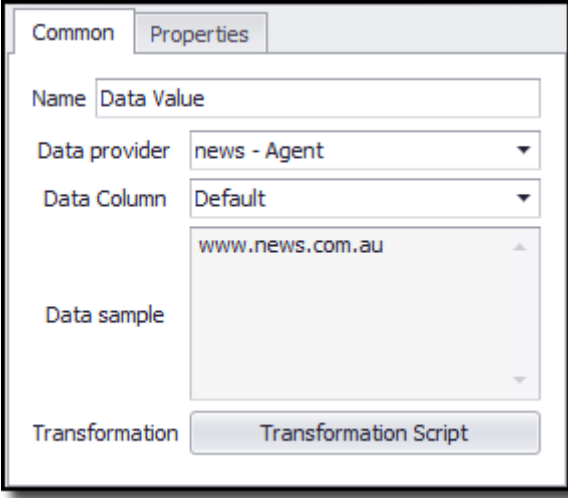
A command that will extract the URL of the current web page

Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the attribute value to meet your requirements. See the Content Transformation Script topic for more information about content transformation scripts.

5.2.9 Data Value

Use this command to extract data from a provider and save it to a data output. The command can save values from data provider commands, such as **URL** commands or **Form Field** commands. For example, if you are using a Set Form Field command to provide input values to a web form, then these input values will not automatically save to a data output unless you use a **Data Value** command.

See Using Data Input for more information about data providers.



The screenshot shows a dialog box with two tabs: 'Common' and 'Properties'. The 'Properties' tab is active. It contains the following fields and controls:

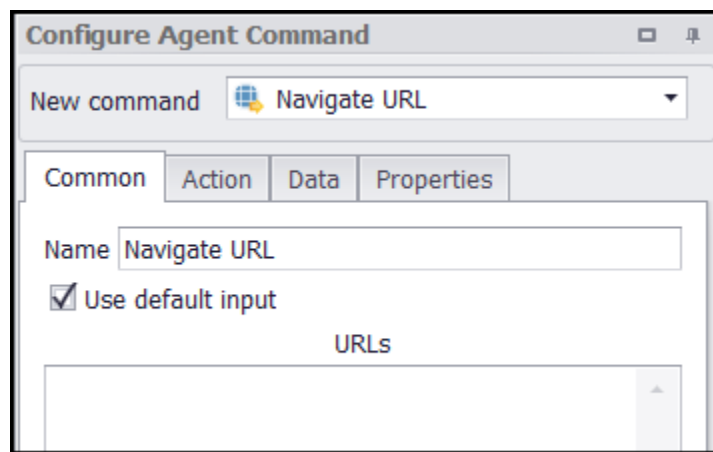
- Name:** A text box containing 'Data Value'.
- Data provider:** A dropdown menu with 'news - Agent' selected.
- Data Column:** A dropdown menu with 'Default' selected.
- Data sample:** A text area containing 'www.news.com.au'.
- Transformation:** A section with a button labeled 'Transformation Script'.

This command extracts the URL given by the parent agent command

Click the **Transformation Script** button to enter regular .NET expressions or write a .NET script that will transform the attribute value to meet your requirements. See the Content Transformation Script topic for more information about content transformation scripts.

5.3 Action Commands

An **Action command** executes an action in the web browser, such as opening a new web page. Most action commands are also container commands, and normally contain the commands that should execute after the web browser action. If, for example, an action command opens a page in a new web browser, the sub-commands will execute in the context of that new web page.



These are the **Action** commands available in Content Grabber:

- Agent
- Navigate Link
- Navigate URL
- Navigate Pagination
- Set Form Field

Below we summarize each of the configuration tabs for **Action commands**, and provide links to each of the subtopics.

Configuration of an Action Command

Explore the options and properties that you can configure for a command by taking these simple steps:

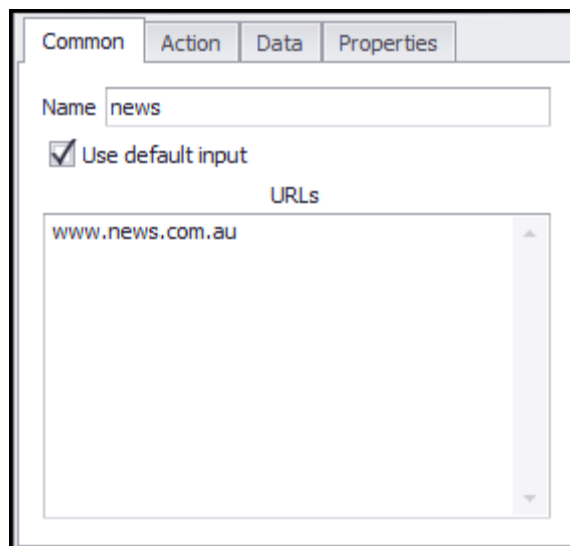
1. Clicking once on a web element in the browser panel.
2. Locate the **New Command** drop-down in the **Configure Agent Command**.
3. Choose a command from the drop-down.
4. Explore the tabs: **Common**, **Action**, **Data** and **Properties**.
5. In the **Common** tab, uncheck the **Use default input** box to reveal more options.
6. In the **Action** tab, uncheck the **Discover Action** box to reveal more options.

In many web-scraping scenarios, the default functionality will be quite sufficient. Should you find the need for more flexibility and control, you can learn how to configure all aspects and properties in the Action Configuration section.

5.3.1 Agent

The **Agent command** is the first command that executes in an agent; all other commands are sub-commands. So, only one **Agent** command can exist in an agent. The **Agent command** loads the start URL, which is the first point of data extraction, and also contains all common agent properties (including data export configuration).

The **Agent** command uses a data provider that provides one or more start URLs, and the command will execute once for each of these URLs.



This Agent command uses a simple data provider to load a single static start URL

NOTE: The **Agent command** derives from the Navigate URL command, which loads one or more URLs.

Command Configuration

The configuration screen for the **Agent** command has four tabs: **Common**, **Action**, **Data**, and **Properties**. We explain the **Properties** in the sections below. In the **Common** tab, you can edit the command name and (optionally) customize the data provider properties.

CSV Data: If you leave a check in the **Use Default Input** box, the command will provide simple CSV data and use that data as input. Simple CSV data consists of values that you enter directly into the command, so no external CSV is necessary.

You can uncheck the **Use Default Input** box and choose the **Data Provider** that will provide the start URLs. The default data provider is a simple data provider that provides a list of static URLs. You can populate the data provider directly by entering the start URLs in the URLs input box.

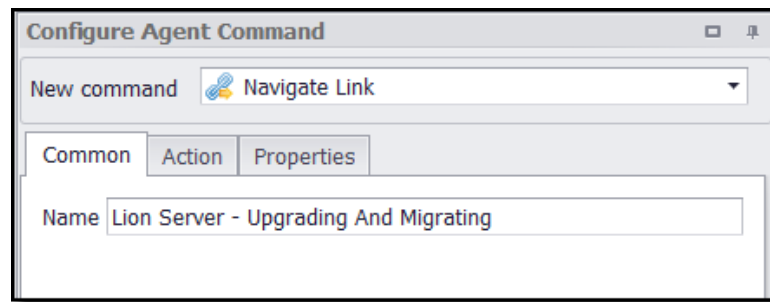
Use the **Action** tab to control how the web browser loads the start URLs. See Action Configuration for more information.

Use the **Data** tab to set the data provider that provides the start URLs. Read more in Using Data Input.

5.3.2 Navigate Link

This command executes an action, such as a mouse click, on a web element. You can apply the **Navigate Link** command to any kind of web element, including links, buttons, and menus.

The figure below shows the **Configure Agent Command** panel after choosing **Navigate Link** from the **New Command** drop-down:



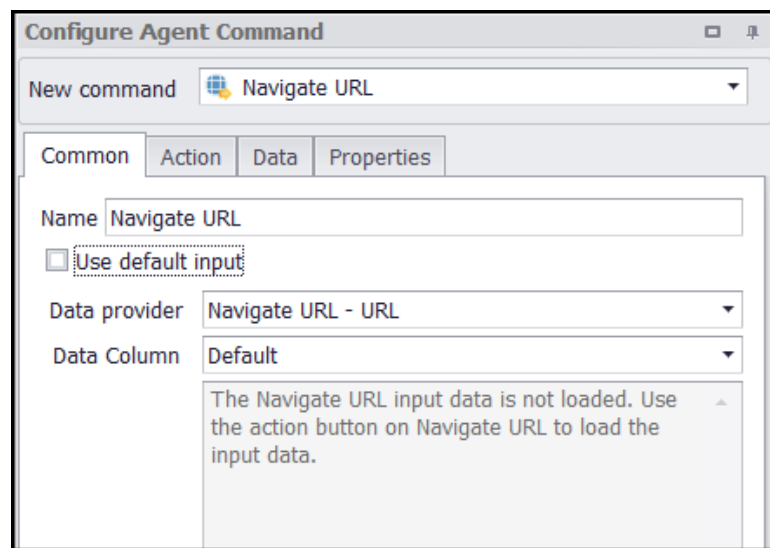
Command Configuration

The configuration screen for the **Navigate Link** command has three tabs. **Common**, **Action**, and **Properties**. We explain the **Properties** in the sections below. Use the **Common** tab to set the command name. Use the **Action** tab to control how the command chooses the web element and loads new content into the web browser. See Action Configuration for more information.

5.3.3 Navigate URL

This command opens a new URL and loads a web page into the web browser. You can specify a single URL, a list of URLs, or intake URLs from an input data source such as a database or CSV file.

The figure below shows the **Configure Agent Command** panel after choosing **Navigate URL** from the **New Command** drop-down:



Command Configuration

The configuration screen for the **Agent** command has four tabs: **Common**, **Action**, **Data**, and **Properties**. We explain the **Properties** in the sections below. In the **Common** tab, you can edit the command name and, optionally, customize the data provider properties.

You can uncheck the **Use default input** box and choose the **Data provider** that will provide the start URLs. The default data provider is a simple data provider that provides a list of static URLs. You can populate the data provider directly by entering the start URLs in the URLs input box.

Use the **Action** tab to control how the web browser loads the start URLs. See Action Configuration for more information. Use the **Data** tab to set the data provider that provides the start URLs. Read more in Using Data Input. The **Navigate URL** command can use the data provider configured on the **Data** tab, or it can use a data provider configured by a parent command. The **Data** tab is not available if the command uses a data provider from a parent command. See Using Data Input for more information about data providers.

Posting Data and Custom Header

The **Navigate URL** command supports URLs with custom parameters that allow you to post data and add custom headers, which is especially useful if you want to load a web page that is the result of a web-form submission. Sometimes, you can request such a web page by putting the form field values on the URL as normal query parameters; but other times the website requires that you post the form field values.

URL Parameters

You can use the following three URL parameters to specify post data, custom headers and request method.

Parameter	Description
??Post=	Specifies data that should be posted to the website.
??Headers=	Specifies additional headers that should be sent to the website. NOTE: This parameter must come <i>after</i> the ??post parameter.
??Method=	The following request methods are supported: GET, POST, DELETE, PUT, PATCH POST is used automatically when no method is specified and the post parameter is not empty.

Example

The URL in this example would post *par2=val2&par3=val3* to the website, while *par1=val1* is just a normal query parameter.

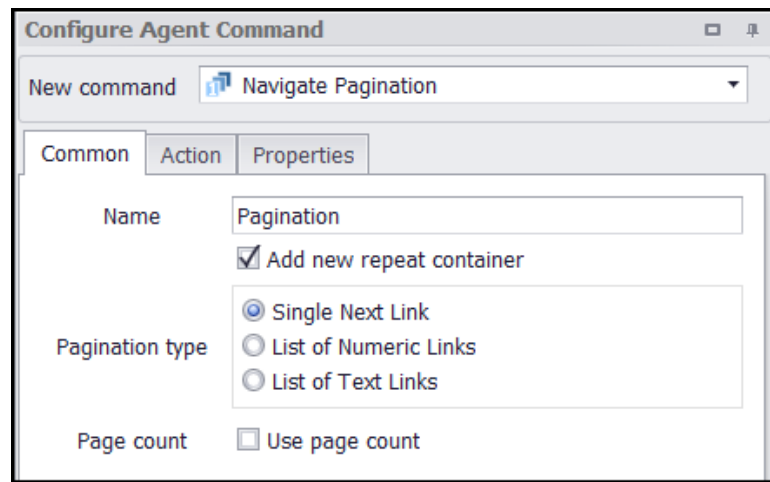
```
http://www.sequentum.com?par1=val1??  
post=par2=val2&par3=val3??headers  
=Content-Type: application/x-www-form-  
urlencoded
```

NOTE: When posting value pairs to a website, you must also add the header *Content-Type: application/x-www-form-urlencoded*. Otherwise the website will not understand the format of the posted data.

5.3.4 Navigate Pagination

Typically, pagination is the feature that gives the user the ability to navigate among multiple pages. The most common example of pagination is a website that displays a search results listing that spans multiple pages. This command follows all pagination links on a web page, and will process a specific container command after opening each pagination link.

The figure below shows the **Configure Agent Command** panel after choosing **Navigate Pagination** from the **New Command** drop-down:



Pagination takes several different forms, but in nearly all cases it will fall under one of the following categories:

- **Next Page** link/button
- List/sequence of numeric links
- List/sequence of text links.

We cover each of these in the sections below.

Single Next Page Link

We recommend that you use the **Single Next Link** pagination type on a website that uses a single link to move to the next page. In addition to a single **Next Page** link, a website may provide a sequence of pagination links. In such cases, you should still choose this pagination type, which will ignore any of the other pagination links. It's only necessary that Content Grabber can move to the next page in the sequence. So, it's more efficient to identify the **Next Page** link.

Content Grabber will continue to follow the **Next Page** link until it no longer appears on the next results page - or it becomes inactive. In very rare cases, a website may continue to display an active **Next Page** link even

when there are no more pages available, and may link back to the first page in the sequence or continue to load the last page.

You can check the box for **Use Page Count** and then configure the agent to stop the pagination command when it reaches the maximum number of pages. The **Page Count** property requires you to specify the Capture Commands that delivers the maximum page number. Most websites that display a search result will also display the total number of available pages in the search result, so you can choose the Web Element Content command to capture the total number of pages. Then, use this command to place a limit on the number of pages that the pagination command will process.

If you want to stop pagination on a fixed page, you can add a **Calculated Value** command that contains the fixed page number, and then use that command as the command that delivers the maximum page number.

List of Numeric Links

This pagination type is the best choice for a website that only displays a sequence of pagination links but lacks a **Next Page** link. Page set navigation links are often found together with the numeric links. For example, if a you find 10 numeric links, the page set navigator links allows a user to navigate to the next 10 numeric links or the previous 10 numeric links. This pagination type requires one web selection for the numeric links and, if applicable, another selection for the page set navigator link that moves to the next set of numeric links.

When Content Grabber processes a list of numeric links, it will extract a specific attribute of the chosen web elements and then convert the content into numeric values. If the agent cannot convert the content into a numeric value, it ignores the corresponding link. Content Grabber expects a sequence of links, so it will continue to the next page number in the sequence after it processes the current page number. You can use a

content transformation script to convert the content into a number. For example, if the agent cannot convert the text attribute of a pagination link directly into a number, it may look for the page number in the URL attribute. In this case, a transformation script would be necessary to transform the URL into a number.

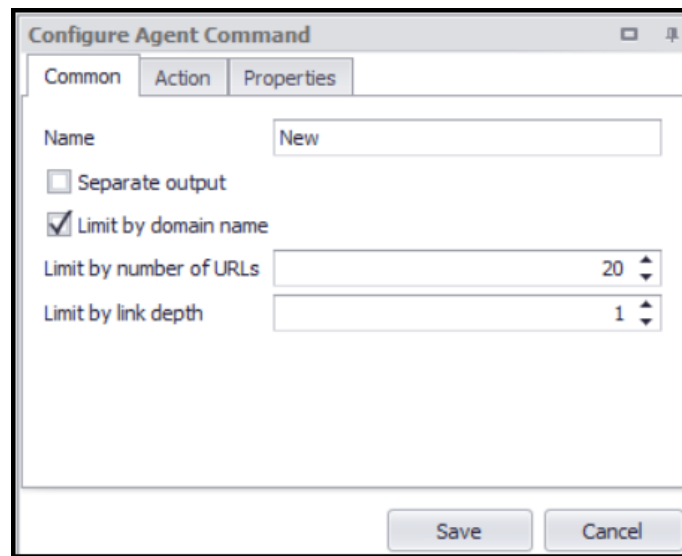
List of Text Links

Use this pagination type on a website that features a list of pagination links but none of the attributes of the link elements will convert into sequential numbers. A common application of this pagination type is on a website that displays items according to the first letter of the item name. In cases like this, the pagination would consist of all the letters in the alphabet (A-Z), and when a user clicks on a letter, all items starting with that letter will appear in the listing.

5.3.5 Crawl Website

The **Crawl Website** command Crawls a website by following all links found on the website. The number of links can be limited by domain name, link depth or number of URLs. The command selects all links by default, but the web selection can be modified, so it only selects specific links.

The **Crawl Website** command is often used to scan a website for certain information, such as emails, addresses and phone numbers.



Crawl Website configuration screen.

The **Crawl Website** command provides the data fields **URL** and **Depth**, which can be captured with a **Data Value** command.

The **Crawl Website** command uses a HTML Parser, not a Dynamic Browser, by default. This can be changed in the Action tab, but a HTML Parser is recommended to ensure good performance.

5.3.6 Set Form Field

Use this command to set input fields in a web form. The command supports the following types of web form fields:

- Input fields
- Select boxes
- Text area fields
- Check boxes
- Radio buttons.

Note: Some web sites use advanced JavaScript to make web elements emulate standard form fields, and the **Set Form Field** command does not support those web elements.

Use a normal Navigate Link command to click on a submit button to submit the web form. The submit button can be a standard web form submit button or any other web element that uses JavaScript to submit the web form. A Navigate Link command is not required if one of the form fields automatically submits the web form when an input value changes.

default data provider for standard text box

default data provider for radio button

default data provider for a select box

Command Configuration

The configuration screen for the **Set Form Field** command has four tabs - **Common**, **Action**, **Data** and **Properties**. Use the **Common** tab to set the command name and select the data provider that provides the input values for the form field. The default data provider for a standard input form field or a radio button is a simple data provider that provides the input values. The input values for a radio button or check box must be either **On** or **Off**. The default data provider for a select box form field is a selection data provider. The default selection data provider will attempt to select all appropriate values in the drop-box (select-box), and the **Set Form Field** command uses those values as input values.

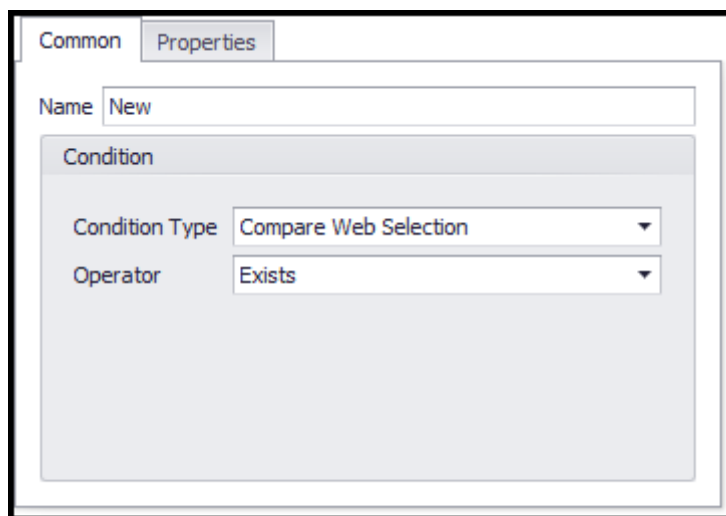
Use the **Action** tab to control how the web browser loads the URLs. See Action Configuration for more information.

Use the **Data** tab to configure a data provider that can provide the form field input values. The **Set Form Field** command can use the data provider configured on the **Data** tab, or it can use a data provider configured by a parent command. The **Data** tab is not available if the command uses a data provider from a parent command. See Using Data Input for more information about data providers.

5.3.7 Manual Navigation

The **Manual Navigation** command pauses an agent and allows the end-user to navigate manually in the web browser. When the end-user has finished navigating, he can press a **Continue** button to continue running the agent.

A condition can be specified, so the the agent only pauses when that condition is true.



The screenshot shows a configuration window for a 'Manual Navigation' command. It has two tabs: 'Common' and 'Properties', with 'Properties' currently selected. Under the 'Properties' tab, there is a 'Name' field containing the text 'New'. Below this is a 'Condition' section, which is expanded to show two dropdown menus. The first dropdown, labeled 'Condition Type', is set to 'Compare Web Selection'. The second dropdown, labeled 'Operator', is set to 'Exists'.

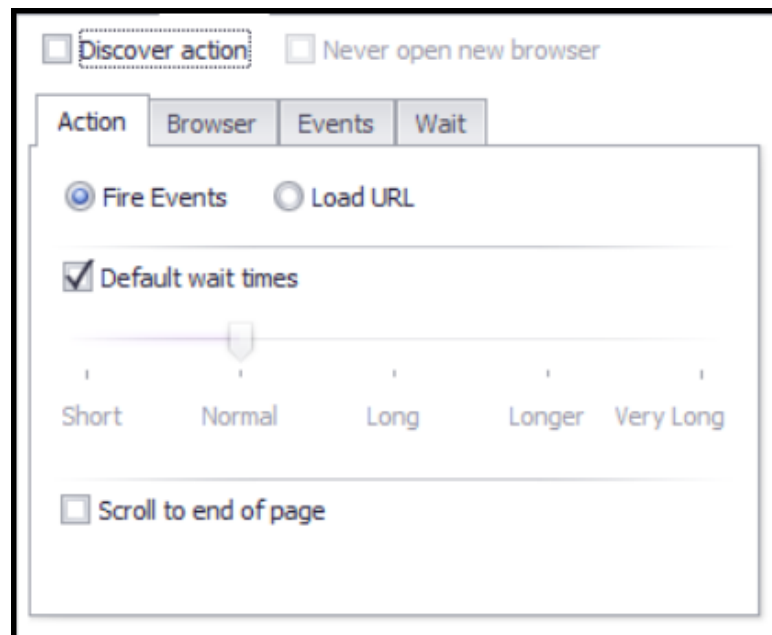
A Manual Navigation command configured to pause the agent when the associated web selection exists.

A **Manual Navigation** command is often used to navigate sections of a website that cannot be automatically processed, such as advanced CAPTCHA pages.

5.3.8 Action Configuration

By default, all action commands are set to **Discover action** settings (the Discover action check box on the **Action** tab is checked). The action settings will automatically be configured when you execute the command the first time. The default action settings are usually quite sufficient, so you will rarely need to worry about the **Action** configuration tab at all. However, you can fine-tune the action settings to achieve better performance, and you may need to adjust the action settings to get the command working correctly.

After choosing a **New Command** type from the drop-down in the **Configure Agent Command** panel, click the **Action** tab.



By default, all action commands are set to automatically discover action settings when you execute the command for the first time. The settings are usually suitable for most scenarios, so you will rarely need to worry about

the **Action** configuration tab at all. After some experience, you may want to fine-tune these settings to achieve better performance, and sometimes it may be necessary to adjust the action settings to get the command to function according to your precise requirements.

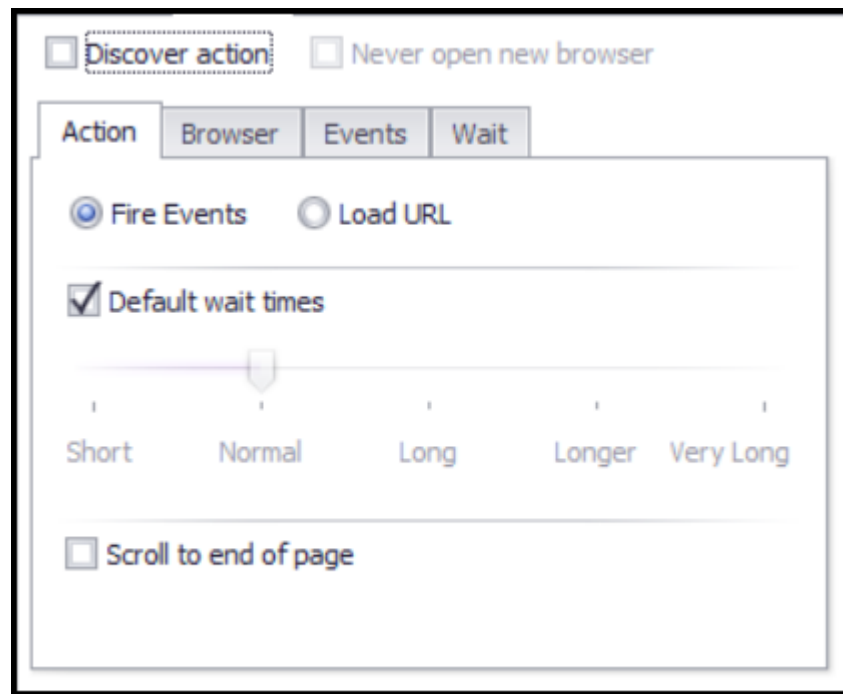
There are several configuration tabs that are available for **Action** commands in the **Configure Agent Command** panel, including:

- **Action** - Specifies the type of action, which can be a **Fire Event**, **URL**, or **No Action**. Not all action types are available for all action commands.
- **Browser** - Specifies the web browser in which the new content should load.
- **Events** - For action set to Fire Events only, this is a list of events that will fire on the chosen web element.
- **Wait** - Specifies the browser activities for which the command will wait before the action is complete.

5.3.8.1 Action

Action commands can execute one of two types, or no action at all:

- **Fire Events** - The command fires events, such as a mouse click, on the selected web element.
- **Load URL** - The command loads a new page into the web browser using a direct URL.
- **No Action** - The command does not execute an action. This is only relevant for form fields which may execute an action when an input values is assigned, but often execute no action at all.



This action command will fire events

Wait Times

An action command uses Wait Time values to determine how long it should wait for activities, such as how long it should wait for a new page to start loading. If you decrease the Wait Time values, the agent will run faster, but it may not work correctly if the website is slow. If the website is very slow you may need to increase the Wait Time values to make sure the agent works correctly.

If the option **Default wait times** is checked, the command will use the same Wait Time values as the parent action command.

Scroll to End of Page

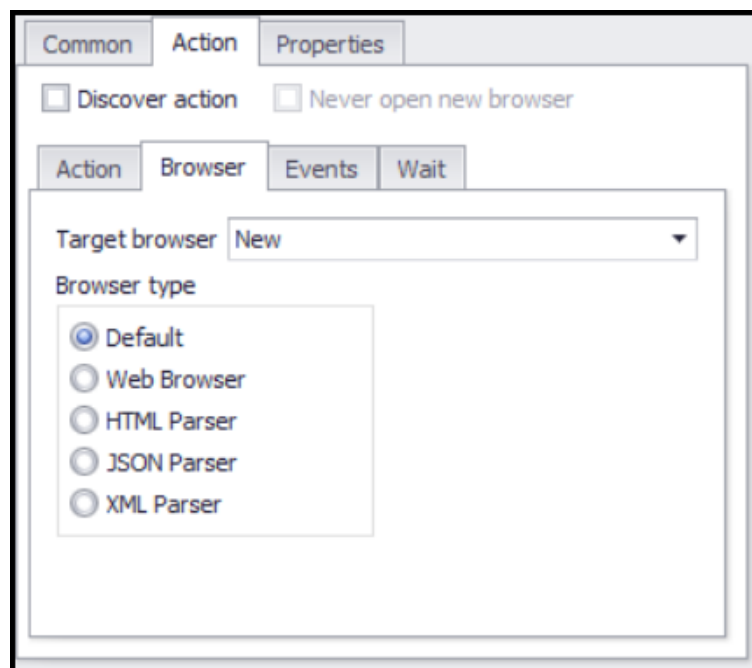
Some web pages load additional content when you scroll the page - either downward or to the right. To extract all content from such pages, you need to include an action that scrolls

down to the end of the page, so all content is available to the agent.

When you set the option **Scroll to end of page**, you will be able to limit the number of times the command scrolls to the end of a page to load new content. This can be important, since some pages will continue loading new content for a single page until Content Grabber finally runs out of memory.

5.3.8.2 Browser

An action command often loads new content. With the **Browser** action type, you can configure how content loads into a new web browser, the current web browser, or the parent web browser. You can also specify a different browser mode, as we explain below.



This Navigate Link command will open a web page in a new web browser

Uncheck the **Discover Action** box and then click the Browser tab to view these for the **Target Browser**:

- **New.** The action command loads content into a new web browser, and all sub-commands will operate in the new browser. To use this option, the action command must load a completely new page. Asynchronous actions, such as AJAX calls, cannot load content into a new web browser.
- **Current.** The action command loads content into the current web browser, and all sub-commands will operate in the new browser. Asynchronous actions, such as AJAX calls, can only load content into the current web browser.
- **Parent.** Some older websites may require child browsers to load content into the parent browser in order to function correctly, so this option should only be chosen in such cases. If an action command loads content into a new web browser, that new web browser becomes a child browser of the current web browser, and actions in the child browser can direct content into the parent browser. To use this option, the action command must load a completely new page.

Browser Mode

If you leave the default for the **Target Browser** (*New*), you can choose the browser mode:

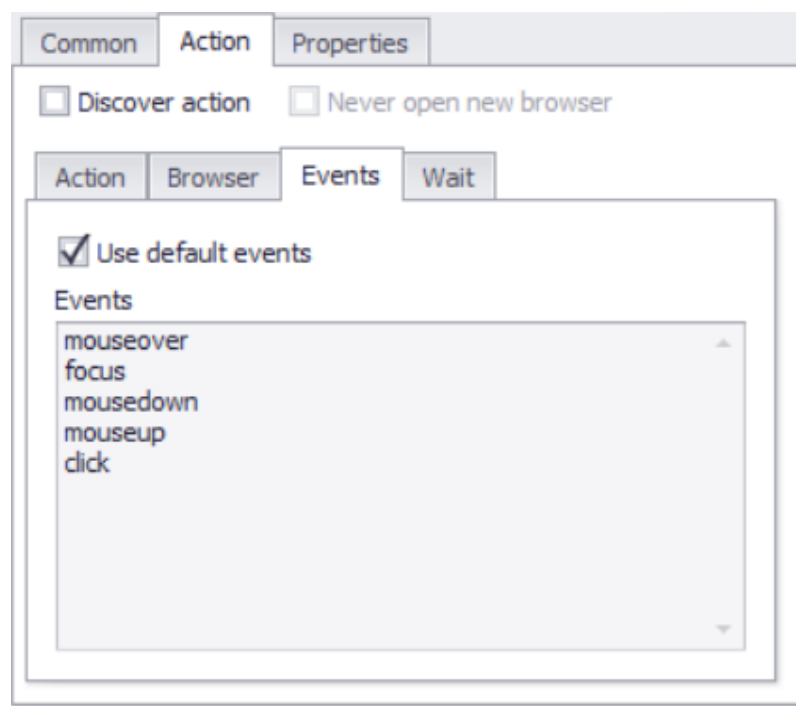
- **Default** - The new web browser will be exactly the same type as the parent browser.
- **Web Browser.** The browser functions as a standard web browser, and it will download images and execute JavaScript.
- **HTML Parser** - the command does not start a new browser. Instead, the web page simply downloads and runs through a HTML parser. The HTML parser does not execute JavaScript and does not load frames, so it is faster and more reliable than a web browser. However, the parser does not work on websites that rely on JavaScript, and the parser may also be unable to submit some web forms (even when they don't rely on JavaScript).

- **JSON Parser** - the command does not start a new browser. Instead, it parses JSON content returned by a web server, and lets you easily extract content elements from the JSON content.
- **XML Parser** - the command does not start a new browser. Instead, it parses XML content returned by a web server, and lets you easily extract content elements from the XML content.

Note: You must reopen browser tab for any change in this setting to take effect.

5.3.8.3 Events

If a command action type is set to fire events, then you can specify the events that should be fired on the selected web element.



The Events tab

In most cases, you can check the **Use default events** check box, which will fire all appropriate events that the web element supports. In special cases, you may want to remove some of the default events. For example, an action may try to open a drop-down box for an input form field.

Firing the **focus** or **click** event on the input form field may cause the drop-down box to open, but the **blur** event may cause the drop-down box to close. In that case, firing all the default events would open the drop-down box but quickly close it again. To prevent this, uncheck the **Use default events** box and remove the **blur** event from the list.

Supported Events and Functions

Content Grabber supports all the default events for the chosen web element and some custom events and functions. The following list includes only the most common events, and is not a complete list of all available events. Please see a JavaScript reference guide for all available events.

Event Name	Description
mousedown	Emulates the press of a mouse button onto the chosen web element without releasing the button.
mouseup	Following a mousedown event, emulates releasing a mouse button onto the chosen web element.
click	In immediate succession, emulates a press and a release of a mouse button on the chosen web element .
keydown	Emulates pressing a key in relation to the chosen web element without releasing it.

keyup	Emulates releasing a key in relation to the chosen web element .
keypress	In immediate succession, emulates a press and a release of a key in relation to the chosen web element.
focus	Emulates bringing input focus to the chosen web element.
blur	Emulates removing input focus to the chosen web element.
change	Emulates changing the input value of the chosen web element.

The following list includes custom functions that can be used along with the standard events:

Function	Description
exec(JavaScript)	<p>Executes a JavaScript on the selected web element.</p> <p>Example 1: <code>exec(\$(element).unbind('blur'))</code> <code>)</code></p> <p>This example removes all blur events from the chosen web element. This example requires JQuery to be available on the web page, but the exec function works on non-JQuery JavaScript as well.</p>

	<p>Example 2: <i>exec(element.click())</i></p> <p>This example fires the click event on the selected web element.</p> <p>Example 3: <i>exec(window.history.back())</i></p> <p>This example moved the current page back to the previous page.</p> <p>The variable element is always defined as the selected web element.</p>
unbind(Event)	<p>Removes all events of type Event from the selected web element. This function requires JQuery to be available on the web page.</p> <p>Example: <i>unbind(blur)</i></p> <p>This example is equivalent to calling <i>exec(\$(element).unbind('blur'))</i> or <i>exec(\$(element).off('blur'))</i></p>
click()	<p>Fires the following 3 events:</p> <p>mousedown</p> <p>mouseup</p> <p>click</p>

delay(milli seconds)	<p>Pauses execution of the command for a specific number of milliseconds.</p> <p>Example: <i>delay(2000)</i></p> <p>This example inserts a delay of 2 seconds.</p> <p>Important note: The activity timeouts include any event delay. So, if you have a single activity that waits 500 milliseconds, and all events take longer than 500 milliseconds to fire, then the action will time out before all the events have had time to fire.</p>
removeCg Attributes()	<p>Content Grabber adds custom attributes to DOM elements in order to keep track of these elements. Very rarely, this causes issues with the target website. This function simply removes these attributes before the action.</p>
setinputte xt() setinputte xt(text)	<p>The function inserts text into the chosen form field, and is only compatible with Form Field commands. If the form field is a select box, the function selects the option with the text attribute equal to the specified text.</p> <p>If this function call contains no text, the function inserts the input data for the Form Field command.</p>

	<p>Example: <code>sendinputtext("hotels")</code></p> <p>Typically, a Form Field command sets the value of the chosen form field and then fires the specifies events. This function gives you the ability to fire events before setting the value of a form field.</p> <p>NOTE: Often, this function is used when the corresponding Form Field command property Set Value is set to false.</p>
keycode(keycode)	Fires keydown, keyup and keypress with the specified key code.
key(key) key("string")	<p>Fires keydown, keyup and keypress with the specified key. The key can be a character or one of the following:</p> <p>enter paste left right up down</p> <p>If a key is a character, it can be preceded with one of the following:</p> <p>ctrl+ shift+ alt+</p>

	<p>Examples:</p> <pre>key(a) key(ctrl+a) key(paste) key(enter)</pre> <p>if a string enclosed in double quotes is specified, the keydown, keyup and keypress events are fired for each character in the string.</p> <p>Example:</p> <pre>key("hotel")</pre>
<p>scroll()</p> <p>scroll(percentage)</p>	<p>This function scrolls downward on the page containing the chosen web element, to ensure adequate coverage for pages that load content dynamically.</p> <p>Often, the function call is used along with the option Repeat Link Action, which will repeat a downward scroll to the bottom-until the scroll no longer loads new content.</p> <p>Example 1: <code>scroll()</code></p> <p>This example will scroll all the way to the bottom of the content. Optionally, you can specify an the amount to scroll in terms of pixels.</p>

	<p>Example 2: <i>scroll(50)</i></p> <p>This example will scroll 50 pixels down through the content.</p>
scrolls(scrollCount)	<p>This function scrolls downward a specified number of times on the page containing the chosen web element, to ensure adequate coverage for pages that load content dynamically.</p>
windowscroll() windowscroll(percentage)	<p>This function scrolls down to the bottom of the page containing the chosen web element, to ensure adequate coverage for pages that load content dynamically.</p> <p>If the web browser contains multiple frames, you should choose a web element within the frame to which you want to scroll. The chosen web element has no other influence on this function.</p> <p>Often, the function call is used along with the option Repeat Link Action, which will repeat a downward scroll to the bottom - until the scroll no longer loads new content.</p> <p>The Scroll to End of Page action option combines the windowscroll event with the action option Repeat Link Action, so this option can be used as an alternative to the windowscroll function.</p>

	<p>Example 1: <code>window.scroll()</code></p> <p>This example will scroll all the way to the bottom of the web page. Optionally, you can specify an the amount to scroll in terms of pixels.</p> <p>Example 2: <code>window.scroll(50)</code></p> <p>This example will scroll 50 pixels down the web page.</p>
window.scrolls(scroll Count)	This function scrolls down, a specified number of times, to the bottom of the page containing the chosen web element, to ensure adequate coverage for pages that load content dynamically.
Back()	Move back to the previous page by calling the JavaScript function <code>window.history.back()</code>

If the action command is a **Form Field**, then the input value is available for all event functions by using the variable **[input]**. For example, you can call the function `sendtext` to set the input value of a form field:

```
sendtext([input])
```

Or, the input value could be set using JQuery:

```
exec$(element).val("[input]")
```

If you set a form field value using any of these event functions, you may want to uncheck the form field box **Set Value**, so that the form field value cannot be set automatically, but rather only by an event function.

Query Selectors & Google's Geolocation Search

JavaScript query selectors can be used to fire events on other web elements than the action web element. Query selectors must be specified before the event name and must be enclosed in double quotes.

Example:

```
"#search".focus
```

The above event configuration fires the **focus** event on a web element with the ID **search**.

One important application for query selectors are websites using Google's Geolocation Search. The following event configuration is required to select a drop down item in the Google's Geolocation Search plugin.

```
"#search".focus
```

```
mouseover
```

```
"#search".blur
```

where **search** is the ID of the input field used by Google's Geolocation Search. This input field is custom defined, so you may need to use different query selectors for different websites. For example, if the Geolocation Search uses an input field with name **city**, the event configuration would look like this:

```
"input[name='city']").focus  
mouseover  
"input[name='city']").blur
```

5.3.8.4 Wait

Typically, you have no concern about the sequence of complex activities during the loading of a web page, since you simply wait for the content that you want to see. The most critical content on a web page will likely load far in advance of the time that you actually get around to viewing a specific part of the web page. Usually, all features function correctly as you fill in web forms or click links.

However, it's very different with web-scraping agents, since these agents are very fast. An agent will attempt to process a web page as quickly as possible and continue onto the next page. A web-scraping agent is so fast that it could easily start processing a page before all of the essential content loads. So, it's important that you configure an action command to wait for all important browser activities to complete and all the content loads before web page processing begins.

When an action command executes, it waits for certain activities to complete in the web browser. For example, if a command executes a click on a link, it may wait for a page to load or an AJAX call to complete. Some actions may result in a very complex set of activities. An action may load a new page that then uses AJAX to load additional dynamic content onto the page.

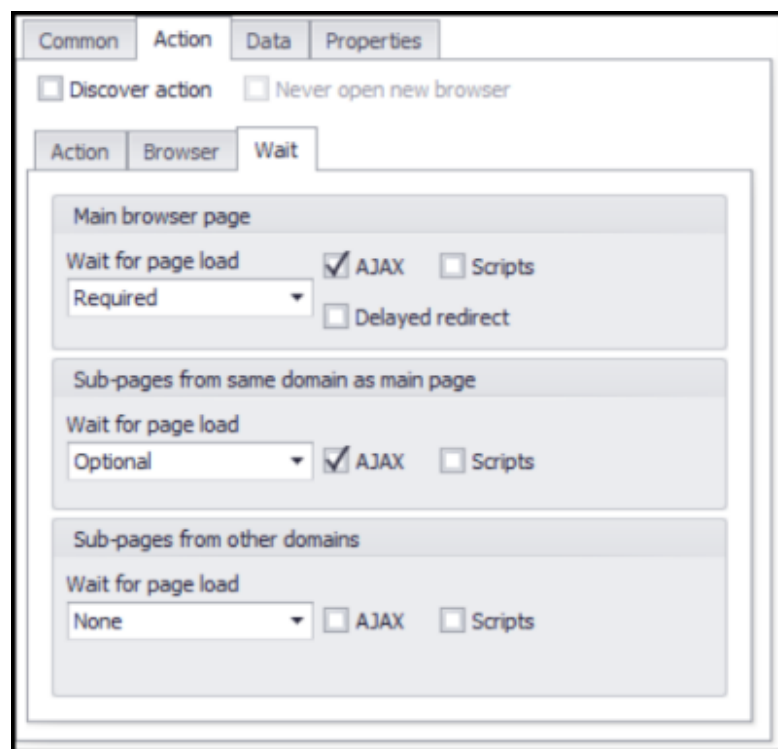
Discovering Activities

Action commands automatically discover web browser activities. After a command fires the action events, it will monitor all activity in the web browser and wait for critical activities to complete. Once no new activities have started for a little while, it will consider the action to be complete.

You can specify which activities an action command should wait for. The command can wait for activities in the main web page and in sub-pages that are loaded in web frames.

Page load activities can be optional or required. An error will be reported if a page load activity is required, but no page load occurs. If **Wait for page load** is set to **None**, the command will not wait for any page load to occur, which is slightly faster than setting **Wait for page load** to **Optional**.

An AJAX activity occurs when a web page loads content from the web server asynchronously. A Script activity occurs when a JavaScript file is loaded by the web page asynchronously. AJAX and Script activities are always optional, which means no error will be reported if a command is configured to wait for AJAX, but no AJAX activities occur.



Wait configuration panel.

Complex Website Activities

Some websites have very complex activities. For example, many travel websites that provide hotel and flight search functionality will load a waiting page and after a while load the actual search result. An action command will often complete the action after

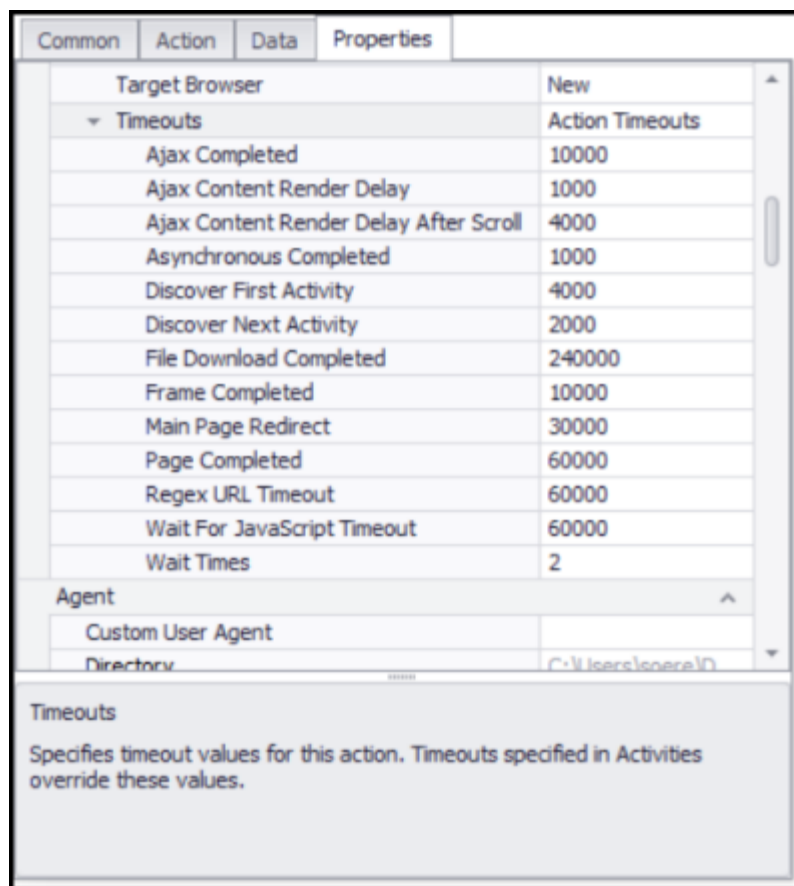
the waiting page is loaded, since it doesn't know that more content will be loaded later. If the website redirects from the waiting page to the search result page, then the Wait option **Delayed redirect** can often be used successfully, but sometimes websites use other techniques and it can be very difficult for the action command to tell when an action has completed.

Sometimes it's possible to determine that a website action has completed when a specific URL has been loaded. This URL could be from a full page load, a frame page load, or an asynchronous AJAX call. A Wait for Content sub-command can be used to wait for a URL that matches a Regular Expression.

Sometimes the only reliable way to determine when an action has completed, is to wait for certain web content to appear on the web page. A Wait for Content sub-command can be used to wait for web content.

Wait Timeouts

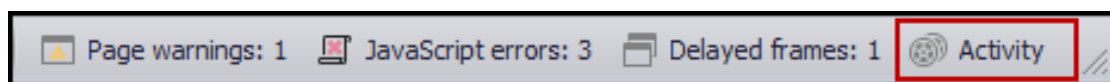
Action commands will wait for browser activities for a certain period of time before the wait times out, and the command either considers the action completed or reports the timeout as a page load error. The default timeout values are usually appropriate, but there will sometimes be situations where some timeout values should be modified. For example, timeout values may need to be increased for a very slow website in order for the agent to work properly, or timeout values could be decreased for a very fast website in order to increase agent performance.



Action timeouts.

Browser Activity Screen

This feature shows all browser activities that occur after the current action executes. You can use this information to determine potential issues with the configuration of the action. Use the **Activity** button on the Content Grabber status bar to open the Browser Activity screen, as shown in the figure below:



The Activity button on the status bar

Critical activities have dark coloring and other activities have light coloring. A blue row appears in the sequence at the point where the command recognizes completion of the action. Activities that occur after the action completes may not necessarily indicate

a problem. If the agent does not work as you expect, then you may need to reconfigure your action in such a way that it waits for some or all of those activities.

Activity	Url	JSON	Timing
Page Completed	https://book.cruisedirect.com/web/cruises/results.aspx?		10608375
Async Script Loading	https://www.barilliance.net/data.js.php?a=pv&id=34782&iid=823770940&pid=8cfp=1&i/vt=...		10608433
Async AJAX Completed	https://book.cruisedirect.com/Web/SearchForms/Templates/cruisedirector/cruise_search_form....		10608454
Async Script Loading	https://lptag.liveperson.net/tag/tag.js?site=13135887		10608455
Async Script Loading	https://book.cruisedirect.com/Veb/data.aspx?type=CruiseFormData&ft=*.*.*.*.*&sid=...		10608455
Async Script Loading	https://lptag.liveperson.net/lptag/api/account/13135887/configuration/applications/tagslets/.jso...		10608474
Frame Loading	https://book.cruisedirect.com/web/cruises/SelectForCompare.aspx?show=true&showHeadFoot...		10608559
Frame Completed	https://book.cruisedirect.com/web/cruises/SelectForCompare.aspx?show=true&showHeadFoot...		10608578
Async Script Loading	https://server.iad.liveperson.net/hcp/html/mTag.js?site=13135887		10608731
Async Script Loading	https://server.iad.liveperson.net/hc/js-13135887/?cmd=lptagTagsSnippets		10608731
Async Script Loading	https://server.iad.liveperson.net/hc/13135887/?visitor=1214501914225763msessionkey=8...		10608731
Async Script Loading	https://server.iad.liveperson.net/hc/13135887/?visitor=1214501914225763msessionkey=8...		10609837
Async Script Loading	https://server.iad.liveperson.net/hc/13135887/?site=13135887&cmd=lptagRepstateMulti&lpc...		10610164
Async Script Loading	https://server.iad.liveperson.net/hc/13135887/?site=13135887&cmd=lptagRepstateMulti&lpc...		10610181
Async Script Loading	https://server.iad.liveperson.net/hc/13135887/?visitor=1214501914225763msessionkey=8...		10610188
Async Script Loading	https://server.iad.liveperson.net/hc/13135887/?site=13135887&cmd=engagementVisitorEvent...		10610401
Async Script Loading	https://cdn.inspectlet.com/inspectlet.js		10610639
Async Script Loading	https://static.criteo.net/js/dld.js		10610642
Async Script Loading	https://sslwidget.criteo.com/event?a=30277&v=4.0.0&p0=e%3Dexd%26site_type%3D&p1...		10610674
Async AJAX Loading	https://frn.inspectlet.com/ginit/202972288		10610719
Parsing Document			10611357
Action Completed			10611565
Async AJAX Completed	https://frn.inspectlet.com/ginit/202972288	{"screencapture": "y..."}	10611695
Async AJAX Loading	https://frn.inspectlet.com/tag		10611770
Async AJAX Loading	https://frn.inspectlet.com/getfid		10611774
Async AJAX Completed	https://frn.inspectlet.com/tag		10613048
Async AJAX Completed	https://frn.inspectlet.com/getfid	{"fid": 1766867965}	10613048

Activity is seen after the action completes, which may not be a problem

5.3.8.5 Wait for Content

Wait for Content commands can be used to wait for web content to appear on a web page, or to wait for a specific URL to complete loading.

A **Wait for Content** command must be a direct sub-command of an action command and will work in conjunction with the action command. When an action command contains multiple **Wait for Content** sub-commands, the action command will wait until a condition specified by any of these commands is satisfied.

A **Wait for Content** command cannot be a sub-command of a HTML List command or a HTML Area command, unless the HTML area is broken by the **Wait for Content** command's parent action command. An action command breaks a HTML area by default, so this limitation is rarely relevant.

The screenshot shows a configuration dialog box with two tabs: 'Common' and 'Properties'. The 'Properties' tab is active. It contains the following fields:

- Name:** A text box containing the word 'New'.
- Wait type:** A dropdown menu with 'Wait for Web Content' selected.
- Shared Action Options:** A sub-section containing:
 - Wait for content:** A dropdown menu with 'Optional' selected.
 - Timeout (milliseconds):** A spinner box set to '10000'.

At the bottom right of the dialog are two buttons: 'Save' (with a green checkmark icon) and 'Cancel' (with a red X icon).

Wait for Web Content

When a **Wait for Content** command is configured to wait for web content, a web element must be selected in the web browser, and the **Wait for Content** command's parent action command will wait until that web element is available on the web page.

The web selection for **Wait for Content** commands works differently than for other commands that require a web

selection. The selection XPath(s) specified by a **Wait for Content** command are used directly in a web page while it's loading, before the page is parsed by Content Grabber, so the XPath(s) cannot use extended Content Grabber syntax, but must follow strict XPath 1.0 syntax. Furthermore, because the XPath(s) are used before Content Grabber has parsed the web page, the XPath(s) cannot work across frames, and the **frame** or **iframe** node names are therefore not valid in these XPath(s). Content Grabber will use the XPath(s) to search for the web content in all available frames and iframes.

Multiple **Wait for Web Content** commands or a single command with multiple selection XPath(s) can be used to cover different scenarios. For example, if an action command must wait for a search result to appear on the web page, but sometimes the search result is empty and a message is displayed instead of the search result, then one selection XPath can be used to select the search result, and another XPath used to select the message that appears when the search result is empty. This means the command will not wait around for the search result to appear on the page if the search result is empty, but instead stop waiting as soon as the empty search message appears on the page.

Wait for URL

When a **Wait for Content** command is configured to wait for a URL, a regular expression must be specified, and the **Wait for Content** command's parent action command will wait for a URL that matches that regular expression. Content Grabber will check the URL of all frames and all web requests that load content asynchronously.

Shared Action Options

The parent action command of one or more **Wait for Content** commands specifies how long the action command should wait for content and if the content is optional or required. For convenience, you can change these action options while editing a **Wait for Content** command. When you change these options for one **Wait for Content** command, the options will automatically change for all **Wait for Content** commands with the same parent action command. You can also configure these options directly on the parent action command.

5.4 List Commands

A **List** command is a special type of container command that iterates through a set of elements and executes each of its sub-commands once for each element. The element list can be taken from an input source, such as a CSV file, or it can come from a web selection that selects multiple web elements on the current web page.

The following commands are list commands:

- Agent
- Data List
- Set Form Field
- Web Element List

The Web Element List command uses a web selection to generate its element list. The Agent, Set Form Field and Data List commands use an input data provider to generate their element lists. A command is not considered a list command if the input data provider is set to **None**. For example, a Set Form Field command that uses input data from a different command would have its own data provider set to **None**, and therefore is not a **List** command in that case. You can learn more by clicking the links to any of the subsections above.

5.4.1 Data List

This command provides input data to sub-commands. The command (and all sub-command) will execute once for each input data value. The figure below shows the **Command Properties** panel after choosing **Data List** from the **New Command** drop-down:

The screenshot shows a 'New command' dialog box with a dropdown menu set to 'Data List'. Below the dropdown are two tabs: 'Common' and 'Properties'. The 'Properties' tab is selected. Inside the 'Properties' tab, there is a 'Name' field with the value 'Data List', a 'Data provider' dropdown menu set to 'Simple', and a large 'Input data' text area. Below the text area is a checkbox labeled 'Input data includes header row' which is currently unchecked. At the bottom of the dialog, there is a 'Refresh data' button labeled 'Load or Refresh Data' and a 'Select data' dropdown menu.

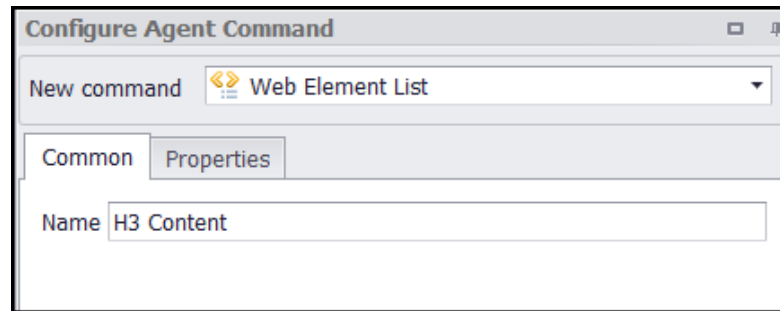
5.4.2 Web Element List

This command selects a list of elements on a web page. The parent command and all of its sub-commands will execute once for each web element in the list, and the web selection of each sub-command will be relative to the current element in the list.

NOTE: All sub-commands will have the same parent

selection.

The figure below shows the **Configure Agent Command** panel after choosing **Web Element List** from the **New command** drop-down:



As we write above, the web selection of each sub-command will be relative to the current element in the list, but some commands can break the web-element list. If a command breaks a parent web-element list, then the sub-commands will no longer be relative to the web-element list. It's important to realize that any command that loads a new web page will always break a web-element list - since the elements in the web-element list don't exist on the new web page.

A command that loads a partial web page, or makes changes to a web page, may break a web-element list if the **Break HTML Area** property has been set to **True**.

5.5 Group Commands

Group commands are simple container commands that you can use to place commands together - which can be a much easier arrangement for navigating and maintaining an agent. You may, for example, want to group together several commands that have various bits of contact data into a **Group** command and give it the name "Content Information".

There are other uses for **Group** commands. Consider this restriction: you can link only one other container command to a container command, such as Navigate Pagination.

So, you would need to use a **Group** command if you want to link to three container commands together.

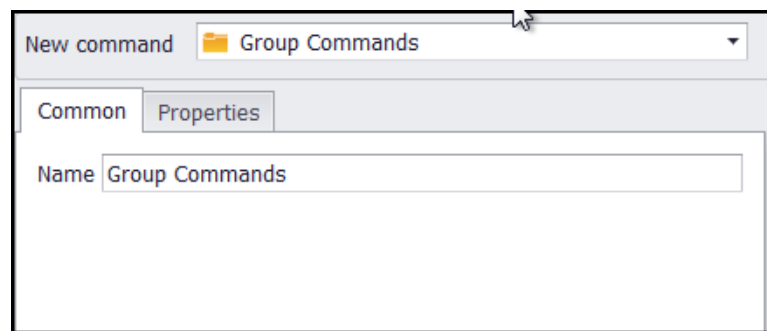
These are the **Group** commands:

- Group
- Group in Page Area

5.5.1 Group

Use this command to group commands together. Although the **Group** command performs no action, it can be useful in the following cases:

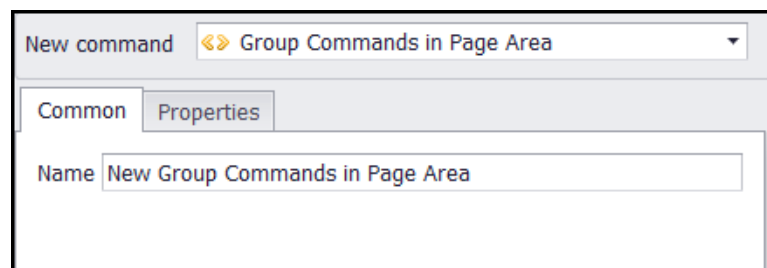
- To group commands that logically belong together. This makes it easier to navigate and maintain an agent. For example, four capture commands used to extract street address, city, zip code and state could be grouped into a **Group** command having the name "Address".
- When adding commands to the template library for future reuse.
- When you want to link a container command to more than one other command. The Navigate Pagination command is an example of a container command that often needs to link to more than one other command.



5.5.2 Group in Page Area

Use a **Group Commands in Page Area** command to group commands together that have the same web selection. The command is similar to the Group command, but it has a corresponding web selection, and any web selection of the sub-commands will be relative to that selection.

Typically, you would choose this command when you need to split a single web element into separate fields. For example, a single web element may contain the entire address of a company, and several capture commands with the same web selection will be necessary to extract and split the address into separate fields such as street address, city, zip code, and state. If you group all the capture commands into a single **Group Commands in Page Area** command, then you only need to select the web element once. Also, if the page location of the address changes, you only need to change the web selection of the **Group Commands in Page Area** command. You can also add this command to your Template Library, so that you can reuse the command in another agent and only need to change the web selection.



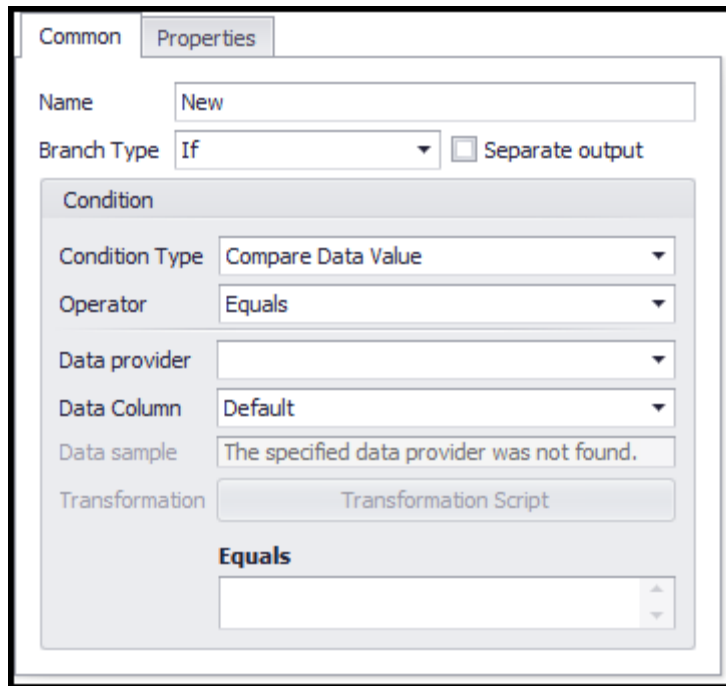
5.6 Branch Commands

Branch commands are used to alter the execution flow of an agent. The following two branch commands are available.

- Else..Elseif..Else
- Exit or Retry

5.6.1 If..Elseif..Else

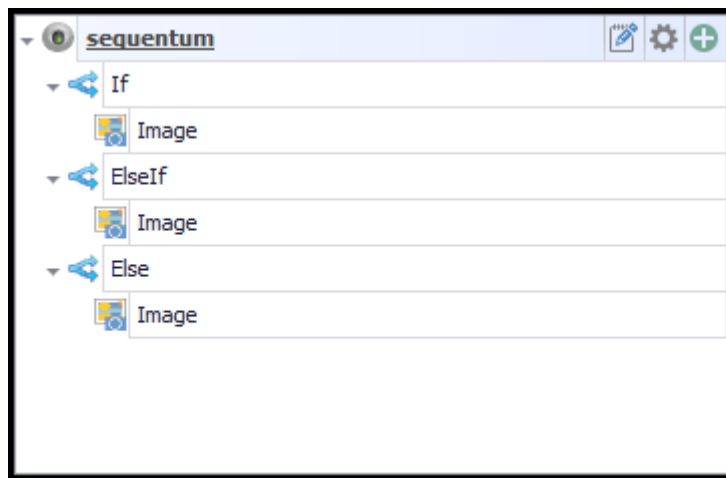
The **If..Elseif..Else** container command provides standard conditional branch functionality. If the condition is true the sub-commands are executed. Otherwise the sub-commands are skipped.



The screenshot shows a configuration window for the 'If..Elseif..Else' branch command. It has two tabs: 'Common' and 'Properties'. The 'Properties' tab is active. The 'Name' field is set to 'New'. The 'Branch Type' is set to 'If'. There is a checkbox for 'Separate output' which is unchecked. The 'Condition' section contains several fields: 'Condition Type' is 'Compare Data Value', 'Operator' is 'Equals', 'Data provider' is empty, 'Data Column' is 'Default', and 'Data sample' is 'The specified data provider was not found.'. Below these is a 'Transformation' button labeled 'Transformation Script'. At the bottom, there is a section labeled 'Equals' with an empty text box.

If..Elseif..Else branch command configuration.

The Branch Type can be **If**, **Elseif** or **Else**. An **Elseif** branch command must come after an **If** branch, and an **Else** branch command must come after an **If** or **Elseif** branch command. **If**, **Elseif** and **Else** branch commands must have the same parent command in order to work together.



Agent with If..ElseIf..Else branch commands.

5.6.2 Exit or Retry

The **Exit or Retry** command is used to exit or retry a container command. If an **Exit or Retry** command exits the Agent command, the agent run completes immediately.

The screenshot shows a configuration dialog for the 'Exit or Retry' command. It has a 'Name' field with the value 'New'. Below this, there are two radio buttons: 'Exit Command' (which is selected) and 'Retry Command'. At the bottom, there is an 'Exit Command' label and a dropdown menu with the value 'Agent'.

A condition can be specified, so the **Exit or Retry** command only exits or retries the container command if the condition is true.

A retry count can be specified when retrying a container command. Once a container command has been retried the specified number of times, the agent will no longer execute the **Exit or Retry** command, but instead exit the retry container or

continue executing the next command, depending on the option

Log error and exit on failure.

The option **Delete on exit can** be set to delete the current data row on exit.

5.7 Other Commands

Click on the links below to learn more about the other commands that are available in **Content Grabber**:

- [Execute Script](#)
- [Transform Page](#)
- [Refresh Document](#)
- [Close Browser](#)

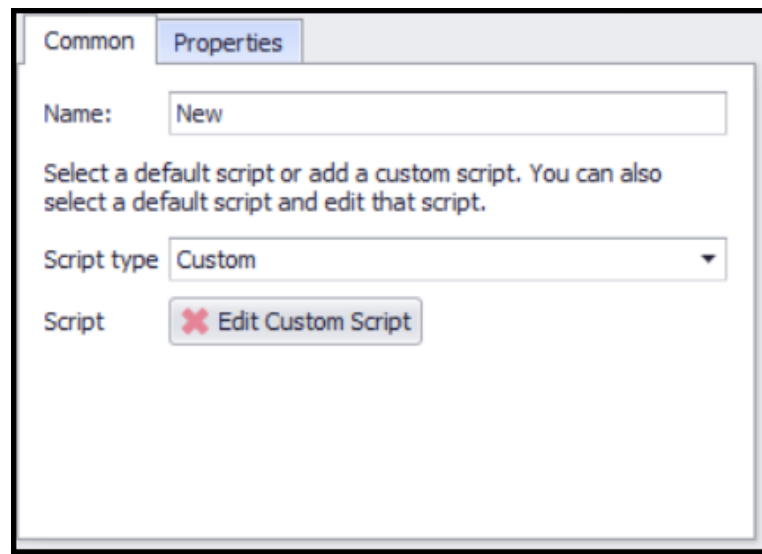
5.7.1 Execute Script

This command executes a custom .NET script, which is useful in these scenarios:

- Control the command execution flow
- Check for duplicates
- Manual interaction with the agent web browser.

Each script command has a corresponding web element selection, and this element has a corresponding entry in the custom script. At run time, if the selection does not exist on the web page, the web element will be NULL in the script. You can check for NULL elements in the script and execute custom functionality if a NULL is found.

The figure below shows the **Configure Agent Command** panel after choosing **Execute Script** from the **New Command** drop-down:



Predefined Scripts

Click the **Script type** drop-down to choose a script containing typical functionality. You can use a script just as it is given in the template, or click the **Edit Custom Script** button to launch a pop-up window to edit the script according to your requirements.

Exit on Duplicate Data

Choose **Exit on Duplicate Data** in the **Script type** drop-down to select a script that will examine the data extraction up to a specific point and check for a matching data row. If a duplicate is found, the agent exits a specified container command.

If you want to check for duplicates in the data extraction from a previous agent run, you must ensure that the agent does not replace data every time it runs. This is done on the **Internal Database** window - which you can open from the **Data** menu.

Common Properties

Name:

Select a default script or add a custom script. You can also select a default script and edit that script.

Script type:

Command:

Condition:

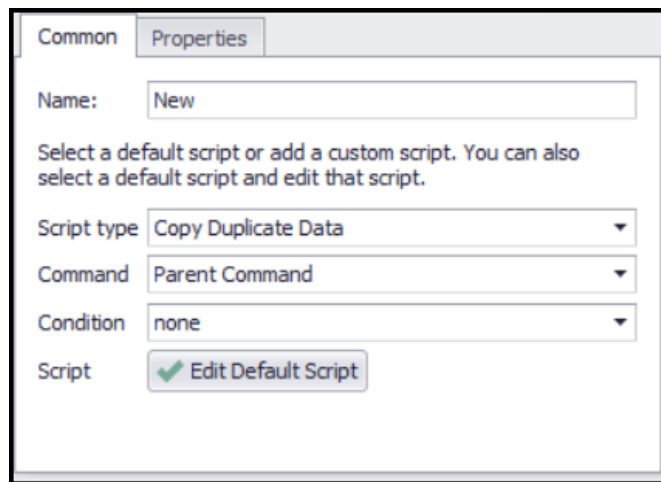
Script:

See Removing Duplicate Data and Extracting New Data Only for more information about how this script can be used.

Copy Duplicate Data

Choose **Copy Duplicate Data** in the **Script type** drop-down to select a script that will examine the data extraction up to a specific point and check for a matching data row. If a duplicate is found, the agent will copy the duplicate data and all child data to the current data row, and then exit a specified container command.

If you want to check for duplicates in the data extraction from a previous agent run, you must ensure that the agent does not replace data every time it runs. This is done on the **Internal Database** window - which you can open from the **Data** menu.



Common Properties

Name:

Select a default script or add a custom script. You can also select a default script and edit that script.

Script type:

Command:

Condition:

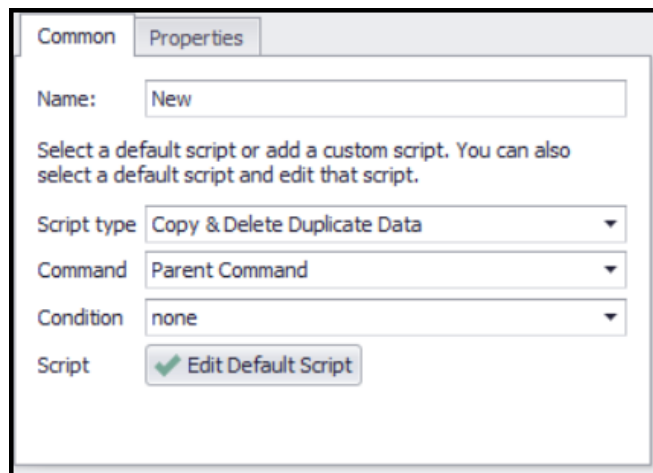
Script:

See Reusing Existing Data for more information about how this script can be used.

Copy & Delete Duplicate Data

Choose **Copy & Delete Duplicate Data** in the **Script type** drop-down to select a script that will examine the data extraction up to a specific point and check for a matching data row. If a duplicate is found, the agent will copy the duplicate data and all child data to the current data row, and then delete the exiting data row, and lastly exit a specified container command.

If you want to check for duplicates in the data extraction from a previous agent run, you must ensure that the agent does not replace data every time it runs. This is done on the **Internal Database** window - which you can open from the **Data** menu.



Common Properties

Name:

Select a default script or add a custom script. You can also select a default script and edit that script.

Script type:

Command:

Condition:

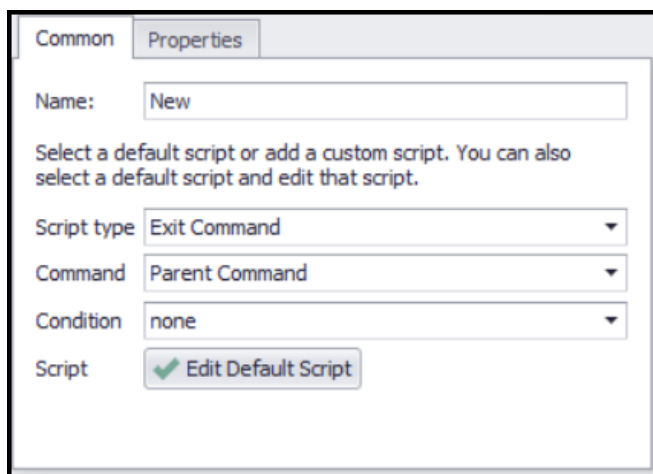
Script: ☒ Edit Default Script

Exit Command

This script exits a container command when a specific condition is met.

Exit Command & Delete Data

This script deletes the current data row and then exits a container command when a specific condition is met.



Common Properties

Name:

Select a default script or add a custom script. You can also select a default script and edit that script.

Script type:

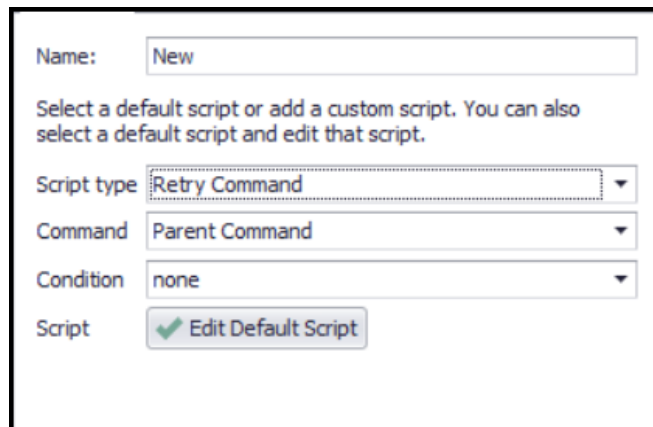
Command:

Condition:

Script: ☒ Edit Default Script

Retry Command

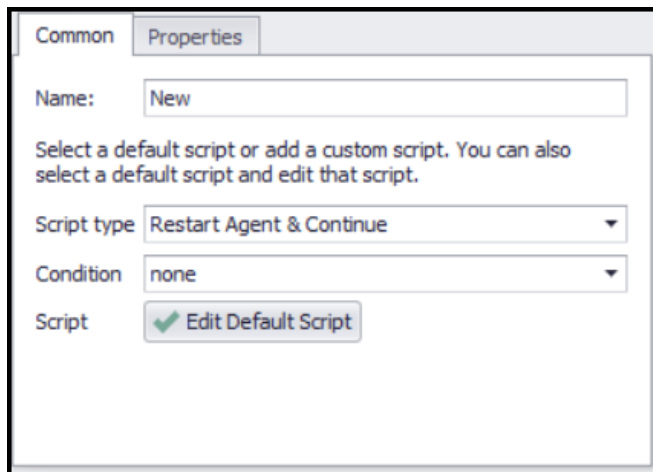
This script retries a container command when a specific condition is met. You can control the number of times that the agent will retry a container command by editing the default script and using the **RetryCount** value in the script parameters.



A screenshot of a configuration window for a script. The window has a title bar and a main content area. At the top, there is a text input field labeled 'Name:' with the value 'New'. Below this is a paragraph of text: 'Select a default script or add a custom script. You can also select a default script and edit that script.' Underneath the text are three dropdown menus: 'Script type' with 'Retry Command' selected, 'Command' with 'Parent Command' selected, and 'Condition' with 'none' selected. At the bottom, there is a 'Script' label followed by a button with a green checkmark icon and the text 'Edit Default Script'.

Retry Agent & Continue

This script restarts the entire agent and continues where it left off when a specified condition is met.



A screenshot of a configuration window for a script, similar to the one above. It has a title bar and a main content area. At the top, there are two tabs: 'Common' and 'Properties', with 'Properties' selected. Below the tabs is a text input field labeled 'Name:' with the value 'New'. Underneath is the same paragraph of text: 'Select a default script or add a custom script. You can also select a default script and edit that script.' Below the text are two dropdown menus: 'Script type' with 'Restart Agent & Continue' selected, and 'Condition' with 'none' selected. At the bottom, there is a 'Script' label followed by a button with a green checkmark icon and the text 'Edit Default Script'.

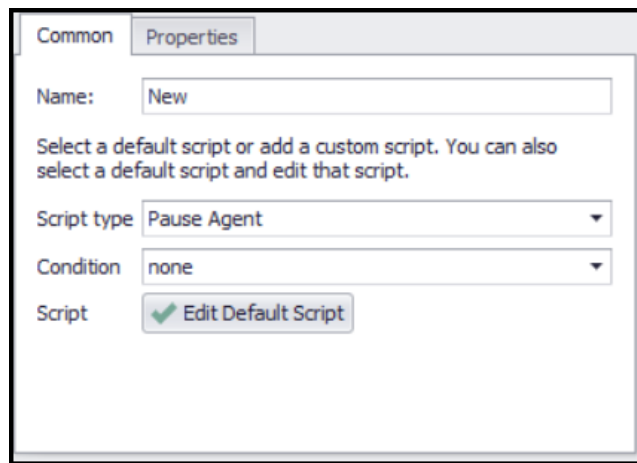
Typically, you would use this script to completely reset a web browser after an error occurs. The agent continues exactly from the point before the reset, so that if an error keeps occurring at this point, the agent will keep restarting in an infinite loop. To avoid this problem, you can set the status **Failed** on the current data row, so the agent will

move to the next data item when it restarts. This code snippet example sets the data row status to **Failed**.

```
args.DataRow.SetStatus(RowStatus.Failed)
```

Pause Agent

This script pauses the agent when a specified condition is met. The agent will display the web browser and allow a user to manually interact with the web browser. The user can press the **Continue** button to continue processing.

The image shows a software configuration window with two tabs: 'Common' and 'Properties'. The 'Properties' tab is active. It contains a 'Name' field with the value 'New'. Below this is a text instruction: 'Select a default script or add a custom script. You can also select a default script and edit that script.' There are two dropdown menus: 'Script type' set to 'Pause Agent' and 'Condition' set to 'none'. At the bottom, there is a 'Script' section with a button that has a green checkmark icon and the text 'Edit Default Script'.

Script Return Value

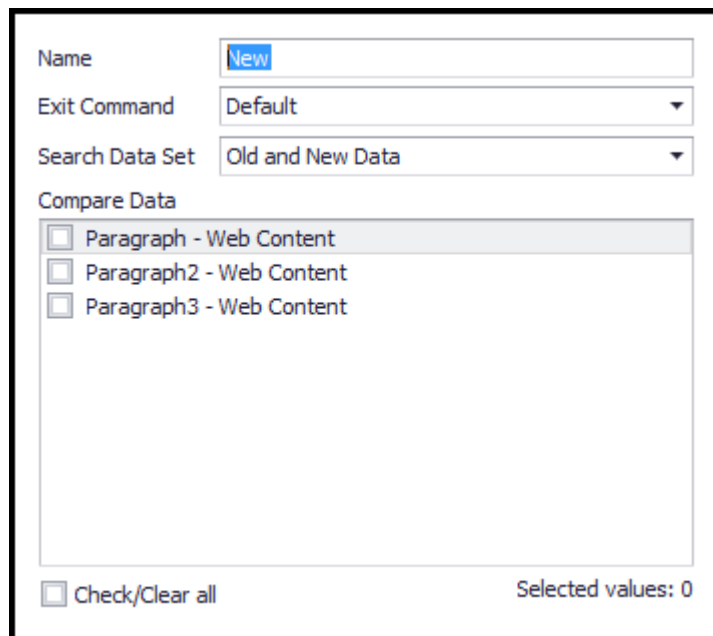
An **Execute Script** command can control the command execution flow by its return value. The command returns an instance of a class **CustomScriptReturn**. This class has the following 4 public static methods:

Method	Description
static CustomScriptReturn RetryContainer(IContainer container)	The agent will retry the specific container command. The container command must

	be a parent of the current command.
static CustomScriptReturn ExitContainer(IContainer container)	The agent will exit the specific container command. The container command must be a parent of the current command.
static CustomScriptReturn Pause()	The agent pauses and displays an agent web browser, which allows a user to interact with the web browser before continuing processing.
static CustomScriptReturn Empty()	The agent continues its normal executing flow.

5.7.2 Remove Duplicate

The **Remove Duplicate** command checks extracted data against previously extracted data and discards the extracted data if it already exists.



The screenshot shows a configuration window for the 'Remove Duplicate' command. It has the following fields and options:

- Name:** A text box containing the word 'New'.
- Exit Command:** A dropdown menu set to 'Default'.
- Search Data Set:** A dropdown menu set to 'Old and New Data'.
- Compare Data:** A list box containing three items, each with an unchecked checkbox:
 - ☐ Paragraph - Web Content
 - ☐ Paragraph2 - Web Content
 - ☐ Paragraph3 - Web Content
- Check/Check all:** A checkbox at the bottom left, which is unchecked.
- Selected values:** A label at the bottom right showing the value '0'.

The **Remove Duplicate** command should be added after the capture commands that extract the data that should be checked, and the **Remove Duplicate** command must have the same parent command as the capture commands.

When a **Remove Duplicate** command detects a duplicate it will exit a specified parent command. By default, a **Remove Duplicate** command exits the current list entry of the nearest parent list command, or the nearest parent command that exports data to a separate data table.

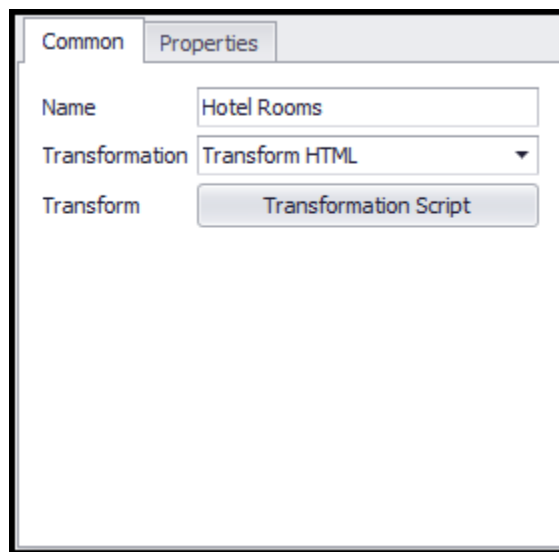
If the target data is displayed by descending order, then you can use a **Remove Duplicate** command to extract only new data by exiting the Agent command when a duplicate is detected. For example, if the target data is ordered by newest date to oldest date and the **Remove Duplicate** command detects that a specific date has already been extracted, then you know that all data before that date has already been extracted and you can exit the agent.

5.7.3 Transform Page

This command transforms part of a web page to facilitate the capture of data. You can choose from the following types of transformations:

- Transform HTML
- Insert HTML
- Replace HTML
- Denormalize HTML Table

Use the **Transform Page** command to transform the structure of a web page into a form that will facilitate easier data extraction. The command uses a transformation script to transform existing HTML, or it can insert or replace HTML on a web page. The corresponding web selection for this command is the HTML of the chosen web element that will be transformed.



The default transformation for this command is **Transform HTML**, which denormalizes an HTML table. When using the **Denormalize HTML Table** transformation, the web selection must be an HTML table. The transformation will repeat some table cells and

rows to make sure that all **rowspan** and **colspan** attributes in the table are set to one.

The screenshot shows a configuration window with two tabs: 'Common' and 'Properties'. The 'Properties' tab is active. It contains two fields: 'Name' with the value 'Hotel Rooms' and 'Transformation' with a dropdown menu set to 'Denormalize HTML Table'.

The figure below depicts an HTML table having some cells that span multiple rows. You can use the **Denormalize HTML Table** transformation to ensure that no cells in the table spans more than one row, which makes it much easier to extract data from the table.

Room Type	Conditions	Max	Price	Nr. rooms	Reservation
Queen Room Flat-screen TV Air conditioning Soundproof Free WiFi Prices are per room Included: 6 % VAT, € 4.50 city tax per night	• FREE cancellation before 6:00 PM on Aug 11, 2014 • PAY LATER • Breakfast AUD 11	2	AUD 142 Value Deal	0	Reserve Confirmation is immediate
	• FREE cancellation before 6:00 PM on Aug 11, 2014 • PAY LATER • Breakfast AUD 11	1	AUD 120	0	
Standard Double Room Flat-screen TV Air conditioning Soundproof Free WiFi Prices are per room Included: 6 % VAT, € 4.50 city tax per night	• FREE cancellation before 6:00 PM on Aug 11, 2014 • PAY LATER • Breakfast AUD 11	2	AUD 142 Value Deal	0	
	• FREE cancellation before 6:00 PM on Aug 11, 2014 • PAY LATER • Breakfast AUD 11	1	AUD 120	0	

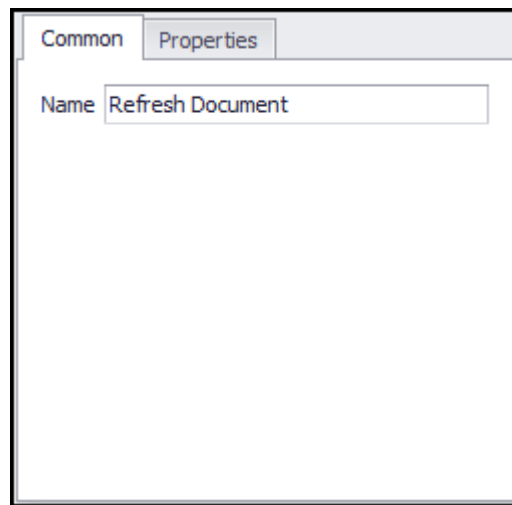
Choose the Denormalize HTML Table transformation on this table to ensure that no cells span

more than one row

5.7.4 Refresh Document

When Content Grabber loads a new web page into the web browser, it parses the dynamic web page and generates an equivalent static version of the page. Agent commands will work on the static version of the web page, since it's much faster and more reliable. All action commands will cause a refresh of the static version of the page, since action commands are likely to either load a new web page or modify the existing page.

The **Refresh Document** command is used to force a refresh of the static version of the web page, so that it corresponds to the dynamic version loaded in the web browser. Usually, Content Grabber will be able to refresh the document when necessary, so you'll only need to use this command in special cases. For example, consider the case where a child web page refreshes the parent web page. After processing the child web page, you may need to perform a **Refresh Document** command before continuing to process the parent web page.

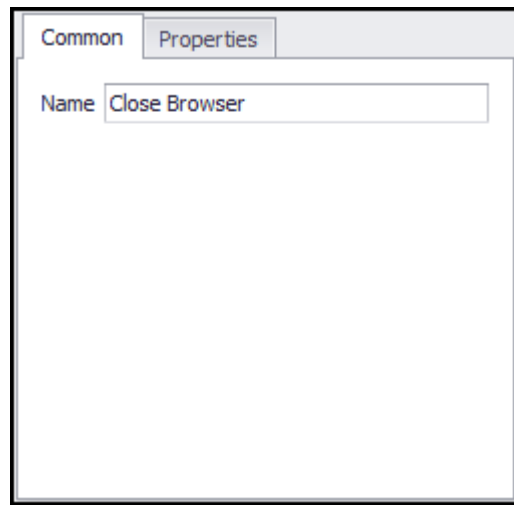


5.7.5 Close Browser

Content Grabber often opens new web pages in a new browser window, and will always open a web page in a new window if the web page design specifies the

opening of a child window. When Content Grabber finishes processing a page in a child window, it will leave the browser window open and reuse the window for the next similar child page. Some websites will close a child window after use, and will only work correctly if the child window does indeed close.

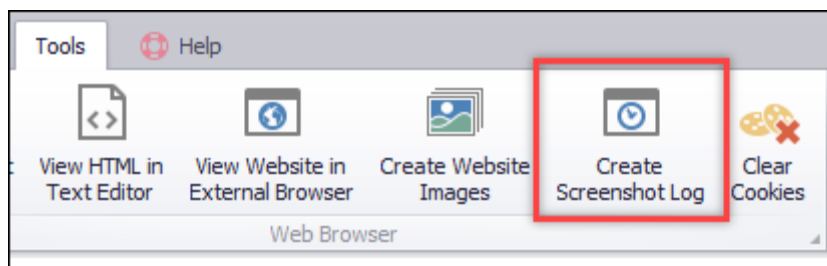
Use the **Close Browser** command to force a browser window to close, so that the website will create a new browser window.



5.7.6 Screenshot Log

Screenshot Log commands can be used to generate a log file that consists of a sequence of text entries with associated screenshots. This can be used in agents designed to test procedures on a website, to also produce simple documentation for the procedure. Please see the topic Website Testing & Documentation for more information.

Screenshot Log commands are ignored unless the agent is run using the button **Create Screenshot Log** from the **Tools** menu.



Press the Create Screenshot Log button to generate documentation.

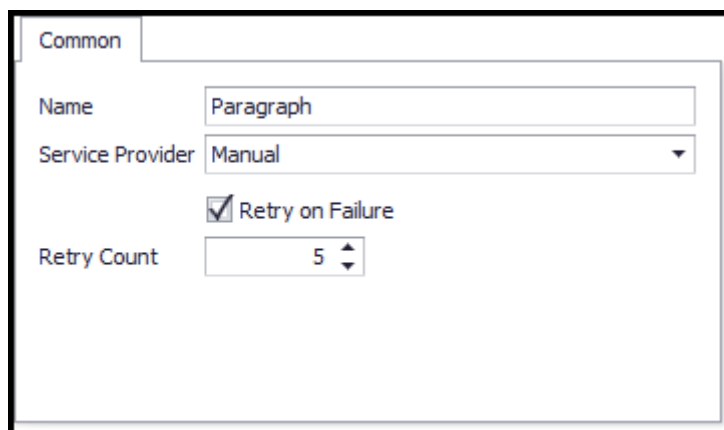
5.8 Composite Commands

A composite command is a command that adds multiple sub-commands to the agent. The following composite commands are available.

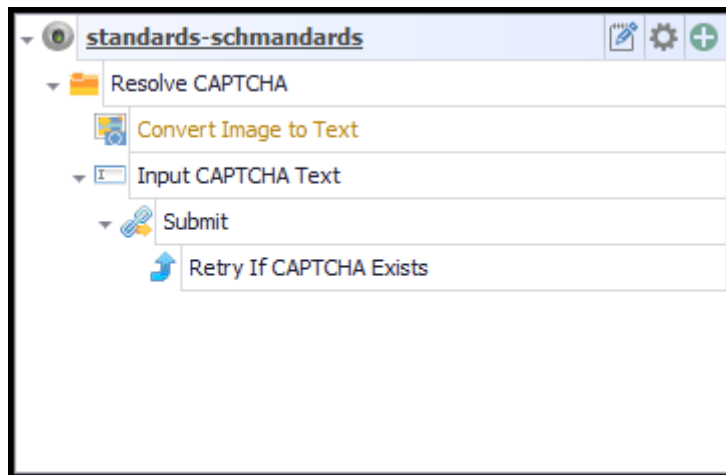
- Resolve CAPTCHA

5.8.1 Resolve CAPTCHA

The Resolve CAPTCHA composite command adds a number of sub-commands that can be used to process standard CAPTCHA images.



To use this composite command, select the CAPTCHA image in the web browser and then add the command. The composite command will add the following sub-commands to the agent.

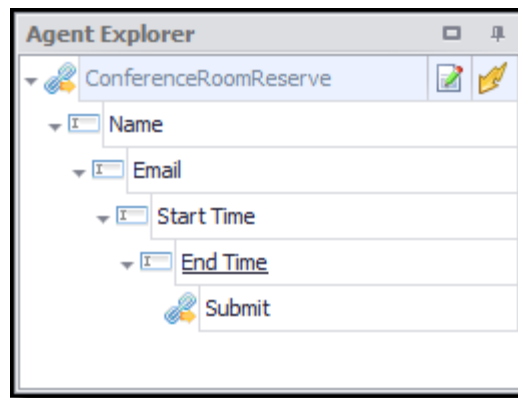


5.9 Web Forms

Content Grabber can submit web forms repeatedly - for any combination of input values. You can specify input values as a static list of values, or you can feed the values from an input data source - such as a database or a CSV file.

Remember, you can use Set Form Field commands to set form fields, and Navigate Link commands to submit web forms. Some web forms don't have a separate **Submit** button, but instead submit the form when a form field value changes. We recommend that you use a Navigate Link command in such cases. Set Form Field commands perform actions, such as loading a new web page.

The figure below depicts an example configuration for an agent that submits a web form:



Typically, Set Form Field commands that submit a web form are nested. So, if the first command is iterating through a list of input values, then all succeeding Set Form Field commands will execute once for each input value. The last command in a web form configuration is typically a Navigate Link command, and since it's nested along with all the preceding Set Form Field commands, it executes once for each combination of input values.

File Download

You can use a web form to download a file, such as an Excel or PDF file. The file download may start when you click the form **Submit** button, or it may start when you change a form field value.

If a file download starts when you click a **Submit** button, you should use a Download Document command instead of a Navigate Link command. The Download Document command should select the **Submit** button and have the **File Download Action** property set to **Click to Download**.

If a file download starts when you change a form field value, you should use a Download Document command instead of a Navigate Link command. Also, the Form Field command that triggers the document download must have the option Download Document command set to the **Download Document**. The Download Document command must be a direct sub-command of the Form Field command that triggers the file download.

Uploading File

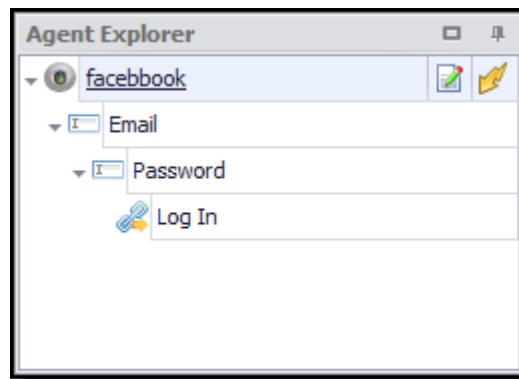
Some web forms contain file upload boxes that allow users to upload files to a website. Scripting cannot modify file upload boxes, so it's necessary to process these differently than other web form input boxes.

Content Grabber will automatically detect file upload form fields and will try to handle the file upload dialog box that appears when clicking on the file upload form field. For the Set Form Field command that handles the file upload form field, the input value should be the full local file path to the file you want to upload.

When the file upload dialog box appears, Content Grabber will automatically enter the file path and click on the button that selects the file. This happens so fast that the dialog box won't appear. Content Grabber needs to click on the button that selects the file, and it does that by looking for a button having an "Open" label. This is the correct button for English language versions of Windows, but will not be the correct button for other language versions of Windows. In such cases, you may need to change the Set Form Field command property **Handle Web Browser Dialog**. The default value is **&OPEN**, which means that the agent will be looking for the text "Open" on the dialog button. The text &O is shown as O and means the button can be activated by pressed the keyboard combination ALT + O.

Website Login Forms

Content Grabber can process website login forms the same way it handles any other web forms. An agent that processes a login web form would normally have a Set Form Field command for the **Username**, and another Set Form Field command for the **Password**, and a Navigate Link command that clicks on the login **Submit** button.



Basic Windows Authentication

Some websites use basic Windows authentication, and they will display a Windows login box. Content Grabber gives you the ability to set the **Username** and **Password** for basic Windows authentication by editing the Agent Command and then setting the **Username** and **Password** in the **Basic Windows Authentication** of the properties tab.

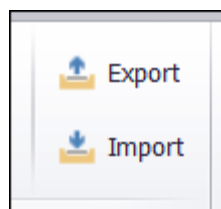
After setting the basic Windows authentication properties, you must reload your agent for the properties to take effect. Basic Windows authentication does not work for in **HTML Parsers, JSON Parsers and XML Parsers**.

5.10 Command Library

Content Grabber contains an agent template library and a command template library for easy reuse of agents and commands.

Exporting and Importing Template Libraries

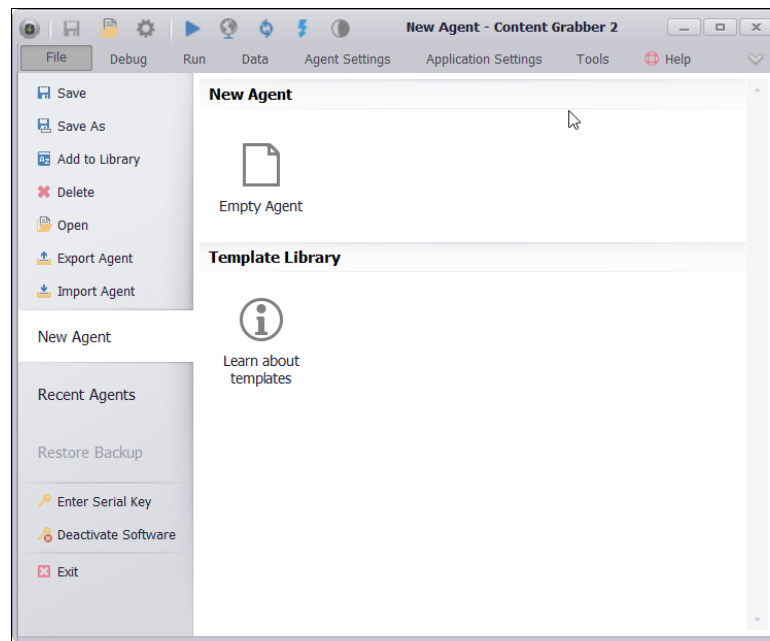
You can export and import all templates from one computer to another.



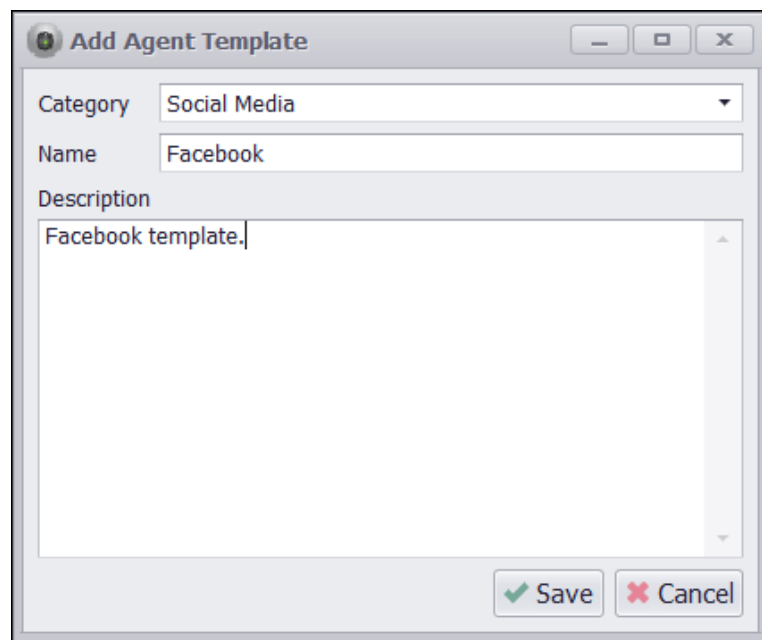
Export or import templates from the Tools menu.

Agent Templates

When creating a new agent, you can derive the new agent from an agent template.



You can add any of your agents to the template library by right-clicking on the agent in the **Agent Explorer** window and then selecting **Add to Agent Template Library** from the context menu.

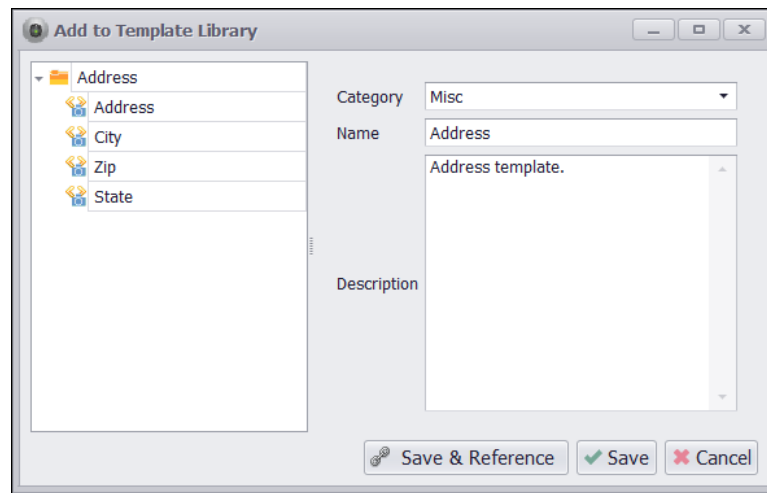


After adding a new agent template, the template will become available in the **Templates Library** when creating a new agent.

Command Templates

You can add any container command to the template library. The container command and all sub-commands will be added to the library, so when you later insert the command from the library into a new agent, the insert task will include the entire command structure.

Add container commands to the library by right-clicking on a command in the **Agent Explorer** window and selecting **Add to Template Library** from the context menu.

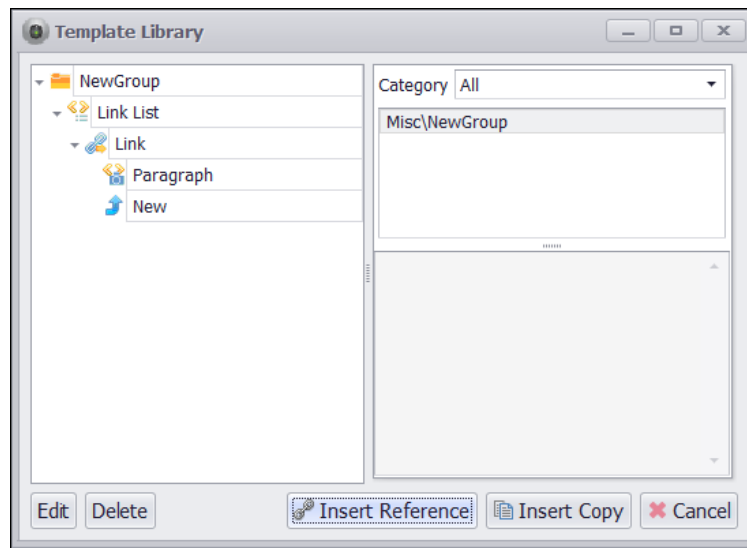


Command templates are categorized to make it easier to find specific templates. When you add a new template, you can place it in an existing category or enter a new category name.

When you save a new container command to the template library, you can choose to just save the template, or you can save and reference the template. Please see **Template References** below for more information about template references.

You can insert a command template into an agent by doing the following:

1. In the **Agent Explorer** window, right-click the container command that will be the parent command.
2. Choose **Insert Template from Library** from the context menu.



Template References

When you insert a command template into an agent, copies of the commands in the template will be inserted into the agent. If the command template later changes, it will not effect the commands that were copied into the agent. If you have many agents that use the same command templates, it's sometimes convenient if any change to a template is reflected in the agents that use the template. This is possible by inserting a reference to the command template instead of a copy.

When adding a template reference to an agent, a local copy of the template will be stored inside the agent. If an agent is copied to another computer where the referenced template doesn't exist, the local copy of the template will be used instead.

When editing commands in an agent, which are part of a referenced template, the local template stored in the agent will be modified, not the computer wide template. If the agent is reopened, the local template stored in the agent will be replaced with the computer wide template, and any changes made to the

local template will be discarded. Any changes made to the local template can be saved to the computer wide template by right-clicking on the referenced command template and selecting **Save Template** from the context menu. Commands that reference a template have a blue font color in the Agent Explorer.

6 Error Handling

Web scraping is frequently and notoriously unreliable, because you must contend with many external factors over which you have no control. A target website may fail because of defects in the web application, or there may be problems with the Internet connectivity anywhere between you and the target website. These problems may seem negligible when you are browsing a few websites with a normal web browser, but a web-scraping agent is capable of navigating more web pages in a few hours than a human can view in an entire year. So, small glitches can become significant problems that inhibit reliable data extraction. You can minimize these factors with error handling, especially for your critical web-scraping tasks.

Error handling can be difficult to implement properly. In cases where your agent only collects non-critical data, you may decide to skip error handling and simply re-run the agent. Error handling is only important when an agent needs to deliver reliable data every time it runs.

For web scraping, there are two aspects to error handling: Agent error handling and Error logs and notifications. Agent error handling happens automatically, when specific errors occur during the execution of an agent. For example, an agent could retry a command if it fails to load a new web page. Error logs and notifications can warn an administrator when an agent encounters trouble and requires attention. Consider the case in which a target website has a new layout and the agent may no longer be able to extract data correctly. You can configure error handling for this agent so that an email notification will be sent to the administrator - who can decide how to update the agent to accommodate the new layout.

We encourage you to learn more about error handling in these sections:

- Agent error handling
- Error logs and notifications

6.1 Agent Error Handling

You can configure an agent to react to errors in two different ways. You can configure the error handling properties for an action command, or you can use a custom script to watch for an error condition and take appropriate action.

You can configure an agent to react to errors in two different ways. You can configure the error handling properties for an action command, or you can use a custom script to watch for an error condition and take appropriate action.

Action Command Error Handling

These are the **Error Handling** properties:

Error Handling	Description
Exit Command	The agent will exit the action command and continue executing the next command. The agent will skip all sub-commands of the action command.
Retry Command	The agent will retry the action command a specified number of times, and if the action command does not succeed, it will skip all sub-commands of the action command and continue executing the next command. Set the property <i>Retry Count</i> to specify the number of retries. If <i>Retry Count</i> is set to zero the agent will keep retrying the command indefinitely.
Restart Agent and	The agent will restart and continue where it left off. This option is useful if an error puts

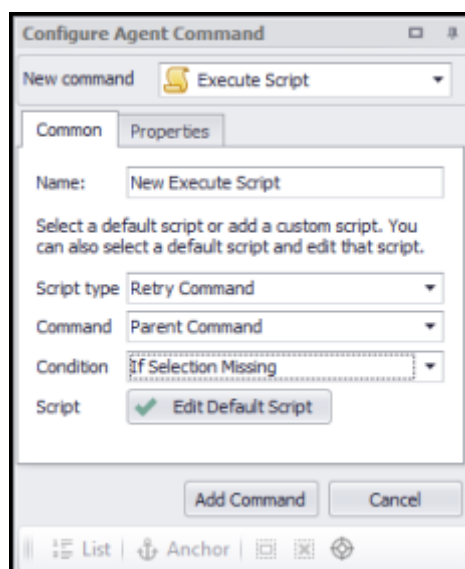
Continue	the website into a state where the agent cannot continue.
Stop Agent	The agent will stop.
No Error Handling	The agent will not handle the error, but will continue to execute the sub-commands of the action command. Use this option if you want to handle the error in a custom script later in the process.

Error Handling In An Execute Script Command

You can use Execute Script Commands to implement advanced error handling, such as in these cases.

- A required element is missing from a web page
- An error message is displayed on a web page
- An error occurs during the execution of an action command.

You can write a custom script in C# or VB.NET, or you can use the default script options to generate a default script without writing any .NET code.



This default script retries a parent command
if a selection is missing

An Execute Script Command has a corresponding web selection, and the script can use this web selection to check for any errors. The script also is given the parameter **IsParentCommandActionError**, which is set to **True** if the parent action command encounters an error while executing the command action.

6.2 Error Logs and Notifications

You can use error notifications to warn when an agent encounters an error, so that the administrator can take appropriate action. The administrator can use the logs to get more information about the errors.

Error Notifications

Notifications are email messages that are sent to a specific email address when certain conditions occur. The notification configuration screen is available by clicking the **Agent Settings > Notifications** menu.

Email Notifications

☒ **Enable notifications** ☐ Default email settings ☐ Default email addresses

Email Configuration

Server:

Port: ☐ Use SSL

Username:

Password:

Sender email address:

Recipient email addresses:

Success email addresses:

[Email Troubleshooting](#) (Please read if you cannot send emails)

Notifications

☐ Always send notification when agent completes a run

☒ Send notification on critical errors

☐ Send notification on unsuccessful data extraction

☐ Send notification on successful data extraction

This agent sends email notifications if it encounters any critical errors

An email notification can be sent when an agent finishes, but never during agent execution or when debugging an agent. You can choose to send email notifications when one or more of these conditions are met:

Condition	Description
Always send notifications when an agent completes a run	A notification email will always be sent when the agent finishes, whether an error occurs or not.
Send notification on critical errors	An email notification will be sent when critical errors occur. A critical error will

	appear as red color on the log screen. These critical errors include page load errors and also situations where <i>required</i> web selections are missing from a web page.
Send notification on unsuccessful data extraction	A notification email will be sent when an agent run doesn't meet a defined success criteria.
Send notification on successful data extraction	A notification email will be sent when an agent run meets a defined success criteria. The notification is sent to Success email addresses if specified.

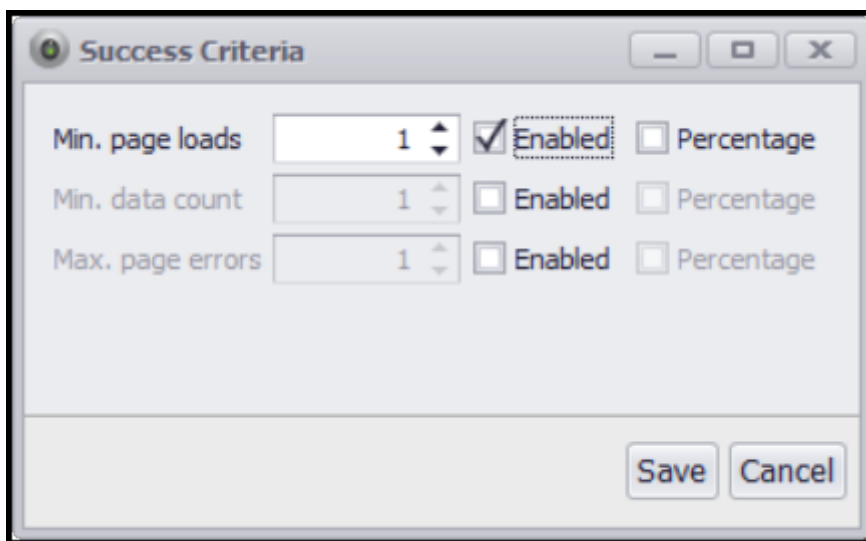
A script can trigger notifications by calling one of these methods on the script parameter object:

Method	Description
void Notify(bool alwaysNotify = true)	Triggers notification at the end of agent execution. If alwaysNotify is set to False , this method only triggers a notification if you configure the agent to Send notification on critical errors (see above).
void Notify(string message, bool alwaysNotify = false)	Triggers notification at the end of an agent execution, and adds a message to the notification email. If alwaysNotify is set to False , this method only triggers a notification if you configure the agent to Send

	notification on critical errors (see above).
--	---

Success Criteria

A success criteria can be defined to tell an agent when it should consider an agent run successful. For example, a success criteria could include that the agent must load a minimum number of pages.



Define criteria for a successful agent run.

The **Success Criteria** window can be opened from the the **Run** menu in the Content Grabber editor, or from the Notification window.

The following three success criteria can be defined.

Criteria	Description
Min. page loads	The minimum number of pages the agent must load.

Min. data count	The minimum number of data entries the agent must extract. A data entry is defined by setting the option Increase data count on any command. The data count is increased every time a command with Increase data count is executed.
Max. page errors	The maximum number of page errors allowed.

Success criteria can be specified as absolute numbers or as a percentage of the value from the last successful agent run. For example, if **Min. page loads** is set to 95% it means the agent must load minimum 95% of the number of pages it loaded at the last successful run. A Percentage criteria always succeeds if the agent has not been successfully run before.

Success criteria can be used to decide when email notifications are sent, but are also used in Change Tracking to decide when it's safe to mark data as deleted.

Email Troubleshooting

Most email servers are very restrictive in regards to who or what is allowed to send emails. If you are getting an error when sending emails from Content Grabber, then it's most likely a problem with the configuration of your account. Here is a list of things you can check:

- Make sure your email account is configured to allow you to send emails from the email address specified in **From email address**.

- Make sure your username and password are correct, and that login to your account is not restricted. Use another email client or your web browser to login to your email account to make sure you can login.
- Use SSL if required. Most online services, such as Gmail, requires SSL.
- Use the correct port. Gmail uses port 587 for example.
- Most online services, such as Gmail, don't allow just any application to login to your email account. Many online services will have a special option to allow any application to login to your account, and you must set this option to allow Content Grabber to login. In Gmail you can set this option on the following page:

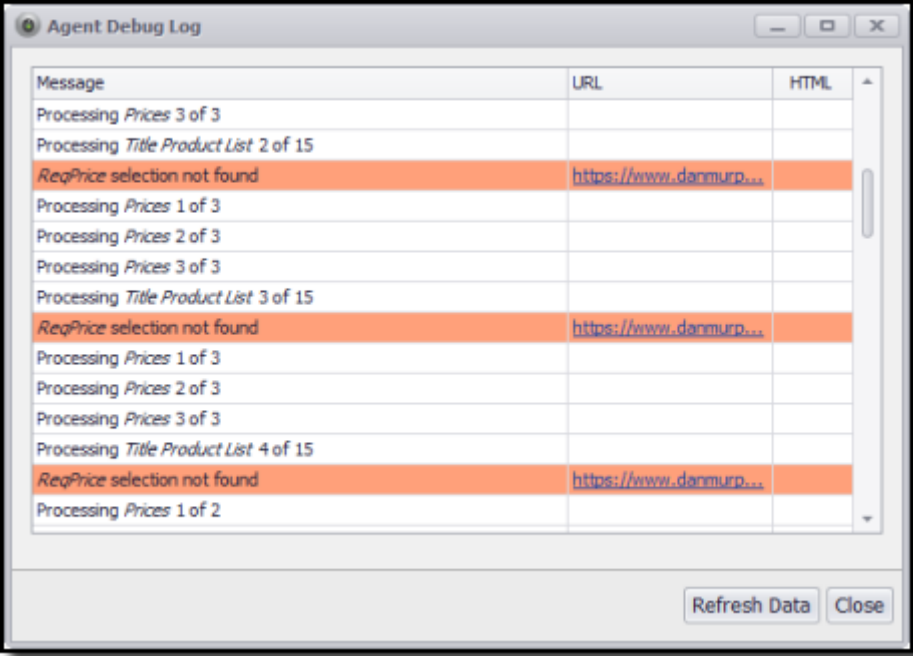
<https://www.google.com/settings/security/lesssecureapps>

Error Logs

Logging is set to low by default when running an agent, and high by default when debugging an agent. Debug logging is always saved to a database table, but runtime logging can be saved to a database table or a text file.

The database table supports the log viewer in the **Content Grabber editor**, which you can access by clicking the **View Log** button in either the **Run** ribbon menu or the **Debug** ribbon menu.

Content Grabber supports 3 log levels, **High**, **Medium** and **Low**. The High log level will log everything. The Medium log level will log errors and warnings, and the Low log level will only log errors.



Message	URL	HTML
Processing Prices 3 of 3		
Processing Title Product List 2 of 15		
ReqPrice selection not found	https://www.danmurg...	
Processing Prices 1 of 3		
Processing Prices 2 of 3		
Processing Prices 3 of 3		
Processing Title Product List 3 of 15		
ReqPrice selection not found	https://www.danmurg...	
Processing Prices 1 of 3		
Processing Prices 2 of 3		
Processing Prices 3 of 3		
Processing Title Product List 4 of 15		
ReqPrice selection not found	https://www.danmurg...	
Processing Prices 1 of 2		

Refresh Data Close

The critical errors in this log are red in color

URL Column

The **URL** column in the **Agent Debug Log** window shows links to all web pages that the agent has processed, and you can double-click on any of these links to open the web page in your default browser. For any failures on data extraction, this is an easy way to open the web page and check if the web page layout is different from the expected layout.

NOTE: Double-clicking on a link to a sub-page only works if the website allows direct links into the sub-pages. Many websites don't allow direct links into sub-pages without going through other pages first. Read below to learn how to directly access the HTML for sub-pages.

HTML Column

If a website does not allow direct links into sub-pages, you can configure an agent to save the entire HTML of all processed pages. When you configure the agent to log all HTML, the HTML column will contain buttons that you can click to view the raw HTML

in the Content Grabber HTML viewer. The raw HTML does not include style sheets and other support files that are necessary to render a complete web page properly, but it will often show you enough information to determine what is causing an error (such a CAPTCHA page).

6.3 Post Status

An agent can be configured to post status information to a web service after it has completed a run. This can be used as an alternative to email notifications.

An agent posts status information as URL parameters or as JSON formatted content. This gives better flexibility than email notifications, since the web service can easily parse the status information and take action depending on that information.

The agent is configured to post JSON formatted status information.

The Post Status screen has the following fields.

Field	Description
-------	-------------

URL	<p>Specifies the method and the URL to the web service. This field cannot be empty. The method must be one of the following:</p> <p>GET POST DELETE PUT PATCH</p>
Post Data	<p>The data to post to the web service. This field can be empty if status information is specified as URL parameters.</p> <p>Content Grabber will automatically replace all parameters enclosed in <>, but any number of custom parameters can be added.</p>
Headers	<p>Optional headers to send with the request. Normally, Content-Type must be specified when posting JSON content.</p>

7 Extracting Data From Non-HTML Documents

Websites generally provide most of their content in HTML format, but some websites may also provide content in other formats - such as **PDF** or **Microsoft Word** documents. Since Content Grabber can only process HTML documents, it will simply download any non-HTML document. Content Grabber can help you extract text and images from within a **PDF** or **Word** document by converting such documents into HTML.

To have Content Grabber convert your non-HTML document, you will need to provide an external document converter. The Content Grabber public website provides a list of open source programs that you can use for this purpose. Please remember that we don't support these tools, and you must comply with the license for any conversion tool.

Limitations

The design of most file formats, including PDF and Word files, doesn't include ease-of-conversion to HTML. So, the conversion output is considerably more difficult to manage than standard HTML. In many cases, you'll have to select the entire HTML page and then use **Regular Expressions** to extract the target content.

Installing a Document Converter

Content Grabber uses a custom script that makes a call to an external document converter, and you can configure this script to call any type of program. The default script can handle the two document converters currently available for download on the Content Grabber website.

Installing the PDF To HTML Converter

Follow these steps to install the **PDF-to-HTML** document converter:

1. Download the *pdftohtml.zip* file from the Content Grabber website:

<https://contentgrabber.com/web-scraping-tools>

2. Extract the content of the zip file into the default **Content Grabber Converters** folder, *My Documents\Content Grabber\Converters*. You can also copy the converter into the corresponding *Public Documents* folder if you need the converter to be available for all users on the computer.
3. The direct path to the document converter should now be:

My Documents\Content Grabber\Converters\pdftohtml\pdftohtml.exe

Installing the Docx To HTML Converter

Follow these steps to install the docx to HTML document converter:

1. Download the *docxtohtml.zip* file from the Content Grabber website:

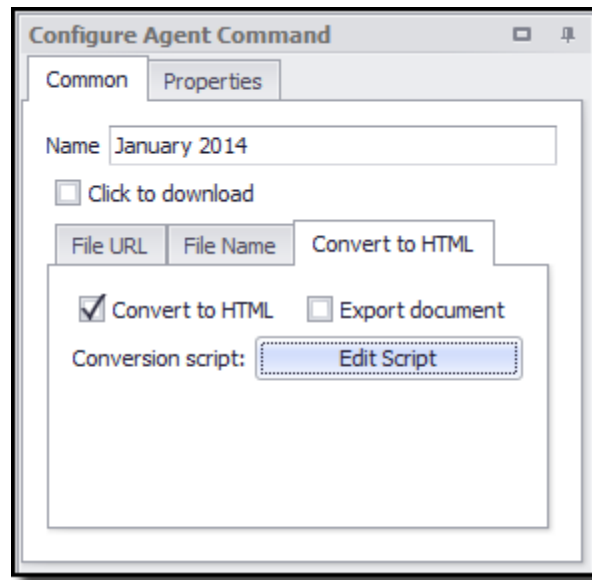
<https://contentgrabber.com/web-scraping-tools>

2. Extract the content of the zip file into the default **Content Grabber Converters** folder, *My Documents\Content Grabber\Converters*. You can also copy the converter into the corresponding *Public Documents* folder if you need the converter to be available for all users on the computer.
3. The direct path to the document converter should now be:

*My Documents\Content
Grabber\Converters\docxtohtml\docxtohtml.exe*

Using a Document Converter

You'll need to add the custom script (that calls the document converters) to a Download Document command, and that same command must also perform the download of the document.



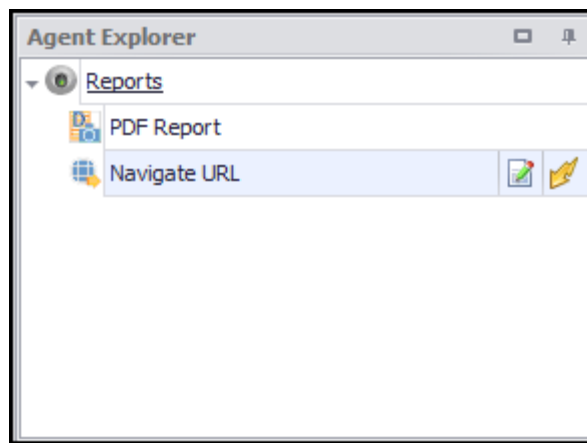
The default conversion script looks like this:

```
using System;
using System.IO;
using Sequentum.ContentGrabber.Api;
public class Script
{
    //See help for a definition of ConvertDocumentToHtmlArguments.
    public static bool ConvertDocumentToHtml(ConvertDocumentToHtmlArguments args)
    {
        if(args.DocumentType=="pdf")
            ScriptUtils.ExecuteCommandLine(@"Converters\pdftohtml\pdftohtml.exe",
                args.DocumentFilePath, args.HtmlFilePath, "-noframes");
        else if(args.DocumentType=="docx")
            ScriptUtils.ExecuteCommandLine(@"Converters\docxtohtml\docxtohtml.exe",
                args.DocumentFilePath, args.HtmlFilePath, "");
        if(!File.Exists(args.HtmlFilePath))
    }
}
```

```
        return false;  
    return true;  
}  
}
```

Extracting Content From a Converted Document

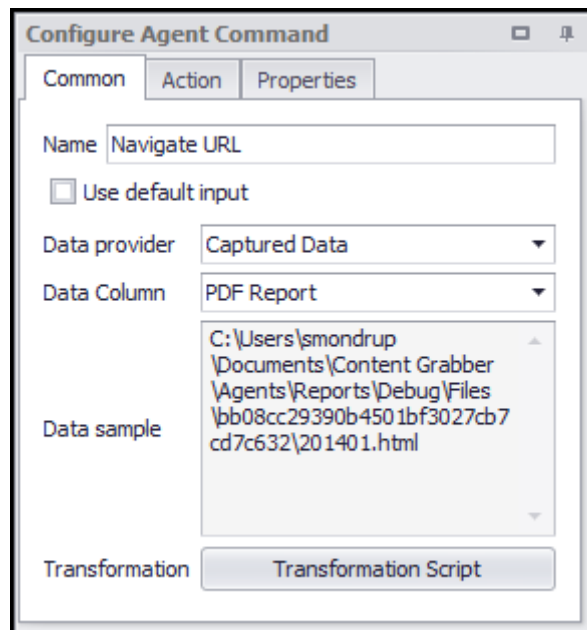
After converting a document to HTML, that document goes into the agent data folder and you can use a Navigate URL command to open the HTML document. The Download Document command that did the conversion will also store the path to the document, and then the Navigate URL command can use that path to get the file URL to the HTML document.



An agent with a URL command that links to a converted HTML document

The Navigate URL command uses data from the Download Document command, so the Download Document command must execute first. Also, both the commands must have the same parent command, or the Navigate URL command must be a child command of the command that contains the Download Document command.

You can execute the Navigate URL command in the editor to open the converted HTML document, but you must first execute the Download Document command to make sure the HTML document is available.



**A Navigate URL command using data captured by a
Download Document command**

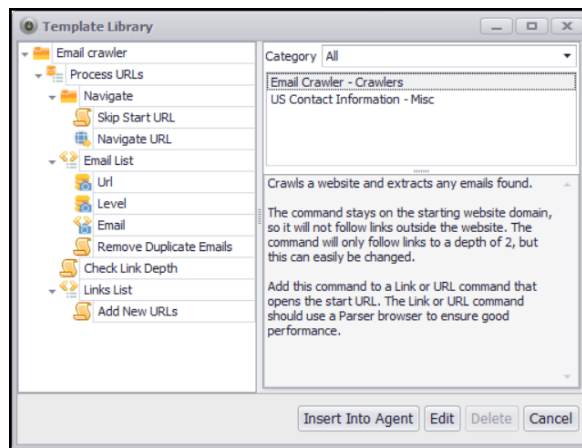
8 Crawling a Website

A Content Grabber agent normally follows a specified path through a website when extracting content. This is much faster than crawling an entire website to look for the content, because page loads are the main performance bottleneck when extracting data, and crawling an entire website will normally result in many more page loads than following a specified path through the website.

Sometimes it may not be possible to specify a fixed path through a website, since you may not know exactly where to find the content that you want to extract. For example, you may want to extract all email addresses on a website, but the email addresses could be located anywhere on the website, so the only option is the crawl the entire website and look for email addresses to extract.

Content Grabber does not contain any built-in web crawling feature, but it's possible to configure a set of commands that works like a web crawler. The command setup is not trivial and requires some scripting, but it's very flexible and performs well.

The Content Grabber template library contains a default set of commands you can use as a basis for your own website crawler. The default command template extracts email addresses from websites, but you can easily change that so it extracts any other content.



Default web crawler template.

9 Data

Content Grabber manages three types of data:

- **Input data** - is used to provide input values to an agent, such as input values for a web form or start URLs. See the topic [Using Data Input](#) for more information.
- **Internal data** - An agent stores data extraction elements in an internal database while the agent is running. When the agent completes, this internal data will become external data. For more information, see the topic [The Internal Database](#).
- **External data** - this is data that goes out to a specific export target. See the topic [Exporting Data](#) for more information about exporting data.

Distributing Data

If an agent is exporting data to a particular file type, such as a spreadsheet, then the agent can also send the file as an attachment to an email message to one or more recipients. The agent can also FTP the files to a remote computer. See the topic [Distributing Data](#) for more information.

We also recommend these topics:

- [Database Connections](#)
- [The Internal Database](#)
- [Using Data Input](#)
- [Exporting Data](#)
- [Distributing Data](#)

9.1 Database Connections

Many professionals who use Content Grabber will normally involve the use of one or more of their database servers, and a web-scraping agent can both import and export data to multiple servers. To increase performance and reliability, you can also

configure an agent to have its primary database on an external server instead of using the default **SQLite** file database.

Development Environments

As you gain proficiency with agent development and deployment, you may find it best to use separate computing environments for the various stages: a developer environment, a testing environment, and a production environment. Each of these environments will typically have its own database, and so the database connections in your agent will need to change when you move an agent from development into testing, and then from testing into production.

Unique Database Names on Your Network

To make it easier to move agents between different computer environments, you can configure network database connections - each of which has a unique name on the network. Using such names will allow for an automatic switch to a new connection if the agent moves to another computer on the network. Also, we recommend that you take this approach, even if you are only using a single computer to design and run agents, since you won't have to configure a new database connection each time you create an agent.

Another benefit of using unique-name database connections is the agent will contain only the *name* of the connection. This design gives additional security when you send agents to others, since the agent will contain no database connection information.

If you don't configure a network database connection, then the connection will be known only to the agent in the location that you configure the agent. You can use the database connection anywhere in the agent, but it will be unknown to other agents.

9.2 The Internal Database

Content Grabber uses two layers of internal data, *internal* data and *export* data. While the agent is running, it continuously saves data in an internal format. When the agent

finishes, it converts the internal data into export data, and then it sends this data to the export target so it becomes external data.

The default internal database is a SQLite database, but this can be changed SQL Server or MySQL. Oracle and OleDb are not currently supported as internal databases.

Internal Data

Content Grabber stores internal data in the database - which contains all the data extraction elements for any agent, but also contains the data which corresponds to the settings for the agent properties. For example, if an agent stops, crashes, or otherwise fails, the internal data is used to start the agent again exactly at the point of failure. Also, the internal data will always contain a data table for each container command in the agent and at least one data field for each capture command in the agent. You can always view the internal data by clicking the **Data > View Internal Data** menu.

If you configure an agent to add new data to the existing data, then the internal data store will continue to grow larger. Otherwise, the internal data store will be recreated every time the agent is run.

Export Data

The agent stores the export data in the internal database, and this data doesn't contain any of the property data for the agent itself. Also, the export data is more readable than the internal data. The export data will usually not contain a data table for each container command in the agent, and some capture commands may not have corresponding data fields.

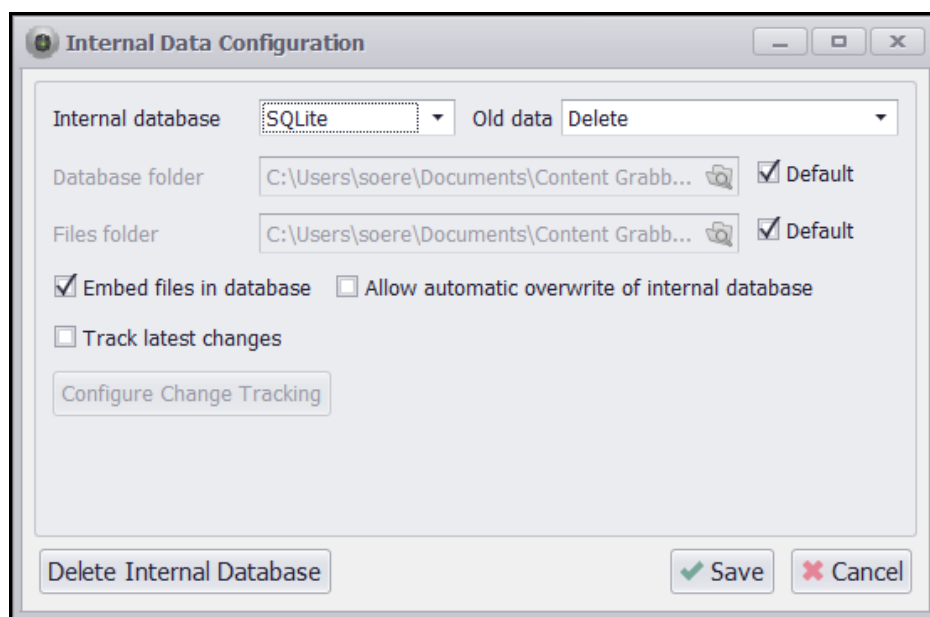
Export data is always overwritten, so you cannot add data to existing export data. You can only add data to existing internal data. You can always view the internal data by clicking the **Data > View Export Data** menu.

External Data

External data is the same as the export data, and the agent delivers it to an external data store chosen by the user. Typically, external data is overwritten, except when you choose **Export last data segment only** and you are exporting to a database. In that case, the operation simply adds the data to the external database tables. You can also use a Data Export Script to customize the export process which will allow you to update or add to existing data. The export data is set in a fixed format, but you have some options for changing this format. You need to use an Data Export Script if you want to automatically create or configure your own data structures within this data.

Internal Database

The Internal Database window can be opened from the **Data** menu in the Content Grabber editor. You can use this window to change the internal database. The default internal database is a **SQLite** file database, but you can change it to either a SQL Server or MySQL database. Changing the internal database from SQLite to SQL Server or MySQL can increase performance of agents significantly.



Internal database settings.

The **Old data** option on the Internal Database window allows you to control how long extracted data is kept in the internal database. The following options are available.

Option	Description
Delete	All previously extracted data is deleted when an agent starts a new run.
Keep All and Export	<p>Extracted data is never deleted from the internal database, and all extracted data is exported to the chosen export target.</p> <p>This option is often used when an agent has been configured to extract only new data. The agent can check previously extracted data and stop when it reaches data that has already been extracted.</p>

**Keep Some and
Don't Export**

The agent will keep data from the last successful run, but it will only export data from the current run.

This option is often used when previously extracted data can be used to increase performance of an agent. For example, if the agent downloads large files, it may be able to use information on the website to see if a file has changes, and if a file has not changed, then copy the file from the previously extracted data rather than downloading the file again.

The **Embed files in database** option is used to control whether extracted files are stored in the internal database or on disk.

The **Track latest changes** option can be used to keep track of the latest changes that have been made to extracted data. The agent will mark extracted data as deleted, modified or added. See the Change Tracking topic for more information.

When changes are made to an agent, the internal database tables may need to change as well. For example, if the agent is modified to extract more data fields, the internal database tables need more columns to store the new data. Content Grabber needs to recreate the internal database tables when such agent changes are made, and this will remove all existing data in those tables. This will reset change tracking, and could have other serious consequences if an agent is configured to use previously extracted data.

The option **Allow automatic overwrite of internal database** can be used to control whether an agent is allowed to recreate the database tables. An error occurs if a change is made to an agent that requires the database tables to be recreated, and the agent is not allowed to do so. In this case a user may be able to manually make changes to the internal database tables to accommodate the modified agent without losing existing data.

Important Notice About SQL Server and MySQL

WARNING: It is important to ensure that no two agents that use the same internal database have the same agent name.

If two agents use the same internal database, it's important they don't use the same agent name. When an agent recreates the internal database it first removes all tables for that agent by deleting tables starting with the name of the agent.

For example, if the agent uses the following internal tables, it would remove all tables starting with SWSCR [Agent Name].

SWSCR [Agent Name] AGENT PROCESS
SWSCR [Agent Name] AGENT PROCESS
SWSCR [Agent Name] AGENT BROWSER
SWSCR [Agent Name]
SWSCR [Agent Name] [Command Name 1]
SWSCR [Agent Name] [Command Name 2]

This approach creates an obvious problem when two agents use the same internal database and also start with the same name, because Content Grabber would remove the tables for both projects when it should only remove tables for one project. It is important to ensure that no two agents that use the same internal database have the same agent name.

9.3 Using Data Input

Content Grabber can take input data from a wide variety of data sources, such as databases and CSV files. Though you can use input data for many different purposes, its most common use is for loading URLs or inputting values for web forms.

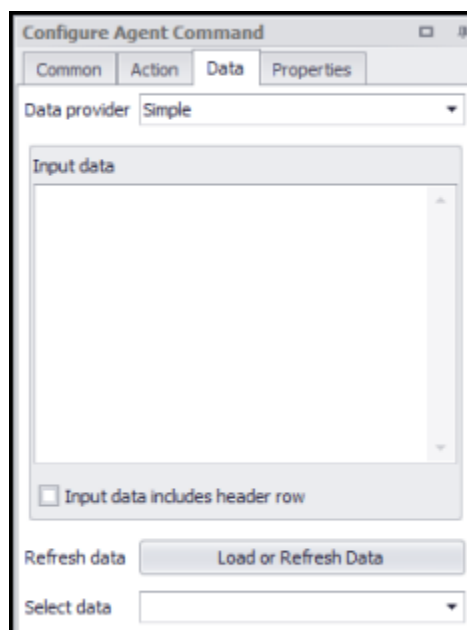
We explain each of the types of input data in these sections:

- Data providers
- Input parameters
- Agent data

9.3.1 Data Providers

You can assign a data provider to any Data List command and can load data from the following data sources:

- Simple
- CSV
- Selection
- Script
- SQL Server
- MySQL
- Oracle
- OleDb



Data List commands include the following:

- Agent
- Set Form Field
- Navigate URL
- Data List

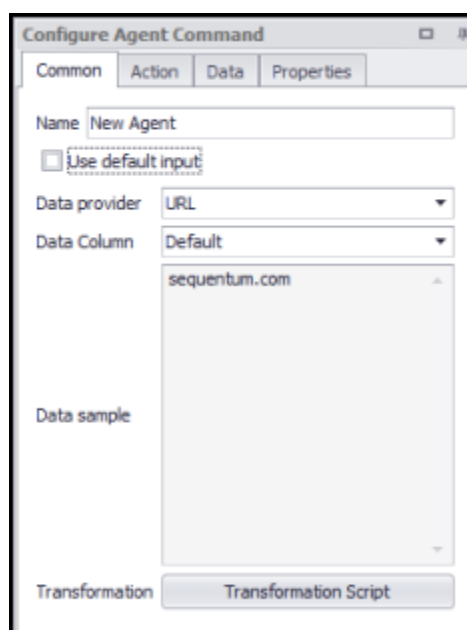
A list command will iterate through all the data rows given by the data provider and execute each of the sub-commands once for each row.

Consuming Data

Consumer commands use the data that comes from the **Data Provider**. The consumer must be a sub-command of the command providing the data. Data consumers include the following types of commands:

- Agent
- Set Form Field
- Navigate URL
- Data Value

Most data providers are also data consumers and, consequently, many commands will retrieve data automatically. For example, an agent command will normally provide at least one start URL to itself - as shown in the figure.



You can choose the **Data Provider** on the **Common** tab of the **Configure Agent Command** panel. If multiple data columns are available, then you can choose a specific **Data Column**. The **Default** data column is simply the first data column that the data provider presents.

A data provider will often provide multiple data columns for data sets that correspond to a web form. An example would be an agent that searches for air fares, which would submit a list of departure and destination cities to a web form. The data provider would then provide two columns: one for departure city and one for destination city.

Consuming Data in a Script

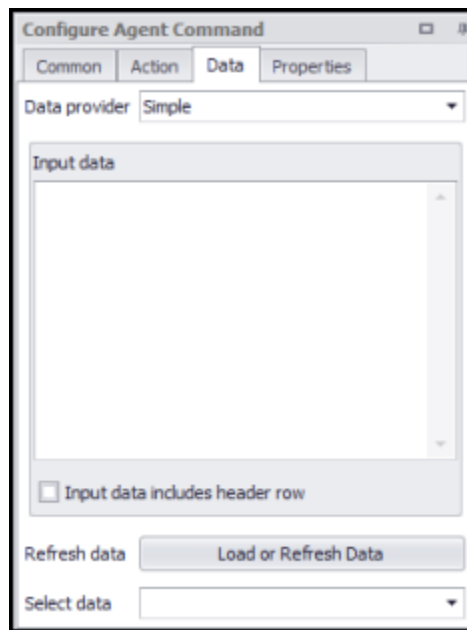
A script can consume data from a data provider as long as the script corresponds with a command that is a sub-command of the data provider. Most scripts provide easy access to input data from the script arguments. The following example retrieves the current data from the data column **DepartureCity** in the **CityData** data provider.

```
args.GetInputData("CityData").GetStringValue("DepartureCity");
```

See the Scripting article for more information.

Simple Data Provider

Choosing **Simple** for the **Data Provider** will cause the agent to extract data from a CSV data set that the agent designer enters at design time. The agent contains the CSV data internally, so that there is no dependency on external files.



The CSV data set may contain multiple rows, each having multiple comma-separated data values. The data follows the same formatting rules as standard CSV data. So, you should enclose a data value in double-quotes if it contains a comma or single quote, and use two double quotes to escape any double quotes in a data value.

The following example contains a header row, although a header row is not mandatory. If the data contains a header row, then you need to check the **Input data includes header row** box on the **Data** tab.

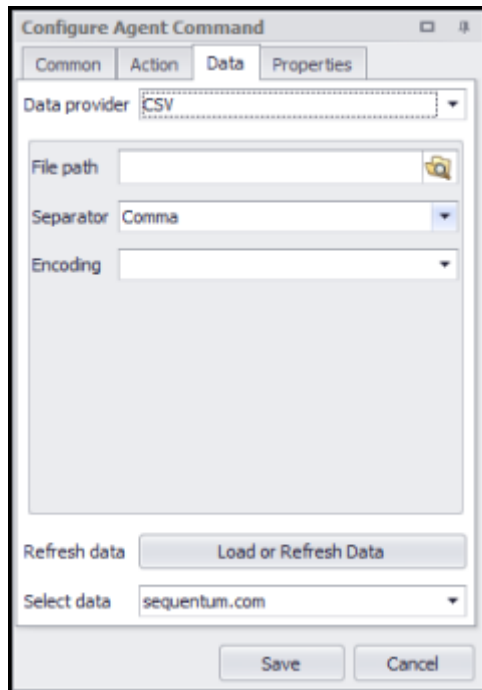
```
Departure City, Destination City
Sydney, Brisbane
Sydney, Melbourne
```

CSV Data Provider

Choosing **CSV** for the **Data Provider** is similar to the **Simple** data provider, but uses an *external* CSV file.

We recommend that you choose the **CSV Data Provider** for large CSV files, since the CSV data provider will perform much better than the **Simple** data provider for large quantities of data.

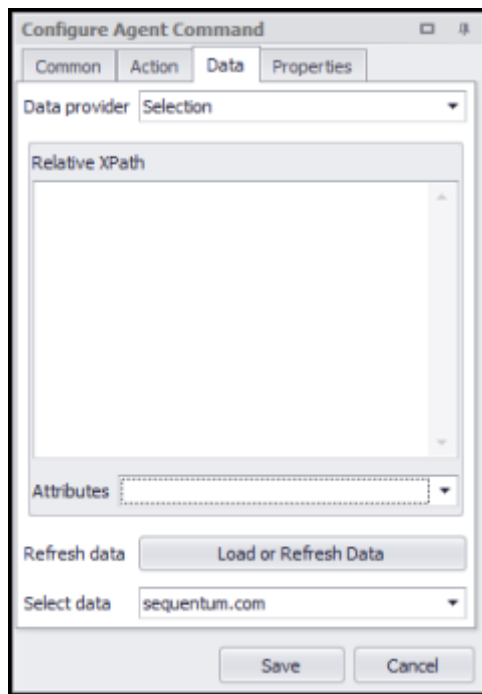
For a **CSV** provider, you can choose the value **Separator** and the text **Encoding** of the CSV file.



You can place CSV files anywhere on your computer, but we recommend you place them in the default input data folder for your agent. Later, if you want to export your agent, then you can include these files along with the export.

Selection Data Provider

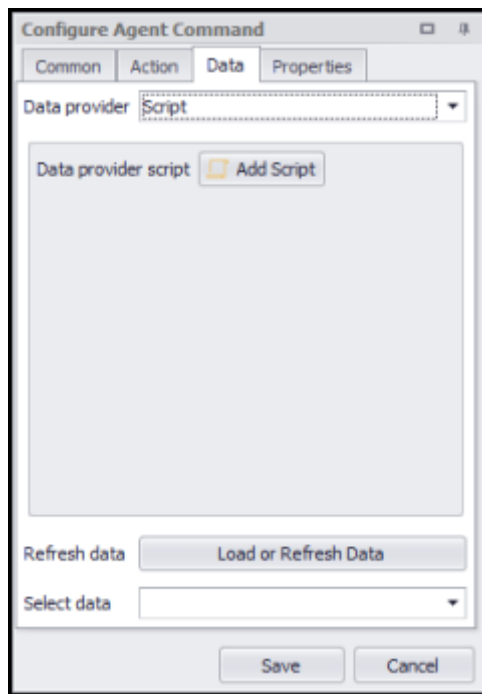
Choosing **Selection** for the **Data Provider** will extract data from elements on the web page. The data provider uses the selection **XPath** of the data provider command and then adds a relative **XPath** to find the web elements that you include in the selection. You can choose which HTML attributes of the web elements to include as data.



The choice of **Selection** for the **Data Provider** is almost exclusively for use with drop-down menus in web forms. A form field command selects the drop down on the web page, and a relative **XPath** selects the option elements within that drop down. The drop-down options then become available to the form field command, which can iterate through the options and ensure that the web form submission is done for each item in the drop down. Although this is somewhat tedious, the benefit is more precise control over which options you want to submit with the web form.

Script Data Provider

Choose **Script** for the **Data Provider** for full customization of the agent input data. This option provides a .NET data table containing the input data, which may contain multiple data columns and rows.



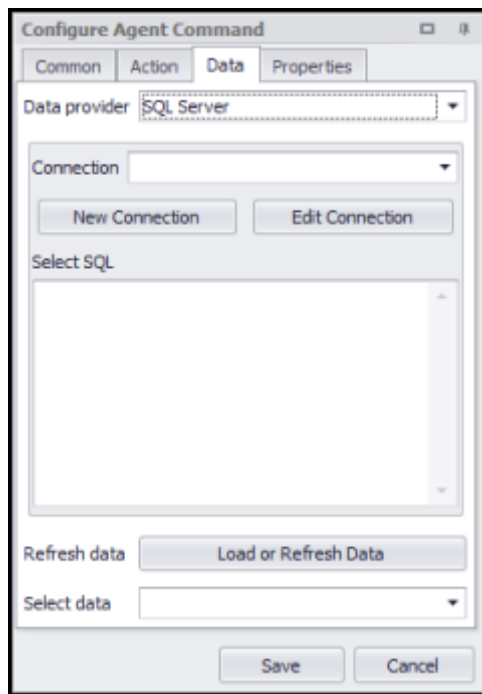
Content Grabber provides some standard .NET libraries which makes it easier to generate .NET data tables for a variety of input data. See the topic [Data Input Scripts](#) for more information and sample code.

Database Data Providers

Choose a **Database** for the **Data Provider** to work any of these database connections:

- SQL Server
- Oracle
- MySQL
- OleDb

In Content Grabber, you can share database connections among all agents on a computer. We recommend that you read more about [create new database connections](#).

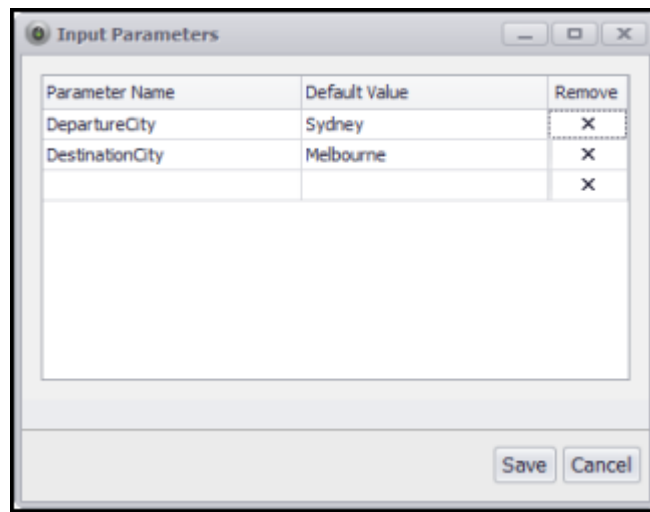


A database data provider requires execution of a SQL *Select* statement on the database connection to retrieve the input data. The following example selects the two data columns, **DepatureCity** and **DestinationCity**, from a table **City**.

```
SELECT DepatureCity, DestinationCity FROM City
```

9.3.2 Input Parameters

Input parameters are named values that get their value assignments when an agent starts. You can specify these as command-line parameters when running an agent from the command line, or specify them in a GUI window when running the agent from the Windows desktop. Any data consumer command within the agent can use these parameters.

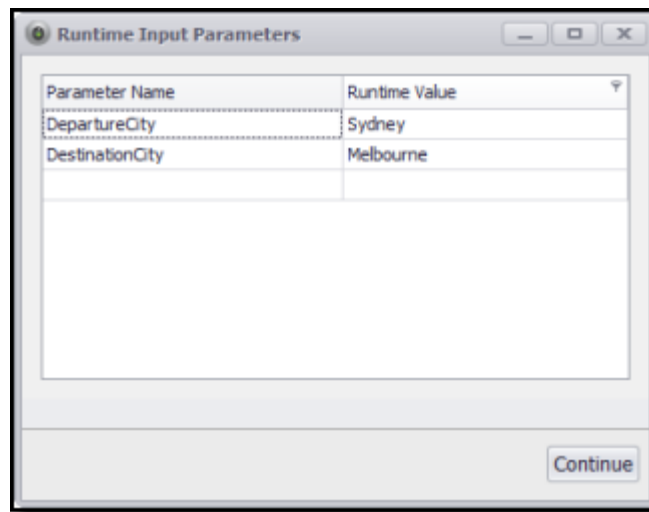


You can configure input parameters to get value assignments at runtime by using one of these methods:

- A GUI window
- Command-line parameters
- The Content Grabber API

Input Parameters with a GUI Window

When running an agent from the Content Grabber editor and the agent has any input parameters, then the **Runtime Input Parameters** window will automatically appear.



Runtime Input Parameters

The user running the agent can enter runtime values for the input parameters and can even add new parameters that are only defined during the current execution of the agent. An agent will always use the default input parameters when debugging. The **Runtime Input Parameters** window will not appear when starting a debugging session, so you will need to change the default values if you want to debug the agent with different input parameters.

Specifying Input Parameters on the Command Line

When running an agent from the command line, you can specify input parameters as command-line arguments. It's unnecessary to specify all of the input parameters. If you don't specify a parameter, the call to run the agent will use a default value for that parameter. See the topic *Running Agent from the Command-Line* for more information.

The following example runs an agent named **Sequentum** on the command line, and then sets the input parameters **DepartureCity** and **DestinationCity**.

```
RunAgent.exe Sequentum -DepartureCity "Sydney" -DestinationCity "Melbourne"
```

Specifying Input Parameters Using the Content Grabber API

When running an agent using the API, you can specify input parameters using the **AgentSettings** class in the Content Grabber API. See the topic Programming Interface for more information.

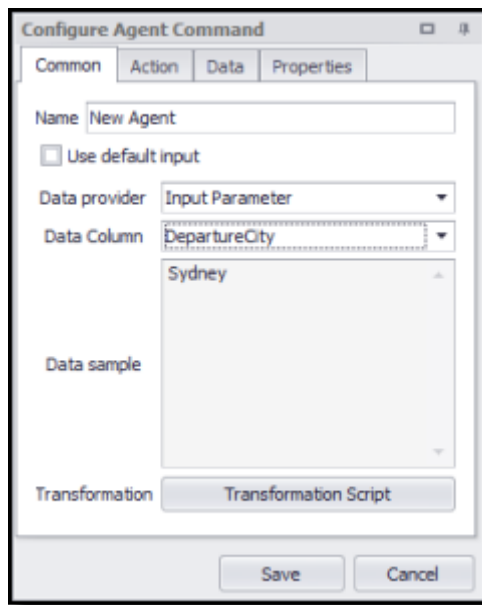
The following example uses the API to add a set of input parameters and then run the agent:

```
AgentApi api = new AgentApi(@"C:\Users\Public\Documents\Content Grabber\Agents\
    qantasApiTest\qantasApiTest.scg");
AgentSettings settings = new AgentSettings();
settings.InputParameters.Add("from", "SYD");
settings.InputParameters.Add("to", "MEL");
settings.InputParameters.Add("departure_date", DateTime.Now.ToString("yyyy-MM-dd"));
settings.InputParameters.Add("return_date", DateTime.Now.ToString("yyyy-MM-dd"));
settings.InputParameters.Add("travel_class", "ECO");
settings.InputParameters.Add("adults", "1");
settings.InputParameters.Add("children", "0");
settings.LogLevel = AgentLogLevel.High;
settings.IsLogToFile = true;
api.RunAgent(settings);
```

Using Input Parameters

Any data consumer command, such as the following, can use input parameters.

- Agent
- Set Form Field
- Navigate URL
- Data Value



Using Input Parameters

Using Input Parameters in a Script

Most scripts provide easy access to input parameters from the script arguments. See the topic [Scripting](#) for more information. The following example retrieves the input parameter **DepartureCity**:

```
args.GlobalData.GetString("DepartureCity");
```

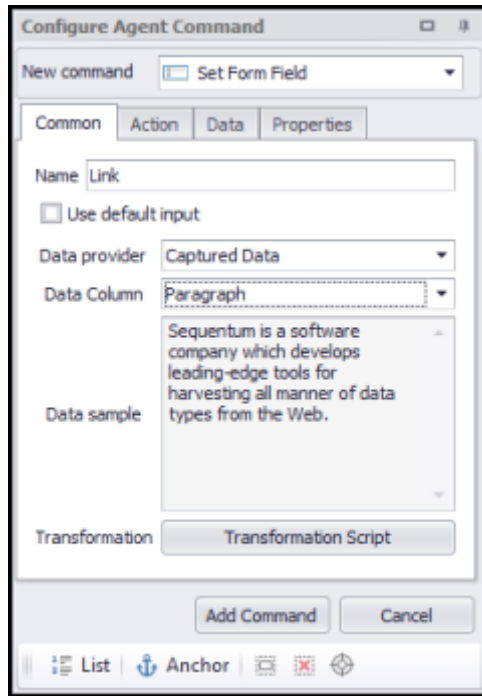
9.3.3 Agent Data

Agent data is data that is currently being extracted by the agent. For example, data extracted by a capture command could be used as input to a form field command.

Agent data can be used by data consumer commands, but the data consumer must be a sub-command of the capture command which is extracting the data. Else, it must have the same parent command as the capture command. See [Agent Data](#) for more information.

Agent data is the data that the agent is currently extracting, and it can be put to use by other commands at runtime. If another command uses such data, then it must execute after the command that extracts the data. Otherwise, the data wouldn't be

available. Also, the command using the data must be a sub-command or have the same parent command as the command that extracts the data.



Using Agent Data

Agent data can be used by any data consumer command. Data consumers include the following types of commands:

- Agent
- Set Form Field
- Navigate URL
- Data Value

CAPTCHA Protection

Agent data is often used when resolving CAPTCHA images. You can automatically process a CAPTCHA page by using an OCR service to convert the CAPTCHA image into text, and then use that text in a form field command to set the CAPTCHA input box.

A CAPTCHA page or section appears on some websites to protect access to a secure area of the website. A CAPTCHA image displays a sequence of numbers and characters that a website user must type into an input box in order to continue to the secure area. This process ensures the website user is a real human and not an automated agent. See the topic CAPTCHA Blocking for more information.

9.4 Exporting Data

While performing a data extraction, an agent saves data to an internal data store and then exports that data to a specific export target when the agent completes or if the user manually stops the agent.

Content Grabber supports many different types of data stores such as databases, CSV files, XML files and spreadsheets. You can use a data export script to gain complete control over the entire export process - including the export of data to data stores that Content Grabber doesn't support.

We recommend that you also read these topics:

- [Selecting an Export Target](#)
- [Exporting Downloaded Images and Files](#)
- [Character Encoding](#)
- [Changing the Default Data Structures](#)
- [Scripting Data Export](#)

9.4.1 Selecting an Export Target

Content Grabber can export data to many different data formats. By default, data will export to an Excel spreadsheet, but you can change the export target by clicking the menu **Data > Data Export**.

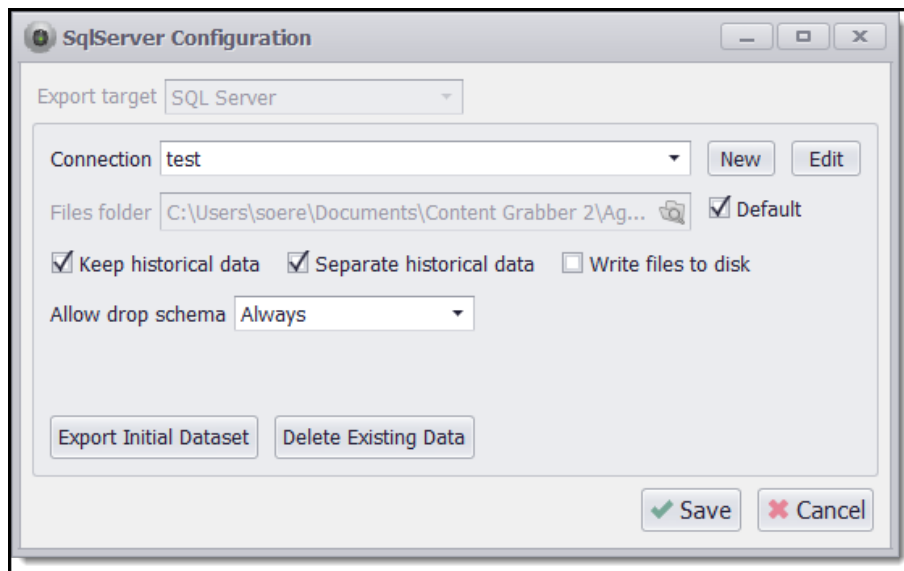
Content Grabber supports the following export targets:

- **None** - data extractions will only remain in the internal database, and the agent won't export the data. This is a useful option if you want to work directly on the internal database and have no need for exporting. **NOTE:** This option is for expert users only.
- **XML** - the data will export to a single XML file.
- **JSON** - the data will export to a single JSON file.
- **SQL Server** - the data will export to one or more SQL Server database tables.
- **MySQL** - the data will export to one or more MySQL database tables.
- **Oracle** - the data will export to one or more Oracle database tables.
- **OleDB** - the data will export to one or more database tables. You can use any database that has an **OleDB** provider.
- **CSV** - the data will export to one or more CSV files. The default character encoding is ASCII, but you can specify another type of encoding.
- **Microsoft Excel 2003** - the data will export to a single Excel spreadsheet, but it may save the data into more than one worksheet. The agent will save any images to the disk.
- **Microsoft Excel 2007 and later** - the data will export to a single Excel spreadsheet. You can save multiple data tables to a single worksheet using grouping and outlining. The agent will save any images to the disk or embed them into the worksheet.
- **PDF** - the data will export to a PDF document. The data will be displayed the same way as for Excel 2007. Only content that is visible in Excel will be written to PDF, and since Excel has a limit to the maximum cell height, long text content may not be exported correctly to PDF.

Content Grabber always stores data in the internal database before exporting the data to the chosen output format. This means you can always export data to a new format without having to extract the data again.

Exporting Data to a Database

When exporting the data to a database, Content Grabber automatically creates new database tables if they don't exist. The option **Allow drop schema** can be set to specify when an agent is allowed to automatically delete and re-create existing database tables if they do not have the correct number of columns.



SQL Server export target.

Use the button **Export Initial Dataset** to write any existing extracted data to your database tables. This is mostly useful when using change tracking and exporting changed data only, so you get any existing data in your database tables before starting to export changes only.

Use the button **Delete Existing Data** to remove all database tables associated with this agent. This will delete all exiting data including any historical data.

Keeping Historical Data

An agent can keep historical data when exporting to a database. The historical data will include the most recent data set and all previously extracted data sets. An agent can save the most recent data set in one set of database tables, and the historical data in a separate set of database tables, or the agent can save only historical data in a single set of database tables.

Exporting Data to Oracle

Content Grabber supports **Oracle** databases but only through the **Oracle OleDb** provider. You must first install the **Oracle** client software before the **OleDb** provider will work. The **OleDb** provider given by **Microsoft** supports **Oracle** only up to version 8i, so you should use the **OleDb** provider from **Oracle**.

Use an Oracle **OleDb** connection string such as this one:

```
Provider=OraOLEDB.Oracle;Data Source=//localhost:1521/OracleTest;  
User Id=test;Password=test;
```

The **Data Source** parameter in the connection string can be set to **TNS**, but only if you configure the TNS configuration file properly.

Exporting Data to MS Access

Content Grabber is a 32-bit application and will only be able to communicate with a 32-bit **OleDb** provider. If you have installed the 64-bit version of **Microsoft Office**, then you won't be able to use the 32-bit **OleDb** provider, and Content Grabber won't be able to export data to **Microsoft Access**. If you are using the 32-bit version of **Office**, you can use a connection string such as this:

```
Provider=Microsoft.ACE.OLEDB.12.0;Data Source=  
C:\Data\MyDatabase.accdb;Persist Security Info=False;
```

You must also change the **Parameters** option from "@" to [@].

9.4.2 Exporting Downloaded Images and Files

An agent will often download images and files during a data extraction. By default, the agent saves these files in the internal database. However, you can configure the agent to save all files to disk by unchecking the **Embed files in database** box on the **Internal Database** configuration screen. When writing files to disk, the agent will store the full path to each file in the internal database.

If you are exporting data and there are corresponding files in the internal database, then these files will also export to the external database. If you uncheck the **Embed files in database** box, then only the file paths will export to the external database. If you are exporting data to a file format that doesn't support embedded files, such as Excel 2003, then only the file paths will export to the external database and the files will be written to disk.

Typically, Content Grabber will use the name of a file when saving it to disk. However, since a website could potentially use the same name for two different files, Content Grabber will set the file name with a unique identifier (GUID) if the file already exists on disk.

You can also use a file name transformation script to name the files, and configure the script to use some of the other extracted data elements for the file name. For example, you could name a product image with the product name or product ID.

9.4.3 Character Encoding

Although the user interface text is English-only, Content Grabber can extract data from websites in any language or encoding. If you have problems with the encoding of your data extraction, then the problem is most likely with your export target. For example, if you are exporting data to a **MySQL** database that uses Latin1 encoding, then you'll find a question mark in the place of each Unicode character.

In accordance with CSV specifications, CSV output files have ASCII encoding and do not support Unicode. However, if you're using a CSV import tool that can read CSV files with other character encodings, then you can specify the encoding when choosing the CSV export target.

MySQL Character Encoding

Many users install their **MySQL** databases with Latin1 character encoding and then encounter some characters appearing as question marks. If you want to export Unicode characters to a **MySQL** database, you'll need to use Unicode character

encoding such as UTF-8. If you configure your **MySQL** database to use Unicode, you must also specify the encoding when configuring the **Export Target** for the agent.

9.4.4 Changing the Default Data Structures

All container commands have a set of properties that you can use to specify how to export data extracted by commands in the containers:

Property	Description
Export Method	<p>Specifies how to export data extracted by commands in the container. The following options are available:</p> <p>Separate - Extracted data is written to a separate child table. The data in the child table will be linked to data in its parent table by a key.</p> <p>Add columns and Rows - Extracted data will be merged with data extracted by the parent container command. Each data entry extracted by this container command will add a new row in the data table, and new data fields will add new columns in the data table.</p> <p>Add columns Only - Extracted data will be merged with data extracted by the parent container command. Data entries extracted by this container will not add new rows to the data table, but new data fields will add new columns to the data table. If</p>

	<p>this container extracts more than one data entry, only the last data entry will be used.</p> <p>Add Columns and Merge Rows - Extracted data will be merged with data extracted by the parent container command. If this container extracts more than one row, the values for each row will be combined to generate a single row, which will then be merged with data from the parent container command. The Capture command option Merge Rows Method specifies how the row values are combined.</p> <p>Convert Rows Into Columns - Extracted data will be merged with data extracted by the parent container command. Each row extracted by this container adds new columns to the data table. The option Export Rows to Columns Name Command can be used to specify which Capture command extracts the data that will be used as the new column name, and Export Rows to Columns Value Command can be used to specify which Capture command extracts the data that will be stored on the column.</p>
Export Rows to Columns	<p>A capture command that will provide the data for column names. This property is optional, but must be set together with a</p>

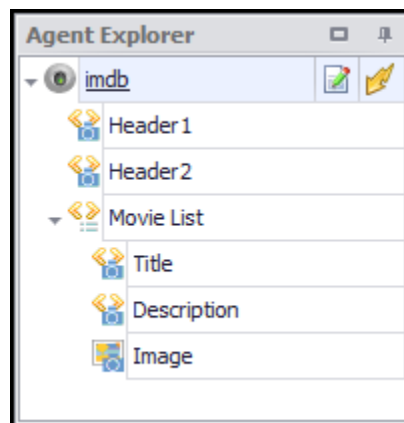
Name Command	command name for the Value Command property.
Export Rows to Columns Value Command	A capture command that will provide the data for column values. This property is optional, but must be set together with a command name for the Name Command property.
Export ID Name	Specifies the name of the primary key column in the exported table (database table, spreadsheet, CSV file, XML Node), if this container generates a new table. The export name post-fixed with ID is used if this property is empty. If multiple agents are exporting data to the same table, then you must set this option to the same value for all those agents.
Export Name	Specifies the name of the exported table (database table, spreadsheet, CSV file, XML Node), if this container generates a new table. The command name is used if this property is empty. If multiple agents are exporting data to the same table, then you must set this option to the same value for all those agents.
Plural Export Name	Specifies the plural name of the exported table if this container generates a new table. The Export Table Name with the added character s is used if this property is empty. This property can be used to control the name of XML nodes when exporting to XML.

Export Empty Row If No Data	Exports a single empty data row if this container extracts no data. Parent and sibling data will be lost if merged with an empty data set, so this option ensures that parent and sibling data is exported when this container extracts no data.
------------------------------------	--

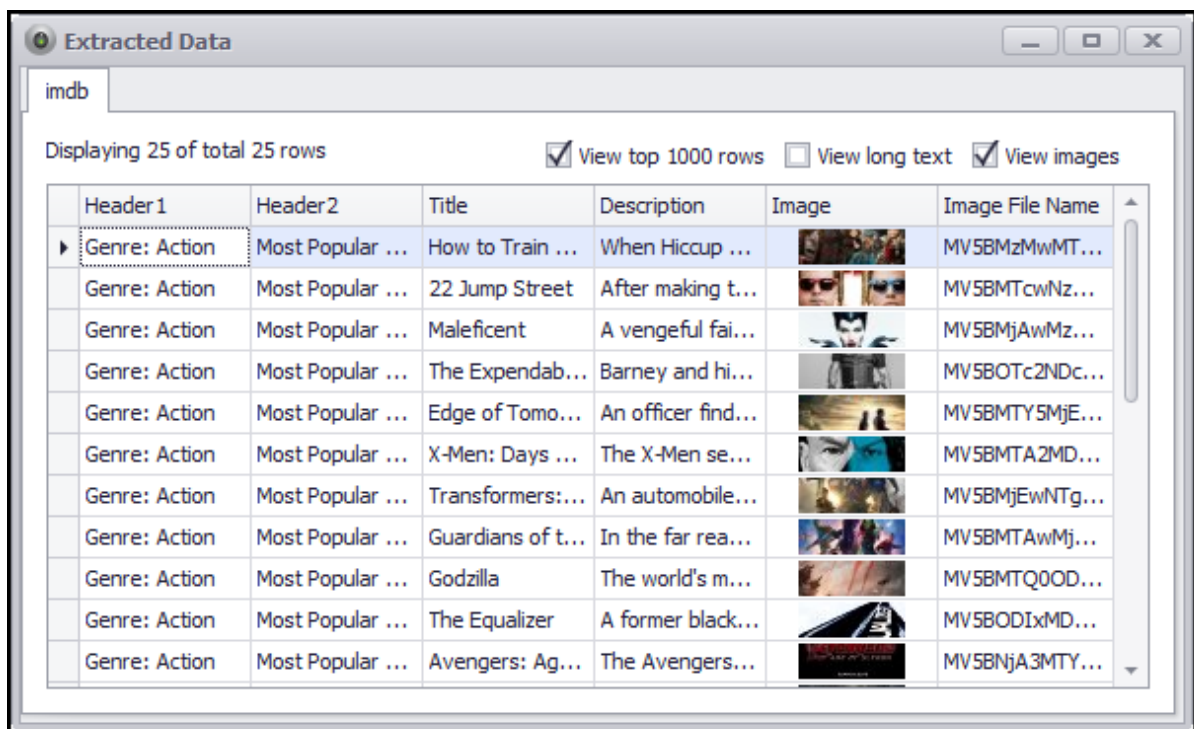
Separate Export Method

When using this method, data extracted by commands in the container will be saved separately. If, for example, the data exports to a database, then the data extracted by the container will go into separate database tables.

The figures below show a simple agent having one list container, along with the data export it generates. By default, the **Movie List** container will merge its data with the data from the **imdb** container, and the result will be a single data table containing the entire data extraction.














A simple agent with one list command



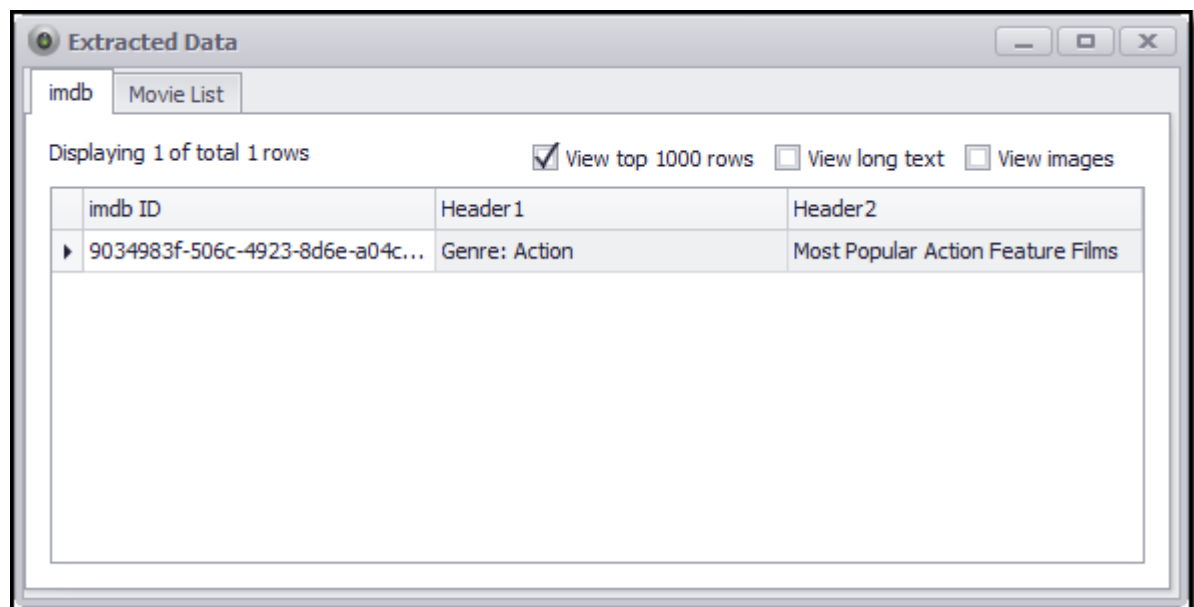
imdb

Displaying 25 of total 25 rows ☒ View top 1000 rows ☐ View long text ☒ View images

Header1	Header2	Title	Description	Image	Image File Name
Genre: Action	Most Popular ...	How to Train ...	When Hiccup ...		MV5BMzMwMT...
Genre: Action	Most Popular ...	22 Jump Street	After making t...		MV5BMTcwNz...
Genre: Action	Most Popular ...	Maleficent	A vengeful fai...		MV5BMjAwMz...
Genre: Action	Most Popular ...	The Expendab...	Barney and hi...		MV5BOTc2NDc...
Genre: Action	Most Popular ...	Edge of Tomo...	An officer find...		MV5BMTY5MjE...
Genre: Action	Most Popular ...	X-Men: Days ...	The X-Men se...		MV5BMTA2MD...
Genre: Action	Most Popular ...	Transformers:...	An automobile...		MV5BMjEwNTg...
Genre: Action	Most Popular ...	Guardians of t...	In the far rea...		MV5BMTAwMjE...
Genre: Action	Most Popular ...	Godzilla	The world's m...		MV5BMTQ0OD...
Genre: Action	Most Popular ...	The Equalizer	A former black...		MV5BODIxMD...
Genre: Action	Most Popular ...	Avengers: Ag...	The Avengers...		MV5BNjA3MTY...

The above agent extracts this data by default. Data from the *Movie List* container merges with data from the "imdb" container.

In this example, you can separate the data from the **Movie List** command by setting the property **Export Method** to **Separate**.



imdb Movie List

Displaying 1 of total 1 rows ☒ View top 1000 rows ☐ View long text ☐ View images

imdb ID	Header1	Header2
9034983f-506c-4923-8d6e-a04c...	Genre: Action	Most Popular Action Feature Films

The data is exported to separate data tables when the property Sub-Container Export method is set to Separate.

Add Columns Only Export Method

When using this export method, data from the container will combine horizontally with data from its parent container as shown in the example below. Only the last data row will be used of the container extracts more than one data row.

Consider an example in which the parent container and the current container generate the following data sets. Here is the data for the parent container:

Column1	Column2
Data1	Data2
Data3	Data4
Data5	Data6
Data7	Data8
Data9	Data10

Here is the data from the current container:

Column3	Column4	Column5
Data11	Data12	Data13

This would be the result of a merge of these two data sets:

Column1	Column2	Column3	Column4	Column5
Data1	Data2	Data11	Data12	Data13
Data3	Data4	Data11	Data12	Data13
Data5	Data6	Data11	Data12	Data13
Data7	Data8	Data11	Data12	Data13

Data9	Data10	Data11	Data12	Data13
-------	--------	--------	--------	--------

Add Columns and Merge Rows Export Method

When using this export method, data from the container will be combined into a single row, which is then merged with data from its parent container as shown in the example below.

Consider an example in which the parent container and the current container generate the following data sets. Here is the data for the parent container:

Column 1	Column 2	Column 3
Data7	Data1	Data2
Data8	Data3	Data4
Data9	Data5	Data6

Here is the data from the sub-container:

Column 4	Column 5
Data10	Data12
Data11	Data13

The result will look like the table below if the option **Merge Rows Method** is set to **Concatenate** for both Column 4 and Column 5:

Column1	Column 2	Column3	Column4	Column5
Data7	Data1	Data2	Data10, Data11	Data12, Data13

Data8	Data3	Data4	Data10, Data11	Data12, Data13
Data9	Data5	Data6	Data10, Data11	Data12, Data13

Convert Rows Into Columns

When using this export method, each data entry extracted by the container adds new data fields to the data extracted by the parent container. By default, the names of the new data fields will be the same as the names of the Capture commands extracting the data plus an index number. For example, if the Capture command name is **image**, the field names will be as follows:

image_1, image_2, image_3,

If the container extracts data such as name/value properties, then you may want the new data fields to have names that correspond to the extracted data. For example, consider the following data extracted by a container command:

Name	Value
Width	100
Height	50
Depth	25

You may want this data to generate the following data fields:

Width	Height	Depth
100	50	25

You can achieve this by setting the property **Export Rows to Columns Name Command** to the name of the command that will extract the values you want to use as field names. Also, set the property **Export Rows to Columns Value Command** to the name of the command that will extract the values you wish to use as field values.

Limitations

Using the export method **Convert Rows Into Columns** can lead to a different number of columns in the exported data table every time you run an agent. This is usually only desirable when exporting data to Excel spreadsheets, and may sometimes be useful when exporting data to CSV files.

9.4.5 Exporting Data with Scripts

With data export scripts, you can customize the export process for extraction data sets. An export script is ideal for exporting to special data structures in a database, or exporting data to a third-party or custom-data store.

An export script gives much more control than the standard export process because the script can control when to insert, update, or delete data in the export target. See the topic [Data Export Scripts](#) for more information.

9.4.6 Export From Multiple Agents

Content Grabber can export data from multiple agent into the same external database tables without having to use a custom export script.

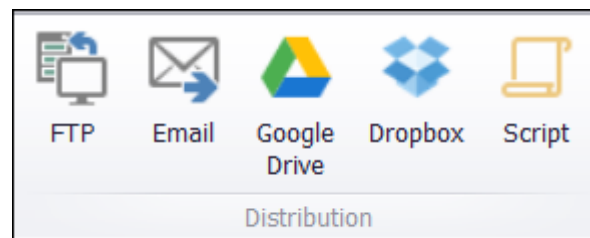
All agents must export data in the exact same format, and export names must be set for all container commands exporting to separate tables, and the export names must be the same for all agents.

The **Database Export** option **Shared Database Tables** must be set to True.

A table column **Agent ID** will be added to the external database tables, so Content Grabber can keep track of what data belongs to which agents.

9.5 Distributing Data

If the export target for an agent is set to a file format such as a spreadsheet, then you can configure an agent to send a data export as a message attachment to one or more email destinations. Alternatively, you can FTP the data to a remote server, save it to Google Drive, or have the agent use a custom script to distribute the export.



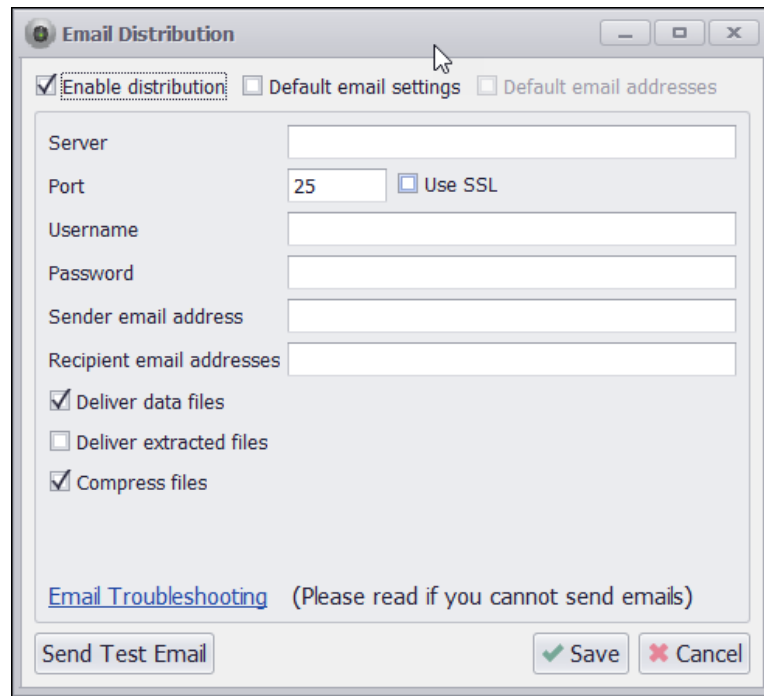
Read these topics for more information:

- Email distribution
- FTP distribution
- Google Drive distribution
- Dropbox distribution
- Scripting Data Distribution

9.5.1 Email Distribution

If the export target for an agent is set to a file format, you can configure an agent to send exported data as a message attachment to one or more email destinations.

When an agent distributes data through email, it attaches the data files to the email in a single compressed file or as individual data files. In the **Email Distribution** panel, you'll need to specify the **Server**, **Login**, and **Email Addresses**.

The image shows a Windows-style dialog box titled "Email Distribution". At the top, there are three checkboxes: "Enable distribution" (checked), "Default email settings" (unchecked), and "Default email addresses" (unchecked). Below these are several input fields: "Server", "Port" (with the value "25" entered), "Username", "Password", "Sender email address", and "Recipient email addresses". To the right of the "Port" field is a checkbox for "Use SSL". Below the input fields are three more checkboxes: "Deliver data files" (checked), "Deliver extracted files" (unchecked), and "Compress files" (checked). At the bottom left is a button labeled "Send Test Email". At the bottom right are two buttons: "Save" (with a green checkmark icon) and "Cancel" (with a red X icon). Above the "Save" and "Cancel" buttons is a link labeled "Email Troubleshooting" followed by the text "(Please read if you cannot send emails)".

Email distribution of exported data

An agent can distribute just data files, or just downloaded images and document files, or both data files and downloaded files.

Email Troubleshooting

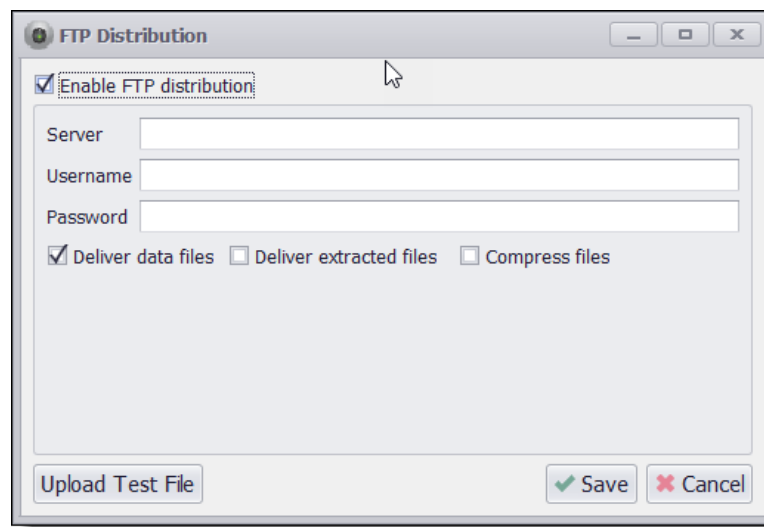
Most email servers are very restrictive in regards to who or what is allowed to send emails. If you are getting an error when sending emails from Content Grabber, then it's most likely a problem with the configuration of your account. Here is a list of things you can check:

- Make sure your email account is configured to allow you to send emails from the email address specified in **From email address**.
- Make sure your username and password are correct, and that login to your account is not restricted. Use another email client or your web browser to login to your email account to make sure you can login.
- Use SSL if required. Most online services, such as Gmail, requires SSL.
- Use the correct port. Gmail uses port 587 for example.
- Most online services, such as Gmail, don't allow just any application to login to your email account. Many online services will have a special option to allow any application to login to your account, and you must set this option to allow Content Grabber to login. In Gmail you can set this option on the following page:

<https://www.google.com/settings/security/lesssecureapps>

FTP Distribution

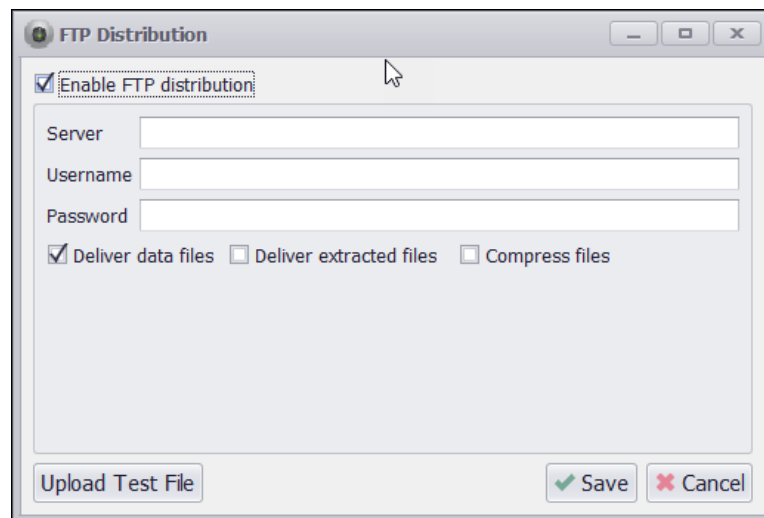
When an agent uses FTP to copy data to a remote server, the agent can copy the data files in a single compressed file or as individual data files. In the **FTP Delivery** window, you must specify the remote **Server** address, **Username**, and **Password**. The remote server must host an FTP service.



FTP distribution of exported data

Google Drive Distribution

When an agent uses FTP to copy data to a remote server, the agent can copy the data files in a single compressed file or as individual data files. In the **FTP Delivery** window, you must specify the remote **Server** address, **Username**, and **Password**. The remote server must host an FTP service.

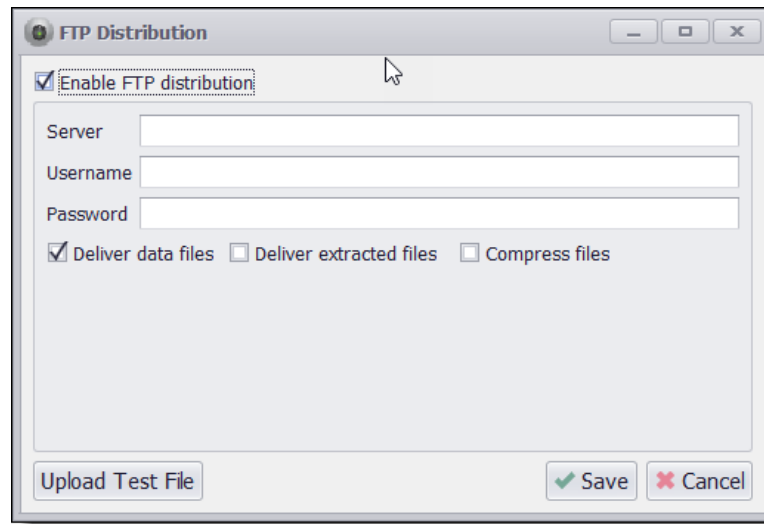


FTP distribution of exported data

9.5.2 FTP Distribution

If the export target for an agent is set to a file format, you can configure an agent to FTP the data to a remote server.

When an agent uses FTP to copy data to a remote server, the agent can copy the data files in a single compressed file or as individual data files. In the **FTP Distribution** window, you must specify the remote **Server** address, **Username**, and **Password**. The remote server must host an FTP service.



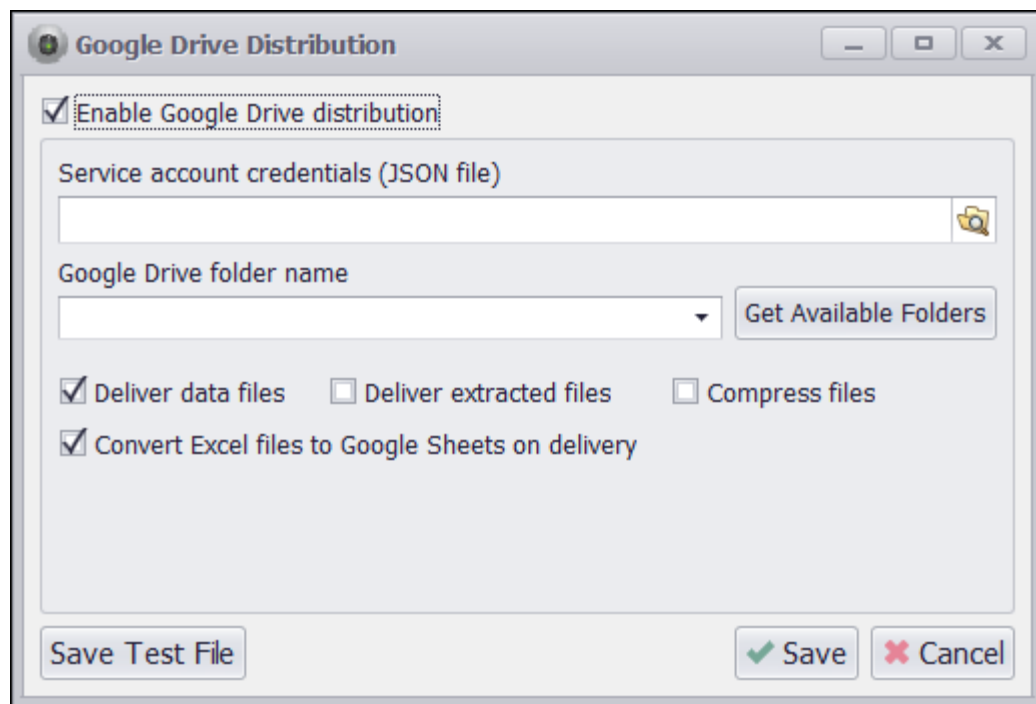
FTP distribution of exported data

An agent can distribute just data files, or just downloaded images and document files, or both data files and downloaded files.

9.5.3 Google Drive Distribution

If the export target for an agent is set to a file format, you can configure an agent to save the data to Google Drive.

When an agent saves data to Google Drive, the agent can save the data files in a single compressed file or as individual data files. In the **Google Drive Distribution** panel, you must specify credentials for a Google Drive service account, and a Google Drive folder that is shared with the service account.



Google Drive distribution of exported data

If an agent is exporting data to Excel, the Excel files can automatically be converted to Google Sheets when saved on Google Drive.

An agent can distribute just data files, or just downloaded images and document files, or both data files and downloaded files.

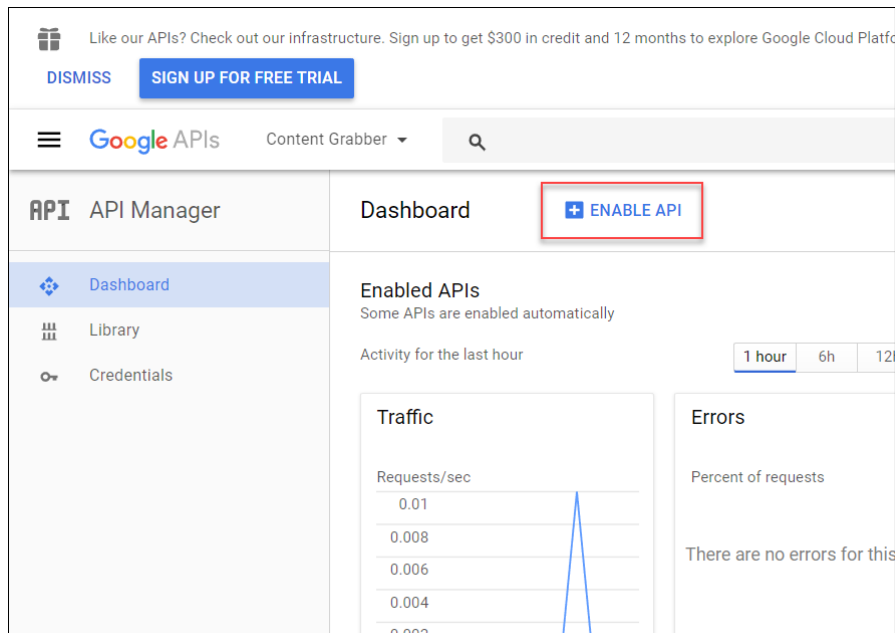
Configuring a Google Drive Service Account

A Google Drive service account is a special account that can be used by applications such as Content Grabber to access Google Drive through the Google Drive API without requiring manual authentication by the account holder.

Creating a Google service account is not as easy as one would have hoped, so here is a detailed step-by-step guide.

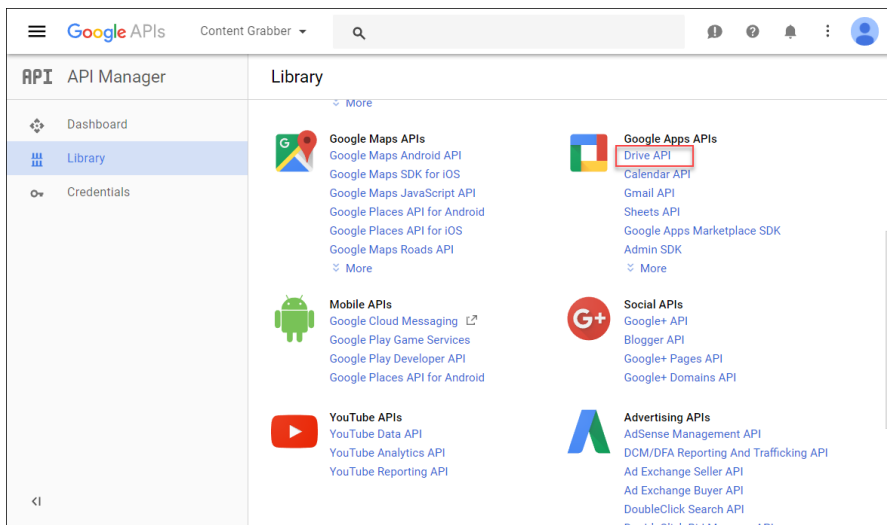
1) Open the Google developers console page.

<https://console.developers.google.com>



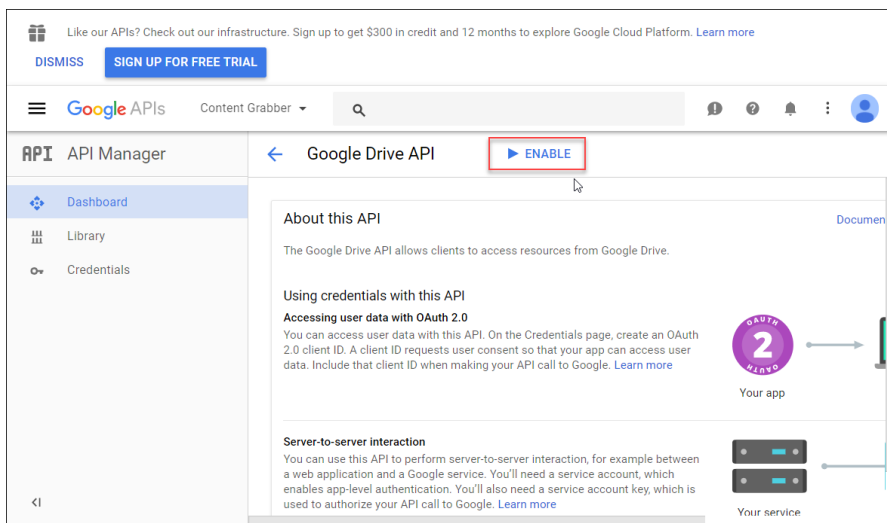
Enable the API on the Google developers console page.

2) If you have already enabled the Google Drive API, then you can go directly to step 4. Otherwise click **Enable API** and select Drive API from the list of available APIs.



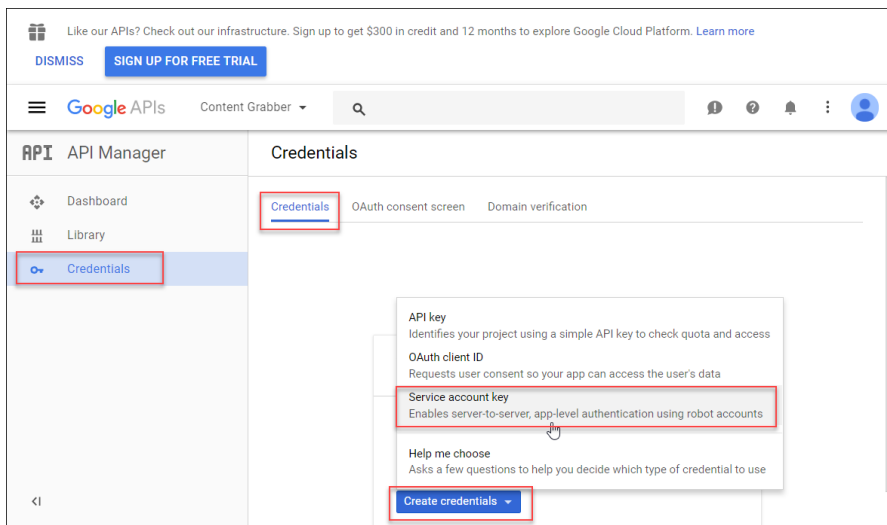
Locate the Drive API from the list of available APIs.

3) Click **Enable** to enable the API.



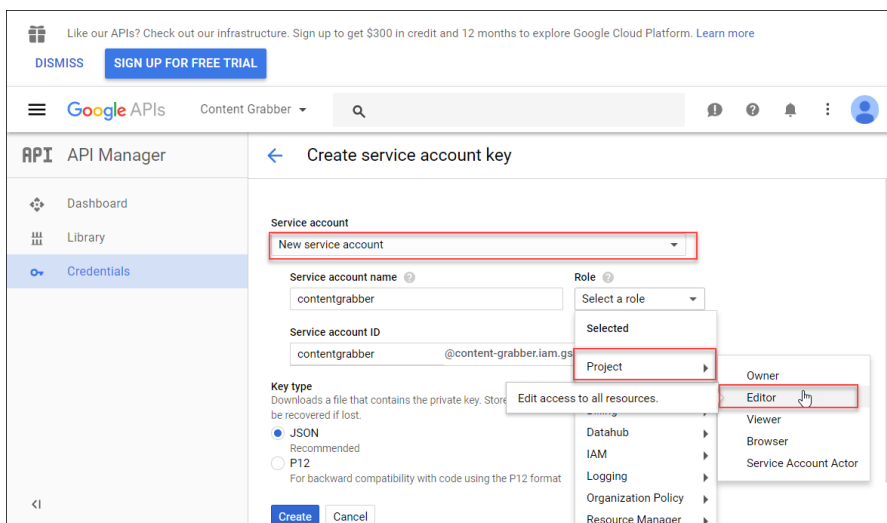
Enable the Drive API.

4) Click **Credentials** from the left hand side menu. Then click the button **Create credentials** and select **Service account key** from the drop down box.



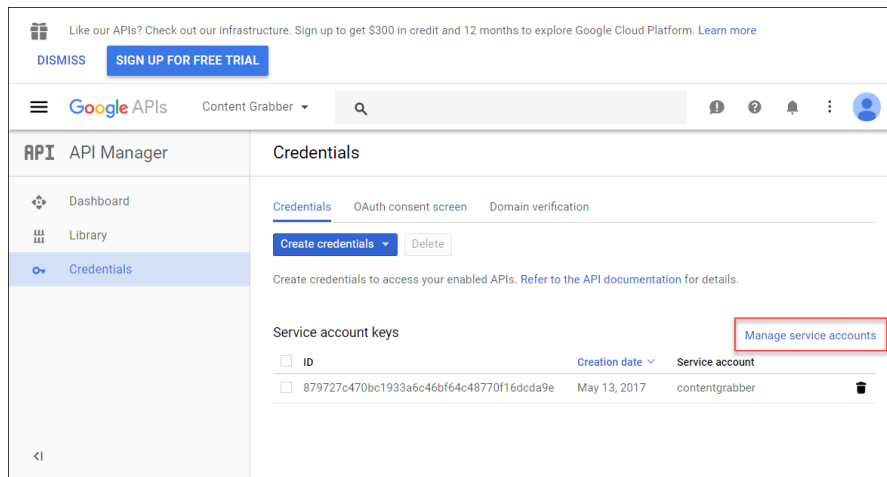
Create a service account.

5) Select **New service account** from the **service account** drop down box. Then enter any service account name, and select **Project - Editor** from the **Role** drop down box. Once you click the **Create** button, a JSON file will be downloaded to your computer. Content Grabber will use this file to gain access to Google Drive, so keep the file in a safe location.



Add the Project Editor role to the service account.

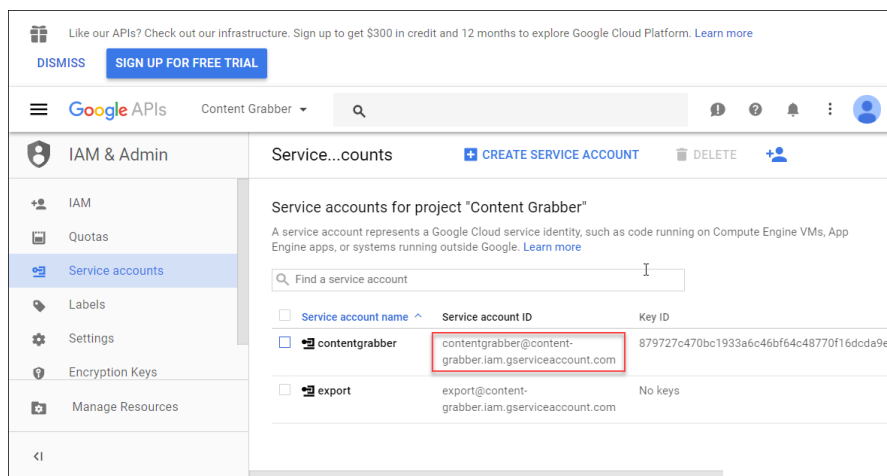
6) We now need to get the service account ID, so follow the link **Manage service accounts**.



The new service account is listed under Credentials.

7) Copy the service account ID to your clipboard (or write it down somewhere). The ID will depend on your account name, but will look something like this:

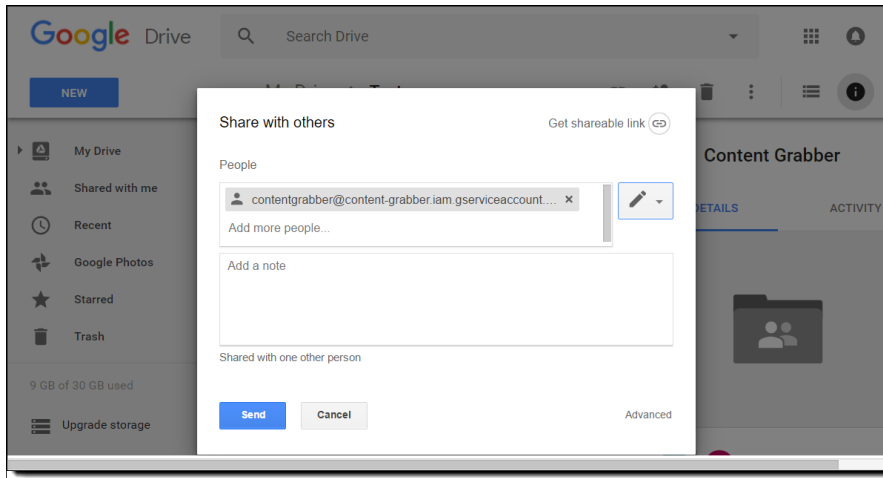
contentgrabber@content-grabber.iam.gserviceaccount.com



Copy the service account ID from the Manage service account screen.

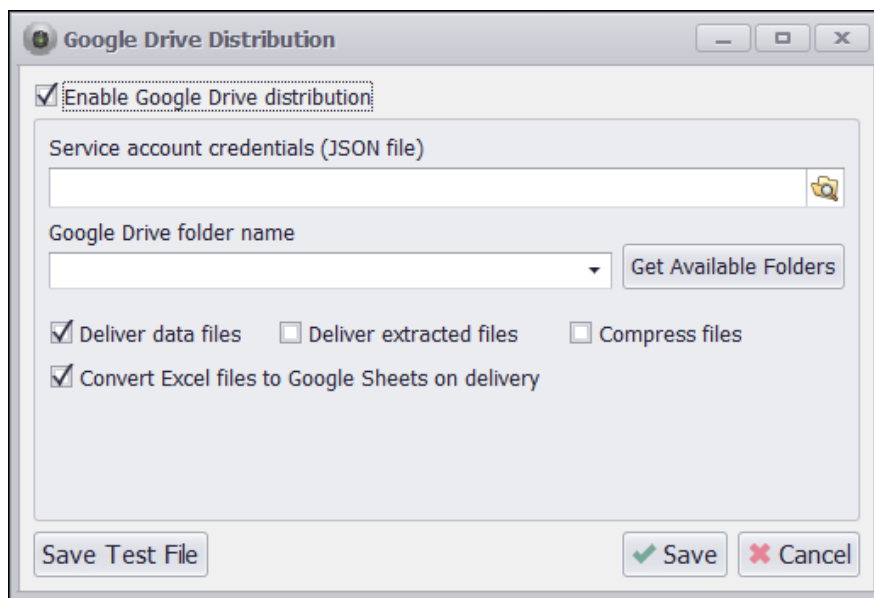
8) Open your Google Drive (<https://drive.google.com>) and create a new folder (or use an existing folder if you like). Right click on the folder and select **Share...** Now enter the service account ID and click **Send**. Your folder is now shared with your

service account, and the service account can save files to that folder.



Share a folder with the new service account.

9) Open an agent in the Content Grabber editor or create a new agent. Click **Google Drive** from the **Data** menu to open the Google Drive configuration screen. Select the service account credentials file that was downloaded to your computer in step 5, and then click **Get Available Folder** to get a list of all folders that are shared with your service account. You can click Upload Test File to make sure everything is configured correctly.

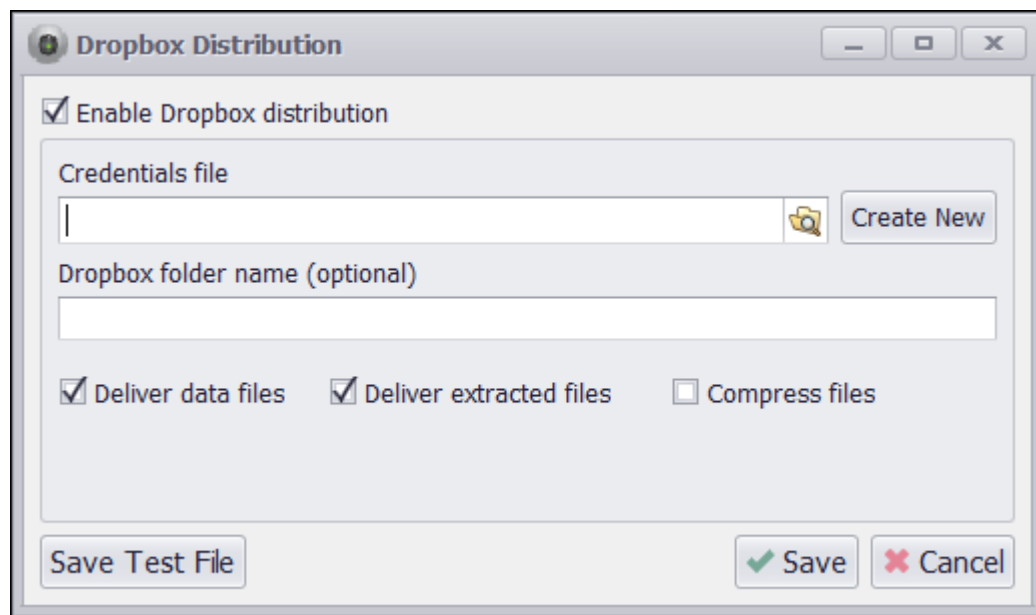


Configure an agent to save extracted data on Google Drive.

9.5.4 Dropbox Distribution

If the export target for an agent is set to a file format, you can configure an agent to save the data to Dropbox.

When an agent saves data to Dropbox, the agent can save the data files in a single compressed file or as individual data files. In the **Dropbox Distribution** panel, you must specify credentials for a Dropbox service account.



Dropbox distribution of exported data

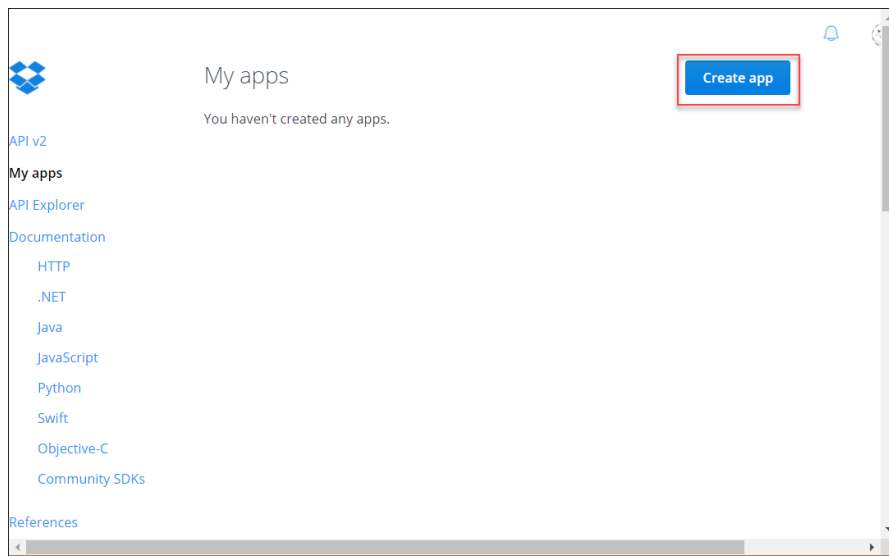
An agent can distribute just data files, or just downloaded images and document files, or both data files and downloaded files.

Configuring a Dropbox Account to Work with Content Grabber

You must create a new Dropbox app for Content Grabber, so Content grabber can access your Dropbox account.

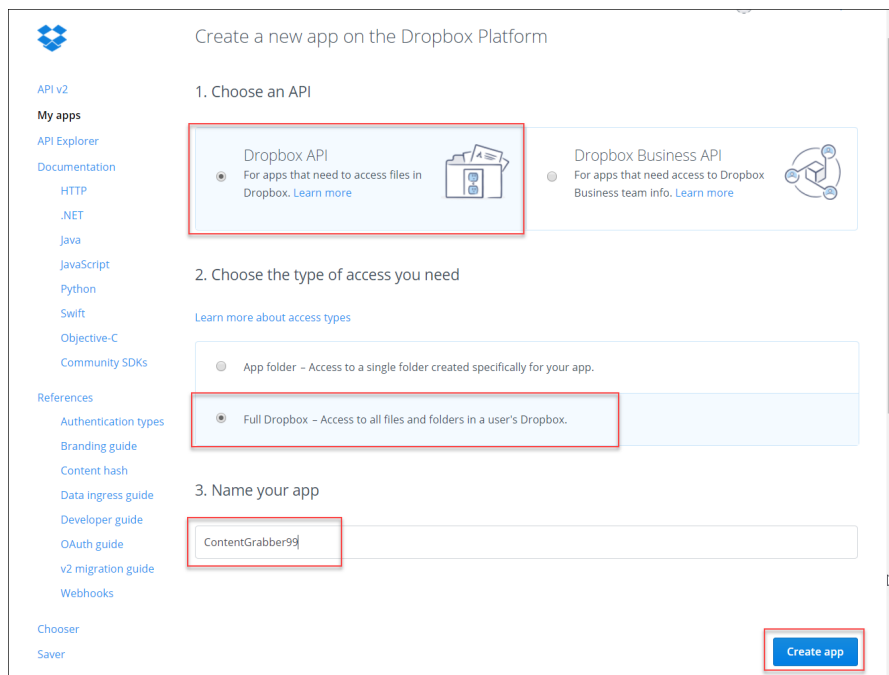
1) Open the **My apps** page on the Dropbox website.

<https://www.dropbox.com/developers/apps>



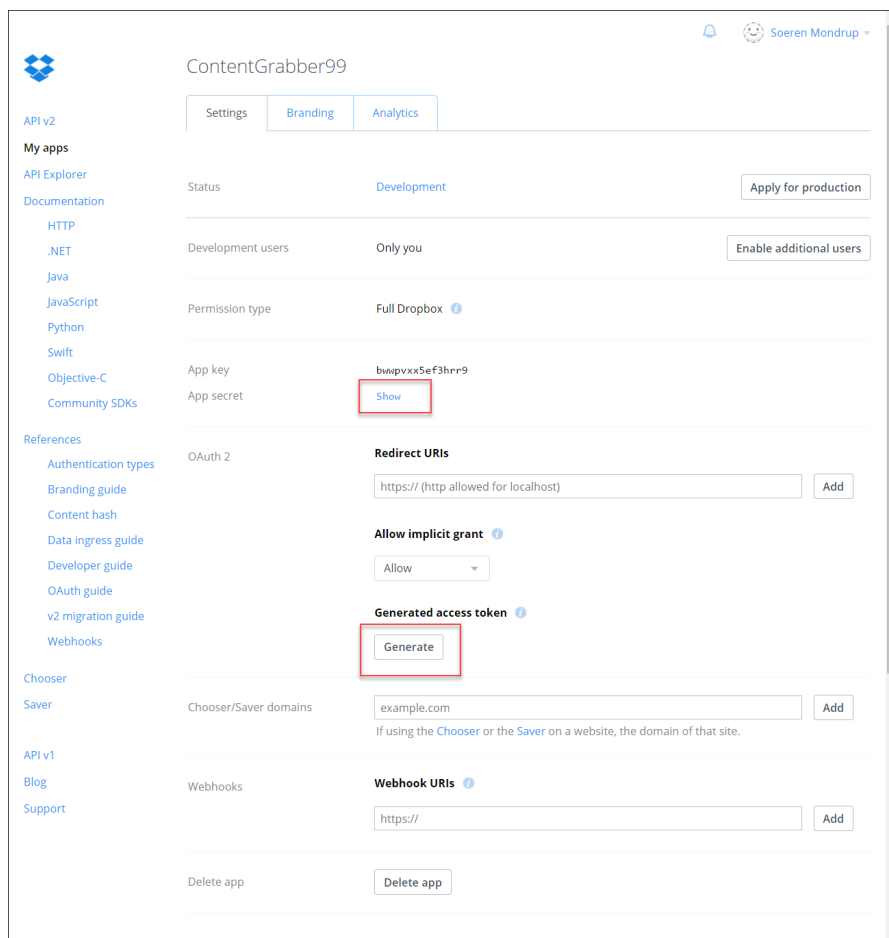
The My apps page on the Dropbox website.

2) Click the **Create App** button to create a new app. Choose **Dropbox API**, then **Full Dropbox**, and select a name for your app. The app name must be unique, so just add a random number after your app name if you get an error telling you the app name is already in use.



Create a new Dropbox app for Content Grabber.

3) Once you have created your app, a page will show you details about your app. Click **Show** under **App secret** to show your app secret, and then click **Generate** under **Generate access token** to generate a new access token.



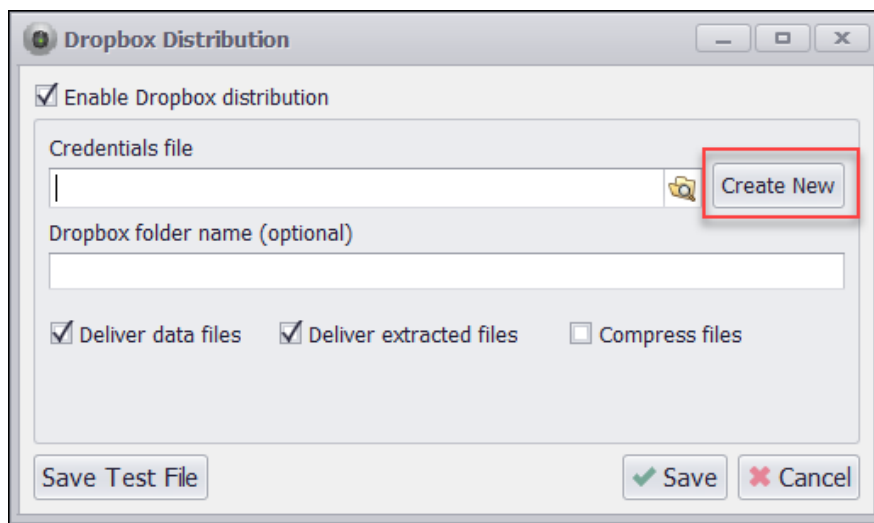
Generate a new access token.

4) Copy the Access token to your clipboard, or just write it down somewhere. You will need this information when you connect Content Grabber to your Dropbox account.

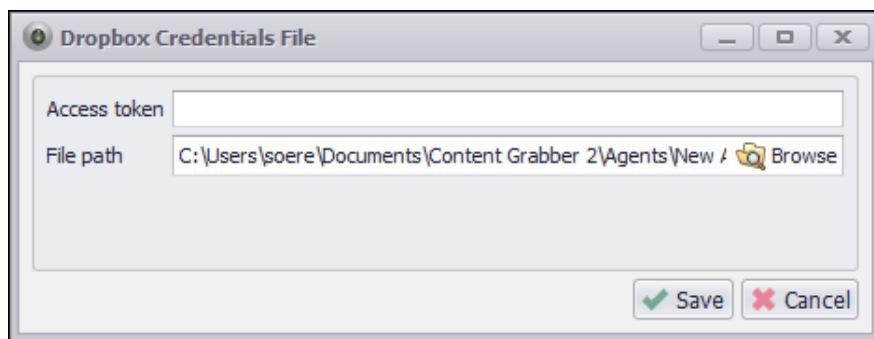
The screenshot shows the 'ContentGrabber99' interface with the 'Branding' tab selected. The left sidebar contains links for 'API v2', 'My apps', 'API Explorer', 'Documentation', 'HTTP', '.NET', 'Java', 'JavaScript', 'Python', 'Swift', 'Objective-C', 'Community SDKs', 'References', 'Authentication types', 'Branding guide', 'Content hash', 'Data ingress guide', 'Developer guide', 'OAuth guide', 'v2 migration guide', 'Webhooks', 'Chooser', 'Saver', 'API v1', 'Blog', and 'Support'. The main content area shows the 'Branding' settings for an application. The 'Status' is 'Development' with an 'Apply for production' button. 'Development users' are listed as 'Only you' with an 'Enable additional users' button. The 'Permission type' is 'Full Dropbox'. The 'App key' is 'bwwpxx5ef3hrr9' and the 'App secret' is 'Show'. Under 'OAuth 2', the 'Redirect URIs' is 'https:// (http allowed for localhost)' with an 'Add' button. The 'Allow implicit grant' is set to 'Allow'. The 'Generated access token' is highlighted with a red box and contains the value 'jbbYpxQ4VSQAAAAAIF1DrArImyZdQYM8chwnmKXBGDm1m7aVsRvxABQK74m5R'. Below the token, a note states: 'This access token can be used to access your account (smondруп@sequentum.com) via the API. Don't share your access token with anyone.' Under 'Chooser/Saver domains', the domain 'example.com' is listed with an 'Add' button. The 'Webhook URIs' is 'https://' with an 'Add' button.

Copy the App key, App secret and Access token.

5) Open the Dropbox distribution panel in the Content Grabber editor and press the **Create New** button to create a new credentials file. You can use the same credentials file for all your Content Grabber agents.

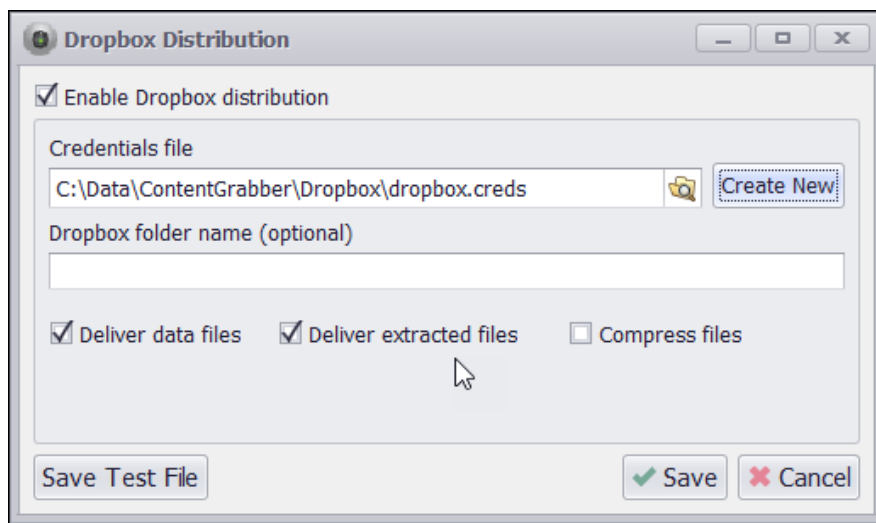


Create a new credentials file.



Enter the Access token to create a new credentials file.

6) Enter the Access token you got from step 4, and choose the file path for the credentials file.



Press Save Test File to check configuration.

7) You can now press the **Save Test File** to test your configuration.

9.5.5 Scripting Data Distribution

An agent can use a custom script to distribute data export files. A data distribution script doesn't need to actually distribute the data, but can be used to perform custom processing tasks after data export is complete.

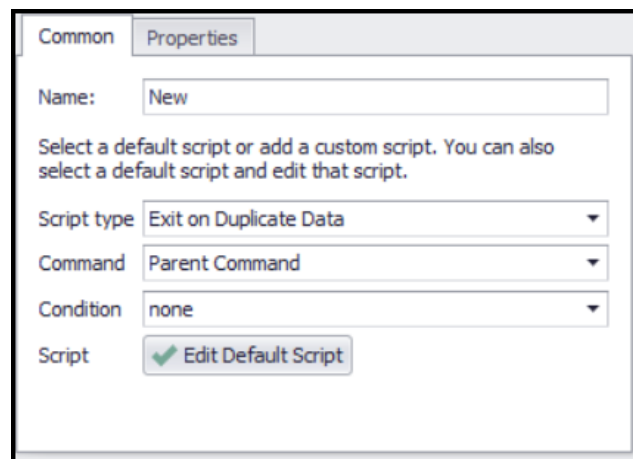
Please see the topic Data Distribution Scripts for more information.

9.6 Removing Duplicate Data

Duplicate data checks can be used to remove duplicate data. There are many ways to incorporate duplicate checks into an agent. The agent could check an external database to see if extracted data already exists, or it could check its own internal database. If an agent is configured to keep previously extracted

data, it could check that data to see if the extracted data already exists there.

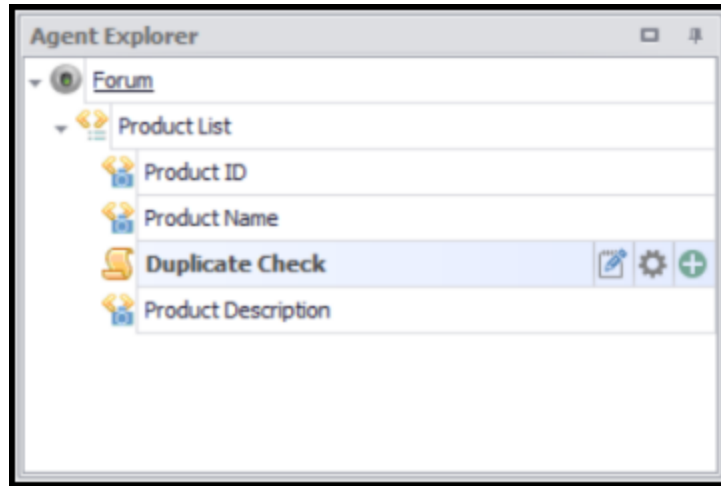
Use a Remove Duplicate command for simple removal of duplicates. For more complex ways of checking for duplicates, the task is best performed by a script. Content Grabber contains a default script that can be used for basic duplicate checks. To use the default duplicate script, add an **Execute Script** command and set the script type to **Exit on Duplicate Data**. This script tries to match data extracted in the current container command with existing data in the internal database. If it finds a match, it deletes the data it has extracted from the current container command and exits the container command.



Default duplicate script.

The position of the duplicate script is important. The script will only try and match data that has already been extracted in the current container command, so data extracted by capture commands that are positioned after the script command will not be used. For example, if a container command has three capture commands that extract Product ID, Product Name and Product Description, and the duplicate check should be done on the Product ID only, then the duplicate script should be

positioned after the capture command that extracts the Product ID, but before the capture commands that extract Product Name and Product Description. If the duplicate check should be done on all the data extracted in the container command, then the duplicate script should be positioned last in the container.



Duplicate check on Product ID and Product Name.

The default duplicate script tries to match extracted data with all existing data in the internal database. If the agent is configured to keep previously extracted data in the internal database, the duplicate check will be done on all data extracted in the current agent run and all data extracted in previous agent runs. If the agent is configured to delete old data, the duplicate check will only be done on data extracted in the current agent run.

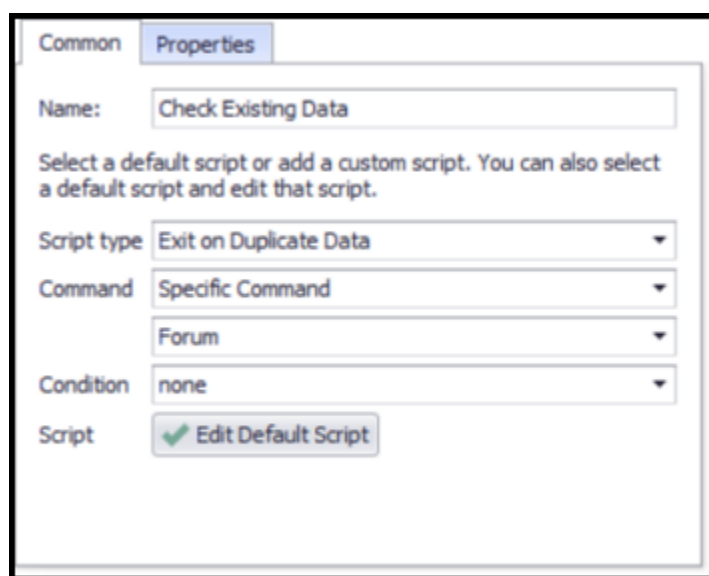
9.7 Extracting New Data Only

When extracting data from some websites, such as large forums, it's sometimes desirable to extract all data only once, and after that just the new data added to the website since last time the agent ran. This can be done by adding an **Execute**

Script command and setting the script type to **Exit on Duplicate Data**. The duplicate script will try and match extracted data with data that already exists in the internal database. If the extracted data already exists in the internal database, the agent knows there's no more new data available and exits.

It's important that data on the target website is sorted in a predictable way. For example, when extracting data from a forum, the forum posts are normally sorted by date, so when the agent reaches a post that was already extracted last time the agent ran, it knows that all other older posts would also have been previously extracted, so the agent can exit.

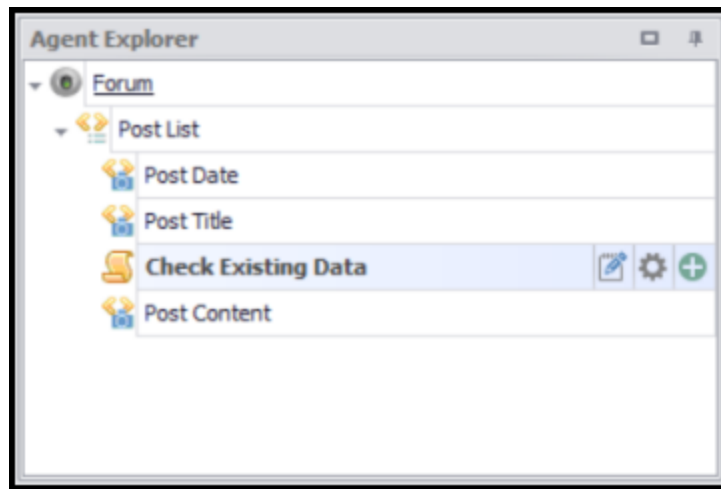
If the target website is not ordered in a predictable way, it not possible to extract only new data in this way, since the agent would have to go through all data on the website to make sure it's not missing any new data.



The screenshot shows a configuration window with two tabs: 'Common' and 'Properties'. The 'Properties' tab is active. The 'Name' field contains 'Check Existing Data'. Below it, a text label reads: 'Select a default script or add a custom script. You can also select a default script and edit that script.' The 'Script type' dropdown menu is set to 'Exit on Duplicate Data'. The 'Command' dropdown menu is set to 'Specific Command'. Below the 'Command' dropdown is a text field containing 'Forum'. The 'Condition' dropdown menu is set to 'none'. At the bottom, the 'Script' field contains a green checkmark icon and the text 'Edit Default Script'.

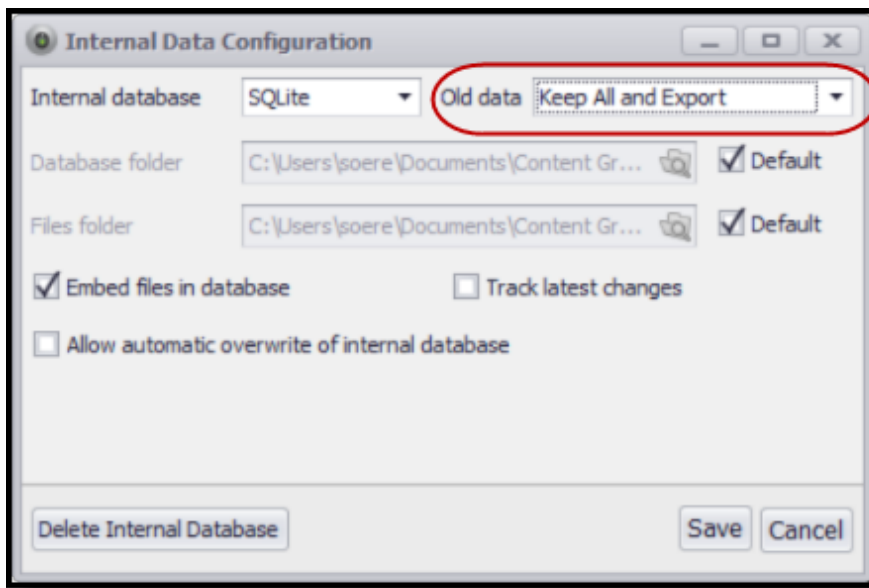
Default duplicate script configured to exit agent when match is found.

The position of the duplicate script is important. The script will only try and match data that has already been extracted in the current container command, so data extracted by capture commands that are positioned after the script command will not be used. For example, if a container command has three capture commands that extract Post Date, Post Title and Post Content, and the match check should be done on Product Date and Post Title only, then the duplicate script should be positioned after the capture commands that extract Post Date and Post Title, but before the capture command that extracts Post Content. If the match check should be done on all the data extracted in the container command, then the duplicate script should be positioned last in the container.



Check for match on Post Date and Post Title.

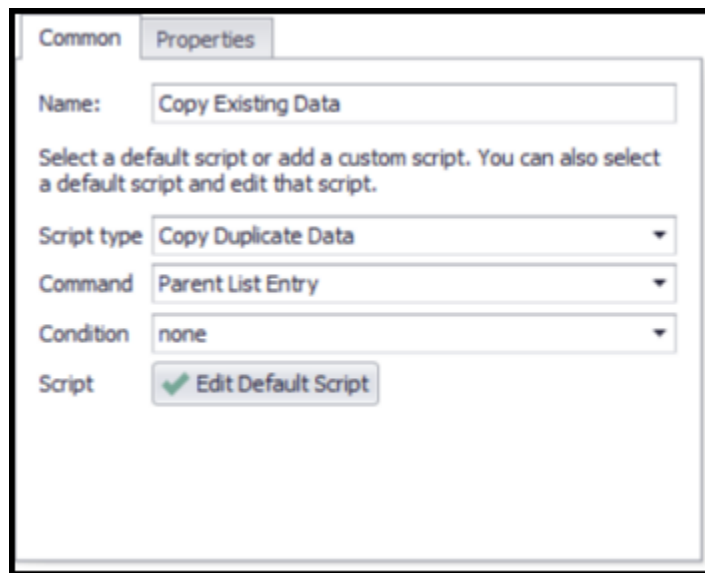
Since the agent will extract only new data, it's important the internal database is configured to keep all previously extracted data.



Internal database configured to keep all previously extracted data.

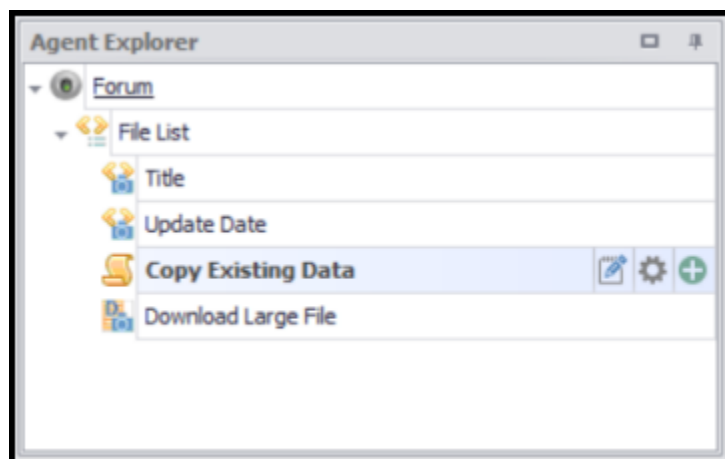
9.8 Reusing Existing Data

Reusing previously extracted data can speed up some agents dramatically. For example, if an agent downloads large files, it may be able to determine that some files have not changed, and then copy the files from the existing data rather than downloading the files again. This can be done by adding an **Execute Script** command and setting the script type to **Copy Duplicate Data**. The duplicate script will try and match extracted data with data that already exists in the internal database. If the extracted data already exists in the internal database, the existing data will be copied to the current data set and the agent will exit the container command.



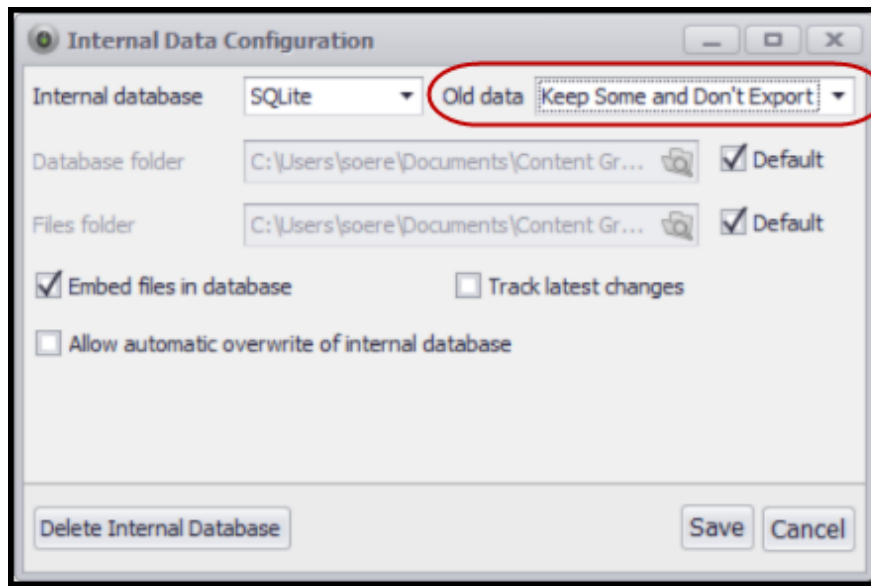
Execute Script command with script type set to Copy Duplicate Data.

The position of the duplicate script is important. The script will only try and match data that has already been extracted in the current container command, so data extracted by capture commands that are positioned after the script command will not be used. For example, if a container command has three capture commands that extract Title, Update Date and Download Large File, and the match check should be done on Title and Update Date only, then the duplicate script should be positioned after the capture commands that extract title and update date, but before the capture command that extracts the large file. Since the duplicate script exists the container command if the title and update date already exists, the Download Large File command will not be processed in that case.



Copy file if title and update date has not changed.

Since the default duplicate script will copy existing data if it has not changed, it's important the internal database is configured to keep previously extracted data. It's not necessary to keep more than one old data set, so the **Old data** option can be set to **Keep Some and Don't Export** which will only keep data from the last successful agent run, and it will only export the currently extracted data plus the old data the duplicate script copied into the current data set.



Internal database configured to keep some old data.

9.9 Change Tracking

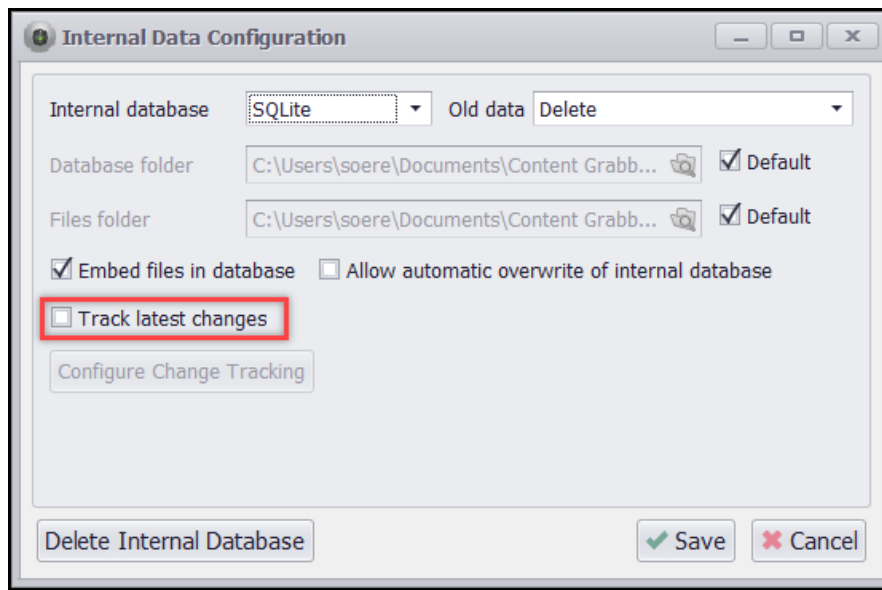
An agent can keep track of the latest changes that have been made to extracted data.

The agent will mark extracted data as deleted, modified or added. If data is deleted but later returned, the data will be marked as "returned", or "returned modified" if the data returned in a modified state.

An agent can be configured to only export data that has changed since last successful run, or only export data that has changed since a specified number of days.

Data will be marked as deleted if it was extracted last time the agent ran, but not during the current run. Data will only be marked as deleted if an agent completes successfully. This prevents data from being incorrectly marked as deleted if an agent fails halfway through a run. The Success Criteria options are used to define when an agent run should be considered successful.

Default change tracking can be enabled on the **Internal Database** window which is available from the **Run** menu in Content Grabber.



Internal database window.

Change tracking can be fine-tuned by setting the advanced change tracking options. These options can be set by clicking the **Configure Change Tracking** button.

The screenshot shows a 'Change Tracking' dialog box with a list of settings. The 'Enabled' option is set to 'True'. Other settings include 'Export Method' (Export All), 'Export Method Days' (7), 'Historical Data Export Method' (Changed Data Only), 'Track Deletes' (True), 'Last Change Enabled' (True), 'Last Change Column Name' (Last Change), 'Change Date Enabled' (True), 'Change Date Column Name' (Last Change Date), 'Insert Date Enabled' (False), 'Insert Date Column Name' (Inserted Date), 'Update Date Enabled' (False), 'Update Date Column Name' (Updated Date), 'Identifier Enabled' (False), 'Identifier Column Name' (Identifier), 'Columns Affected Enabled' (True), 'Columns Affected Column Name' (Columns Affected), 'Changed Last Run Enabled' (False), and 'Changed Last Run Column Name' (Changed Last Run). At the bottom, there is a 'Save' button and a 'Cancel' button.

Option	Value
Enabled	True
Export Method	Export All
Export Method Days	7
Historical Data Export Method	Changed Data Only
Track Deletes	True
Last Change Enabled	True
Last Change Column Name	Last Change
Change Date Enabled	True
Change Date Column Name	Last Change Date
Insert Date Enabled	False
Insert Date Column Name	Inserted Date
Update Date Enabled	False
Update Date Column Name	Updated Date
Identifier Enabled	False
Identifier Column Name	Identifier
Columns Affected Enabled	True
Columns Affected Column Name	Columns Affected
Changed Last Run Enabled	False
Changed Last Run Column Name	Changed Last Run

Enabled
Tracks changes made to exported data.

Save Cancel

Advanced change tracking options.

The following advanced change tracking options are available.

Option	Description
Enabled	Enables or disables change tracking.
Export Method	Specifies how to export data when an agent is not exporting historical data to a database. The option Historical Data Export Method is used instead when exporting historical data to a

Option	Description
	<p>database. The following options are available:</p> <p>Export All. Exports all data no matter if the data has changed or not.</p> <p>Since Last Successful Run. Exports all data that has changed since last successful run.</p> <p>Since Number of Days. Exports all data that has changed since a specified number of days.</p>
Export Method Days	When Export Method is set to Since Number of Days , only data that has changed since the specified time period is exported.
Historical Data Export Method	<p>This option is used instead of Export Method when exporting historical data to a database. The following options are available:</p> <p>All Data. Exports all data no matter if the data has changed or not.</p> <p>Changed Data Only. Exports all data that has changed since last agent run.</p>
Track Deletes	Specifies if an agent should track deleted data.

Option	Description
	<p>If an agent does not track deleted data, the last change status will not change for data that was not found in the last successful agent run.</p> <p>If an agent is tracking deleted data, the last change status will be set to Deleted for data that was not found in the last successful agent run.</p>
Last Change Enabled	Exports the type of change that was last made to a data row.
Last Change Column Name	The name of the data column where the type of change is stored.
Change Date Enabled	Exports the date a data row was last changed.
Change Date Column Name	The name of the data column where the change date is stored.
Insert Date Enabled	Exports the date a data row was first inserted. This is the date the data was first extracted.
Insert Date Column Name	The name of the data column where the insert date is stored.
Update Date Enabled	Exports the date a data row was last processed. This is the date the data was last extracted and compared to existing data. Notice that data may not have changed at this date.

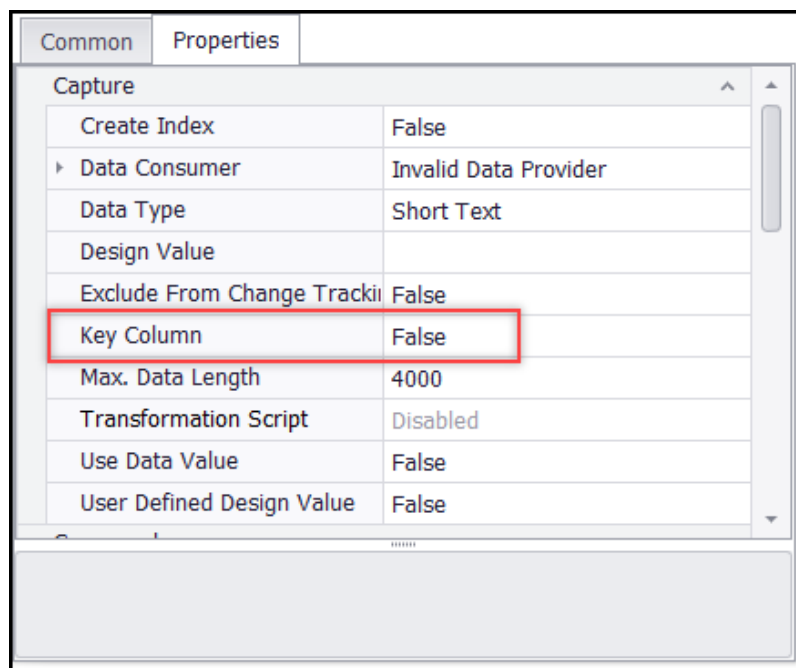
Option	Description
Update Date Column Name	The name of the data column where the update date is stored.
Identifier Enabled	Exports the object identifier used in the internal database. This value uniquely identifies the data row and will not change unless the internal database is recreated.
Identifier Column Name	The name of the data column where the object identifier is stored.
Columns Affected Enabled	Exports a column that contains the names of columns affected by a change.
Columns Affected Column Name	The name of the data column where the columns affected value is stored.
Changed Last Run Enabled	Exports a value indicating if a data row changed last time the agent was run.
Changed Last Run Column Name	The name of the data column to store the value indicating if a data row changed last time an agent was run.

Key Columns

When an agent exports data, it compares extracted data with existing data to see if the data has been added, changed or deleted. In order to see if a data entry has changed, the agent needs to be able to uniquely identify a data entry. By default, an agent uses all captured data to identify a data entry, but this

means that every time any data changes, a data entry is always identified as a new data entry because it never matches any existing data entry.

To get change tracking working properly, so the agent correctly identifies modified data entries, it's important to mark capture commands that extracts data that uniquely identifies a data entry. For example, when extracting product data, a website may display a product ID that a capture command can extract and the agent use as a unique identifier. Set the command option **Key Column** to mark a capture command as a command that extracts data that can be used to uniquely identify a data entry.



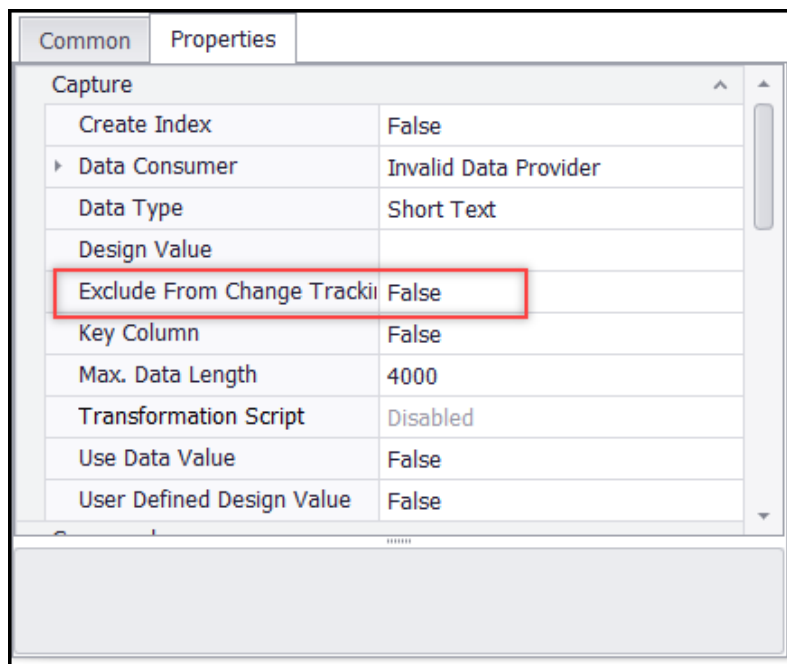
Key Column command option.

Multiple capture commands can be marked as key columns to combine extracted data from multiple commands into a value that uniquely identifies a data entry.

Each container command that is configured to generate a separate data table should have one or more capture commands that are marked as key columns.

Exclude From Change Tracking

Capture commands can be excluded from change tracking, so if the captured data changes, it will not cause the last change status for the data entry to change.



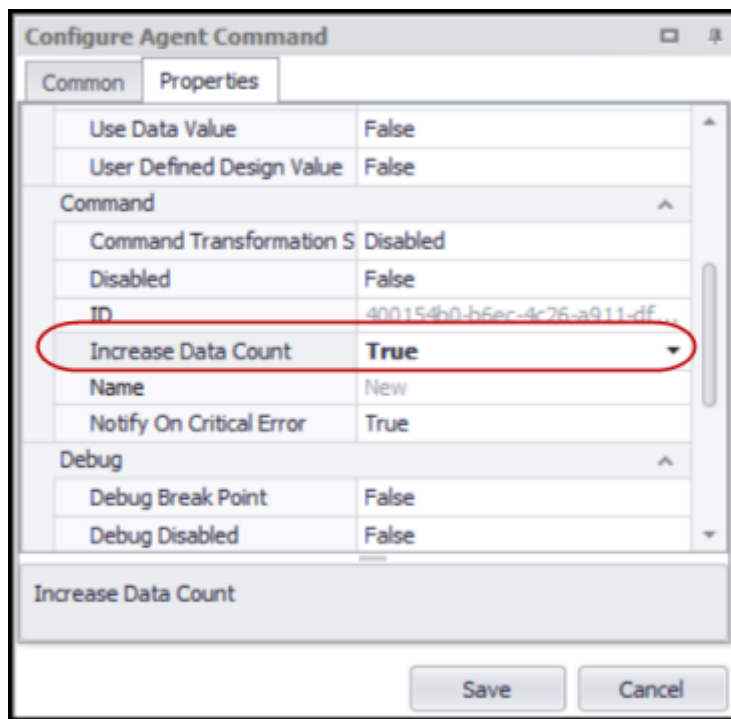
Exclude captured data from change tracking.

9.10 Data Counting

Content Grabber can count extracted data in two different ways. It can count the number of data rows exported to a specified data table, or the agent can be configured to count data in a custom defined way. The two data counts are named **Export Row Count** and **Data Count** respectively.

Content Grabber can extract any kind of data from a website, so it's impossible for the software to know exactly when it has

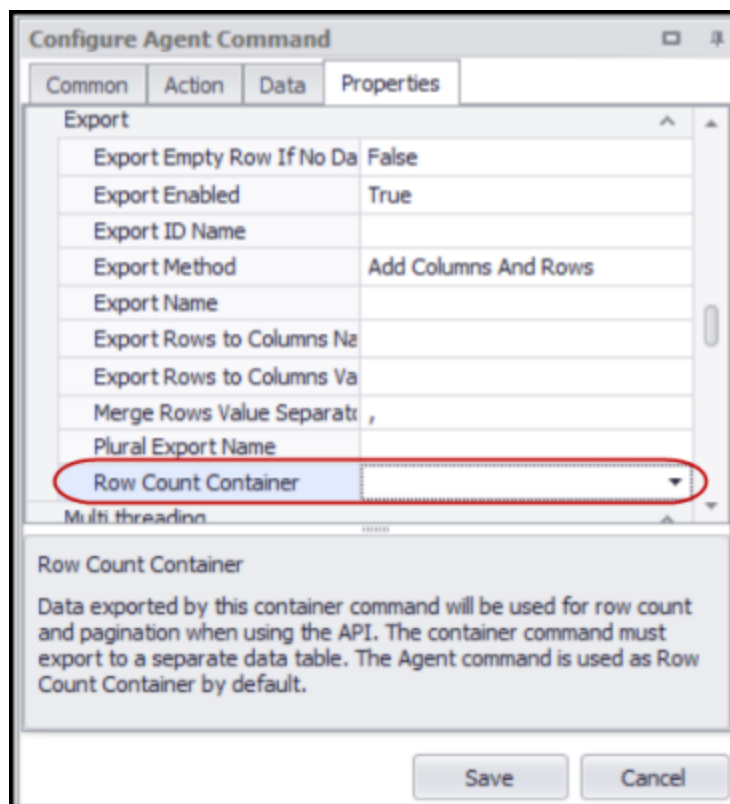
extracted enough data to consider it a logical data entry. For example, you may extract product data that consist of a product category, product details and product images, but the software doesn't know exactly what needs to be extracted before it can consider a product entry completely extracted. To help the software know when it should increase the data count, you can set the Command option **Increase Data Count**. Every time the agent encounters a command with the option **Increase Data Count**, it will increase the data count. You can also increase the data count in a script. If you don't use the **Increase Data Count** option and don't use a script to increase data count, then **Data Count** will always be zero.



The Increase Data Count command option.

The **Export Row Count** is the number of rows in a specified export data table. If no data table has been specified, the number of rows in the primary data table will be used. The primary data table is the export table generated by the Agent

command. You can use the Agent option **Row Count Container** to specify the container command that should be used. The **Row Count Container** command must be a container command that exports to a separate data table.



The Row Count Container agent option.

The **Export Row Count** value is calculated after data has been exported, so it's not available during data extraction and it cannot be used as a success criteria.

The **Data Count** value is increased during data extraction, so it can be used to measure agent progress, and it can also be used as a success criteria.

10 Anonymous Web Scraping

Many users prefer to browse the web in private. Some website owners do not want web scraping tools to extract data from their websites, and their administrators may attempt to block access to their website for any non-human users. Content Grabber will behave exactly like a normal Chrome user when your agent uses a **Web Browser**.

When you visit a website with a web browser or a web scraping tool such as Content Grabber, the website owner can record your IP address and attempt to identify you with the intent to block your access to the website.

If you do not want a website owner to be able to identify you when you visit a website, you can use a proxy server to hide your IP address. The proxy server will visit the website for you. There are many different types of proxy servers, but Content Grabber supports only HTTP proxy servers. It does not support other types of proxy servers, such as SOCKS proxies.

See the topic CAPTCHA & IP Blocking for more information about configuring proxy servers in Content Grabber.

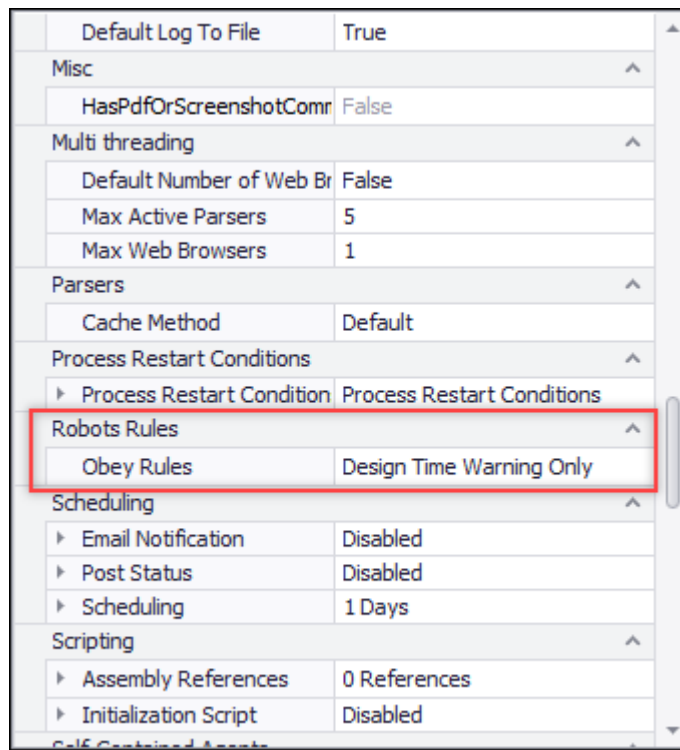
11 Obeying Robots Rules

Some websites use a robots.txt file to tell robots how they are allowed to navigate a website. A robots.txt file contains a list of User Agents and associated paths the User Agents are allowed, or disallowed, to follow. If you decide to obey robots.txt rules, you should configure your agent to use a custom User Agent that describes your business, so website publishers can decide to allow or disallow your agents to follow certain paths on their websites.

You can configure an agent to always obey Robots rules and block URLs that a robots.txt file disallow, or a page warning can be generated during design time when navigating to a URL that is disallowed. You can also choose to never obey Robots rules.

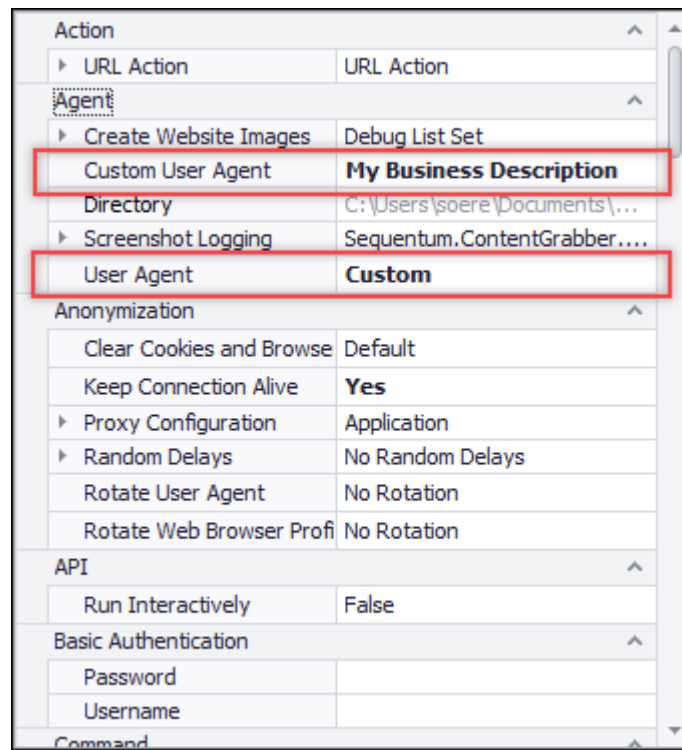
If an agent is using a Dynamic Browser, Content Grabber will only check Robots rules on URLs loaded in the main browser frame. URLs loaded in sub-frames and IFrames will not be validated.

Robots rules can be configured on the Agent Properties panel.



Agent Properties panel.

A custom User Agent can also be configured on the Agent Properties panel.



12 CAPTCHA & IP Blocking

Some website owners do not want web-scraping agents to extract data from their websites. They may try to block access to their website if they believe the user is not human.

CAPTCHA and IP blocking are the two most common ways of blocking web scraping agents. You can configure a Content Grabber agent to manually or automatically bypass CAPTCHA, and also configure an agent to use proxy servers - which prevents IP blocking.

Read these topics for more information:

- CAPTCHA Blocking
- IP Blocking & Proxy Server

12.1 CAPTCHA Blocking

A website can implement CAPTCHA blocking by using a web form that the user must submit to gain access to any restricted areas of the site. The web form is usually quite simple, consisting of an image element and a text box element. The image displays some characters which the user must enter into the text box in the exact sequence as given in the image. A human user can read the text in the CAPTCHA image, but a web-scraping agent requires special character recognition software to successfully discern the characters in the image.

A screenshot of a web registration form titled "New registration:". It contains three input fields: "email address" with a red asterisk and a note "password will be sent to this address", "login:", and "enter text from the captcha:". The captcha image shows the word "Ship" in a stylized, green, wavy font. Below the captcha is a text box for the user's input. At the bottom is a "register" button.

A typical registration form with CAPTCHA blocking

Content Grabber performs both manual and automatic data extraction from websites that implement CAPTCHA blocking. Automatic data extraction requires an account with a third-party CAPTCHA recognition service and, typically, there is a small fee for processing each CAPTCHA image. Manual data extraction is free, but requires you to manually decode CAPTCHA images while running a data extraction agent.

Manual CAPTCHA Configuration

You have two options when configuring manual CAPTCHA. The easiest one to configure is the option we describe below. The other option uses the same approach as automatic CAPTCHA configuration, but instead of using a script to resolve the CAPTCHA, a window is displayed allowing the user to manually resolve the CAPTCHA.

Manual CAPTCHA processing is easy to configure, but requires you to manually decode CAPTCHA images while the agent is running. The agent will pause and display the browser window where a user can view the CAPTCHA image and enter the CAPTCHA text in the text box. You can configure the agent to

automatically submit the web form after the user has entered the CAPTCHA text, or you can reply on the user to submit the form while the agent is paused.

If CAPTCHA blocking is part of a larger registration form, you can process the CAPTCHA part manually and let the agent process the rest of the form automatically. In this case you should let the agent submit the form automatically rather than relying on the user to submit the form while the agent is paused.

Follow these steps to pause an agent when a CAPTCHA image is displayed:

1. Add an Execute Script command to your agent.
2. Select the CAPTCHA image element in the web browser.
This sets the command's web selection.
3. Select the default script type Pause Agent.
4. Select the default script condition If Selection Exists.
5. Save the command.

This command will pause the agent when the CAPTCHA image element exists on the web page, and allow a user to enter the CAPTCHA text.

You must add this command to all locations in the agent where CAPTCHA blocking could be encountered.

Important: Manual CAPTCHA processing relies on a human user to decode the CAPTCHA image, so an agent using manual CAPTCHA configuration cannot be run from the scheduler or API, or any other fully automated way.

Automatic CAPTCHA Configuration

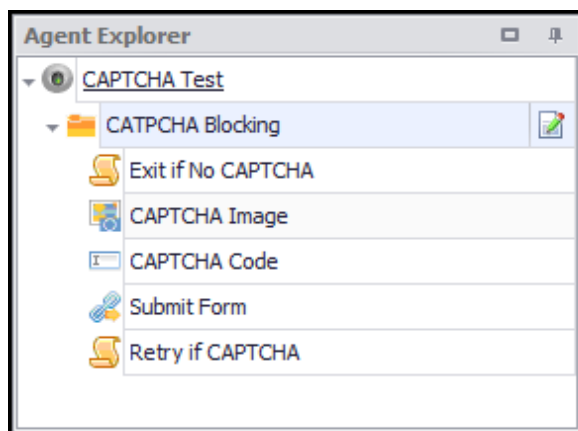
Automatic CAPTCHA processing requires an account with a third party CAPTCHA recognition service. The third party recognition service must provide a .NET API and you must add an OCR script that uses this API to call the service. See the section below for two examples of CAPTCHA recognition services.

Follow these steps to configure an agent for automatic CAPTCHA processing:

1. Add a new Group Commands command. This group of commands will handle CAPTCHA.
2. Add an Execute Script command to the group. This command will skip CAPTCHA processing if the CAPTCHA image doesn't exist in the web page.
 - i. Select the CAPTCHA image in the web browser. This sets the command's web selection.
 - ii. Select the default script type Exit Command.
 - iii. Select the default command Parent Command.
 - iv. Select the default condition If Selection Missing.
3. Add a Download Image command to the group. This command will download the CAPTCHA image and use an OCR script to decode the image into plain text.
 - i. Select the CAPTCHA image in the web browser. This sets the command's web selection.
 - ii. Open the OCR tab and check the option Convert image to text.
 - iii. Add an OCR script. See below for script examples. If you want to manually resolve the CAPTCHA, you can

check the option **Convert image manually** in which case you should not specify a script.

4. Add a Set Form Field command to the group. This command will use the converted image text to set the CAPTCHA text box.
 - i. Select the CAPTCHA form field on the web page. This sets the command's web selection.
 - ii. Clear the command option Use default input.
 - iii. Set the data provider to Captured Data and select the CAPTCHA image command from step 3.
5. Add a Navigate Link command to the group. This command will submit the CAPTCHA form.
 - i. Select the CAPTCHA form submit button. This sets the command's web selection.
6. Add an Execute Script command to the group. This command will retry CAPTCHA processing if the CAPTCHA image still exists on the web page. If the CAPTCHA image still exists we assume it's because the CAPTCHA recognition service decoded the CAPTCHA image incorrectly and we'll try again.
 - i. Select the CAPTCHA image in the web browser. This sets the command's web selection.
 - ii. Select the default script type Retry Command.
 - iii. Select the default command Parent Command.
 - iv. Select the default condition If Selection Exists.



Automatic CAPTCHA configuration

The command library includes the group of commands listed above. Select the *Automatic CAPTCHA* command from the library and add it to your agent. After you have added the group command from the library, you need to set the web selection for all the commands that require a web selection.

You must add this group of command to all locations in the agent where CAPTCHA blocking could be encountered.

CAPTCHA OCR Scripts

Content Grabber includes the API and standard OCR scripts to call the following CAPTCHA recognition services.

<http://www.deathbycaptcha.com>

<http://bypasscaptcha.com>

At the time of writing, the Death by CAPTCHA service charges US\$6.95 for 5000 CAPTCHAs and Bypass CAPTCHA charges US\$34 for 5000 CAPTCHAs. We are not affiliated with these

companies in any way and don't charge any additional fees for these services.

The following OCR script uses the Death by CAPTCHA service to decode CAPTCHA images.

```
public static string
ConvertImageToText(ConvertImageToTextArguments args)
{
    string captcha =
    DeathByCaptchaService.DecodeCaptcha(args.Image, "login",
    "password");
    return captcha;
}
```

The *login* and *password* in the script above is provided by Death by CAPTCHA.

The following OCR script uses the Bypass CAPTCHA service to decode CAPTCHA images.

```
public static string
ConvertImageToText(ConvertImageToTextArguments args)
{
    string captcha =
    BypassCaptchaService.DecodeCaptcha(args.Image, "key");
    return captcha;
}
```

The *key* in the script above is provided by Bypass CAPTCHA.

Troubleshooting

If CAPTCHA fails when entering a correct CAPTCHA it may be caused by the following issue.

Content Grabber cannot directly get the CAPTCHA image from the web browser, so it downloads the image a second time, and that may be disallowed by the web server, especially if you are using a proxy rotation service where a new IP address maybe assigned for the image download. To overcome this problem, you can use a **Download Screenshot** command instead of a **Download Image** command, in which case a second image download is not required.

12.2 IP Blocking & Proxy Servers

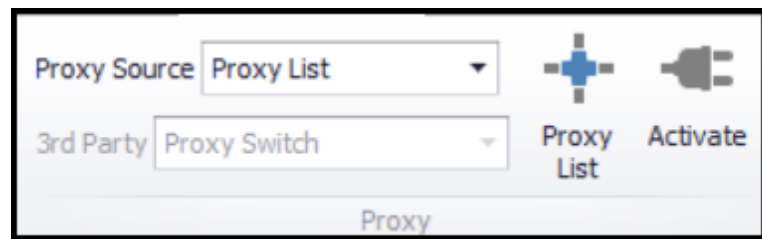
When you visit a website with a web browser or a web scraping tool, such as Content Grabber, the website owner can record your IP address and may be able to use this information to identify you or block your access to the website.

If you do not want a website owner to be able to identify you while you visit a website, you can use a proxy server to hide your IP address. When you use a proxy server, you do not visit the target website directly, but instead request that the proxy server visit the website for you.

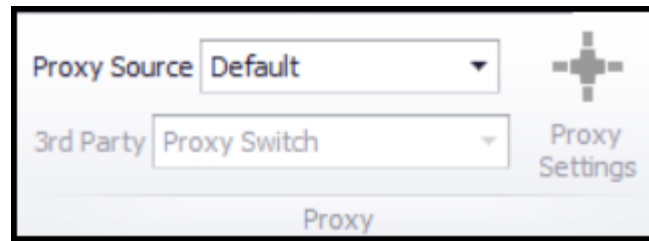
There are many different types of proxy server, but Content Grabber supports only HTTP proxy servers. It does not support other types of proxy servers, such as SOCKS proxies.

How to Configure Proxy Servers

After you have purchased proxy server access or found freely available proxy servers on the web, you will receive one or more proxy server IP addresses and possibly a username and password to access the proxies. You need to enter this information into the Content Grabber agent by selecting *Agent Settings* from the ribbon menu, or you can setup default proxies in the *Application Settings* ribbon menu.



Agent proxy settings



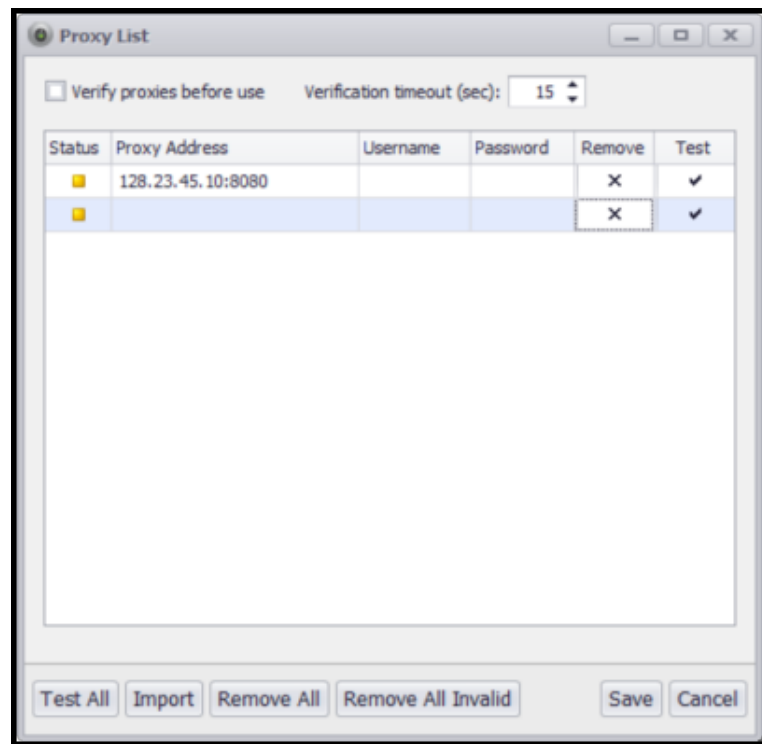
Application proxy settings

The proxy menus allow you to set the **Proxy source**. You can select one of the following proxy sources:

Proxy source	Description
Default	The agent will use the proxy configured in Internet Explorer, or no proxies if no proxies have been configured in Internet Explorer.
Application	The agent will use the default proxy settings configured in the <i>Application Settings</i> menu.
Proxy List	The agent will use a specific list of proxies. Click the <i>Proxy List</i> button to add proxies to the list.
Gateway	Use this proxy setting in <i>Application Settings</i> if you must connect to a specific proxy to access the Internet.
3rd Party	Content Grabber can integrate with the following 3rd party proxy providers:

	<p>Luminati</p> <p>Nohodo</p> <p>Private Proxy Switch</p> <p>If you set the 3rd party proxy to Proxy API, you can specify API configuration to download a list of proxies from a 3rd party API.</p> <p>You can also set the 3rd party proxy to Fiddler, which will allow you to view web traffic between Content Grabber and the target web server. This can sometimes be useful when debugging hard to process websites. This requires the Fiddler software to be running on the computer.</p>
Direct	The agent will not use any proxies.

The Proxy List screen is used to specify a list of proxies and set the proxy verification option:



Add proxies on the Proxy list screen

You must specify the **Proxy Address** in the following format, including the port number:

206.118.215.245:60099

In the example above, 206.118.215.245 is the IP address of the proxy server and 60099 is the port number.

Proxy Verification

Content Grabber can automatically verify if a proxy is available before switching to the proxy. This allows agents to switch to the next proxy if a proxy is not available, and thereby avoid stopping the agent prematurely just because a proxy is unavailable.

Proxy verification	Description

option	
Verify proxy before use	The agent will verify a proxy before switching to the proxy.
Verification timeout	The number of seconds the agent will wait for a successful proxy verification.

Importing Proxy Servers

If you are using a large number of proxy servers, it can be tedious to add them all to each project. You can use the **Import Proxies** button to import a list of proxies from a CSV file. The CSV file must have the following format:

Proxy Address, Username, Password

The *Username* and *Password* are optional columns.

CSV Example 1:

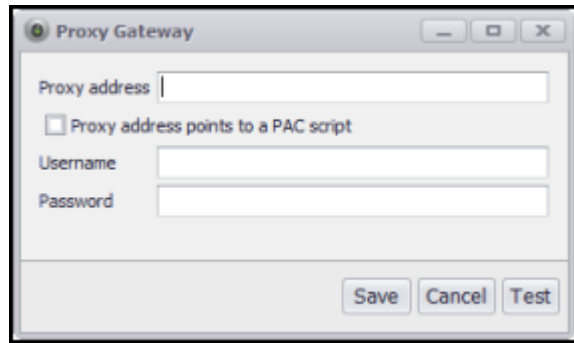
proxy
173.244.220.185:8800
50.21.10.78:8800
134.22.166.242:8800

CSV Example 2:

proxy, username, password
173.244.220.185:8800, user1, pass1
50.21.10.78:8800, user1, pass1
134.22.166.242:8800, user1, pass1

Proxy Gateway

Some company network configurations require all users to access the Internet through a specific proxy. Use the *Gateway* proxy option in *Application Settings* to open the *Proxy Gateway* screen.



Some companies require force Internet access
only through a proxy gateway

The **Proxy address** can specify a proxy directly or it can be a URL to a proxy configuration script (PAC).

13 Improving Agent Performance and Reliability

It is important to understand that the complexity of web scraping increases significantly when extracting large amounts of data. The time it takes to perform certain operations may take so long that the whole web scraping process becomes impractical. Web scraping is unreliable in nature, and the longer a web scraping project runs, the higher the chance something fails and interrupts the web scraping process.

The following topics list a few steps you can take to increase performance and reliability. Performance tuning is not for the novice user, and if you have no IT experience, you may not be able to implement the recommendations in this topic.

- Using MS SQL as Internal Database
- Using the Static Parser Browser
- Optimizing Agent Commands

13.1 Using MS SQL as Internal Database

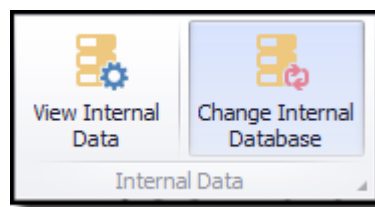
A Content Grabber agent saves all extracted data to an internal database while running. The default internal database is a SQLite file database, and file databases don't perform as well as real databases, such as SQL Server or MySQL. You are likely to see significant performance improvements and lower CPU usage when switching to a real database server depending on your system and the data you're extracting.

We recommend that you change the internal database to **SQL Server Express** if you don't already have an existing database server. **SQL Server Express** is a free database server from Microsoft and can contain up to 10GB of data in a single database. You can download **SQL Server Express** from this URL:

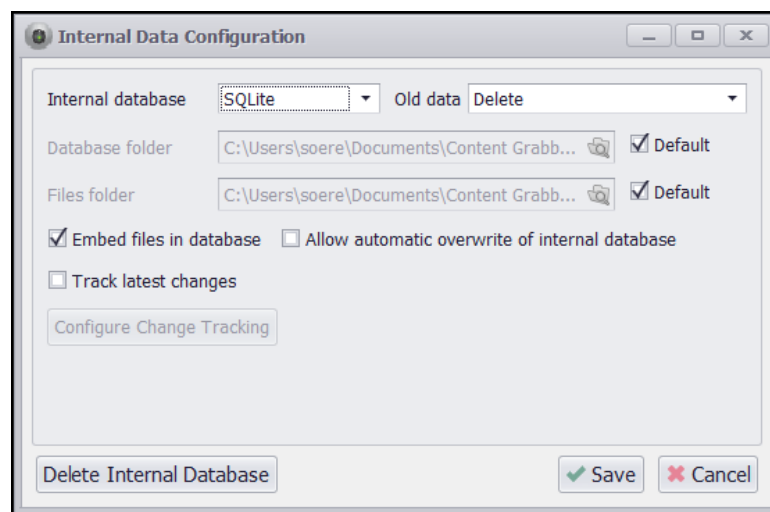
<http://www.microsoft.com/sqlserver/en/us/editions/express.aspx>

Read the **SQL Server Express** documentation to learn how to install and operate the database server.

Once you have installed SQL Server Express and added a database, you can change the internal database from the menu **Data > Change Internal Database**:



For example, you can improve performance by choosing **SQL Server** from the **Internal Database** drop-down:



13.2 Using the HTML Parser

When Content Grabber extracts data from a web page, it first loads the web page into a web browser. The web browser parses and renders the web page, and executes any JavaScript the page contains. This is a very safe approach, since

Content Grabber uses an embedded version of Chrome as it's web browser. Therefore if the target website is working in Chrome, Content Grabber can usually extract data from the website. However, the approach is also slow and may cause instability.

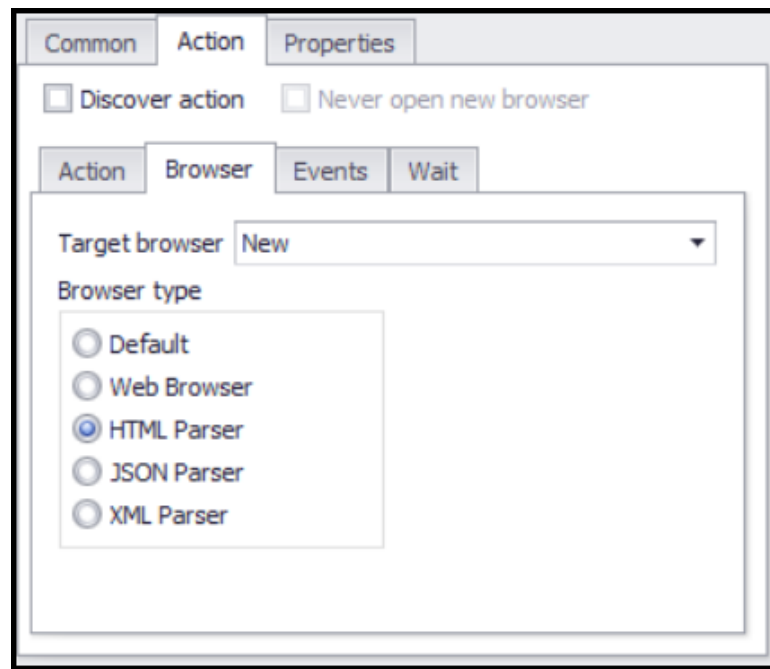
If you have been using Chrome to browse the web, you may have sometimes experienced problems, such as hanging websites or program crashes. This may occur very rarely (say once a year), so it may not be a problem during normal usage of Chrome. When Content Grabber uses Chrome to browse a website, it may access more web pages in a few hours than you access in a year, so stability issues are magnified significantly.

The main source of website instability is JavaScript. A website developer can use JavaScript to implement dynamic features on the website, but JavaScript bugs may lead to memory leaks, hanging websites or even program crashes.

All action commands in a Content Grabber agent, that open a new web browser, can be configured to open a specific type of web browser. The default browser is an embedded version of Chrome, but you can change this to a **HTML Parser**. The **HTML Parser** does not use Chrome at all, and it completely ignores JavaScript, so it's generally much more reliable.

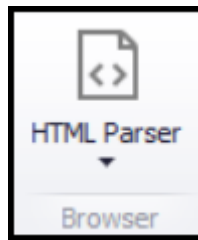
JavaScript is always single threaded, so many operations cannot be performed simultaneously when using Chrome web browsers. Since the Static Parser does not execute JavaScript, it can often process web pages much faster than a Chrome web browser.

Many websites don't work properly if JavaScript is disabled, so the **HTML Parser** will not work for all websites, but many websites can be partly processed with a **HTML Parser**, so you should always switch to a **HTML Parser** if a particular web page can be processed without JavaScript.



Configure an Action command to use a specific web browser type

If you want an agent to use the **HTML Parser** by default, then you can set the web browser type on the **Agent Settings > Browser > HTML Parser**:



13.3 Optimizing Agent Commands

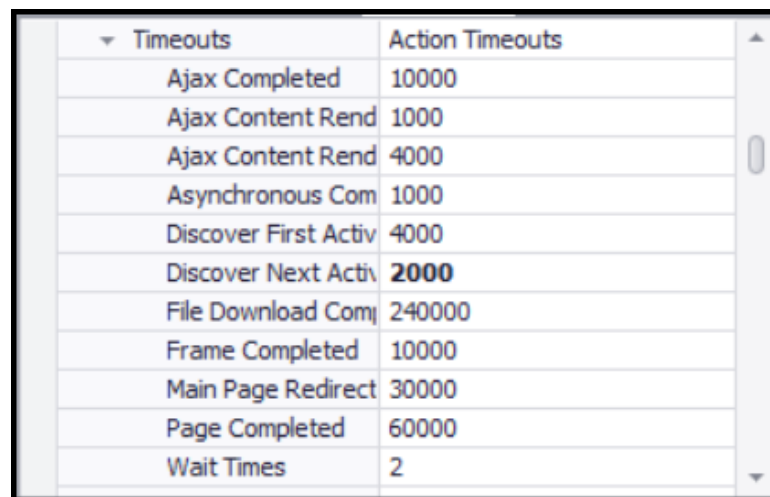
Action commands such as Navigate Link and Navigate URL commands are much slower to execute than commands that simply capture content, so it's important to try and limit the number of action commands in your agents. If you can access a target web page directly with a direct URL, rather than clicking multiple links, then you should always opt for the direct URL.

Many dynamic websites show and hide content when you click on buttons or links. Content Grabber can extract both visible and hidden content, so it's often not necessary to add an action command that simply makes content visible.

Optimizing Browser Activities

An agent spends most of its time waiting for web pages to load, and Content Grabber cannot always determine exactly when a page load or page action has completed, so it sometimes ends up waiting longer than it has to. If the agent starts extracting data too early, it may not be able to extract the data correctly, since some data may not have loaded onto the web page yet.

An action command uses a set of timeout values to determine how long it should wait for new activities. These timeout values are set fairly conservatively to make sure they are long enough for slow computers and slow Internet connections. You may be able to lower these timeout values, so an action command completes faster without failing to extract content correctly.



Timeouts	Action Timeouts
Ajax Completed	10000
Ajax Content Rend	1000
Ajax Content Rend	4000
Asynchronous Com	1000
Discover First Activ	4000
Discover Next Activ	2000
File Download Com	240000
Frame Completed	10000
Main Page Redirect	30000
Page Completed	60000
Wait Times	2

Default activity timeout values

13.4 Performance Sessions

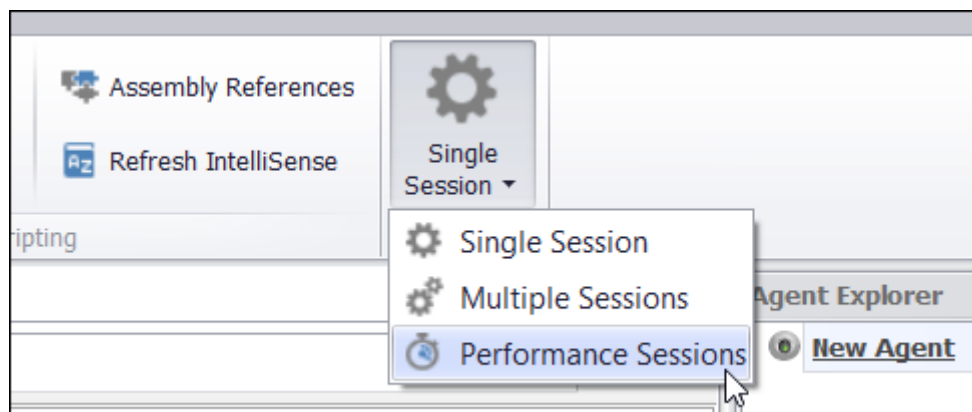
Sessions allow you to run multiple instances of the same agent at the same time. This can be used to split up large web scraping tasks and have multiple instances of an agent working on the task.

Content Grabber splits up a large task by dividing list entries into subsets, and each instance of the agent will then work on one of those subsets. For example, if you are processing a long list of start URLs, Content Grabber could divide the list into two and have one instance of the agent go through the first half of the list and a second instance of the agent go through the second half of the list.

Agents already use multithreading internally to split up work in a similar fashion as performance sessions, but sometimes it's faster to have multiple processes working on a task rather than multiple threads, especially if you run the processes on multiple computers. A single instance of an agent processing a website using multithreading needs to wait for threads to catch up at certain points. For example, when processing pagination with multiple threads going through a list of links on each page, some threads may finish before others, but they'll need to wait for all threads to finish before the agent can move to the next page in pagination. Multiple instances of the same agent run completely independent, so one instance will never have to wait for other instances to catch up.

Running an Agent in a Session

To run an agent in a session, you must specify a session ID when you run the agent and the agent must be configured to support sessions. To configure an agent to support sessions, set the agent option **Support Sessions** to **Performance Sessions** Agent Settings tab.



When using **Performance Sessions**, the session ID must be in a special format that dictates how work is divided between sessions. The input list associated with the Agent command (start command) will be divided by default, but you can specify any list command in an agent by setting the option **Process in Sessions** on the list command. You can only set this option on one command in an agent.

The special format of the session ID specifies how many sessions will work on the input list, and the subset of list entries the current instance of the agent should work on. The session ID must be in the following format.

[Subset to Process]/[Total Number of Sessions]

For example, if you have an agent that processes a list of 10 start URL, and want 5 instances of an agent to each process 2 URLs, then the session ID "3/5" would start an instance of the agent that processes URL number 5 and 6.

You can start multiple sessions at once by specifying a comma separated list of numbers, or a range of numbers. Here are a few examples:

"1-10/10" starts all 10 sessions.

"1,3-6/10" starts session 1 and 3 to 6.

1,4,5/10" starts session 1, 4 and 5.

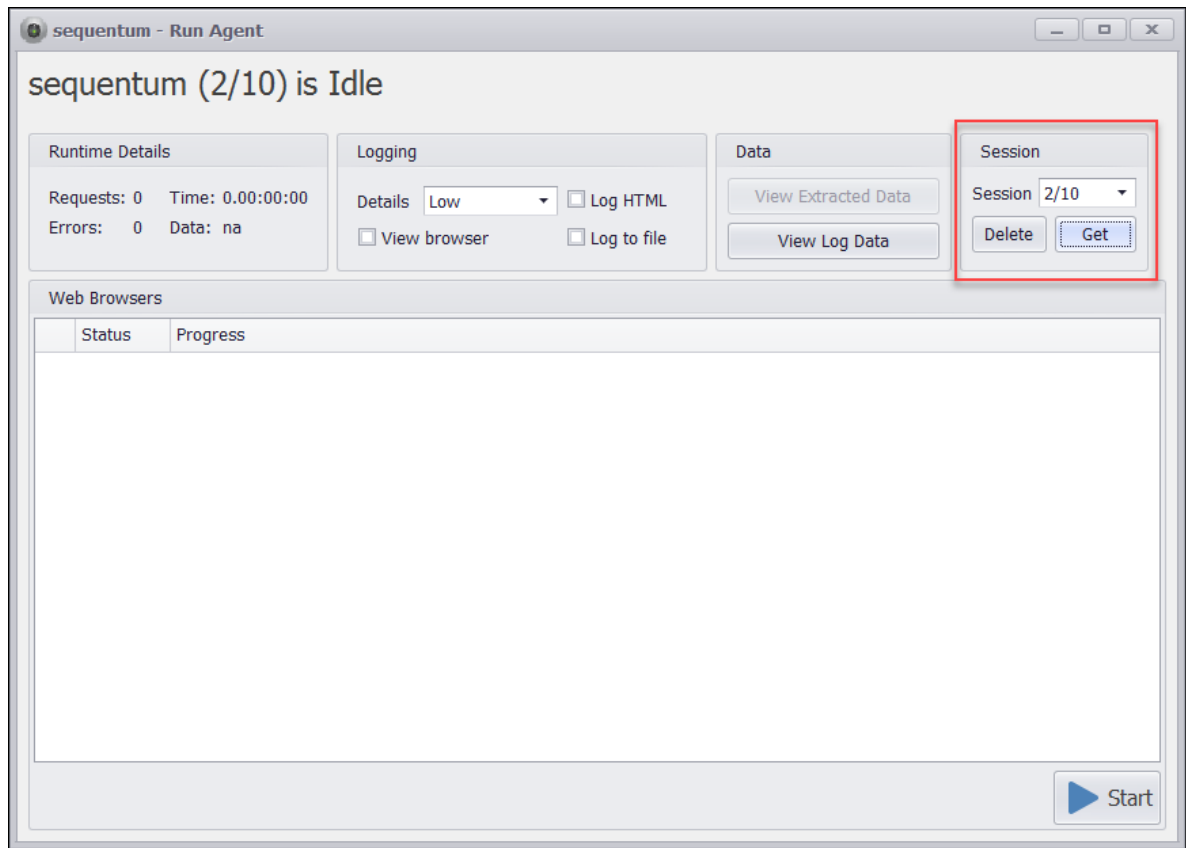
A session ID can be specified when running an agent from the commandline by using the command-line option *session_id*. The following command-line runs an agent named **sequentum** with a session ID "2/10".

```
RunAgent.exe "sequentum" session_id "2/10"
```

You can also specify a session ID when you run an agent from the Content Grabber editor. Open the **Run** window and enter a session ID or select an existing session ID

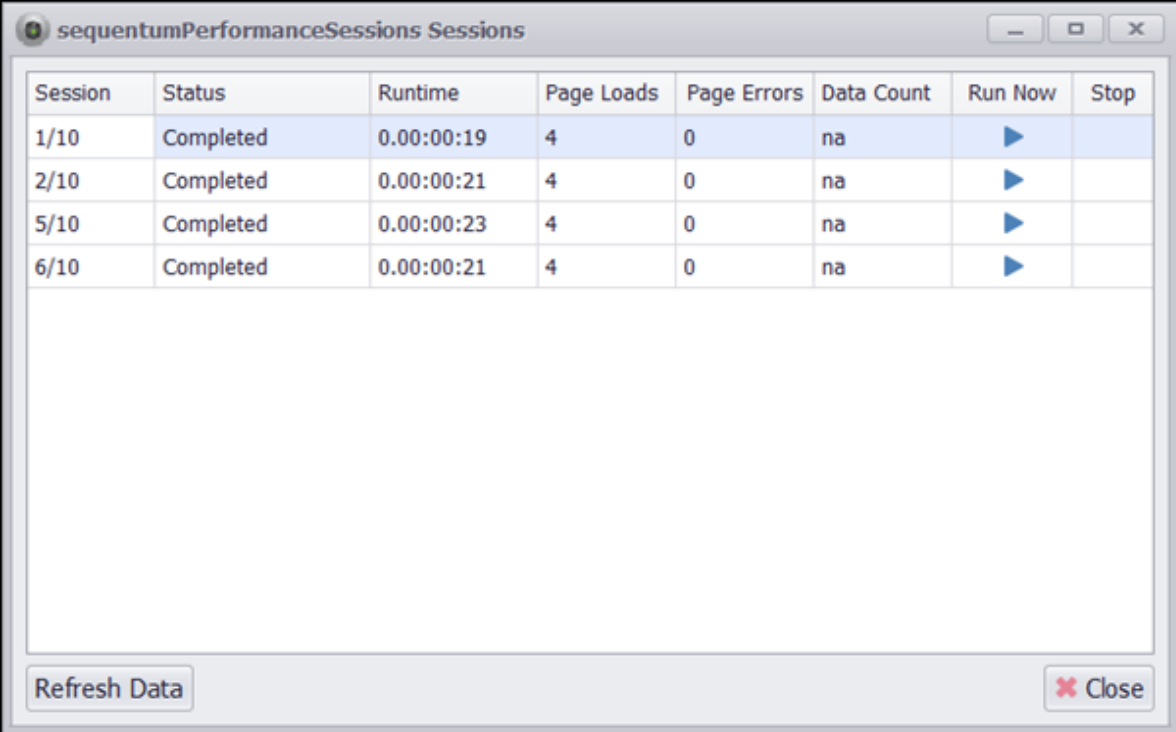
from the drop down box, and then press the **Get** button to make that session active.

Notice: The session panel is only visible if the agent supports sessions.



Run an agent with a session from the Content Grabber editor.

You can select **All Sessions** from the session dropdown list and press the **Get** button to open a window that displays status information for all agent sessions.



Session	Status	Runtime	Page Loads	Page Errors	Data Count	Run Now	Stop
1/10	Completed	0.00:00:19	4	0	na	▶	
2/10	Completed	0.00:00:21	4	0	na	▶	
5/10	Completed	0.00:00:23	4	0	na	▶	
6/10	Completed	0.00:00:21	4	0	na	▶	

Refresh Data Close

You delete all sessions by selecting **All Sessions** from the session dropdown list and then press the **Delete** button. You can also delete a range of sessions by specifying a session range.

Session Data Cleanup

Normally, data generated by sessions is cleaned up when the session expires. This is because sessions are normally always new sessions with a new session ID, so to avoid having old data hanging around forever, Content Grabber will remove session data periodically, unless you specifically tell it not to. However, performance sessions are different, since they are not always new sessions with new session IDs, so by default, Content grabber will not remove data generated by performance sessions.

You can manually delete one or more sessions from the **Run** window in the Content Grabber editor.

When you delete a session, Content Grabber will only clean up externally exported session data if the agent is configured to export data to a database. If you are using

an **Export Script** or if you are exporting to a file format, then you are responsible for any cleanup of exported session data.

You may not always want to remove externally exported session data when you delete a session. To prevent session data from being removed, set the agent option **Cleanup External Session Data** to false. This option can be found in the **Sessions** section on the advanced options tab.

13.5 Multithreading

Multithreading refers to multiple tasks running concurrently, and in Content Grabber that usually mean multiple HTML parsers and web browsers loading and processing multiple web pages from a target website at the same time.

Content Grabber uses a maximum of 5 active HTML parsers and 1 active dynamic browser to process a website by default, but that can be changed by setting the agent options **Max Active Parsers** and **Max Active Browsers**. An additional active parser or browser will sometimes be used by an agent to avoid deadlocks while waiting for an available parser or browser.

The **Max Active Parsers** option specified the number of active HTML, XML and JSON parsers, and **Max Active Browsers** specifies the number of active dynamic web browser.

The best number for **Max Active Parsers** and **Max Active Browsers** depends on how hard you can hit the target website without making the website unstable and without getting blocked by the website. The number also depends on how much memory and how many CPU cores are available on the computer running Content Grabber.

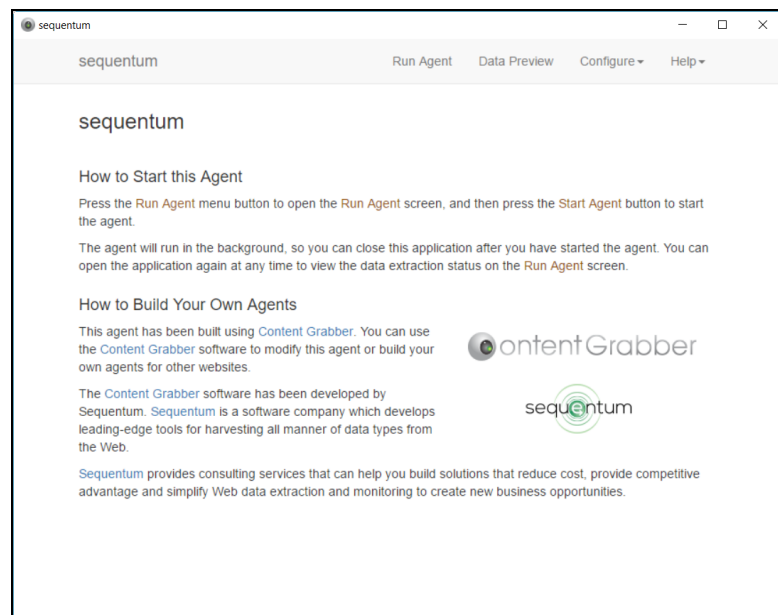
A dynamic web browser uses a significant amount of memory and CPU, and often uses many concurrent web connections to download content from a website, so the **Max Active Browsers** options should not be set too high. A HTML parser uses little memory and CPU, and only a single web connection, so the **Max Active Parsers** option can often be set much higher.

14 Building Self-Contained Agents

You can build self-contained agents that will run without the need for the Content Grabber application to be present on the target computer. Self-contained agents can be distributed royalty-free, and includes a user interface that allows end users to configure and run the agent.

Note: the self-contained user interface has an introduction screen that contains a promotional message for Content Grabber and Sequentum. This message can only be removed if you are using a premium version of Content Grabber.

The following figure shows the default self-contained user interface, in which you can add your own text and logo:



An end user can configure the following agent features:

- Export target (only file formats are supported)

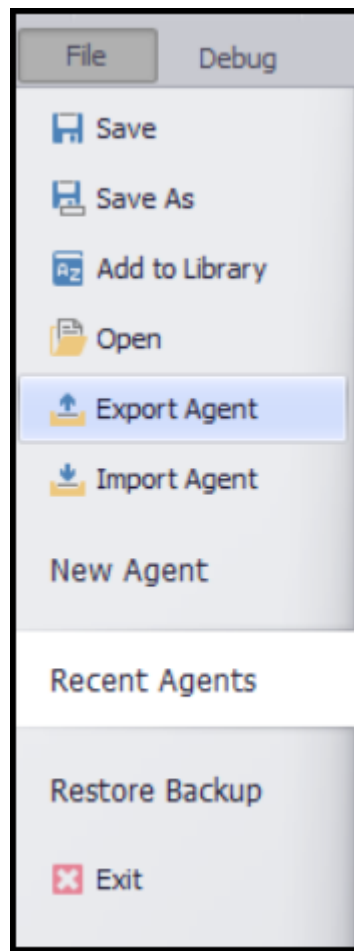
- Scheduling
- Input parameters
- Notifications
- Proxies.

A self-contained agent can only export to file formats (Excel, CSV or XML). It cannot export to databases or execute a data export script. Self-contained agents require .NET v4.5+. The self-contained agents will not attempt to download and install the required .NET version if it doesn't exist on the target computer, but most Windows computers will already have this version of .NET installed.

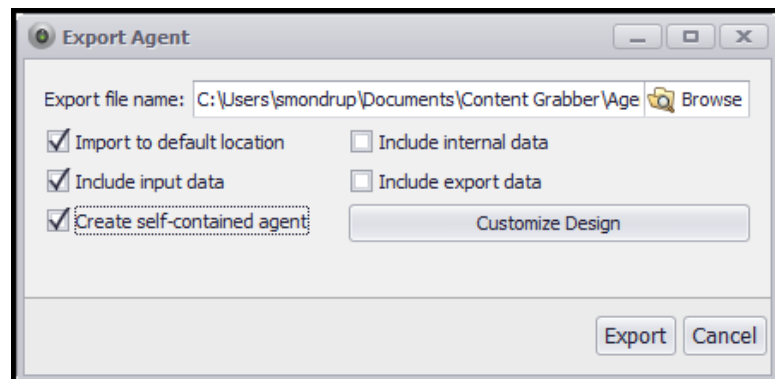
Building a Self-Contained Agent

Follow the steps below to build a self-contained agent:

1. Click the **Export Agent** application menu to open the **Export Agent** screen.



2. Make sure you check the option **Create self-contained agent**.



3. You can now click **Export** to create the self-contained agent. Before creating the self-contained agent you can click **Customize Design** to customize the design and text displayed on the agent's user interface.

For more information, see the topic [Customizing the User Interface](#).

Self-Contained Agent Files

All files required to configure and run a self-contained agent are included in a single executable file which can be distributed royalty free. Any custom assemblies used by an agent are included in the executable file, but only if they are located in the *Assemblies* folder for the agent. Assemblies shared by agents and located in the shared *Assemblies* folder are not included in the executable file, and should not be used in self-contained agents.

Upgrading a Self-Contained Agent

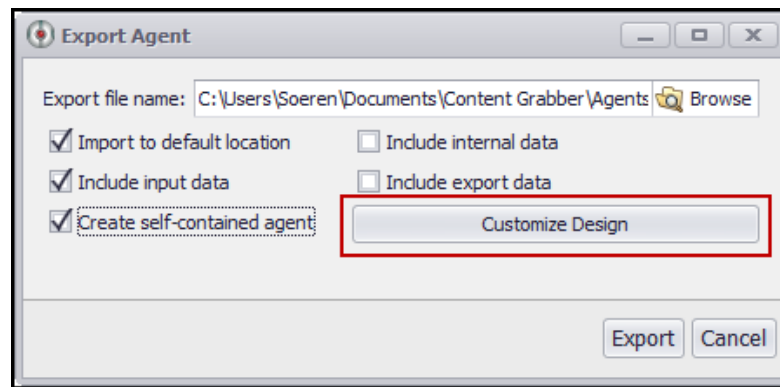
You can upgrade an agent if the target website of a self-contained agent has changed and you need to make changes to the agent in order for it to work correctly. Simply make the changes to your agent and export the agent again. The end-user can override his existing self-contained agent with your new agent. Any configuration changes the end-user has made to the agent will not be lost.

For more information, see the topic [Using a Self-Contained Agent](#).

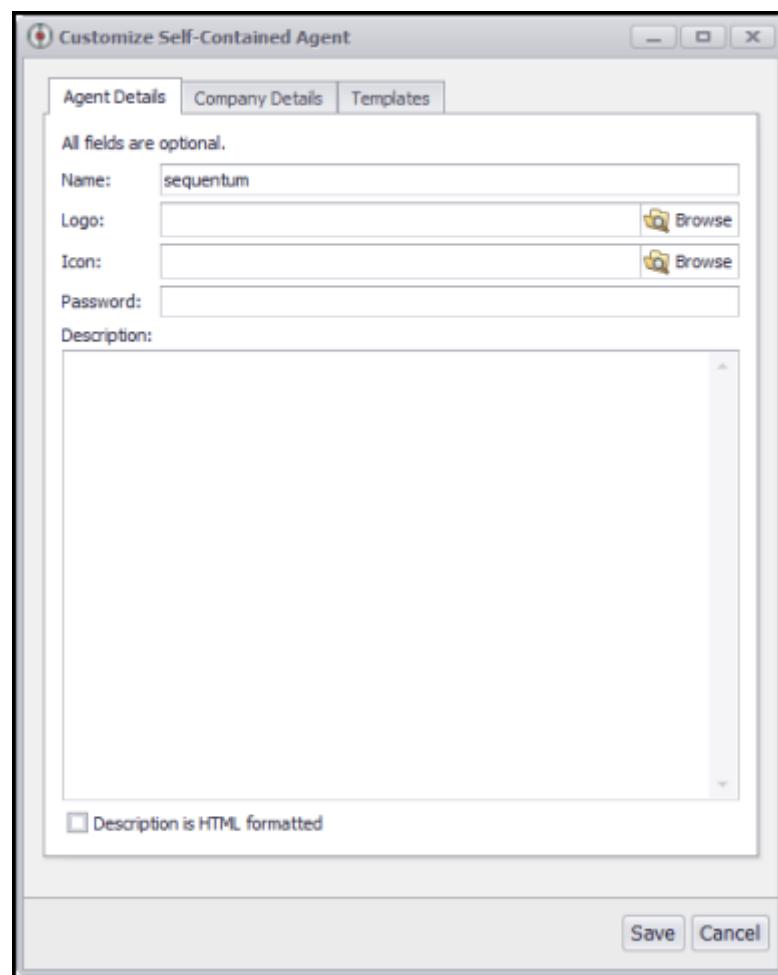
14.1 Customizing the User Interface

A self-contained agent includes a user interface that allows users to configure and run the agent. You can control some of the text and images displayed on this user interface, and you can also control which agent options the user can configure.

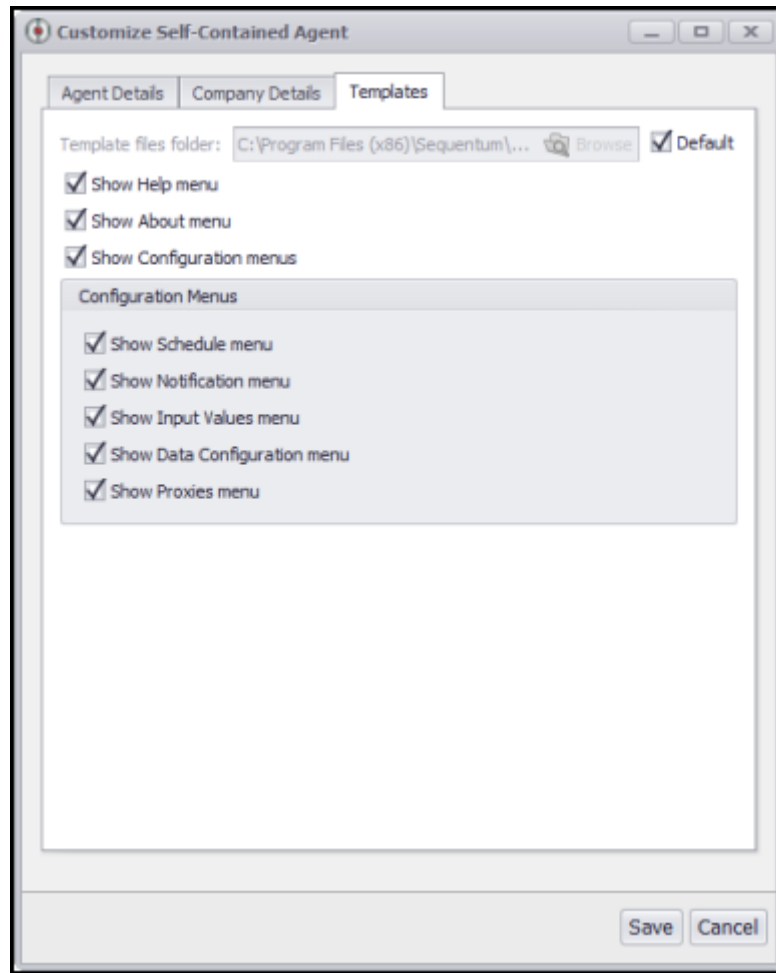
Click the **Customize Design** button to open the **Customize Self-Contained Agent** screen:



The **Customize Self-Contained Agent** screen has three tabs, in which you can specify the text and images that should be used on the user interface of the self-contained agent:



The **Agent Details** tab allows you to specify text and images that are displayed on the agent start-up screen. The **Company Details** tab allows you to specify text and images that are displayed on the agent About screen. The **Templates** tab allows you to specify which configuration options should be available on the user interface:



The **Templates** tab also allows you to specify custom template files. The template files are HTML files that controls the user interface layout. The template files use JavaScript in a file AgentProxy.js to control the agent and get information about the agent. For example, the JavaScript has methods to get the information you provide on the **Agent Detail** and **Company Details** tabs.

If you want to provide your own template files, we recommend you copy the default template files to a new location and modify them to suit your needs. The default files are located in:

[Content Grabber Installation Folder]\Resources\SelfContained\Docs

This folder contains a number of files and sub-folders. You can remove or add files to your custom folders. All files and sub-folders in your custom folder will be included in the executable agent file.

14.2 Using Input Data

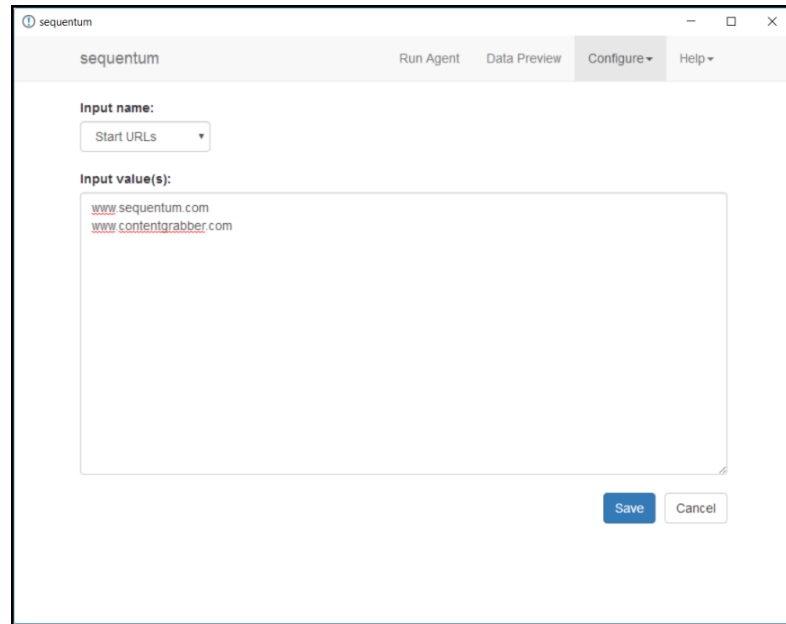
A self-contained agent can use any input data that is supplied by a database. If an agent uses data from a CSV file, the file should be placed in the default agent input folder, and you should set the option to include input data when exporting the agent.

If an agent uses data from a Simple Data Provider, the standard self-contained user interface can be used by the end-user to edit the input data. A Simple Data Provider is only available in the self-contained user interface if the data provider property *Public Provider* is set to True. A public name for the data provider can also be specified. This public name will only be used in the self-contained user interface, and not in the Content Grabber editor.

▼ Data Provider	Simple
▶ CSV provider	
▶ Database Provider	None
Design Row Index	1
Provide Data As List	False
Provider Type	Simple
Public Provider	True ▼
Public Provider Name	Start URLs
▶ Script Data Provider	Disabled
▶ Selection Provider	URL
▶ Simple Provider	www.sequentum....

Allow Simple input data to be edited using
the self-contained user interface

An agent can have multiple public data providers, and each data provider can be selected from a drop down box as shown on the following figure.



Simple input data can be edited using the self-contained user interface

14.3 Using a Self-Contained Agent

A self-contained agent is a single executable file that includes all the required files to configure and run the agent. A self-contained agent does not need to be installed. When you want to configure or run the agent, simply click on the executable agent file, and the agent user interface will be displayed.

The first time the executable agent file is run, a sub-folder with the same name as the agent will be created in the folder where the executable file is located. This folder will store configuration information for the agent, so the folder should not be removed. If the folder is removed, all configuration changes you have made to the agent will be lost, and the agent will start with its default configuration.



Upgrading a Self-Contained Agent

You can upgrade a self-contained agent by simply overriding the existing executable agent file with a new agent file. Any configuration changes you have made to the agent will not be lost, unless you delete the configuration folder.

15 Running Agents from the Command-Line

You can run agents from the command line using the **RunAgent.exe** program, which you can find in the Content Grabber installation folder.

RunAgent.exe agentName

You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the program. If the agent is not located in that folder, Content Grabber will look in the default public agent location. The default public agent location is:

C:\Users\Public\Documents\Content Grabber 2\Agents

If your agent is not located in any of the default agent locations, you need to specify the full path to the agent file:

RunAgent.exe "C:\My Custom Folder\agentName.scg"

When running multiple agents in a batch file, you can use the **Start** command to run the agents asynchronously:

Start "WindowTitle" RunAgent.exe agentName1
Start "WindowTitle" RunAgent.exe agentName2

Exit Codes

The **RunAgent.exe** command line program returns one of the following exit codes:

Exit Code	Status	Description
0	Success	Data extraction was completed successfully.

Exit Code	Status	Description
1	Failed	A critical application error occurred during data extraction.
2	Incomplete	Data extraction was interrupted.
3	Completed with errors	Data extraction was completed, but with one or more page load errors or missing required elements.
4	Incomplete with errors	Data extraction was interrupted, and encountered one or more page load errors or missing required elements.
5	Export Failed	Data extraction failed because it was unable to export data. You can manually open Content Grabber and attempt to export.
6	License Restriction	The agent contains features that are only available to agents created with a Premium Edition of Content Grabber when using the royalty free runtime.
7	Completed Unsuccessfully	Agent run completed without meeting defined success criteria.
8	Already Running	Trying to start an agent that is already running.
9	Invalid Data Cannot Continue	Retrying or continuing an agent that has an invalid internal data structure.
10	Initialization Script Failed	The agent initialization script failed.
11	Process Start Timeout	Timeout waiting for agent process to start.
12	Unknown Exception Check Log	Unknown error. Look for details in the error log.
13	Invalid Parameter	An invalid agent command-line parameter was specified. Look for details in the

Exit Code	Status	Description
		error log.
14	Process Error Exit Before Start	The agent process exited with an error before the agent started successfully.
15	Agent Import Folder not Found	An agent import folder has been specified, but the folder does not exist.
16	Agent Import Name Empty	An agent import name has been specified, but the name is empty.
17	Error Creating Default Document Folders	An error occurred while trying to create default Content Grabber folders in the current users document folder. This error will only be returned from the Content Grabber editor, never from the Content Grabber runtime.
18	Default Log Folder Missing	The log_to_file or log_path has been specified without a log path.
19	ErrorCreatingDefaultLogPath	Error creation specified log path.

Command Line Arguments

Input parameters can be added as command line arguments. The input parameter name must be preceded with a dash. For example:

```
RunAgent.exe agentName -username "test" -password "test"
```

Please see the topic Input Parameters for more information.

The following command line switches can be used, and should not have a preceding dash:

Switch	Description
log_level	Log detail level. Logging is turned off by default. Example: <i>RunAgent.exe agentName log_level High</i>
log_html	Logs the HTML of all loaded web pages to files. Example: <i>RunAgent.exe agentName log_html</i>
log_to_file	Logs information to a file instead of a database. A log path must be specified. Specify an empty string to use the default log path. Example: <i>RunAgent.exe agentName log_to_file ""</i>
view_browser	Displays the web browsers used to navigate the target website. Example: <i>RunAgent.exe agentName view_browser</i>
no_ui	No user interface will be displayed. This will cause an agent pause command to fail. Example: <i>RunAgent.exe agentName no_ui</i>
session_id	The agent will run in a specified Session. Example: <i>RunAgent.exe agentName session_id sdgfd4353</i>
session_timeout	If an agent runs in a Session, this is the session timeout in minutes. The default session timeout is 30 minutes. Example: <i>RunAgent.exe agentName session_id sdgfd4353 session_timeout 10</i>
run_method	Specifies how to run an agent. One of the following values is expected.

Switch	Description
	<ul style="list-style-type: none">• Restart• Continue• ContinueRefreshAgent• ContinueAndRetryErrors• ContinueAndRetryErrorsRefreshAgent <p>Example: <i>RunAgent.exe agentName run_method continue</i></p>
agent_import_folder	If the specified agent is an exported agent, the agent will be imported to this folder.
agent_import_name	If the specified agent is an exported agent, the agent will be renamed to this name. Only applicable when agent_import_folder is specified.
log_path	Use this parameter instead of log_to_file if you want to log to database, but want to specify the log path for any errors that occur before the agent is able to log to database. Don't specify this parameter if you have already specified log_to_file .

15.1 Using the Content Grabber runtime

NOTICE: This functionality is only available with a **Content Grabber OEM License**.

You can use the Content Grabber command-line program on a computer without installing the entire Content Grabber software environment. You must first copy the Content Grabber runtime files to any folder on the computer.

The Content Grabber runtime files can be generated in the Content Grabber application by choosing **Runtime Package** in the **Application** menu. This will generate a zip file with all required files and folders. All these files and folders must be copied to a folder on the target computer.

The **RunAgent.exe** command-line program is part of the Content Grabber runtime and can be used in the same way **RunAgent.exe** program in the Content Grabber installation folder.

These are the minimum computer specifications for running the Content Grabber runtime:

- **Windows 7/8/10/2008R2/2012/2012R2**
- **.NET framework version 4.5+**

16 Running Multiple Instances of the Same Agent

Content Grabber can run as many agents concurrently as your computer can handle, but by default, you can only run a single instance of each agent at the same time. There are a few scenarios where it's important to be able to run multiple concurrent instances of the same agent, and Content Grabber allows you to configure an agent to support this.

Web Applications

Running multiple instances of the same agent is particularly useful when running an agent directly from a website where you can have many web users visiting the website at the same time. Each web user can start the agent in their own session and the user will only see agent progress status and extracted data belonging to his session.

Please see the topic Sessions in the Programming Interface chapter for more information.

Splitting Large Web Scraping Tasks

Large web scraping tasks can take a long time to complete, but you can decrease this time by splitting up the task and have multiple instances of an agent working on the task. You can run instances of the same agent on multiple computers to further optimize the process.

Please see the topic Sessions in the Improving Performance and Reliability chapter for more information.

17 Scripting

You can use scripting in a variety of ways to customize Content Grabber behavior, or to extend standard functionality. Content Grabber scripts are .NET functions written in C# or VB.NET, or regular expressions.

Content Grabber scripts can be divided into three categories:

- Content transformation scripts.
- Command scripts.
- Extension scripts.

Content Transformation Scripts

Content transformation scripts are used to transform a piece of text. For example, if you extract an address from a web page, but want only the postal code, you can transform the address into just the postal code. You can also view this process as extracting sub-text, but sometimes you may do more than merely extract sub-text. That is why we call this transformation.

Content transformation scripts are used in the following situations:

- Transformation of extracted content.
- Transformation of file names when downloading files.
- Transformation of URLs when using Navigate Link commands that extract and navigate to direct URLs.
- Transformation of input data.

See the topic Content Transformation Scripts for more information.

Command Scripts

Command scripts are general purpose scripts that are executed as part of the normal agent command execution flow. The scripts are often used to alter the execution flow of an agent.

See the topic Command Scripts for more information.

Extension Scripts

Extension scripts are used to extend Content Grabber functionality. A **Data Export** script is an example of an extension script that can be used to apply some custom business logic to extracted data.

The following scripts are extension scripts:

- Agent Initialization Scripts
- Data Export Scripts
- Data Distribution Scripts
- Data Input Scripts
- Image OCR Scripts
- Convert Document to HTML Scripts

17.1 Script Languages

The Content Grabber scripting engine supports C#, VB.NET and Regular Expressions. C# and VB.NET can be used for all types of scripts, but Regular Expressions can only be used by content transformation scripts.

Regular Expressions

This manual does not explain regular expression syntax. Please visit the following website for more information about regular expressions:

Regular Expressions Reference Guide

Regular expression scripts in Content Grabber can include any number of regex match and replace operations. Each regex operation must be specified on two lines. The first line must contain the regex pattern and the second line must contain the operation.

The following operations are supported.

Operation	Description
return	Returns the first match in the original content or a selected group within the match. The returned match can be combined with static text.
return all	Returns all matches or the specified group within all matches.
return table	Returns all matches in a data table. Each captured group becomes a column in the data table. Named capture groups can be used to name the data columns.
return if match otherwise	<p>Return a specific value if a match is found and another value if a match is not found. For example, True could be returned if a match is found and False if a match is not found.</p> <p>Example:</p> <p>return if match "True" otherwise "False"</p>
replace with	Replaces all matches - or a selected group within the first match, in the original content and then returns the content.

A group within a match must be specified by the group number, and cannot be specified by a group name, except when using the **return table** operator. The group number 0 specifies the entire match, and the number 1 specifies the first group in the match. A group number must be preceded by the character \$, so for example, **\$1** specifies the first group in a match. Use two \$ characters (\$\$) to escape the \$ character in an operation.

If a regex script contains more than one regex operation, the next operation will work on the output from the preceding operation. All regex operations are case insensitive and line breaks are ignored. The following eight special operations can be specified on a single line:

- **strip_html** removes all HTML tags from the content.
- **url_decode** decodes an encoded URL.
- **html_decode** decodes encoded HTML.
- **unescape_string** unescapes content that has been string encoded. For example, \" will be converted to " and \\ to \.
- **trim** removes line breaks and white spaces from the beginning and end of the content.
- **line_breaks** converts some HTML tags such as <P>,
 and into standard Windows line breaks.
- **to_lower** converts text to lower case.
- **to_upper** converts text to upper case.
- **capitalize_words** capitalizes all words in the text.
- **aggregate [separator]** if *return all* has been used to return a list, this operation adds all strings in the list together separated by the specified separator.
- **insert_data** inserts data into data templates defined in the content.

The syntax `{$content_name}` can be used to reference extracted data, input data, global data or input parameters. For example, if you had a capture command named

product_id, you could construct the following regular expression to extract all text between the product ID and the first white space:

```
{$product_id}(.*)\s
```

The specified name may reference a capture command name, a data provider command name, an input parameter name, or a global data name. The script will first try and find a matching capture command, then a matching data provider command, then an input parameter name, and lastly a global data name. If the data from one source does not exist or is empty, the script will proceed to the next data source.

Examples

Here are a number of examples:

Regular Expression	Description
.* return	Returns the entire match, so everything in this case.
A(.*)B return \$1	Returns the group 1 match, so everything between A and B.
(A).*(B) return \$1\$2	Returns group 1 and 2 matches, so AB in this case.
A(.*)B replace with	Replaces every instance of everything between A and B (including A and B) with nothing.
A(.*)B	Replaces every instance of everything between A and B (including A and B) with "some new text".

replace with some new text	
A(.*)B replace \$1 with some new text	Replaces the first instance of everything between A and B (excluding A and B) with "some new text".
A(.*)B replace with \$1	Replaces every instance of everything between A and B (including A and B) with the text between A and B, so in effect it removes A and B.
A(.*)B return \$1 C(.*)D replace with	First extracts everything between A and B, and then replaces every instance of everything between C and D (including C and D) with nothing.
^(.*) replace with A\$1B	Inserts A at the beginning of the string and B at the end of the string.

 replace with \r\n	Replaces all HTML tags with standard Windows line breaks.
A(.*)B return \$1 url_decode	Returns the group 1 match, so everything between A and B, and then URL_decodes the result.
A(.*)B return C\$1D	Returns the group 1 match, but adds C to the beginning and D to the end before returning the match.

C# and VB.NET

This manual does not explain C# and VB.NET syntax. Please visit an online reference guide for more information about these languages.

C# and VB.NET scripts must have one class named **Script** with a static function that is executed by Content Grabber. The signature of the static method depends on the type of script. The following example is for a Data Input script.

```
using System;
using System.Data;
using Sequentum.ContentGrabber.Api;
public class Script
{
    public static DataTable ProvideData(DataProviderArguments args)
    {
        DataTable data = args.ScriptUtilities.LoadCsvFileFromDefaultInputFolder("inputData.scv");
        return data;
    }
}
```

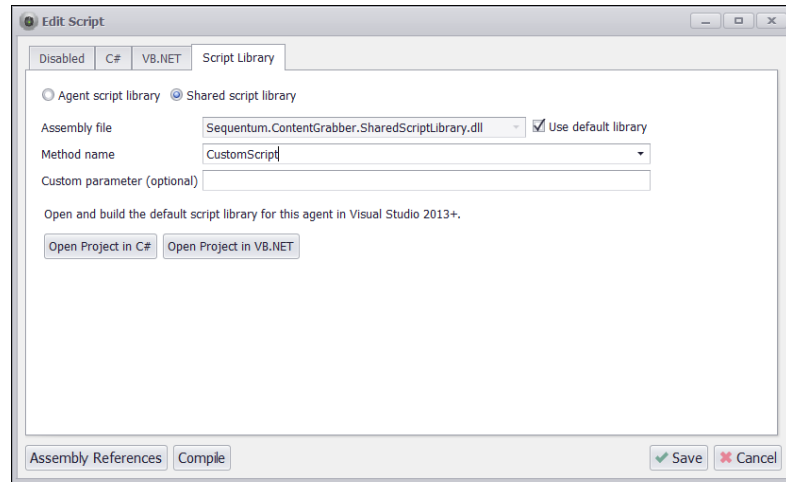
A script can have more than one function in the **Script** class, and can also use functionality from external .NET libraries. All external .NET libraries must be added to the agent's assembly references. See the topic Assembly References for more information.

17.2 Script Library

Content Grabber provides a default procedure for developing scripts in **Visual Studio**. The build-in script editor in Content Grabber does not provide advanced development tools such as a debugger, so we recommend you build large scripts in Visual Studio and smaller scripts in the built-in script editor.

When you add a script in Content Grabber, you can select to use a script library. The script library is a normal .NET class library that contains classes and methods for all

types of scripts in Content Grabber. When using a Script Library, Content Grabber will automatically call the specific methods in the .NET class library.



Agent or Shared Script Library

Content Grabber supports two types of script libraries, a script library that is shared between all agents in your Content Grabber document folder, and a script library that is used only for a specific agent.

The shared script library assembly must be placed in the folder *Content Grabber\Assemblies* and the script library is shared by all agents located in *Content Grabber\Agents*. The *Content Grabber* folder can be located anywhere, but by default it's located in *My Documents\Content Grabber*.

The agent script library assembly must be placed in the **Assemblies** sub-folder of an agent folder.

Visual Studio Solution Template

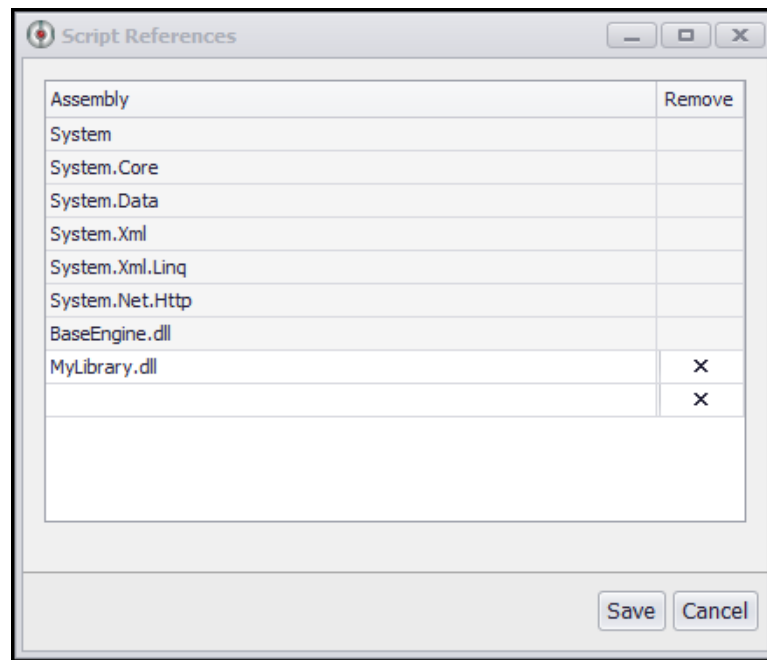
Content Grabber provides a Visual Studio 2013+ solution with a project that contains all the required classes and methods for a default script library. When you click the button **Open Project in C#** or **Open Project in Vb.NET**, Content Grabber will

automatically copy the solution and projects to your agent's **Visual Studio** folder or to the shared **Visual Studio** folder if you are using a shared script library. The **Visual Studio** solution also contains a test project you can use to test the functionality in your script library.

When you build the **Visual Studio** solution in release mode, the script library assembly is automatically copied to the right location so it's picked up by Content Grabber when you run an agent. When Content Grabber loads the script library it will become locked and you will need to close Content Grabber before you can build the library in release mode. The debug version of the library is not locked by Content Grabber, because it's not copied to any of the **Content Grabber Assemblies** folders, and will not be used by your agents by default.

17.3 Assembly References

You can use external assemblies in your scripts by adding assembly references to your agent. Click the **Assembly References** button when editing a script to open this screen:



You should only enter the file name of your assembly file when adding a reference, not the full file path. You must copy the assembly file to the agent's *Assemblies* folder or to the shared *Assemblies* folder. Content Grabber will look for unresolved assemblies in those two folder.

The shared *Assemblies* folder is located in *Content Grabber\Assemblies* and assemblies in this folder are shared by all agents in *Content Grabber\Agents*. The *Content Grabber* folder can be located anywhere, but by default it's located in *My Documents\Content Grabber*.

17.4 Script Utilities

All scripts have access to a *ScriptUtils* class through the script arguments. This class contains the following utility functions:

Function	Description
DataTable LoadCsvFile(string path)	Loads a CSV file from a specified location and returns the data in a DataTable.
DataTable LoadCsvFile(string path, char separator)	Loads a CSV file from a specified location using a specified value separator and returns the data in a DataTable.
DataTable LoadCsvFileFromDefaultInputFolder(string filename)	Loads a CSV file with a specified name from the default input data folder and returns the data in a DataTable.
DataTable LoadCsvFileFromDefaultInput	Loads a CSV file with a specified name from the default input data folder using a

Folder(string filename, char separator)	specified value separator and returns the data in a DataTable.
void ExecuteCommandLine(string programFilePath, string inputFilePath, string outputFilePath, string options)	Executes a program on the command line. The parameters inputFilePath, outputFilePath and options are added as command line parameters.
void ExecuteCommandLine(string programFilePath, string arguments)	Executes a program on the command line with the specified command line parameters.
int ClearBrowserCookies(string url)	Clears all Internet Explorer cookies on the computer associated with the specified URL.
void ResetBrowserSession()	Clears the Internet Explorer session cookies. This is equivalent to restarting an Internet Explorer browser.
string PostData(string url, string postData, int timeout = 0)	Posts data to a web server.

string PostData(string url, byte[] postData, int timeout = 0)	Posts data to a web server.
string PostData(string url, string postData, string headers, int timeout = 0)	Posts data to a web server.
string PostData(string url, byte[] postData, string headers, int timeout = 0)	Posts data to a web server.
string GetData(string url, int timeout = 0)	Gets data from a web server.
string GetData(string url, string headers, int timeout = 0)	Gets data from a web server.
string TransformContent(string	Transforms data with a Content Grabber regex script. The regex script must follow the exact format of a Content

input, string regexScript)	Grabber transformation script. Use \r\n for line breaks.
---------------------------------------	--

Example

The following example loads data from a CSV file located in the agent's default input folder:

```
using System;
using System.Data;
using Sequentum.ContentGrabber.Api;
public class Script
{
    public static DataTable ProvideData(DataProviderArguments args)
    {
        DataTable data = args.ScriptUtilities.LoadCsvFileFromDefaultInputFolder("inputData.scv");
        return data;
    }
}
```

Database Utilities

All scripts have access to the following function through the script arguments:

```
public IConnection GetDatabaseConnection(string connectionName)
```

The function returns a predefined database connection. See the topic Database Connections for more information about predefined database connections.

The **IConnection**, **ICommand** and **IReader** interfaces are Content Grabber interfaces that are meant to make it easier to write and read data to a database, but you can always use the standard .NET libraries if you prefer.

The following example writes some extracted data to a database:

```

using System;
using Sequentum.ContentGrabber.Api;
using Sequentum.ContentGrabber.Commands;
public class Script
{
    public static CustomScriptReturn CustomScript(CustomScriptArguments args)
    {
        IConnection connection = args.GetDatabaseConnection("exportDatabase");
        connection.OpenDatabase();
        try
        {
            ICommand command = connection.GetNewCommand();
            command.SetSql("insert into export_table values
                (@title, @description)");
            command.AddParameterWithValue("title", args.DataRow["title"], CaptureDataType.S
            command.AddParameterWithValue("description",
                args.DataRow["description"], CaptureDataType.ShortText);
            command.ExecuteNonQuery();
        }
        finally
        {
            connection.CloseDatabase();
        }
        return CustomScriptReturn.Empty();
    }
}

```

The **IConnection** interface has the following functions and properties:

Function or Property	Description
IReader GetNewReader(string sql)	Returns an IReader interface to a new Reader object that uses the current connection.

ICommand GetNewCommand()	Returns an ICommand interface to a new Command object that uses the current connection.
void OpenDatabase()	Opens the database connection.
void CloseDatabase()	Closes the database connection.
object GetConnection()	Return the underlying database connection that is specific to the database type used. For example, if the database is a MySQL database, the object returned is a MySqlConnection .
void ExecuteNonQuery(string sql)	Executes a SQL statement that doesn't return a result set.
object ExecuteScalar(string sql)	Executes a SQL statement that returns a single scalar value.
long ExecuteCount(string sql)	Executes a SQL statement that returns a single long value. This function should be used when using the SQL function "Count".
bool HasTable(string tableName)	Return true if the specified table exists in the database.
bool HasColumn(string tableName,	Return true if the specified columns exists in the specified table.

string columnName)	
DataRow[] GetTableColumns(string tableName)	Return all columns in the specified database table.
void DropTable(string tableName)	Drops the specified database table.
void TruncateTable(string tableName)	Truncates the specified database table.
void Lock()	Locks the database connection, so it cannot be used by any other running threads.
void Release()	Releases a lock on the database connection.
IReader ExecuteReader(string sql, params object[] pars)	Executes a SQL with a list of SQL parameters and returns a IReader.
DataTable ExecuteDataTable(string sql, params object[] pars)	Executes a SQL with a list of SQL parameters and returns a DataTable.

void ExecuteNonQuery (string sql, params object[] pars)	Executes a SQL with a list of SQL parameters that doesn't return a result.
object ExecuteScalar (string sql, params object[] pars)	Executes a SQL with a list of SQL parameters that returns a single scalar value.
long ExecuteCount (string sql, params object[] pars)	Executes a SQL with a list of SQL parameters that returns a single long value. This function should be used when using the SQL function "Count".

The **ICommand** interface has the following functions and properties:

Function or Property	Description
void AddParameterWithValue (string pName, object pValue, CaptureDataType type)	Adds a SQL parameter to the Command object. A matching parameter must exist in the associated SQL statement. The value of a parameter is updated if the parameter already exists.
void AddParameter (string pName,	Adds a SQL parameter to the Command object. A matching parameter must exist in the associated

CaptureDataType e type)	SQL statement. The value of a parameter is updated if the parameter already exists.
void SetParameterVa lue(string pName, object pValue)	Sets the value of an existing SQL parameter. A matching parameter must exist in the associated SQL statement.
void ExecuteNonQue ry(string sql)	Executes a SQL statement that doesn't return a result set.
void ExecuteNonQue ry()	Executes an existing SQL statement that doesn't return a result set.
object ExecuteScalar(s tring sql)	Executes a SQL statement that returns a single scalar value.
object ExecuteScalar()	Executes an existing SQL statement that returns a single scalar value.
long ExecuteCount(s tring sql)	Executes a SQL statement that returns a single long value. This function should be used when using the SQL function "Count".
long ExecuteCount()	Executes an existing SQL statement that returns a single long value. This function should be used when using the SQL function "Count".
void SetSql(string	Set the SQL statement associated with the SQL command.

sql)	
------	--

The **IReader** interface has the following functions and properties:

Function or Property	Description
void SetSql(string sql)	Sets the SQL statement associated with this data reader.
void AddParameterWithValue(string pName, object pValue)	Adds a SQL parameter to the Command object. A matching parameter must exist in the associated SQL statement. The value of a parameter is updated if the parameter already exists.
bool Read()	Reads a row of data. Executes an existing SQL statement if it has not already been executed.
void Close()	Closes the data reader.
DateTime GetDateTimeValue(int columnIndex)	Gets a DateTime value from the current row.
string GetStringValue(int columnIndex)	Gets a String value from the current row.
int GetIntValue(int columnIndex)	Gets a Integer value from the current row.

Guid GetGuidValue(int columnIndex)	Gets a Guid value from the current row.
Type GetFieldType(int columnIndex)	Gets the data type of a specified column.
IDataReader GetDataReader()	Returns the underlying .NET data reader.
object GetFieldValue(int columnIndex)	Gets a data value from the current data row.

Extension Methods

Content Grabber provides a few extension methods that can be used in all scripts:

Function	Description
static DataTable ToDataTable(this string stringValue, string columnName)	Converts a single string value into a DataTable with one data column and one data row.
static DataTable ToDataTable(this string[] dataRows, string columnName)	Converts an array of strings into a DataTable with one data column and one data row for each string value.

Function	Description
static DataTable ToDataTable(this string[] dataRows, string columnName, string stringFormat)	Converts an array of strings into a DataTable with one data column and one data row for each string value. A standard .NET format string can be used to format the string value before it is inserted into the DataTable.
static DataTable ToDataTable(this List<string> dataRows, string columnName)	Converts a list of strings into a DataTable with one data column and one data row for each string value.
static DataTable ToDataTable(this List<string> dataRows, string columnName, string stringFormat)	Converts a list of strings into a DataTable with one data column and one data row for each string value. A standard .NET format string can be used to format the string value before it is inserted into the DataTable.

Example

The following script generates a list of URLs and uses an extension method to convert the list of URLs into a DataTable.

```
using System;
using System.Collections.Generic;
using System.Data;
using Sequentum.ContentGrabber.Api;
public class Script
{
    public static DataTable ProvideData(DataProviderArguments args)
    {
        List<string> urls = new List<string>();
        for(int i=1;i<1000;i++)
        {
            urls.Add("http://www.domain.com/page.php?ID="+i.ToString());
        }
        return urls.ToDataTable("url");
    }
}
```

```

    }
}

```

Runtime Data

Extracted data can be accessed at run-time using the `RuntimeData` class available through the script arguments:

Function	Description
InternalReader GetInternalReaderForRow(string tableName, Guid rowId)	Returns a data reader for a specified row in a specified internal data table.
GetInternalReader(string tableName, Guid? parentRowId)	Returns a data reader for a specified internal data table with the specified parent row ID.
InternalReader GetInternalReader(string tableName, Guid? parentRowId, SortedList<string, object> searchColumnValues)	Returns a data reader for a specified internal data table with the specified parent row ID and column values.
InternalReader GetInternalReader(string tableName, SortedList<string, object> searchColumnValues)	Returns a data reader for a specified internal data table matching the specified column values.
InternalReader GetInternalReader(string tableName, SortedList<string, object> searchColumnValues, int	Returns a data reader for a specified internal data table matching the specified column values. Selects a specified number of data rows starting at a specified index. The result will be

Function	Description
startIndex, int count, string orderColumnName = null)	ordered by Row ID if the order column name is not specified.
InternalReader GetInternalReader(string tableName)	Returns a data reader for a specified internal data table.
InternalReader GetInternalReader(string tableName, int startIndex, int count, string orderColumnName = null)	Returns a data reader for a specified internal data table matching the specified column values. Selects a specified number of data rows starting at a specified index. The result will be ordered by Row ID if the order column name is not specified.
InternalReader GetInternalReaderForCurrentRow()	Returns a data reader for the current row in the current internal data table.
InternalReader GetInternalReaderForRow(Guid rowId)	Returns a data reader for the specified row in the current internal data table.
InternalReader GetInternalReader(Guid? parentRowId)	Returns a data reader for the current internal data table with the specified parent row ID.
InternalReader GetInternalReader(Guid? parentRowId, SortedList<string, object> searchColumnValues)	Returns a data reader for the current internal data table with the specified parent row ID and column values.
InternalReader GetInternalReader(SortedLis	Returns a data reader for the current internal data table with the specified column values.

Function	Description
t<string, object> searchColumnValues)	

The methods in the RuntimeData class all returns an internal data reader with the following methods and properties:

Function	Description
bool Read()	Reads the next data row. Returns false if there are no more data rows available.
void Close()	Closes the data reader.
string GetStringValue(string columnName)	Gets a String value from the current row.
int? GetIntegerValue(string columnName)	Gets a Integer value from the current row.
Guid? GetGuidValue(string columnName)	Gets a Guid value from the current row.
DateTime? GetDateTimeValue(string columnName)	Gets a Date/Time value from the current row.
double? GetFloatValue(string columnName)	Gets a Float value from the current row.
double? GetFloatValue(string columnName)	Gets a Boolean value from the current row.
string GetStringValue(int index)	Gets a String value from the current row.

Function	Description
int? GetIntegerValue(int index)	Gets a Integer value from the current row.
Guid? GetGuidValue(int index)	Gets a Guid value from the current row.
DateTime? GetDateTimeValue(int index)	Gets a Date/Time value from the current row.
double? GetFloatValue(int index)	Gets a Float value from the current row.
bool? GetBooleanValue(int index)	Gets a Boolean value from the current row.
Type GetFieldType(int index)	Gets the data type of a specified column.
object GetFieldValue(string columnName)	Returns a data value from the current data row.
object GetFieldValue(int index)	Returns a data value from the current data row.
Guid GetKey()	Returns the primary key of the current row.
Guid? GetParentKey()	Returns parent key of the current row or null if the row has no parent data.
int GetSegment()	Returns the data segment of the current row.
int GetStatus()	Returns the status of the current row.
string GetSessionId()	Returns the session ID or null if no session is used.
object this[string columnName]	Returns a data value from the current data row.
object this[int index]	Returns a data value from the current data row.

Function	Description
InternalReader GetChildTable(string tableName)	Returns a data reader that reads data from a specified child table of the table associated with this data reader. The returned data reader will only read data rows that are child data to the current data row.
InternalReader GetChildTable(string tableName, Guid parentRowId, SortedList<string, object> searchColumnValues)	Returns a data reader that reads data from a specified child table of the table associated with this data reader.
InternalReader GetChildTable(string tableName, SortedList<string, object> searchColumnValues)	Returns a data reader that reads data from a specified child table of the table associated with this data reader.
InternalReader GetChildTable(string tableName, Guid parentRowId)	Returns a data reader that reads data from a specified child table of the table associated with this data reader.
InternalReader GetChildTableRow(string tableName, Guid rowId)	Returns a data reader that reads data from a specified child table of the table associated with this data reader.
InternalReader GetParentTableRow()	Returns a data reader that reads the parent data row of the table associated with this data reader.
InternalReader GetParentTableRow(Guid	Returns a data reader that reads a specified data row from the parent table of the table

Function	Description
rowId)	associated with this data reader.
InternalReader GetParentTable(Guid parentRowId, SortedList<string, object> searchColumnValues)	Returns a data reader that reads data from the parent table of the table associated with this data reader.
InternalReader GetParentTable(SortedList< string, object> searchColumnValues)	Returns a data reader that reads data from the parent table of the table associated with this data reader.
InternalReader GetParentTable(Guid parentRowId)	Returns a data reader that reads data from the parent table of the table associated with this data reader.
string[] ChildTableNames	An array containing the names of all child tables of the table associated with this data reader.
InternalDataColumn[] Columns	Return all columns in the current table.
IDataReader GetDataReader()	Returns the underlying .NET data reader.
string ReadAllToJson(bool isIncludeChildData = false, bool isIncludeFiles = false)	Reads all rows and returns them as a JSON string.
string ReadAllToJsonList(bool isIncludeChildData = false, bool isIncludeFiles = false)	Reads all rows and returns them as a JSON list.
string ToJson(bool isIncludeChildData = false,	Returns the current row as a JSON string.

Function	Description
bool isIncludeFiles = false)	

Examples

The following example reads all data from the current row, including child data, and returns the data as a JSON string.

```
IRuntimeData data = args.RuntimeData;  
InternalReader reader = data.GetInternalReaderForCurrentRow();  
string json;  
if(reader.read())  
{  
    reader.ToJson(true, false);  
}  
else  
{  
    json = "No data available";  
}  
return json;
```

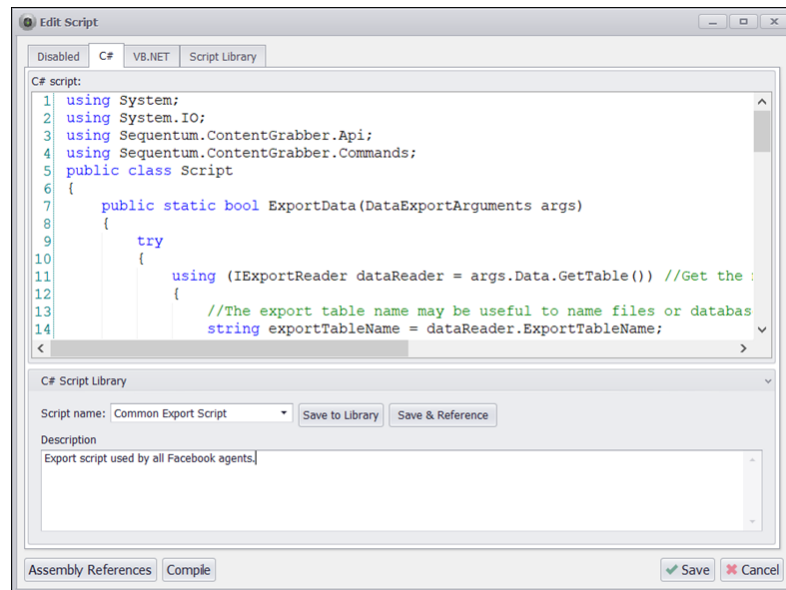
The following example reads all data associated with the current row from the first child table, and returns the data as a JSON string.

```
IRuntimeData data = args.RuntimeData;  
InternalReader reader = data.GetInternalReaderForCurrentRow();  
InternalReader childReader = reader.GetChildTable(reader.ChildTableNames[0],  
args.DataRow.RowId);  
string json = childReader.ReadAllToJsonList(true, false);  
return json;
```

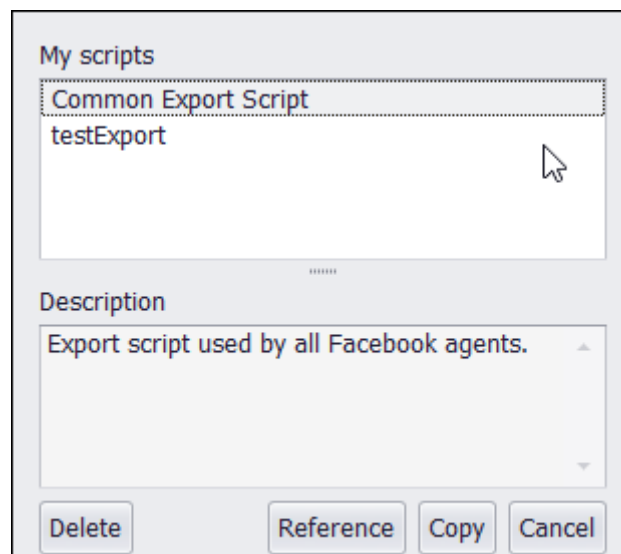
17.5 Script Template Library

Content Grabber has a Script Template Library that allows you to easily share scripts between agents. This library also contains some standard scripts for common operations, such as regular expressions used to extract emails or phone numbers from a piece of text.

You can save a script to the library at anytime when writing a script. Simply enter a script name and a description, and click **Save to Library** or **Save & Reference**. See **Template References** below for more information about template references. If you save a script with the same name as an existing script, the existing script will be overwritten.



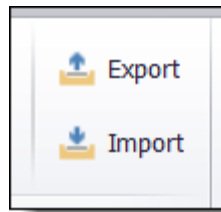
You can load a script from the library by opening the **Script name** drop down box, and then selecting the script you want to load.



Press the **Copy** button to copy the script from the template into the script editor. Press the **Reference** button to copy the template script into the script editor and reference the script template. See **Template References** below for more information about template references.

Exporting and Importing Template Libraries

All templates can be exported and imported, so you can move the templates from one computer to another. Content Grabber exports or imports all templates, including agent, command and script templates. You cannot export or import just script templates.



Export or import templates from the Tools menu.

Template References

When using a script template, a copy of the script will be inserted into the script editor and can then be modified and saved to an agent. If the script template later changes, it will not effect the script that was saved to the agent. If you have many agents that use the same script templates, it's sometimes convenient if any change to a template is reflected in the agents that use the template. This is possible by using a reference to the script template instead of a copy.

When adding a script template reference, a local copy of the template will be stored inside the agent. If an agent is copied to

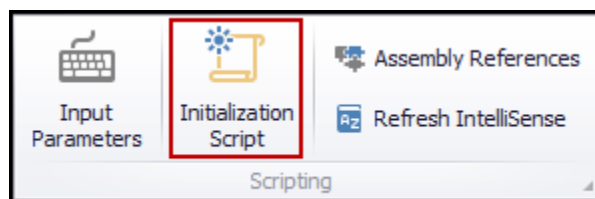
another computer where the referenced template doesn't exist, the local copy of the template will be used instead.

When editing a script that has a script template reference, the local template stored in the agent will be modified, not the computer wide template. If the agent is reopened, the local template stored in the agent will be replaced with the computer wide template, and any changes made to the local template will be discarded. When saving a script that has a script template reference, you will be given a choice to save changes to both the local script template and the computer wide script template.

17.6 Agent Initialization Scripts

An agent initialization script is run before the agent starts. The script can be used to initialize data before an agent is run. The script is also often used to set agent properties, such as database connection parameters, but we don't recommend you do this, since the *Agent* class is undocumented and unsupported.

You can add an agent initialization script to an agent by clicking the menu **Agent > Initialization Script**:



Agent initialization scripts can be written in C# or VB.NET.

The following example initializes a global counter.

```
using System;
using Sequentum.ContentGrabber.Api;
public class Script
{
```

```

        public static bool InitializeAgent(AgentInitializationArguments args)
        {
            args.GlobalData.AddOrUpdateData("counter", 1);
            return true;
        }
    }
}

```

The following example makes changes to an agent before it's run. The changes to the agent are not permanent, but only in effect at runtime. When you are debugging an agent, the debugger works directly on the agent in the editor, so any changes to an agent are permanent. We therefore recommend you use the **args.IsDebug** variable to turn off any changes to an agent during debugging.

```

using System;
using Sequentum.ContentGrabber.Api;
using Sequentum.ContentGrabber.Commands;
public class Script
{
    public static bool InitializeAgent(AgentInitializationArguments args)
    {
        //Don't make changes to the agent while debugging
        If(args.IsDebug)
            return true;

        //Set the file path and download folder if we're exporting to CSV
        if(args.Agent.ExportDestination.ExportTargetType == ExportTargetType.Csv)
        {
            if(args.GlobalData.HasData("CsvFilePath"))
            {
                args.Agent.ExportDestination.CsvExport.UseDefaultPath = false;
                args.Agent.ExportDestination.CsvExport.FilePath = args.GlobalData.GetStri

            }

            if(args.GlobalData.HasData("DownloadFolder"))
            {
                args.Agent.ExportDestination.UseDefaultDownloadPath = false;
                args.Agent.ExportDestination.DownloadDirectoryName = args.GlobalData.G
            }
        }
    }
}

```

```

    }
    //Set the database connection if we're exporting to a SQL Server database
    else if(args.Agent.ExportDestination.ExportTargetType == ExportTargetType.SqlServer)
    {
        if(args.GlobalData.HasData("DatabaseName"))
        {
            args.Agent.ExportDestination.DatabaseExport.DatabaseConnectionName =
        }
    }

    //Set the database connection and SQL for the input data provider in the Agent command
    if(args.GlobalData.HasData("DatabaseName") && args.GlobalData.HasData("SQL"))
    {
        args.Agent.DataProvider.DatabaseProvider.DatabaseConnectionName = args.Globa
        args.Agent.DataProvider.DatabaseProvider.Sql = args.GlobalData.GetString("SQL")
    }

    return true;
}
}

```

If you continue an agent, the initialization script is not run, but any changes that was made to global data or the agent are retained.

NOTE: The script must have a static method with the following signature:

```
public static bool InitializeAgent(AgentInitializationArguments args)
```

All logging in an agent initialization script is always written to file and never to database. This is because the script runs before the agent starts, and the internal database is not available at that time.

An instance of the *AgentInitializationArguments* class is provided by Content Grabber and has the following functions and properties:

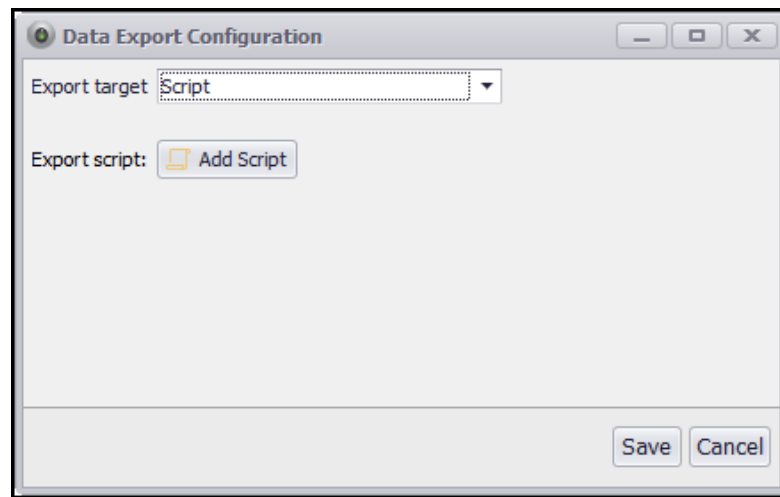
Property or Function	Description
Agent	The current agent that is executing the script.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
bool IsDebug	True if the agent is running in debug mode.
InputData InputDataCache	All input data available to the agent command.
void WriteDebug(string debugMessage, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void WriteDebug(string debugMessage, bool showMessageInDesignMode, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void Notify(bool alwaysNotify)	Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
void Notify(string message, bool alwaysNotify)	Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.

Property or Function	Description
GlobalDataDictionary GlobalData	Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts. Input Parameters are also stored in this dictionary.
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.

17.7 Data Export Scripts

Data Export scripts can be used to customize the export process. An export script could be used to export extracted data to custom data structures in a database, or it could export data to a custom data source.

A **Data Export** script can be added from the **Data Export Configuration** screen:



The following example writes data from two export tables to two CSV files:

```
using System;
using System.IO;
using Sequentum.ContentGrabber.Api;
using Sequentum.ContentGrabber.Commands;
public class Script
{
    public static bool ExportData(DataExportArguments args)
    {
        try
        {
            StreamWriter writer1 = File.CreateText(@"c:\temp\data1.csv");
            try
            {
                StreamWriter writer2 = File.CreateText(@"c:\temp\data2.csv");
                try
                {
                    IExportReader mainTableReader = args.Data.GetTable();
                    try
                    {
                        while(mainTableReader.Read())
                        {
                            string csvData1 = childTableReader.
                                GetStringValue("Field Name 1");

                            csvData1 += ",";
                        }
                    }
                }
            }
        }
    }
}
```



```
csvData1 += childTableReader.GetStringValue  
("Field Name 2");  
  
writer1.WriteLine(csvData1);  
  
IExportReader childTableReader =  
mainTableReader.GetChildTable("Child table name");  
try  
{  
    while(childTableReader.Read())  
    {  
        string csvData2 =  
childTableReader.GetStringValue("Child Field Name 1");  
        csvData2 += ",";  
        csvData2 += childTableReader.  
GetStringValue("Child Field Name 2");  
        writer2.WriteLine(csvData2);  
    }  
}  
finally  
{  
    childTableReader.Close();  
}  
}  
finally  
{  
    mainTableReader.Close();  
}  
}  
finally  
{  
    writer1.Close();  
}  
}  
finally  
{  
    writer2.Close();  
}  
return true;  
}
```

```
        catch(Exception ex)
        {
            args.WriteDebug(ex.Message, DebugMessageType.Error);
            return false;
        }
    }
}
```

The script must have a static method with the following signature:

```
public static bool ExportData(DataExportArguments args)
```

The function should return **True** if it succeeds and **False** if it fails. In this example, an instance of the **DataExportArguments** class will be given by Content Grabber and has the following functions and properties:

Property or Function	Description
Agent Agent	The current agent.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
IExportData Data	The data to export.
bool IsDebug	True if the agent is running in debug mode.
ExtendedSorted List<string, string> InputParameters	All input parameters defined by the agent.

void WriteDebug(string debugMessage, DebugMessageType messageType = DebugMessageType type.Information)	<p>Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.</p>
void WriteDebug(string debugMessage, bool showMessageIn DesignMode, DebugMessageType type messageType = DebugMessageType type.Information)	<p>Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.</p>
void Notify(bool alwaysNotify)	<p>Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.</p>
void Notify(string message, bool alwaysNotify)	<p>Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a</p>

	notification if the agent has been configured to send notifications on critical errors.
GlobalDataDictionary GlobalData	Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts. Input Parameters are also stored in this dictionary.
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.
string[] ExportData(ExportTarget exportTarget, string sessionId = null)	Exports data to the specified export target. Returns the names of any exported data files if exporting to a file format.
void ExportToDatabase(string databaseConnectionName, string sessionId = null)	Exports data to the specified database.

string[] ExportToCsv(string sessionId = null)	Exports data to a CSV file using the default export folders and default options. Use the <code>ExportData</code> method to specify CSV export options. Returns the names of all exported CSV files.
string ExportToExcel(string sessionId = null)	Exports data to an Excel file using the default export folders and default options. Use the <code>ExportData</code> method to specify Excel export options. Returns the name of the exported Excel file.
string ExportToXml(string sessionId = null)	Exports data to an XML file using the default export folders and default options. Use the <code>ExportData</code> method to specify XML export options. Returns the name of the exported XML file.
void DistributeData(string file)	Distributes data to the target configured in the agent.
void DistributeData(string[] files)	Distributes data to the target configured in the agent.

The **IExportData** interface has the following functions and properties:

Property or Function	Description
IExportReader GetTable()	Returns a data reader that reads data from the main table containing exported data. All other data tables will be child tables of this table.

IExportReader this [string name] { get; }	Returns a data reader that reads data from a specified data table.
IExportReader GetTable (string name)	Returns a data reader that reads data from a specified data table.
bool ValidateData ()	Validates the export data and returns True if the data is valid and False if the data is invalid. If the agent that extracted this data has changed since the data was extracted, then the data may have become invalid.

The **IExportReader** interface has the following functions and properties:

Property or Function	Description
bool Read ()	Reads the next data row. Returns False if there are no more data rows available.
void Close ()	Closes the data reader.
string GetStringValue (string columnName)	Gets a String value from the current row.
int GetIntValue (stri ng columnName)	Gets a Integer value from the current row.

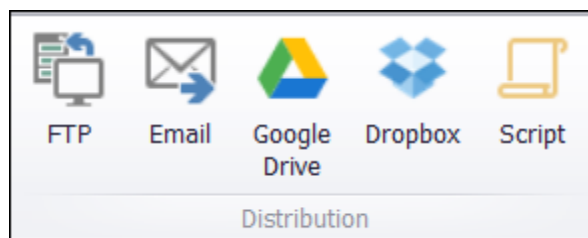
Guid GetGuidValue(string columnName)	Gets a GUID value from the current row.
string GetStringValue(int index)	Gets a String value from the current row.
int GetIntValue(int index)	Gets a Integer value from the current row.
Guid GetGuidValue(int index)	Gets a GUID value from the current row.
Type GetFieldType(int index)	Gets the data type of a specified column.
object GetFieldValue(string columnName)	Return a data value from the current data row.
object GetFieldValue(int index)	Return a data value from the current data row.
object this[string columnName]	Return a data value from the current data row.
object this[int index]	Return a data value from the current data row.

IExportReader GetChildTable(s tring tableName)	Return a data reader that reads data from a specified child table of the table associated with this data reader. The returned data reader will only read data rows that are child data to the current data row.
string[] ChildTableName s	An array containing the names of all child tables of the table associated with this data reader.

17.8 Data Distribution Scripts

Content Grabber has built-in support for data distribution by email or to an FTP server when data is exported to file formats such as CSV or Excel. A **Data Distribution** script can be used to customize data distribution or distribute data to an unsupported media. A Data Distribution script could also be used to run some custom functionality after data has been exported. For example, a script could run a stored procedure in a database after data has been exported to the database.

You can add a **Data Distribution** script to an agent by clicking the **Script** button in the **Data** ribbon menu.



The following example executes a stored procedure in a SQL Server database:

```
using System;
using Sequentum.ContentGrabber.Api;
public class Script
```



```

{
    public static bool DeliverData(DataDeliveryArguments args)
    {
        args.GetDatabaseConnection("test").ExecuteNonQuery("exec updateData");
        return true;
    }
}

```

The script must have a static method which has the following signature:

```
public static bool DeliverData(DataDeliveryArguments args)
```

The function should return **True** if it succeeds and **False** if it fails.

An instance of the **DataDeliveryArguments** class is provided by Content Grabber and has the following functions and properties:

Property or Function	Description
Agent Agent	The current agent.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
string[] Files	The exported files that need to be distributed.
bool IsDebug	True if the agent is running in debug mode.
void WriteDebug(string debugMessage ,	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if it is called during design time.

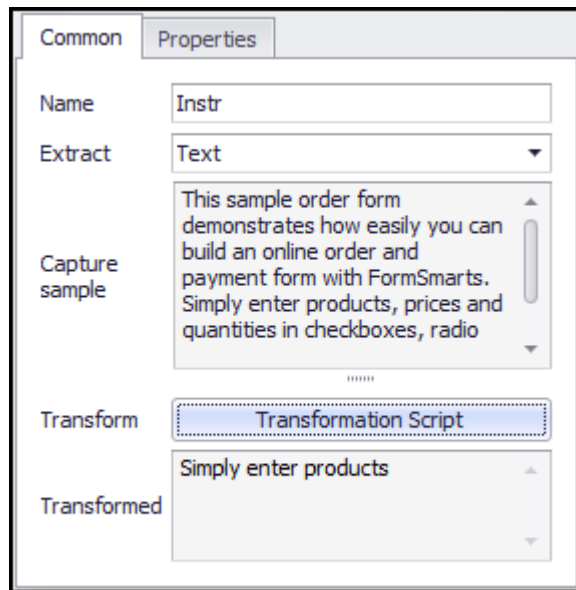
DebugMessage Type messageType = DebugMessage Type.Information on)	
void WriteDebug(string debugMessage , bool showMessageI nDesignMode, DebugMessage Type messageType = DebugMessage Type.Information on)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void Notify(bool alwaysNotify)	Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
void Notify(string message, bool alwaysNotify)	Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a

	notification if the agent has been configured to send notifications on critical errors.
GlobalDataDictionary GlobalData	Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts. Input Parameters are also stored in this dictionary.
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.

17.9 Content Transformation Scripts

Content transformation scripts are used to transform content after it has been extracted from a web page. Content transformation is often used on HTML elements to extract information that is not placed in individual elements and therefore cannot be selected in the web browser. For example, content transformation can be used to extract parts of an address, such as a postal code, from a single HTML element containing the full address.

Content transformation scripts can be used in most Capture Commands to transform the content extracted by the commands, but can also be used in other types of commands, such as in a Navigate Link command to transform an extracted URL.



A content transformation can be defined as a regular expression or as a C# or VB.NET script. Regular expressions are often used when you wish to extract sub-text from a larger piece of extracted text.

The following regular expression example extracts all the text until the first '<' character:

```
(.*?)<
return $1
```

See the topic Script Languages for information about how to use regular expressions in Content Grabber.

The following script also extracts all text until the first '<' character, but uses C# instead of regular expressions:

```
using System;
using Sequentum.ContentGrabber.Api;
public class Script
{
    public static string TransformContent(ContentTransformationArguments args)
```

```

    {
        return args.Content.Remove(args.Content.IndexOf('<'));
    }
}

```

The script must have a static method with the following signature:

```
public static string TransformContent(ContentTransformationArguments args)
```

The function will return the transformed content.

An instance of the **ContentTransformationArguments** class is provided by Content Grabber and has the following functions and properties:

Property or Function	Description
Agent Agent	The current agent.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
Command Command	The current agent command being executed.
IConnection DatabaseConnection	The current internal database connection used by the agent. This connection is already open and should not be closed by your script.
string Content	The extracted content that should be transformed.
IHtmlNode HtmlNode	The extracted HTML node.

InternalDataRow DataRow	The current internal data row containing the data that has been extracted so far in the current container command.
bool IsDebug	True if the agent is running in debug mode.
InputData InputDataCache	All input data available to the current command.
void WriteDebug(string debugMessage, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void WriteDebug(string debugMessage, bool showMessageIn DesignMode, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.

void Notify(bool alwaysNotify)	Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
void Notify(string message, bool alwaysNotify)	Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
GlobalDataDictionary GlobalData	Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts. Input Parameters are also stored in this dictionary.
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.
IInputDataRow GetInputData()	If the current command is a data provider, the data for that command is returned. Otherwise this function

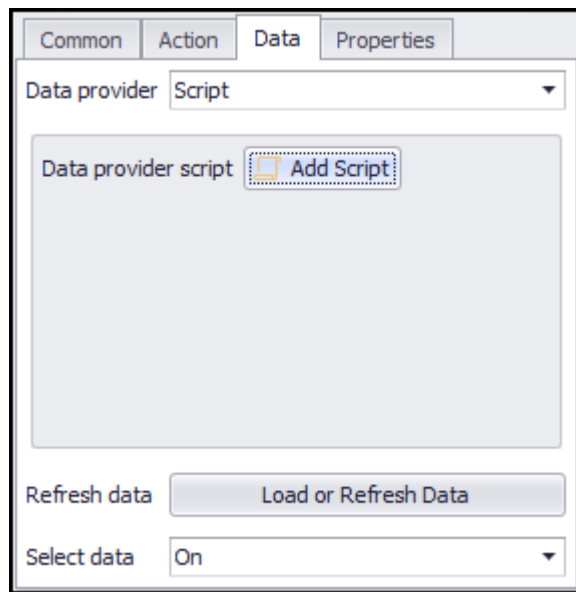
	searches the command's parents and returns the first found input data.
InputDataRow GetInputData(Command command)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
InputDataRow GetInputData(string commandName)	If the specified command is a data provider, the data for that command is returned. Otherwise searches the command's parents and returns the first found input data.
InputDataRow GetInputData(Guid commandId)	If the specified command is a data provider, the data for that command is returned. Otherwise the function throws an error.

17.10 Data Input Scripts

Data Input scripts can be used to generate data for data provider commands. Data provider commands include the following types of commands:

- Agent
- Navigate URL
- Set Form Field
- Data List

A **Data Input** script can be added to a command by selecting the data provider **Script** from the **Data** configuration tab:



The following example generates a list of URLs that could be used in an Agent command to provide start URLs:

```
using System;
using System.Collections.Generic;
using System.Data;
using Sequentum.ContentGrabber.Api;
public class Script
{
    public static DataTable ProvideData(DataProviderArguments args)
    {
        List<string> urls = new List<string>();
        for(int i=1;i<1000;i++)
        {
            urls.Add("http://www.domain.com/page.php?ID="+i.ToString());
        }
        return urls.ToDataTable("url");
    }
}
```

This script makes use of the extension method **ToDataTable** which is part of the Script Utilities.

The script must have a static method with the following signature:

```
public static DataTable ProvideData(DataProviderArguments args)
```

The function must return a **DataTable** that contains the input data. The **DataTable** can have one or more data columns, and all columns will be available to the data provider command that is using the script.

An instance of the **DataProviderArguments** class is provided by Content Grabber and has the following functions and properties:

Property or Function	Description
Agent	The current agent.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
Command Command	The current agent command being executed.
IContainer ParentContainer	The parent container command of the current command.
IConnection DatabaseConnection	The current internal database connection used by the agent. This connection is already open and should not be closed by your script.
IHtmlNode HtmlNode	The extracted HTML node.
IInternalDataRow DataRow	The current internal data row containing the data that has been extracted so far in the current container command.

bool IsDebug	True if the agent is running in debug mode.
bool IsSchemaOnly	If true, only the data schema is required, so you can optimize processing by only returning the data schema with no data.
InputData InputDataCache	All input data available to the current command.
void WriteDebug(string debugMessage, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void WriteDebug(string debugMessage, bool showMessageIn DesignMode, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.

void Notify(bool alwaysNotify)	Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
void Notify(string message, bool alwaysNotify)	Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
GlobalDataDictionary GlobalData	Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts. Input Parameters are also stored in this dictionary.
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.
IInputDataRow GetInputData()	If the current command is a data provider, the data for that command is returned. Otherwise this function

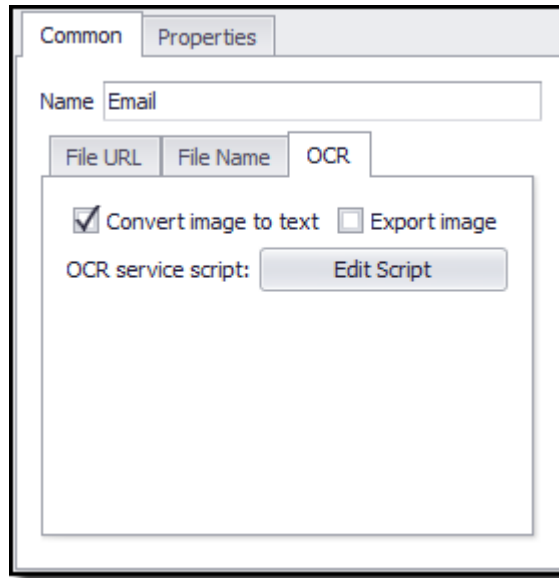
	searches the command's parents and returns the first found input data.
InputDataRow GetInputData(Command command)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
InputDataRow GetInputData(string commandName)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
InputDataRow GetInputData(Guid commandId)	If the specified command is a data provider, the data for that command is returned. Otherwise the function throws an error.

17.11 Image OCR Scripts

An Image OCR script is used to convert an image into plain text. This is often used when a website uses a CAPTCHA page to block automated access to the website. A CAPTCHA page contains an image with numbers and characters that a user must recognize and enter in plain text. Some websites also display emails and phone numbers as images, and an Image OCR script can be used to convert those images into plain text, making the data more useful.

Content Grabber does not include any OCR functionality, so you must use a 3rd party OCR service. The Image OCR script is used to integrate with a 3rd party service.

An Image OCR script can be added to a Download Image command by selecting the OCR configuration tab and setting the option **Convert image to text**.



The following two examples show how to integrate with two popular CAPTCHA recognition services:

```
public static string ConvertImageToText(ConvertImageToTextArguments args)
{
    string captcha = DeathByCaptchaService.DecodeCaptcha(args.Image, "login",
"password");
    return captcha;
}
```

The script above integrates with the 3rd party service
<http://www.deathbycaptcha.com>:

```
public static string ConvertImageToText(ConvertImageToTextArguments args)
{
    string captcha = BypassCaptchaService.DecodeCaptcha(args.Image, "key");
    return captcha;
}
```

The script above integrates with the 3rd party service <http://bypasscaptcha.com/>

The Image OCR script must have a static function with the following signature:

```
public static string ConvertImageToText(ConvertImageToTextArguments args)
```

The function must return the converted image as a string.

An instance of the **ConvertImageToTextArguments** class is provided by Content Grabber and has the following functions and properties:

Property or Function	Description
byte[] Image	The image that needs to be converted to text.
Agent Agent	The current agent.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
Command Command	The current agent command being executed.
IContainer ParentContainer	The parent container command of the current command.
IConnection DatabaseConnection	The current internal database connection used by the agent. This connection is already open and should not be closed by your script.
IHtmlNode HtmlNode	The extracted HTML node.
IInternalDataRow DataRow	The current internal data row containing the data that has been

	extracted so far in the current container command.
bool IsDebug	True if the agent is running in debug mode.
bool IsSchemaOnly	If true, only the data schema is required, so you can optimize processing by only returning the data schema with no data.
IInputData InputDataCache	All input data available to the current command.
void WriteDebug(string debugMessage, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void WriteDebug(string debugMessage, bool showMessageInDesignMode, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.

void Notify(bool alwaysNotify)	Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
void Notify(string message, bool alwaysNotify)	Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
GlobalDataDictionary GlobalData	Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts. Input Parameters are also stored in this dictionary.
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.
IInputDataRow GetInputData()	If the current command is a data provider, the data for that command is returned. Otherwise this function

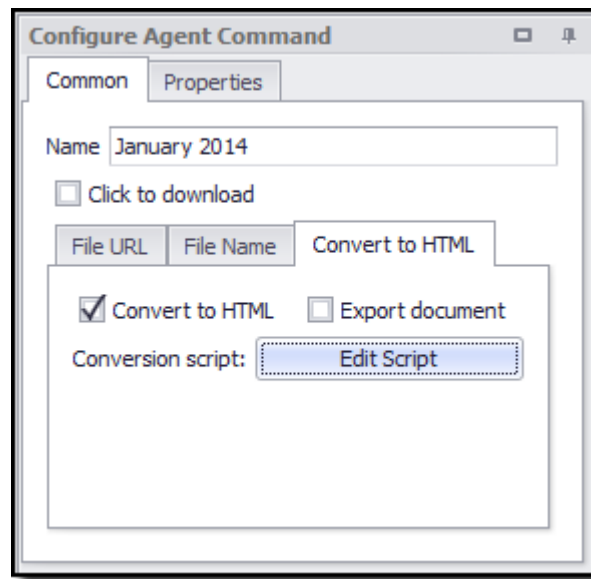
	searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(Command command)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(string commandName)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(Guid commandId)	If the specified command is a data provider, the data for that command is returned. Otherwise the function throws an error.

17.12 Convert Document to HTML Scripts

A **Convert Document to HTML** script is used to convert a downloaded document into a HTML page, so Content Grabber can extract data from the document the same way as for any other HTML page.

Please see the topic [Extracting Data From Non-HTML Documents](#) for more information.

A **Convert Document to HTML** script can be added to a Download Document command by selecting the **Convert to HTML** configuration tab and setting the option **Convert to HTML**:



The following example is the default **Convert Document to HTML** script. This script checks the type of the downloaded document, and uses an appropriate document converter to convert the document to a HTML page:

```
using System;
using System.IO;
using Sequentum.ContentGrabber.Api;
public class Script
{
    public static bool ConvertDocumentToHtml(ConvertDocumentToHtmlArguments args)
    {
        if(args.DocumentType=="pdf")
            ScriptUtils.ExecuteCommandLine(@"Converters\pdftohtml\pdftohtml.exe",
                args.DocumentFilePath, args.HtmlFilePath, "-noframes");
        else if(args.DocumentType=="docx")
            ScriptUtils.ExecuteCommandLine(@"Converters\docxtohtml\docxtohtml.exe",
                args.DocumentFilePath, args.HtmlFilePath, "");
        if(!File.Exists(args.HtmlFilePath))
            return false;
        return true;
    }
}
```

The function should return **True** if the conversion succeeds or **False** if the conversion failed.

An instance of the **ConvertDocumentToHtmlArguments** class is provided by Content Grabber and has the following functions and properties:

Property or Function	Description
string DocumentFilePath	The path of the document that needs to be converted to HTML.
string DocumentType	The type of document that needs to be converted to HTML. For example, if the document is a PDF document, the document type will be pdf.
string HtmlFilePath	The file path the script should use for the converted HTML file.
Agent Agent	The current agent.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
Command Command	The current agent command being executed.
IContainer ParentContainer	The parent container command of the current command.
IConnection DatabaseConnection	The current internal database connection used by the agent. This

	connection is already open and should not be closed by your script.
IHtmlNode HtmlNode	The extracted HTML node.
IInternalDataRow DataRow	The current internal data row containing the data that has been extracted so far in the current container command.
bool IsDebug	True if the agent is running in debug mode.
bool IsSchemaOnly	If true, only the data schema is required, so you can optimize processing by only returning the data schema with no data.
IInputData InputDataCache	All input data available to the current command.
void WriteDebug(string debugMessage, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.

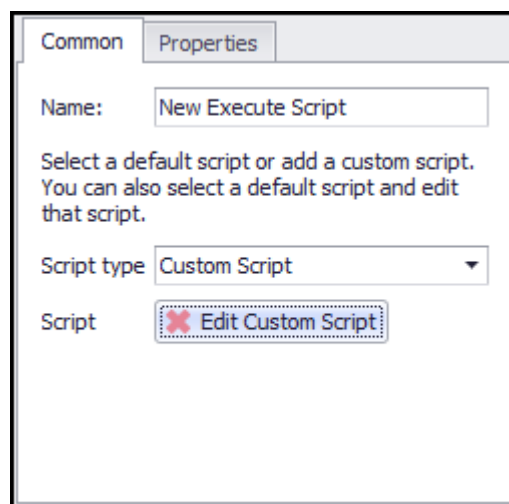
void WriteDebug(string debugMessage , bool showMessageIn DesignMode, DebugMessage Type messageType = DebugMessage Type.Information on)	<p>Writes log information to the agent log.</p> <p>This method has no effect if agent logging is disabled, or if called during design time.</p>
void Notify(bool alwaysNotify)	<p>Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.</p>
void Notify(string message, bool alwaysNotify)	<p>Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.</p>
GlobalDataDictionary GlobalData	<p>Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts.</p>

	Input Parameters are also stored in this dictionary.
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.
IInputDataRow GetInputData()	If the current command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(Command command)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(string commandName)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(Guid commandId)	If the specified command is a data provider, the data for that command is returned. Otherwise the function throws an error.

17.13 Custom Scripts

A custom script is the script used by an Execute Script command to execute custom functionality. The Execute Script command provides a list of predefined scripts that can be used directly or edited to suit specific needs. Please see the Execute Script topic for more information about how to use the predefined scripts.

A custom script can be added to an Execute Script command by setting the **Script type** to **Custom Script**:



The following example writes extracted data to a database. Normally a Data Export script would be used to write extracted data after an agent has completed, but an Execute Script command could be used to write data as it's being extracted:

```
using System;
using Sequentum.ContentGrabber.Api;
using Sequentum.ContentGrabber.Commands;
public class Script
{
    public static CustomScriptReturn CustomScript(CustomScriptArguments args)
    {
```



```

        IConnection connection = args.GetDatabaseConnection("exportDatabase");
        connection.OpenDatabase();
        try
        {
            ICommand command = connection.GetNewCommand();
            command.SetSql("insert into export_table values (@title,
@description)");
            command.AddParameterWithValue("title", args.DataRow["title"],
CaptureDataType.ShortText);
            command.AddParameterWithValue("description",
args.DataRow["description"], CaptureDataType.ShortText);
            command.ExecuteNonQuery();
        }
        finally
        {
            connection.CloseDatabase();
        }
        return CustomScriptReturn.Empty();
    }
}

```

The above script uses the *IConnection* and *ICommand* interfaces which are part of Content Grabber's Script Utilities.

The script must have a static method with the following signature.

```
public static CustomScriptReturn CustomScript(CustomScriptArguments args)
```

The script must return an instance of a class **CustomScriptReturn**. This class has the following 4 public static methods:

Method	Description
static CustomScriptReturn RetryContainer(IContainer container)	The agent will retry the specific container command. The container command must be a parent of the current command.

static CustomScriptReturn ExitContainer(IContainer container)	The agent will exit the specific container command. The container command must be a parent of the current command.
static CustomScriptReturn Pause()	The agent pauses and displays an agent web browser, which allows a user to interact with the web browser before continuing processing.
static CustomScriptReturn Empty()	The agent continues its normal executing flow.

An instance of the **CustomScriptArguments** class is provided by Content Grabber and has the following functions and properties:

Property or Function	Description
Agent Agent	The current agent.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
Command Command	The current agent command being executed.
IContainer ParentContainer	The parent container command of the current command.

ICconnection DatabaseConnection	The current internal database connection used by the agent. This connection is already open and should not be closed by your script.
IHtmlNode HtmlNode	The extracted HTML node.
IInternalDataRow DataRow	The current internal data row containing the data that has been extracted so far in the current container command.
bool IsDebug	True if the agent is running in debug mode.
bool IsSchemaOnly	If true, only the data schema is required, so you can optimize processing by only returning the data schema with no data.
IInputData InputDataCache	All input data available to the current command.
void WriteDebug(string debugMessage, DebugMessageType messageType = DebugMessageType.Infor mation)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void WriteDebug(string debugMessage, bool showMessageInDesignMo de, DebugMessageType	Writes log information to the agent log. This method has no effect if agent logging is

messageType = DebugMessageType.Information)	disabled, or if called during design time.
void Notify(bool alwaysNotify)	Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
void Notify(string message, bool alwaysNotify)	Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
GlobalDataDictionary GlobalData	Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts. Input Parameters are also stored in this dictionary.
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined

	for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.
IInputDataRow GetInputData()	If the current command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(Command command)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(string commandName)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(Guid	If the specified command is a data provider, the data for

commandId)

that command is returned.
Otherwise the function
throws an error.

17.14 Condition Scripts

A condition script can be used by commands that use a condition to determine a particular outcome, such as commands that control execution flow.

The following script is an example of a condition script:

```
using System;
using Sequentum.ContentGrabber.Api;
using Sequentum.ContentGrabber.Commands;
public class Script
{
    public static bool ConditionScript(ConditionScriptArguments args)
    {
        if(args.GlobalData["test"].ToString().Length() == 10)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

The script must have a static method with the following signature.

```
public static bool ConditionScript(ConditionScriptArguments args)
```

The script must return true or false.

An instance of the *ConditionScriptArguments* class is provided by Content Grabber and has the following functions and properties:

Property or Function	Description
Agent Agent	The current agent.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
Command Command	The current agent command being executed.
IContainer ParentContainer	The parent container command of the current command.
IConnection DatabaseConnection	The current internal database connection used by the agent. This connection is already open and should not be closed by your script.
IHtmlNode HtmlNode	The extracted HTML node.
IInternalDataRow DataRow	The current internal data row containing the data that has been extracted so far in the current container command.
bool IsDebug	True if the agent is running in debug mode.
bool IsSchemaOnly	If true, only the data schema is required, so you can optimize processing by only returning the data schema with no data.

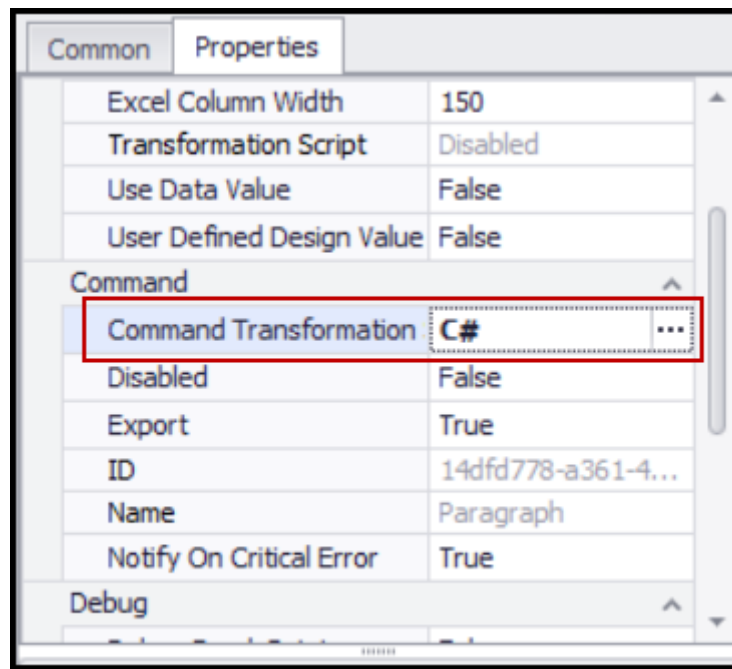
InputData InputDataCache	All input data available to the current command.
void WriteDebug(string debugMessage, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void WriteDebug(string debugMessage, bool showMessageInDesignMode, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void Notify(bool alwaysNotify)	Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
void Notify(string message, bool alwaysNotify)	Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a notification if the agent has

	been configured to send notifications on critical errors.
GlobalDataDictionary GlobalData	<p>Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts.</p> <p>Input Parameters are also stored in this dictionary.</p>
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.
IInputDataRow GetInputData()	<p>If the current command is a data provider, the data for that command is returned.</p> <p>Otherwise this function searches the command's parents and returns the first found input data.</p>
IInputDataRow GetInputData(Command	If the specified command is a data provider, the data for

command)	that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(string commandName)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
IInputDataRow GetInputData(Guid commandId)	If the specified command is a data provider, the data for that command is returned. Otherwise the function throws an error.

17.15 Command Transformation

A Command Transformation script can be used to change command properties at runtime. It's only safe to modify agent commands in Command Transformation scripts and Agent Initialization scripts. It's not safe to modify sub-commands in a Command Transformation script. You should only modify the current command.



Command Transformation is available from the command Properties tab.

The following example sets the selection XPath of the current command:

```
using System;
using Sequentum.ContentGrabber.Api;
using Sequentum.ContentGrabber.Commands;
public class Script
{
    public static bool TransformCommand(CommandTransformationScriptArguments args)
    {
        ISelection selection = args.Command as ISelection;
        selection.Selection.SelectionPaths[0].XPath =
            "//div[@id='body']/section[1]/p[1]";
        return true;
    }
}
```

The script must have a static method with the following signature.

```
public static bool TransformCommand(CommandTransformationScriptArguments args)
```

The function must return true for success and false for failure.

An instance of the **CommandTransformationScriptArguments** class is provided by Content Grabber and has the following functions and properties:

Property or Function	Description
Agent Agent	The current agent.
ScriptUtils ScriptUtilities	A script utility class with helper methods. See Script Utilities for more information.
Command Command	The current agent command being executed. You can change the properties of this command, but not on any sub-commands.
IContainer ParentContainer	The parent container command of the current command.
IConnection DatabaseConnection	The current internal database connection used by the agent. This connection is already open and should not be closed by your script.
IHtmlNode HtmlNode	The extracted HTML node.
IInternalDataRow DataRow	The current internal data row containing the data that has been extracted so far in the current container command.
bool IsDebug	True if the agent is running in debug mode.
bool IsSchemaOnly	If true, only the data schema is required, so you can optimize processing by only

	returning the data schema with no data.
InputData InputDataCache	All input data available to the current command.
void WriteDebug(string debugMessage, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void WriteDebug(string debugMessage, bool showMessageIn DesignMode, DebugMessageType messageType = DebugMessageType.Information)	Writes log information to the agent log. This method has no effect if agent logging is disabled, or if called during design time.
void Notify(bool alwaysNotify)	Triggers notification at the end of an agent run. If alwaysNotify is set to false, this method only triggers a notification if

	the agent has been configured to send notifications on critical errors.
void Notify(string message, bool alwaysNotify)	Triggers notification at the end of an agent run, and adds the message to the notification email. If alwaysNotify is set to false, this method only triggers a notification if the agent has been configured to send notifications on critical errors.
GlobalDataDictionary GlobalData	Global data dictionary that can be used to store data that needs to be available in all scripts and after agent restarts. Input Parameters are also stored in this dictionary.
IConnection GetDatabaseConnection(string connectionName)	Returns the specified database connection. The database connection must have been previously defined for the agent or be a shared connection for all agents on the computer. Your script is responsible for opening and closing the connection by calling the OpenDatabase and CloseDatabase methods.
IInputDataRow GetInputData()	If the current command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.

InputDataRow GetInputData(Command command)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
InputDataRow GetInputData(string commandName)	If the specified command is a data provider, the data for that command is returned. Otherwise this function searches the command's parents and returns the first found input data.
InputDataRow GetInputData(Guid commandId)	If the specified command is a data provider, the data for that command is returned. Otherwise the function throws an error.

18 Programming Interface

The Content Grabber programming interface (API) provides access to the Content Grabber runtime from your own applications. The Content Grabber runtime can be distributed with your applications royalty free and does not require the Content Grabber application to be installed on the target computer. The Content Grabber runtime requires .NET version 4.5 or higher.

If you want to run agents from a web application, you must use the Content Grabber proxy API. The proxy API interfaces with the Content Grabber runtime using a **Windows** service. The **Windows** service is installed as part of the Content Grabber application, so Content Grabber must be installed on the web server.

You can also use simple web requests to call the Windows service without the use of the Content Grabber proxy API. This means that you can easily execute agents and retrieve extracted data from non-Windows environments, such as from a PHP page on a Linux server.

In this chapter, you can learn more about Building a Desktop Application and Building a Web Application.

API Restrictions

The API cannot change the agent command structure, or alter the agent in a way that would change the output data structure. You cannot add, delete, move or copy commands, and you cannot change the **Disabled** and **Export** command properties.

18.1 Building a Desktop Application

The Content Grabber application can build stand-alone agents. This is a convenient way to distribute agents that can be configured and run without requiring the full Content Grabber application on the target computer. However, stand-alone agents have a standardized user interface and can only export data to file formats. If you

want full control of the user interface and export features, you can build your own desktop application using the Content Grabber API.

See these topics for more information:

- Visual Studio Configuration
- Distributing Your Application

Examples

The following example uses the API to run an agent with a set of input parameters. It sets the log level to high and specifies that log information should be written to file.

```
AgentApi api = new AgentApi(@"C:\Users\Public\Documents\Content Grabber\Agents\
  qantasApiTest\qantasApiTest.scg");
AgentSettings settings = new AgentSettings();
settings.InputParameters.Add("from", "SYD");
settings.InputParameters.Add("to", "MEL");
settings.InputParameters.Add("departure_date", DateTime.Now.ToString("yyyy-MM-dd"));
settings.InputParameters.Add("return_date", DateTime.Now.ToString("yyyy-MM-dd"));
settings.InputParameters.Add("travel_class", "ECO");
settings.InputParameters.Add("adults", "1");
settings.InputParameters.Add("children", "0");
settings.LogLevel = AgentLogLevel.High;
settings.IsLogToFile = true;
api.RunAgent(settings);
```

The following example uses the API to run an agent asynchronously and checks the agent for progress.

```
AgentApi api = new AgentApi(@"C:\Users\Public\Documents\Content
  Grabber\Agents\test\test.scg");
api.StartAgent();

AgentStatus status = api.GetAgentStatus();
while(status.IsRunning())
{
    Thread.Sleep(2000);
```

```
        status = api.GetAgentStatus();
    }
    if (status.IsCompletedWithFailure())
    {
        Console.WriteLine("Failure");
    }
    else
    {
        Console.WriteLine(api.GetAgentDataAsJson());
    }
}
```

Sending Messages to the Host Application

An agent that runs asynchronously on the same computer as the host application can send custom messages to the host application. The agent and the host application run in different processes, so inter-process communication is required for the agent to send messages to the host application. The Content Grabber API provides inter-process communication functionality that is based on named pipes.

The following example uses the API method `StartPipe` to start a named pipe, and then waits for messages to arrive from agent. The agent will always send the message **Completed** when the agent has completed its run. All other messages must be sent from custom scripts in the agent.

```
using (var api = new AgentApi(@"C:\Users\Public\Documents\Content
Grabber\Agents\test\test.scg"))
{
    var pipe = api.StartPipe();
    api.StartAgent();
    while (true)
    {
        var message = pipe.GetNextMessage();

        if (message == null)
            continue;

        if (message.MessageType == CgMessageType.Completed)
        {
            break;
        }
        Console.WriteLine(message.Message);
    }
}
```

```
}
```

The following **Execute Script** command sends the currently extracted data entry to the host application.

```
using System;
using Sequentum.ContentGrabber.Api;
public class Script
{
    public static CustomScriptReturn CustomScript(CustomScriptArguments args)
    {
        args.ScriptUtilities.SendMessage(CgMessageType.Data,
args.DataRow.ToJson());

        return CustomScriptReturn.Empty();
    }
}
```

Agent API Functions

Function	Description
AgentApi(string agentNameOrPath, string sessionId)	<p>Instantiates a new API class with the specified agent and session ID. You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p>C:\Users\Public\Documents\Content Grabber\Agents</p>

AgentApi(string agentNameOrPath)	<p>Instantiates a new API class without a session. You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p>C:\Users\Public\Documents\Content Grabber\Agents</p>
AgentApi(string key, string agentName, string sessionId)	<p>Instantiates a new API class with the specified agent, session ID and access key. The agentName parameter should be the agent name without the full path. Content Grabber will use the Access Key to determine the agent's full path.</p> <p>The sessionId can be set to null if no sessionId is required.</p>
void Connect(string endPointAddress)	<p>Connects to a Content Grabber agent service. You can specify the server name or IP address and port number. The default connection string for a local service is:</p> <p>http://localhost:8000/ContentGrabber</p>

void CloseConnection()	Closes the connection to the Content Grabber agent service.
void StartAgent()	Runs the agent specified when instantiating the API. The agent will run asynchronously.
void StartAgent(AgentSettings settings)	Runs the agent with additional settings. The agent will run asynchronously. See below for more information about the <i>AgentSettings</i> class.
void StopAgent()	Stops the agent if it is currently running.
void CloseAgentSession()	<p>Closes an agent session. When you close an agent session, all data associated with that session is removed and you will not be able to retrieve status information about the agent that ran in this session. You can only close a session if an agent is not currently running in the session.</p> <p>You don't need to close a session. Session data will be removed automatically after the agent has completed running and the session timeout has elapsed. The default session timeout is 30 minutes, so by default session data will be removed automatically 30 minutes after the agent has completed.</p>

AgentStatus GetAgentStatus()	Returns status information about an agent that has been run asynchronously. See below for more information about the AgentStatus class.
DataTable GetAgentProgressAsDataTable()	Returns progress information in a DataTable about an agent running in asynchronously. See below for more information about the information returned.
DataTable GetAgentProgressAsJson()	Returns progress information as a JSON string about an agent running in asynchronously. See below for more information about the information returned.
DataTable GetAgentLogsDataTable(offset, limit)	<p>Returns log data in a DataTable for an agent that has been run asynchronously. This function does not return any data if logging is disabled or if logging is written to file. See below for more information about the information returned.</p> <p>offset (optional): Index of the first log entry to return.</p> <p>Limit (optional): Index of the last log entry to return.</p>
string GetAgentLogsAsJson(offset, limit)	Returns log data as a JSON string for an agent that has been run asynchronously. This function does not return any data if logging is disabled or if logging is written to file. See below for

	<p>more information about the information returned.</p> <p>offset (optional): Index of the first log entry to return.</p> <p>Limit (optional): Index of the last log entry to return.</p>
DataSet GetAgentExportDataAsDataSet(offset, limit)	<p>Returns extracted data in a DataSet for an agent that has been run asynchronously.</p> <p>offset (optional): Index of the first data entry to return.</p> <p>Limit (optional): Index of the last data entry to return.</p>
string GetAgentExportDataAsJson(offset, limit)	<p>Returns extracted data as a JSON string for an agent that has been run asynchronously.</p> <p>offset (optional): Index of the first data entry to return.</p> <p>Limit (optional): Index of the last data entry to return.</p>
string GetAgentExportDataAsXml(offset, limit)	<p>Returns extracted data as an XML string for an agent that has been run asynchronously.</p> <p>offset (optional): Index of the first data entry to return.</p> <p>Limit (optional): Index of the last data entry to return.</p>

RunAgentReturnJson(string agentNameOrPath, limit)	<p>Runs an agent synchronously and returns extracted data as a JSON string. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>Limit (optional): Maximum number of data rows to return.</p>
RunAgentReturnXml(string agentNameOrPath, limit)	<p>Runs an agent synchronously and returns extracted data as an XML string. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>Limit (optional): Maximum number of data rows to return.</p>
RunAgentReturnDataSet(string agentNameOrPath, limit)	<p>Runs an agent synchronously and returns extracted data in a DataSet. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>Limit (optional): Maximum number of data rows to return.</p>
RunAgentReturnJson(AgentSe	Runs an agent synchronously with additional settings and returns extracted

RunAgentReturnJson(settings, limit)	<p>data as a JSON string. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>See below for more information about the AgentSettings class.</p> <p>Limit (optional): Maximum number of data rows to return.</p>
RunAgentReturnXml(AgentSettings settings, limit)	<p>Runs an agent synchronously with additional settings and returns extracted data as an XML string. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>See below for more information about the AgentSettings class.</p> <p>Limit (optional): Maximum number of data rows to return.</p>
RunAgentReturnDataSet(AgentSettings settings, limit)	<p>Runs an agent synchronously with additional settings and returns extracted data in a DataSet. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p>

	See below for more information about the AgentSettings class. Limit (optional): Maximum number of data rows to return.
Agent GetAgent()	Returns the agent specified when instantiating the API class.
SaveAgent(Agent agent)	Saves the specified agent.
CgPipeIn StartPipe()	Starts a pipe that can be used to send messages from a running agent to the host application. The pipe must be started before the agent is started.

AgentSettings

The following agent settings can be specified when running an agent:

Property	Description
bool IsViewBrowser	The web browser is displayed while the agent runs. This option has no effect when an agent is being run from a Windows service.
bool IsNoUi	No User Interfaces will be displayed while an agent runs, and the web browser will not render web content on the screen.
AgentRunMethod RunMethod	Specifies how to run the agent. Restart, Continue, or Continue And Retry Errors

string ScheduleUserName	The full user name, including domain name, used when retrieving schedule information.
string SchedulePassword	The password used when retrieving schedule information.
GlobalDataDictionary GlobalData	Any serializable data object can be stored in this dictionary and will be available to all scripts in an agent. Notice that input parameters will eventually be stored in this dictionary as well, so it doesn't matter if you use input parameters or global data to store your input data.
bool LogLevel	Log detail level. Set the log level to None to turn off logging.
bool IsLogHtml	Logs the raw HTML of all web pages processed by the agent.
bool IsLogToFile	Logs data to a file instead of a database table.
string LogFilePath	The log path if logging to a file. The path can be a directory and a specific file. In both cases the directory must exist.
int Timeout	This value specifies the session timeout in minutes when an agent is run asynchronously. All session data is removed automatically when the agent has completed and this

	<p>timeout has elapsed. The default session timeout is 30 minutes.</p> <p>This value specifies the maximum number of seconds an agent will run when it's run synchronously. When the timeout is reached, the agent will stop and close its session if it's run in a session. The default timeout is 30 seconds.</p>
Dictionary<string, string> InputParameters	A list of input parameters.

AgentStatus

An agent can provide the following status information:

Property	Description
PageLoads	<p>The RunStatus can be one of the following values:</p> <ul style="list-style-type: none"> • Completed. The agent has completed successfully. • Incomplete. The agent has completed, but stopped prematurely. The agent may have been stopped manually. • Failed. The agent has completed, but a critical error occurred. • Idle. The agent has never been run. • Starting. The agent is starting.

	<ul style="list-style-type: none"> • ExportingData. The agent is exporting data to the specified export target. • Stopping. The agent is in the process of stopping. • Restarting. The agent is restarting. This usually occurs when the agent needs to clear JavaScript memory leaks. • ExportFailed. The agent completed, but failed to export data.
MissingElements	The number of page loads. This includes AJAX calls triggered by agent actions.
StartTime	The amount of time the agent has run.
int MissingElements	The number of times an agent command could not find it's specified content where the content was not specified as optional.
int PageErrors	The number of page load errors. This includes errors loading content from AJAX calls that were triggered by agent actions.
DateTime StartTime	The time the agent started.

Agent Progress Data

An agent can provide progress data in a **DataTable**. The **DataTable** contains a **DataRow** for each web browser the agent is using to extract data. Each **DataRow**

contains a status column and a description column. The progress data is the same information displayed when running an agent in the Content Grabber agent editor.

Agent Log Data

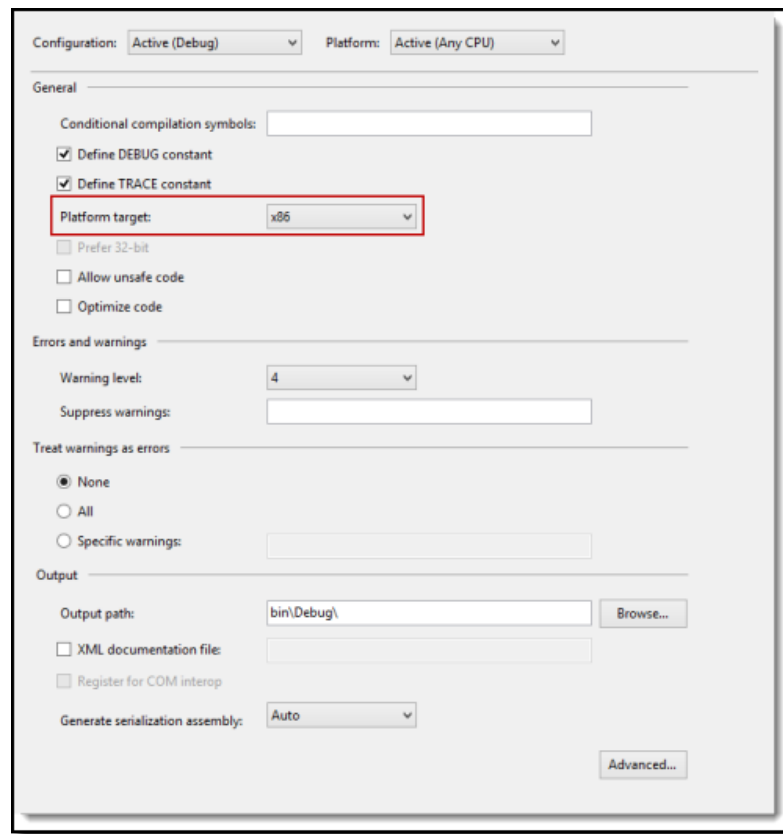
An agent can provide log data in a **DataTable**. The **DataTable** contains a log level column and a description column. A log level of 1 means an error, 2 means a warning and 3 means information. The log data is the same data you can view in the Content Grabber agent editor.

Agent Export Data

The API can provide extracted data in a **DataSet**, as an XML string or as a JSON string. For large amount of data, use the parameters **offset** and **limit** to page through the data. **Offset** is the index of the first data entry to return and **limit** is the index of the last data entry to return. The API method **GetAgentStatus** returns a value **Export Row Count** which contains the total number of data entries available. See Data Counting for more information about the **Export Row Count** value.

18.1.1 Visual Studio Configuration

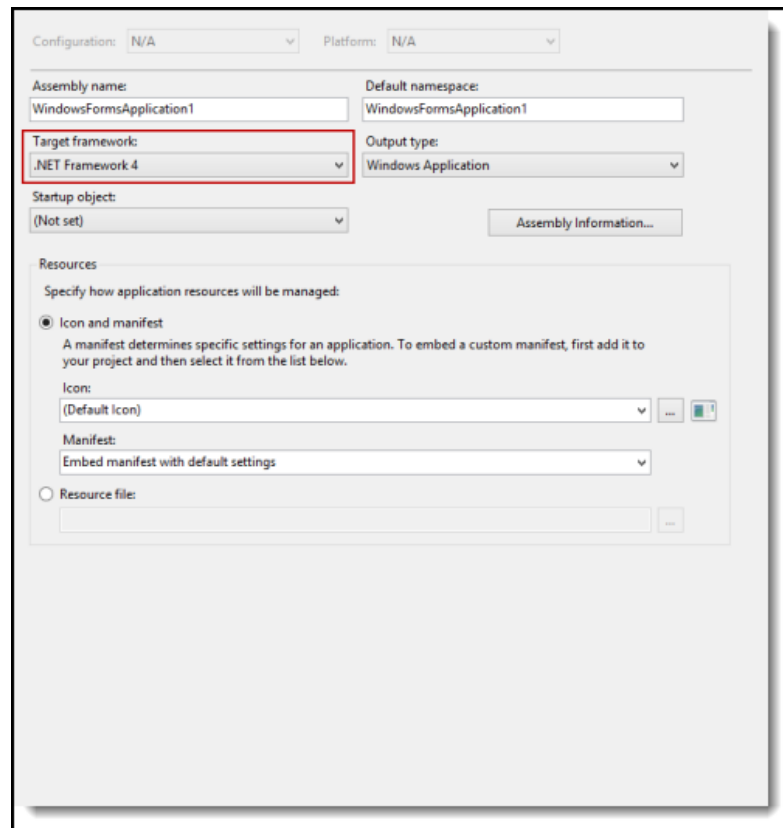
The Content Grabber API comes on both a 32-bit version and a 64-bit version, and since the library will be embedded into your application, you need to select the appropriate platform target for your application. For example, if you're using the 32-bit runtime, your application must be a 32-bit application. You can make sure your application is a 32-bit application by opening project properties in **Visual Studio** and setting the platform target to x86.



If you're using the 64-bit API, set the platform target to x64.

The Content Grabber .NET API files, which you will reference from your Visual Studio project, are compiled with the platform target **Any CPU**, so they will work in both 32-bit and 64-bit applications, but the Content Grabber native runtime files will only work on the chosen platform. If you're building an advanced installer for your application, you could use the platform target **Any CPU**, and then configure your installer to install the appropriate native runtime files. For example, the Content Grabber installer checks the platform of the client computer and then downloads and installs either the 32-bit or 64-bit runtime files depending on the platform.

The Content Grabber API uses the .NET v4.5 framework, so your application should be configured to use the same or a higher version of the .NET framework. To set the .NET framework version in **Visual Studio**, go to project properties in **Visual Studio** and set the target framework to .NET v4.5 or higher.



Assembly References

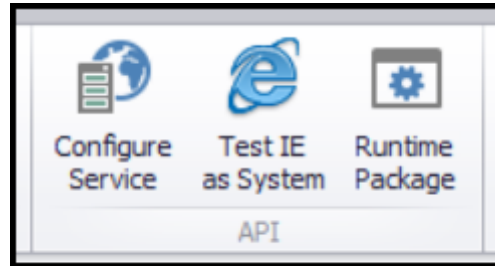
You must add the following references to your project to use the The Content Grabber API:

- AgentApi.dll
- AgentProxy.dll

Once you add these assemblies to your project references, **Visual Studio** will automatically copy these files and some dependent assemblies to your project's BIN folder. However, not all required assemblies will be copied to your BIN folder, so you must manually ensure all the Content Grabber runtime files are in your BIN folder.

The Content Grabber runtime files can be generated in the Content Grabber application by choosing **Runtime Package** in the **Application** menu. This will generate

a zip file with all required files and folders. All these files and folders must be copied to your BIN folder.



18.1.2 Distributing Your Application

NOTICE: This functionality is only available with a **Content Grabber OEM License**.

You can distribute your desktop applications without having to install Content Grabber on the target computers. You just have to make sure the Content Grabber runtime files are in the same folder as your executable program.

The Content Grabber runtime files can be generated in the Content Grabber application by choosing **Runtime Package** in the **Application** menu. This will generate a zip file with all required files and folders. All these files and folders must be copied to the folder of your executable program.

18.2 Building a Web Application

The Content Grabber API can be used to run agents from your own applications. When using the API directly, your application needs security privileges that are standard for desktop applications, but not for web applications. It can be a complex task to ensure a web application has the required security privileges, and you may end up giving the application too many security privileges, making it vulnerable to security breaches.

Important: We don't support using the full API from a web application. We only support the use of the API proxy in web applications. You may be able to use the full API, depending on the security policies on the web server. We are unable to help you configure a web server for this purpose.

We recommend you use the Content Grabber agent service when using the API in web applications. The Content Grabber agent service provides most API functionality through a simple proxy assembly that requires no special security settings and depends on no other files than the single proxy assembly file. The proxy assembly can run in both 32-bit and 64-bit applications, which makes it even easier to use this assembly rather than the full API. Read more in the topic [Using the Content Grabber Agent Service](#).

18.2.1 Using the Content Grabber Agent Service

Content Grabber includes a Windows service that can be used to run agents. Your web application communicates with the Windows service using a small proxy assembly that requires no special security privileges and depends on no other files. You simply add the proxy assembly to your web application's assembly references and use the proxy to call the API functions. The proxy assembly contains the same methods as the standard API, except for functions used to load and save agents. The proxy can only execute agents, not load agents, because the proxy is designed to rely on no other assemblies and loading an agent would require the agent definition classes found in the full Content Grabber API.

Before you can use the proxy to communicate with the **Windows Service**, you need to connect to the service. You use the proxy method **Connect** to connect to the service. The default connection string is:

`http://localhost:8003/ContentGrabber`

If your Content Grabber service is located on a remote server, you can change localhost to the name or IP address of the remote server. The service is listening on port 8003 by default, but you can change the port in the Content Grabber editor. The Content Grabber service is stopped by default and configured to start manually. If you are going to use this service, you should configure the service to start automatically.

All Windows services, including the Content Grabber agent service, run in a special Windows session that cannot interact with users and cannot display user interfaces. JavaScript on some websites does not work correctly without at least a hidden web browser window. Such websites are rare, but if you need to run such a website through the Content Grabber agent service, the service needs to run the agent in a normal user session. In order for the Windows service to start an agent in a normal user session the following three conditions must be met.

1. You must set the agent option **Run Interactively**.
2. The Content Grabber agent service must run under the **System** account.
3. A user must be logged onto the computer while the agent runs. The agent will run in the Windows session of the logged in user, but in the security context of the System account.

Example

The following example connects to a local Content Grabber agent service and runs an agent in a new session. It also sets the log level to high and specifies that log information should be written to file.

```
string sessionId = Guid.NewGuid().ToString();
AgentProxy proxy = new AgentProxy(@"C:\Users\Public\Documents\Content Grabber\
  Agents\qantas\qantas.scg", sessionId);
proxy.Connect("http://localhost:8003/ContentGrabber");
AgentSettings settings = new AgentSettings();
```

```
settings.LogLevel = AgentLogLevel.High;
settings.IsLogToFile = true;
proxy.StartAgent(settings);
```

You can connect to the Content Grabber agent service again at any time to get status information about a specified agent running in a specified session.

```
AgentProxy qantasProxy = new AgentProxy(@"C:\Users\Public\Documents\Content Grabber\
  Agents\qantas\qantas.scg", sessionId);
qantasProxy.Connect("http://localhost:8003/ContentGrabber");
AgentStatus status = qantasProxy.GetAgentStatus();
```

If you are running an agent from a web application, you could use AJAX callbacks to get status information about a running agent. You only have to keep track of the session ID between callbacks.

Proxy Functions

The following functions are available in the proxy assembly:

Function	Description
AgentProxy(string agentNameOrPath , string sessionId)	<p>Instantiates a new proxy with the specified agent and session ID. You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p>

AgentProxy(string agentNameOrPath)	<p>Instantiates a new proxy without a session. You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p>
void Connect(string endPointAddress)	<p>Connects to the Content Grabber agent service. You can specify the server name or IP address and port number. The default connection string for a local service is:</p> <p><i>http://localhost:8000/ContentGrabber</i></p>
void CloseConnection()	<p>Closes the connection to the Content Grabber agent service.</p>
void StartAgent()	<p>Starts the agent specified when instantiating the proxy. The agent will run asynchronously.</p>
void StartAgent(Agent Settings settings)	<p>Starts the agent with additional settings. The agent will run asynchronously. See below for more</p>

	information about the AgentSettings class.
void StopAgent()	Stops the agent if it is currently running asynchronously.
void CloseAgentSession()	<p>Closes an agent session after the agent has been run asynchronously. When you close an agent session, all data associated with that session is removed and you will not be able to retrieve status information about the agent that ran in this session. You can only close a session if an agent is not currently running in the session.</p> <p>You don't need to close a session. Session data will be removed automatically after the agent has completed running and the session timeout has elapsed. The default session timeout is 30 minutes, so by default session data will be removed automatically 30 minutes after the agent has completed.</p>
AgentStatus GetAgentStatus()	Returns status information about an agent that has been run asynchronously. See below for more information about the AgentStatus class.
DataTable GetAgentProgressAsDataTable()	Returns progress information in a DataTable about an agent running in asynchronously. See below for more

	information about the information returned.
DataTable GetAgentProgressAsJson()	Returns progress information as JSON about an agent running in asynchronously. See below for more information about the information returned.
DataTable GetAgentLogAsDataTable(int offset, int limit)	<p>Returns log data in a DataTable for an agent that has been run asynchronously. This function does not return any data if logging is disabled or if logging is written to file. See below for more information about the information returned.</p> <p>offset (optional): Index of the first log entry to return.</p> <p>Limit (optional): Index of the last log entry to return.</p>
string GetAgentLogAsJson(int offset, int limit)	<p>Returns log data as JSON for an agent that has been run asynchronously. This function does not return any data if logging is disabled or if logging is written to file. See below for more information about the information returned.</p> <p>offset (optional): Index of the first log entry to return.</p> <p>Limit (optional): Index of the last log entry to return.</p>

DataSet GetAgentDataAsDataSet(int offset, int limit)	<p>Returns extracted data in a DataSet for an agent that has been run asynchronously.</p> <p>offset (optional): Index of the first data entry to return.</p> <p>Limit (optional): Index of the last data entry to return.</p>
string GetAgentDataAsJson(int offset, int limit)	<p>Returns extracted data as a JSON string for an agent that has been run asynchronously.</p> <p>offset (optional): Index of the first data entry to return.</p> <p>Limit (optional): Index of the last data entry to return.</p>
string GetAgentDataAsXml(int offset, int limit)	<p>Returns extracted data as an XML string for an agent that has been run asynchronously.</p> <p>offset (optional): Index of the first data entry to return.</p> <p>Limit (optional): Index of the last data entry to return.</p>
RunAgentReturnJson(string agentNameOrPath, string sessionId, int limit)	<p>Runs an agent synchronously and returns extracted data as a JSON string. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p>

	<p>Limit (optional): Maximum number of data rows to return.</p>
<p>RunAgentReturnXml(string agentNameOrPath, string sessionId, int limit)</p>	<p>Runs an agent synchronously and returns extracted data as an XML string. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>Limit (optional): Maximum number of data rows to return.</p>
<p>RunAgentReturnDataSet(string agentNameOrPath, string sessionId, int limit)</p>	<p>Runs an agent synchronously and returns extracted data in a DataSet. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>Limit (optional): Maximum number of data rows to return.</p>
<p>RunAgentReturnJson(AgentSettings settings, int limit)</p>	<p>Runs an agent synchronously with additional settings and returns extracted data as a JSON string. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p>

	<p>See below for more information about the AgentSettings class.</p> <p>Limit (optional): Maximum number of data rows to return.</p>
RunAgentReturnXml(AgentSettings settings, int limit)	<p>Runs an agent synchronously with additional settings and returns extracted data as an XML string. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>See below for more information about the AgentSettings class.</p> <p>Limit (optional): Maximum number of data rows to return.</p>
RunAgentReturnDataSet(AgentSettings settings, int limit)	<p>Runs an agent synchronously with additional settings and returns extracted data in a DataSet. The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>See below for more information about the AgentSettings class.</p>

	Limit (optional): Maximum number of data rows to return.
--	---

AgentSettings

The following agent settings can specified when running an agent:

Property	Description
bool LogLevel	Log detail level. Set the log level to None to turn off logging.
bool IsLogHtml	Logs the raw HTML of all web pages processed by the agent.
bool IsLogToFile	Logs data to a file instead of a database table.
int Timeout	<p>This value specifies the session timeout in minutes when an agent is run asynchronously. All session data is removed automatically when the agent has completed and this timeout has elapsed. The default session timeout is 30 minutes.</p> <p>This value specifies the maximum number of seconds an agent will run when it's run synchronously. When the timeout is reached, the agent will stop and close its session if it's run in a session. The default timeout is 30 seconds.</p>
Dictionary<string, string>	A list of input parameters.

InputParameters	
GlobalData	Any serializable data object can be stored in this dictionary and will be available to all scripts in an agent. Notice that input parameters will eventually be stored in this dictionary as well, so it doesn't matter if you use input parameters or global data to store your input data.
ProxyList	A list of web proxies that will be used by the agent. If a proxy list is specified, it overrides any default proxy settings in the agent.

AgentStatus

An agent can provide the following status information:

Property	Description
AgentRunningStatus RunStatus	<p>The RunStatus can be one of the following values.</p> <ul style="list-style-type: none">• Completed. The agent has completed successfully.• Incomplete. The agent has completed, but stopped prematurely. The agent may have been stopped manually.• Failed. The agent has completed, but a critical error occurred.

	<ul style="list-style-type: none">• Idle. The agent has never been run.• Starting. The agent is starting.• ExportingData. The agent is exporting data to the specified export target.• Stopping. The agent is in the process of stopping.• Restarting. The agent is restarting. This usually occurs when the agent needs to clear JavaScript memory leaks.• ExportFailed. The agent completed, but failed to export data.
int PageLoads	The number of page loads. This includes AJAX calls triggered by agent actions.
TimeSpan Runtime	The amount of time the agent has run.
int MissingElements	The number of times an agent command could not find it's specified content where the content was not specified as optional.
int PageErrors	The number of page load errors. This includes errors loading content from AJAX calls that were triggered by agent actions.
DateTime StartTime	The time the agent started.

Agent Progress Data

An agent can provide progress data in a **DataTable**. The **DataTable** contains a **DataRow** for each web browser the agent is using to extract data. Each **DataRow** contains a status column and a description column. The progress data is the same information displayed when running an agent in the Content Grabber agent editor.

Agent Log Data

An agent can provide log data in a **DataTable**. The **DataTable** contains a log level column and a description column. A log level of 1 means an error, 2 means a warning and 3 means information. The log data is the same data you can view in the Content Grabber agent editor.

Agent Export Data

The API can provide extracted data in a **DataSet**, as an XML string or as a JSON string. For large amount of data, use the parameters **offset** and **limit** to page through the data. **Offset** is the index of the first data entry to return and **limit** is the index of the last data entry to return. The API method **GetAgentStatus** returns a value **Export Row Count** which contains the total number of data entries available. See Data Counting for more information about the **Export Row Count** value.

18.2.2 Using Simple Web Requests

The Content Grabber Windows service supports simple web requests, so you can execute agents and retrieve extracted data from non-Windows environments, such as from a PHP page on a Linux server.

The Windows service is listening for web requests on port 8004 by default. You can change this port number in the Content Grabber editor. The service is stopped by default and configured to start manually. If you are going to use this service, you should configure the service to start automatically. Please see

Using the Content Grabber Agent Service for more information about the Windows service.

This is an example of a web request that executes an agent named Sequentum and provides some input values for the agent.

```
http://localhost:8004/ContentGrabber/RunAgentReturnJson?  
agent=sequentum&pars={"StartDate":"2015-10-15","EndDate":"2015-12-15"}
```

The above web request executes the agent synchronously and returns the extracted data as a JSON string.

Web Request Functions

The following functions are available when using web requests:

Function	Description
StartAgent? agent={agentNameOrPath} &sessionId={sessionId} &sessionTimeout={sessionTimeout} &runMethod={runMethod} &logLevel={logLevel} &logHtml={isLogHtml} &logToFile={isLogToFile}	<p>Starts an agent that will run asynchronously.</p> <p>This function supports both GET and POST requests. If you need to specify a long list of input parameters you must use POST requests, since GET requests are limited in length.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will</p>

&pars={inputParameters}
&key={key}

look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:

C:\Users\Public\Documents\Content Grabber\Agents

sessionId (optional): The agent will run in a session with the specified session ID.

sessionTimeout (optional): If the agent is running in a session, this value specifies the number of minutes the agent data will be available before it's deleted. The default session timeout is 30 minutes.

runMethod (optional): The run method specifies how an agent should be run, and must be one of the following values:

- Restart
- Continue
- ContinueAndRetryErrors

logLevel (optional): Log detail level. Set the log level to None to turn off logging. The default log level is None. Accepted values are None, Low, Medium and High.

	<p>logHtml (optional): Logs the raw HTML of all web pages processed by the agent. The default value is False.</p> <p>logToFile (optional): Logs data to a file instead of a database table. The default value is False.</p> <p>Pars (optional): A JSON formatted list of input values that can be used by the agent. The JSON string should be URL encoded.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>RunAgentReturnJson? agent={agentNameOrPath} &sessionId={sessionId} &runMethod={runMethod} &timeout={timeout}</p>	<p>Runs an agent synchronously and returns the extracted data as a JSON string.</p> <p>The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p>

&logLevel={logLevel}
&logHtml={isLogHtml}
&logToFile={isLogToFile}
&pars={inputParameters}
&limit={limit}
&key={key}

This function supports both GET and POST requests. If you need to specify a long list of input parameters you must use POST requests, since GET requests are limited in length.

This function accepts the following parameters:

agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:

C:\Users\Public\Documents\Content Grabber\Agents

runMethod (optional): The run method specifies how an agent should be run, and must be one of the following values:

- Restart
- Continue
- ContinueAndRetryErrors

Timeout (optional): This maximum number of seconds an agent will run. When the timeout is reached, the

agent will stop and close its session if it's run in a session. The default timeout is 30 seconds.

logLevel (optional): Log detail level. Set the log level to None to turn off logging. The default log level is None. Accepted values are None, Low, Medium and High.

logHtml (optional): Logs the raw HTML of all web pages processed by the agent. The default value is False.

logToFile (optional): Logs data to a file instead of a database table. The default value is False.

Pars (optional): A JSON formatted list of input values that can be used by the agent. The JSON string should be URL encoded.

Limit (optional): The maximum number of results to return.

Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the **agentNameOrPath** parameter

	should specify the agent name only without a path.
RunAgentReturnXML? agent={agentNameOrPath} &sessionId={sessionId} &runMethod={runMethod} &timeout={timeout} &logLevel={logLevel} &logHtml={isLogHtml} &logToFile={isLogToFile} &pars={inputParameters} &limit={limit} &key={key}	<p>Runs an agent synchronously and returns the extracted data as an XML string.</p> <p>The agent is always run in a session when the agent supports sessions, and the session is closed automatically after the agent has completed its run.</p> <p>This function supports both GET and POST requests. If you need to specify a long list of input parameters you must use POST requests, since GET requests are limited in length.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p>

runMethod (optional): The run method specifies how an agent should be run, and must be one of the following values:

- Restart
- Continue
- ContinueAndRetryErrors

Timeout (optional): This maximum number of seconds an agent will run. When the timeout is reached, the agent will stop and close its session if it's run in a session. The default timeout is 30 seconds.

logLevel (optional): Log detail level. Set the log level to None to turn off logging. The default log level is None. Accepted values are None, Low, Medium and High.

logHtml (optional): Logs the raw HTML of all web pages processed by the agent. The default value is False.

logToFile (optional): Logs data to a file instead of a database table. The default value is False.

Pars (optional): A JSON formatted list of input values that can be used

	<p>by the agent. The JSON string should be URL encoded.</p> <p>Limit (optional): The maximum number of results to return.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>StopAgent? agent={agentNameOrPath} &sessionId={sessionId}&key={key}</p>	<p>Stops the agent if it is currently running asynchronously.</p> <p>This function supports only GET requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p>

	<p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p> <p>sessionId (optional): The agent session you want to stop if the agent runs in sessions.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>CloseAgentSession? agent={agentNameOrPath} &sessionId={sessionId}&key={key}</p>	<p>Closes an agent session after the agent has been run asynchronously. When you close an agent session, all data associated with that session is removed and you will not be able to retrieve status information about the agent that ran in this session. You can only close a session if an agent is not currently running in the session.</p> <p>You don't need to close a session. Session data will be removed automatically after the agent has completed running and the session timeout has elapsed. The default session timeout is 30 minutes, so by default session data will be removed</p>

automatically 30 minutes after the agent has completed.

This function supports only GET requests.

This function accepts the following parameters:

agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:

C:\Users\Public\Documents\Content Grabber\Agents

sessionId: The agent session you want to close.

Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the **agentNameOrPath** parameter should specify the agent name only without a path.

GetAgentProgressAsJson?
agent={agentNameOrPath}
&sessionId={sessionId}&key={key}

Returns progress information as JSON about an agent running in a asynchronously. See below for more information about the information returned.

This function supports only GET requests.

This function accepts the following parameters:

agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:

C:\Users\Public\Documents\Content Grabber\Agents

sessionId (optional): The agent session if the agent runs in sessions.

Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the **agentNameOrPath** parameter

	should specify the agent name only without a path.
GetAgentLogAsJson? agent={agentNameOrPath} &sessionId={sessionId} &offset={offset} &limit={limit} &key={key}	<p>Returns log data as JSON for an agent that has been run asynchronously. This function does not return any data if logging is disabled or if logging is written to file. See below for more information about the information returned.</p> <p>This function supports only GET requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p> <p>sessionId (optional): The agent session if the agent runs in sessions.</p> <p>offset (optional): Index of the first log entry to return.</p>

	<p>Limit (optional): Index of the last log entry to return.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>GetAgentDataAsJson? agent={agentNameOrPath} &sessionId={sessionId} &offset={offset} &limit={limit} &key={key}</p>	<p>Returns extracted data as a JSON string for an agent that has been run asynchronously.</p> <p>This function supports only GET requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p>

	<p>sessionId (optional): The agent session if the agent runs in sessions.</p> <p>offset (optional): Index of the first data entry to return.</p> <p>Limit (optional): Index of the last data entry to return.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>GetAgentDataAsXML? agent={agentNameOrPath} &sessionId={sessionId} &offset={offset} &limit={limit} &key={key}</p>	<p>Returns extracted data as an XML string for an agent that has been run asynchronously.</p> <p>This function supports only GET requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will</p>

	<p>look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p> <p>sessionId (optional): The agent session if the agent runs in sessions.</p> <p>offset (optional): Index of the first data entry to return.</p> <p>Limit (optional): Index of the last data entry to return.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>GetInfoForAllAgentsAsJson? directory={directory} &userName={userName}</p>	<p>Returns all information about all agents in a specified directory. Username and password of a Windows user is required to retrieve scheduling information.</p> <p>This function supports only GET requests.</p>

&password={password}&key={key}

Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. When using the Windows Task Scheduler a username and password of a Windows user is required to retrieve scheduling information. If username and password is not specified, this function returns scheduling information from the internal scheduler.

This function accepts the following parameters:

directory (required): The root directory of all agents to retrieve information from.

userName (required): The user name of a Windows account that can access scheduled tasks.

password (required): The password of a Windows account that can access scheduled tasks.

Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a

	<p>key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>GetAgentInfoAsJson? agent={agentNameOrPath} &sessionId={sessionId} &userName={userName} &password={password}&key={key}</p>	<p>Returns all information about a specified agent. Username and password of a Windows user is required to retrieve scheduling information.</p> <p>This function supports only GET requests.</p> <p>Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. When using the Windows Task Scheduler a username and password of a Windows user is required to retrieve scheduling information. If username and password is not specified, this function returns scheduling information from the internal scheduler.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will</p>

	<p>look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p> <p>sessionId (optional): The agent session if the agent runs in sessions.</p> <p>userName (required): The user name of a Windows account that can access scheduled tasks.</p> <p>password (required): The password of a Windows account that can access scheduled tasks.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>GetAgentScheduleInfoAsJson? agent={agentNameOrPath} &sessionId={sessionId}</p>	<p>Returns scheduling information about a specified agent.</p> <p>Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. When</p>

onId}
&userName={user
Name}
&password={pass
word}&key={key}

using the Windows Task Scheduler a username and password of a Windows user is required to retrieve scheduling information. If username and password is not specified, this function returns scheduling information from the internal scheduler.

This function supports only GET requests.

This function accepts the following parameters:

agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:

C:\Users\Public\Documents\Content Grabber\Agents

sessionId (optional): The agent session if the agent runs in sessions.

userName (required): The user name of a Windows account that can access scheduled tasks.

	<p>password (required): The password of a Windows account that can access scheduled tasks.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>GetAgentRuntimeInfoAsJson? agent={agentNameOrPath} &sessionId={sessionId}&key={key}</p>	<p>Returns runtime information about a specified agent.</p> <p>This function supports only GET requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p>

	<p>sessionId (optional): The agent session if the agent runs in sessions.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>SetAgentSchedule ?</p> <p>agent={agentNameOrPath}</p> <p>&isScheduleEnabled={isScheduleEnabled}</p> <p>&interval={interval}</p> <p>&intervalType={intervalType}</p> <p>&nextRunTime={nextRunTime}</p> <p>&logLevel={logLevel}</p> <p>&isLogToFile={isLogToFile}</p> <p>&userName={userName}</p>	<p>Sets the schedule of a specified agent. Username and password of a Windows user is required to set a schedule.</p> <p>Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. This function adds a schedule using the Windows Task Scheduler.</p> <p>This function supports both GET and POST requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify</p>

&password={password}
&isRunAsCurrentUser={isRunAsCurrentUser}
&isRunOnlyWhenUserLoggedIn={isRunOnlyWhenUserLoggedIn}
&key={key}

the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:

C:\Users\Public\Documents\Content Grabber\Agents

isScheduleEnabled (required):

Specifies if the scheduled task should be added or removed. If the agent is already scheduled, the scheduled task will be replaced if this parameter is **true** and removed if the parameter is **false**.

interval (required): The scheduled interval. This must be a positive integer value.

intervalType (required): The type of interval. Accepted values are Seconds, Minutes, Hours or Days.

nextRunTime (required): The first time to run the scheduled task.

logLevel (required): Log detail level. Set the log level to None to turn off logging. The default log level is None. Accepted values are None, Low, Medium and High.

logHtml (required): Logs the raw HTML of all web pages processed by the agent. The default value is False.

logToFile (required): Logs data to a file instead of a database table. The default value is False.

userName (required): The password of a Windows account that can access scheduled tasks.

password (required): The password of a Windows account that can access scheduled tasks.

isRunAsCurrentUser (required): This parameter should be set to false, and a user name and password of an existing Windows user should be used instead.

isRunOnlyWhenUserLoggedIn (required): Specifies if the agent should only run when the specified Windows user is logged in. This parameter should normally be set to **false**.

Key (optional): The access key used to determine which agents can be accessed if the Content Grabber

	<p>API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>AddAgentSchedule? agent={agentNameOrPath} &sessionId={sessionId} &logLevel={logLevel} logFilePath={logFilePath} &cron={cron} &startTime={startTime} &inputParameters={inputParameters}&key={key}</p>	<p>Adds a schedule to a specified agent.</p> <p>Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. This function adds a schedule using the internal scheduler.</p> <p>This function supports both GET and POST requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p>

sessionId (optional): The agent will run in a session with the specified session ID.

logLevel (required): Log detail level. Set the log level to None to turn off logging. The default log level is None. Accepted values are None, Low, Medium and High.

logFilePath (optional): Logs to this file path. Logs to a default file path if this parameter is not specified.

cron (required): A Cron expression that specifies when to run the agent. See Cron Expressions for more information.

startTime (required): The first time to run the agent.

inputParameters (optional): JSON formatted input parameters that will be provided to the agent.

Example:

```
{"par1":"value1","par2":"value2"}
```

Key (optional): The access key used to determine which agents can

	<p>be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>UpdateAgentSchedule? agent={agentNameOrPath} &sessionId={sessionId} &scheduleId={scheduleId} &logLevel={logLevel} &logFilePath={logFilePath} &cron={cron} &startTime={startTime} &inputParameters={inputParameters}&key={key}</p>	<p>Updates a schedule for a specified agent.</p> <p>Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. This function updates a schedule in the internal scheduler.</p> <p>This function supports both GET and POST requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p>

sessionId (optional): The agent will run in a session with the specified session ID.

scheduleId (required): The ID of the schedule to update. The ID can be retrieved from a call to `GetAgentScheduleInfoAsJson`.

logLevel (required): Log detail level. Set the log level to None to turn off logging. The default log level is None. Accepted values are None, Low, Medium and High.

logFilePath (optional): Logs to this file path. Logs to a default file path if this parameter is not specified.

cron (required): A Cron expression that specifies when to run the agent. See Cron Expressions for more information.

startTime (required): The first time to run the agent.

inputParameters (optional): JSON formatted input parameters that will be provided to the agent.

	<p>Example:</p> <pre>{"par1":"value1","par2":"value2"}</pre> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>UpdateOrAddAgentSchedule? agent={agentNameOrPath} &sessionId={sessionId} &logLevel={logLevel} logFilePath={logFilePath} &cron={cron} &startTime={startTime} &inputParameters={inputParameters} &key={key}</p>	<p>Updates a schedule for the specified agent if there's exactly one schedule matching the agent path and session ID. Adds a schedule to the agent if there are no schedules matching the agent path and session ID. Returns an error if there's more than one schedule matching the agent path and session ID.</p> <p>Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. This function updates or adds a schedule to the internal scheduler.</p> <p>This function supports both GET and POST requests.</p>

This function accepts the following parameters:

agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:

C:\Users\Public\Documents\Content Grabber\Agents

sessionId (optional): The agent will run in a session with the specified session ID.

scheduleId (required): The ID of the schedule to update. The ID can be retrieved from a call to `GetAgentScheduleInfoAsJson`.

logLevel (required): Log detail level. Set the log level to `None` to turn off logging. The default log level is `None`. Accepted values are `None`, `Low`, `Medium` and `High`.

	<p>logFilePath (optional): Logs to this file path. Logs to a default file path if this parameter is not specified.</p> <p>cron (required): A Cron expression that specifies when to run the agent. See Cron Expressions for more information.</p> <p>startTime (required): The first time to run the agent.</p> <p>inputParameters (optional): JSON formatted input parameters that will be provided to the agent.</p> <p>Example:</p> <pre>{"par1":"value1","par2":"value2"}</pre> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
DeleteAgentSchedule?key={key}&agent={agentNameOrPath}	Deletes a schedule for a specified agent.

&scheduleId={scheduleId}

Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. This function deletes a schedule from the internal scheduler.

This function supports both GET and POST requests.

This function accepts the following parameters:

agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:

C:\Users\Public\Documents\Content Grabber\Agents

scheduleId (required): The ID of the schedule to delete. The ID can be retrieved from a call to `GetAgentScheduleInfoAsJson`.

Key (optional): The access key used to determine which agents can be accessed if the Content Grabber

	<p>API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
DeleteAgentSchedules?key={key}&agent={agentNameOrPath}&sessionId={sessionId}	<p>Deletes all schedules for a specified agent with the specified session ID.</p> <p>Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. This function deletes a schedule from the internal scheduler.</p> <p>This function supports both GET and POST requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p>

	<p>sessionId (optional): The session ID of the schedules to delete.</p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
DeleteAllAgentSchedules?key={key}&agent={agentNameOrPath}	<p>Deletes all schedules for a specified agent.</p> <p>Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. This function deletes a schedule from the internal scheduler.</p> <p>This function supports both GET and POST requests.</p> <p>This function accepts the following parameters:</p> <p>agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent</p>

	<p>service. The default agent location for the local System account is:</p> <p><i>C:\Users\Public\Documents\Content Grabber\Agents</i></p> <p>Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the agentNameOrPath parameter should specify the agent name only without a path.</p>
<p>RunAgentScheduleNow?key={key}&agent={agentNameOrPath}&sessionId={sessionId}&scheduleId={scheduleId}</p>	<p>Runs a schedule for a specified agent with the specified session ID and schedule ID. If no schedule ID is specified, an error is returned if there is more than one schedule that matches the agent and session ID.</p> <p>Content Grabber can schedule agents using its internal scheduler or the Windows Task Scheduler. This function deletes a schedule from the internal scheduler.</p> <p>This function supports both GET and POST requests.</p> <p>This function accepts the following parameters:</p>

agent (required): You can specify the full path to an agent file or just the name of the agent. If you only specify the agent name, Content Grabber will look for the agent in the default location for the user running the agent service. The default agent location for the local System account is:

C:\Users\Public\Documents\Content Grabber\Agents

sessionId (optional): The session ID of the schedule to run.

scheduleId (optional): The ID of the schedule to run. The ID can be retrieved from a call to `GetAgentScheduleInfoAsJson`.

Key (optional): The access key used to determine which agents can be accessed if the Content Grabber API is restricted by access key. If a key is specified, the **agentNameOrPath** parameter should specify the agent name only without a path.

18.3 Sessions

Sessions allow you to run multiple instances of the same agent at the same time. This

is particularly useful when running an agent directly from a website where you can have many web users visiting the website at the same time. Each web user can start the agent in their own session and the user will only see agent progress status and extracted data belonging to his session.

Running an Agent in a Session

To run an agent in a session, you must specify a session ID when you run the agent and the agent must be configured to support sessions. To configure an agent to support sessions, set the agent option **Support Sessions** to **Multiple Sessions** in the *Sessions* section on the **Advanced** options tab.

A session ID can be specified when running an agent via the Agent API or the Agent Proxy. You can use any string as session ID, but each session must have a unique ID, so you would normally use a unique identifier. The following C# code snippet runs an agent with a unique session ID:

```
string sessionId = Guid.NewGuid().ToString();  
AgentApi agent = new AgentApi("sequentum", sessionId);  
agent.StartAgent(new AgentSettings());
```

A session ID can also be specified when running an agent from the commandline by using the command-line option *session_id*. The following command line input runs an agent named **sequentum** with a session ID *ry37456r*.

```
RunAgent.exe "sequentum" session_id "ry37456r"
```

Session Data Cleanup

When running an agent normally without sessions, the agent can cleanup previously extracted data and status information before it starts because only data and status from the last run is important. Data cleanup becomes much more complex when running agent sessions. An agent session should only cleanup data from its own session, so it cannot do a general cleanup of all data belonging to the agent. Since all

sessions are normally new sessions there is no previous session data to cleanup, and the agent cannot cleanup data when it completes a session since you'll need access to the data for some time, so session data risks hanging around forever.

You can manually trigger a session cleanup by calling the API method **CloseSession**, but in a web application you may not always be able to tell when a user session ends. When running an agent in a session you can specify a session timeout, and the session cleanup will start automatically after the agent session completes and the specified timeout has passed. This gives your application a minimum amount of time to use the session data before it's removed.

An agent only cleans up externally exported session data if the agent is configured to export data to a database. If you are using an **Export Script** or if you are exporting to a file format, then you are responsible for any cleanup of exported session data.

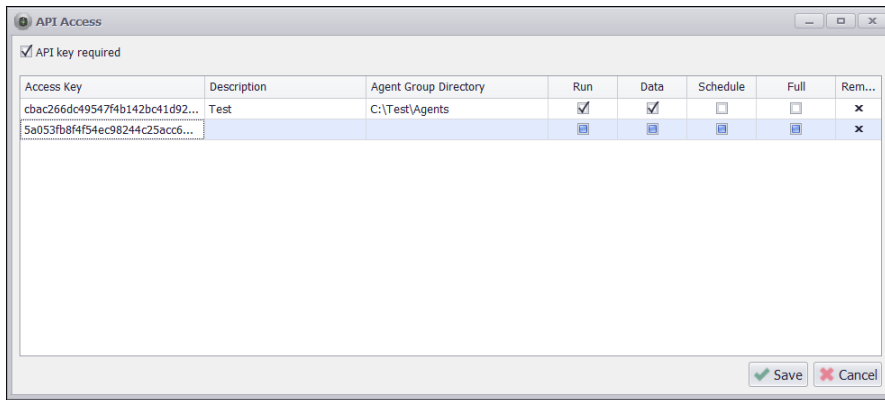
You may not always want to remove externally exported session data automatically when an agent session ends. To prevent session data from being removed, set the agent option **Cleanup External Session Data** to false. This option can be found in the **Sessions** section on the advanced options tab.

If you are not using the API to retrieve extracted data, but are only working with externally exported session data, then you can set the option **Remove Session** to **Immediately** to remove internal session data immediately after it has been externally exported. This will reduce the size and increase performance of the internal database. Session status information is not removed and will still be available until the session timeout has passed.

18.4 API Access Keys

The Content Grabber API can be restricted to certain groups of agents. An access key can be mapped to a directory of agents, and the key will then need to be specified with all API calls to access agents in that directory.

The API Access screen can be opened from the Application menu in the Content Grabber editor.



The option **API key required** can be set to require a key for all API access.

A key can be assigned the following access to agents.

Run - the key provides access to run agents and view agent statuses.

Data - the key provides access to data extracted by agents.

Schedule - the key provides access to schedule agents.

Full - the key provides full access to agents.

If an access key is specified with an API call, Content Grabber will determine the agent's path from the key, so only the agent's name should be specified, not the full agent path.

Example:

*`http://localhost:8004/ContentGrabber/RunAgentReturnJson?
key=cbac266dc49547f4b142bc41d929eb0f&agent=linkedin`*

19 Website Automation

Content Grabber can navigate through a website, submit web forms, and upload and download files, so it's perfectly suited for web automation tasks. One example of such a task is shown in the following blog post:

Automating File Uploads to Web Sites

This blog post shows how to automatically upload products to a Shopify shopping cart.

Website testing is another example of an important web automation task. Content Grabber provides good logging and notification features, so it's well suited for testing. In addition, it provides a framework for generating simple documentation for website procedures. Please see the topic Website Testing & Documentation for more information about website testing and documentation.

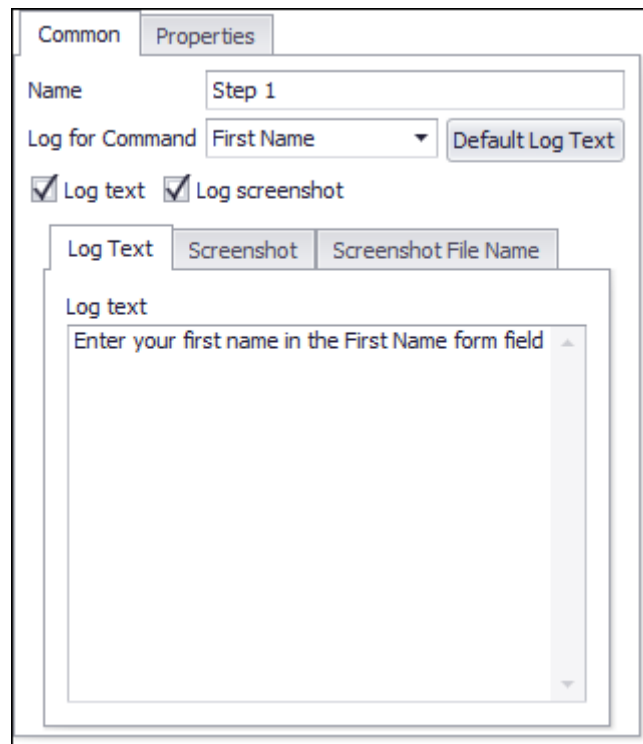
19.1 Website Testing & Documentation

Content Grabber can be used for automated website testing and simple documentation. For example, an agent can be used to test and document a sequence of steps to perform on a website to complete a task, such as submitting a series of web forms. Every time the website is modified, the agent can then be run to ensure the steps are still working in accordance with the documentation. If the agent fails because the sequence of steps required to complete the task has changed, then an administrator can modify the agent to work with the modified procedure and automatically update the documentation for the procedure. If the agent fails because the website is broken, the developer team can be notified to correct the problem.

Configuring an Agent to Generate Documentation

An agent can be configured to generate documentation that consists of a number of steps to complete a procedure. Each step consists of a text description and an optional screenshot of a selected area or the full web page, and a selected web element can be marked with a red border on the screenshot.

Each step in the documentation must be configured individually using a **Screenshot Log** command.

The image shows a software configuration window with two tabs: 'Common' and 'Properties'. The 'Properties' tab is active. It contains a 'Name' field with the value 'Step 1'. Below it is a 'Log for Command' dropdown menu set to 'First Name', with a 'Default Log Text' button to its right. Two checkboxes are checked: 'Log text' and 'Log screenshot'. Below these are three sub-tabs: 'Log Text', 'Screenshot', and 'Screenshot File Name'. The 'Log Text' sub-tab is active, showing a text area with the text 'Enter your first name in the First Name form field'.

Log text configuration tab.

A **Screenshot Log** command can be associated with an action command to specify that the **Screenshot Log** command should provide documentation for that action command. By setting the associated action command, the **Screenshot Log** command can generate default log text, and mark the action command's

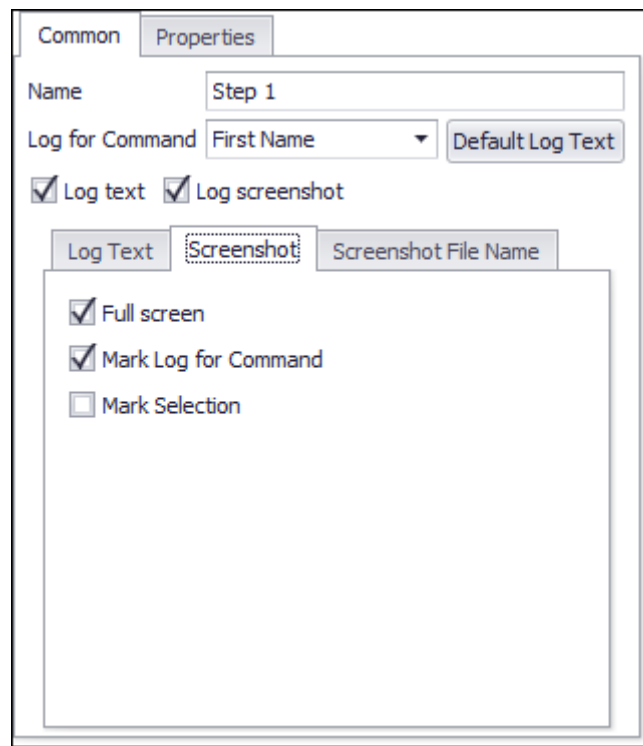
web selection on the screenshot. Log text is the text that describes each step in the documentation.

The **Log Text** tab provides a single input field for log text. The **Default Log Text** button can be used to set default log text if an associated action command has been specified.

Log text can contain special templates to insert extracted data, input data, global data or input parameters.

Data	Syntax
Input data	<code>{ \$command_name field_name }</code> Example: <i>Select { \$Product 7 Default } from the Product 7 select box.</i>
Global data or input parameter	<code>{ \$field_name }</code> Example: <i>Click on the { \$name } button.</i>
HTML attribute of web element selected by the associated action command.	<code>{ \$attribute_name }</code> Example: <i>Click on the { \$value } button.</i>

The **Screenshot** tab specifies where to take a screenshot and what to mark on the screenshot.

The image shows a dialog box titled 'Common Properties'. It has two tabs: 'Common' and 'Properties'. The 'Properties' tab is selected. Inside the 'Properties' tab, there is a 'Name' field with the text 'Step 1'. Below it is a 'Log for Command' dropdown menu with 'First Name' selected, and a 'Default Log Text' button. There are two checked checkboxes: 'Log text' and 'Log screenshot'. Below these are three sub-tabs: 'Log Text', 'Screenshot' (which is selected and highlighted with a dashed border), and 'Screenshot File Name'. Under the 'Screenshot' sub-tab, there are three checkboxes: 'Full screen' (checked), 'Mark Log for Command' (checked), and 'Mark Selection' (unchecked).

Screenshot configuration tab.

The **Screenshot** tab provides the following options.

Option	Desscription
Full screen	Takes a screenshot of the entire web page. If this option is cleared, a screenshot is taken of the web element selected by the Screenshot Log command.
Mark Log for Command	Marks the web element, selected by the associated action command, on the screenshot with a red border.

Mark selection	Marks the web element selected by the Screenshot Log command with a red border. If the option Full screen is cleared, the Screenshot Log command must have two selections. The first selection will be used for the screenshot, and the second selection will be marked on the screenshot.
-----------------------	---

The **Screenshot File Name** tab specifies how to name screenshot files on disk. By default the files are named using a unique identifier, but a data value can be used instead, and Content Transformation can be used to generate a file name.

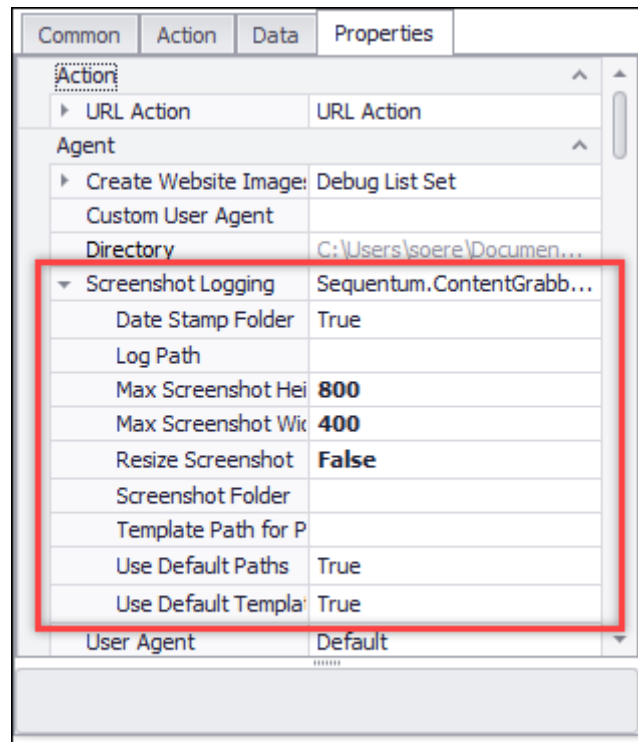
The image shows a software configuration window with two tabs: 'Common' and 'Properties'. The 'Properties' tab is active. It contains the following elements:

- Name:** A text field containing 'Step 1'.
- Log for Command:** A dropdown menu showing 'First Name'.
- Default Log Text:** A button next to the dropdown.
- Log text:** A checked checkbox.
- Log screenshot:** A checked checkbox.
- Log Text, Screenshot, Screenshot File Name:** Three sub-tabs. The 'Screenshot File Name' tab is selected and highlighted with a dashed border.
- Data value:** A checkbox that is currently unchecked.
- File Name:** A large text area containing '[Unique Identifier]'.
- Transformation Script:** A button at the bottom of the 'File Name' section.

Screenshot file name configuration tab.

General Settings

General settings that apply to all **Screenshot Log** commands in an agent are available in the **Screenshot Logging** section of Agent properties.



General Screenshot Log settings.

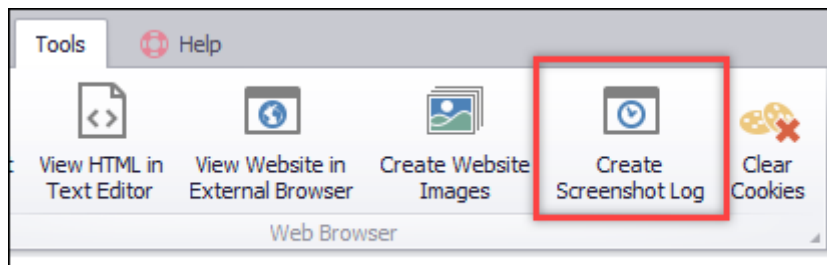
The following general options are available.

Data	Syntax
Date stamp folder	Log file and screenshots will be placed in a sub-folder folder that is named with today's date.
Use default paths	Log file and screenshots will be placed in a sub-folder of the agent folder named ScreenshotLog .

Log path	Specifies the log path if Use default paths is set to false.
Screenshot folder	Specifies the directory where screenshots will be saved if Use default paths is set to false.
Use default template	The log template can be any text file that specifies the layout of the log file. A HTML log template is used by default and is located by default in <i>C:\Program Files\Sequentum\Content Grabber 2\Resources\LogTemplates\ScreenshotLogPage.html</i> .
Template path	Specifies the template path if Use default template is set to false.
Resize screenshot	Resizes the screenshot image files to the size specified by Max Screenshot Width and Max Screenshot Height . If this option is set to false, the default log template will display screenshots resized and provide links to the full sized image files, but will not resize the actual image files.
Max screenshot width	The maximum width of a screenshots.
Max screenshot height	The maximum height of screenshots.

Generating Documentation

Screenshot Log commands are ignored during a normal agent run. To generate the documentation, the agent must be run using the button **Create Screenshot Log** from the **Tools** menu.



Press the **Create Screenshot Log** button to generate documentation.

Index

- & -

&OPEN 195

- (-

(percentage) 157

- . -

.NET class library 343
 .NET functions 337
 .NET v4.0 client profile framework 434
 .NET version 4 420

- ? -

??Headers= 142
 ??Post= 142

- 3 -

32-bit library 434

- A -

Action Command Error Handling 205
 Action Commands 138
 Action Completed 167
 Action Configuration 152
 Action Events 157
 Action Type 138
 Action Types 153
 Activities 138
 Activity 167
 Activity Status 138
 Activity Type 138
 Add Force Refresh Header 138
 Add Horizontally 248
 Add Horizontally Export Method 248
 Add to Agent Template Library 198
 Add Vertically 248

Add Vertically Export Method 248
 Address Bar 34
 Agent 174, 241
 Agent command 140
 Agent Configuration 34
 Agent Data 241
 Agent Debug Log window 207
 Agent Details 324
 Agent Development 30
 Agent Error Handling 205
 Agent Explorer 30, 34
 Agent Export Data 438
 Agent Initialization Scripts 367
 Agent Log Data 438
 Agent Settings 53, 304
 Agent Templates 198
 AgentSettings 438
 AJAX 167
 Ajax Completed Timeout 138
 Ajax Content Render Delay 138
 Ajax Content Render Delay After Scroll 138
 Ajax Start Timeout 138
 Always Merge Data 248
 Always send notifications when an agent completes a run 207
 alwaysNotify 207
 Anonymous Web Scraping 293
 API 297, 420, 434
 API Reference 438
 API Restrictions 420
 Assembly References 140, 345, 434
 Asynchronous Timeout 138
 Attribute Name 136
 Author Details 140
 Auto Detect File Extension 123
 Automatic activity discovery 167
 Automatic CAPTCHA Configuration 297

- B -

Basic Windows Authentication 195
 Block Known Ad Servers 138
 blur 157
 bool HasColumn(string tableName, string columnName) 346
 bool HasTable(string tableName) 346
 bool Read() 346
 Browser 155
 Browser action type 155

Browser Activities 167, 313
 Browser Activity 313
 Browser Activity Screen 167
 Browser Mode 138, 155
 Browser Target 155
 Building a Self-Contained Agent 321
 Building a Web Application 437
 Building Self-Contained Agents 321
 Building Your First Agent 35
 Bypass CAPTCHA 297
 bypasscaptcha.com 393

- C -

C# 337, 338, 367
 Calculated Value 136, 145
 capitalize_words 338
 CAPTCHA 241, 297, 393
 CAPTCHA Blocking 297
 CAPTCHA images 241
 CAPTCHA OCR Scripts 297
 CAPTCHA Protection 241
 Capture Commands 118, 383
 change 157
 Change Export Target 48
 Changing the Default Data Structures 248
 Character Encoding 247
 CityData 230
 Cleanup External Session Data 140, 485
 Clear Cookies 138
 Clear Session 138
 click 157
 click() 157
 Close Browser 181
 CloseSession 485
 Command Library 198
 Command Line Arguments 330
 Command object 346
 Command Properties 178
 Command scripts 337
 Command Templates 198
 Command-line parameters 237
 Company Details 324
 Completed 167
 Condition Scripts 410
 Configuration of an Action Command 138
 Configure Agent Command 30
 Consuming Data 230
 Consuming Data in a Script 230

Container Command Properties 178
 Container Commands 117
 Content Grabber agent service 437
 Content Grabber API 237
 Content Grabber Basics 30
 Content Grabber Message window 30, 38
 Content Grabber Premium Edition 334
 Content Grabber Runtime 8
 Content Transformation Scripts 337, 383
 ContentTransformationArguments 383
 continue 330
 Conversion Script 128
 Convert Document to HTML 398
 Convert Document to HTML Scripts 398
 Convert image to text 297, 393
 Convert to HTML 128
 Convert to Text 123, 132, 134
 ConvertDocumentToHtmlArguments 398
 ConvertImageToTextArguments 393
 Cruise Direct example 38
 CSV 48, 230, 243
 CSV Data Provider 230
 Current 155
 Custom Scripts 404, 414
 Customize Design 324
 Customize Self-Contained Agent 324
 Customizing the User Interface 324

- D -

Data 223
 Data Consumer 137, 142, 149
 Data Distribution 380
 Data Distribution Scripts 380
 Data Export 371
 Data Export script 337
 Data Export Scripts 371
 Data Input 388
 Data Input Scripts 388
 Data List 174, 175
 Data List command 230
 Data Missing Action 175
 Data output 48
 Data Outputs 30
 Data Provider 140, 175
 Data Providers 230
 Data Source 243
 Data Value 137, 241
 Database 230

Database Connections 140, 223
 Database Data Providers 230
 database types 48
 Database Utilities 346
 DataDeliveryArguments 380
 DataExportArguments 371
 DataProviderArguments 388, 404, 414
 DataRow[] GetTableColumns(string tableName) 346
 DataTable LoadCsvFile(string path) 346
 DataTable LoadCsvFile(string path, char separator) 346
 DataTable LoadCsvFileFromDefaultInputFolder(string filename) 346
 DataTable LoadCsvFileFromDefaultInputFolder(string filename, char separator) 346
 DateTime GetDateTimeValue(int columnIndex) 346
 Death by CAPTCHA 297
 deathbycaptcha.com 393
 Debug 50
 Debug Disabled 207
 Debugger 50
 debugging 50
 Default 155, 175, 248
 Default Events 138
 Default Export Method 248
 delay(milliseconds) 157
 DepartureCity 230
 Development Environments 223
 Directory 140
 Discover action 138, 152
 Discover activities 313
 Discover Activities box 167
 Discover Activity Timeout 138
 Discover First Activity Timeout 138
 distributed royalty-free 321
 Distributing Data 223, 257
 Distributing Your Application 437
 Docx To HTML 216
 Docx To HTML Converter 216
 down arrow 157
 down arrow key 157
 Download Document 128, 195
 Download Document Command 149
 Download Image 123
 Download Screenshot 132, 134
 Download Video 126
 Dynamic Page Change 167
 Dynamic Page Changes 167
 Dynamic Websites 18

- E -

Editor Action 138
 Editor Index 175
 Else 410
 Elself 410
 Email and FTP distribution 257, 260, 261, 268
 Email Delivery 257, 260, 261, 268
 Email Distribution 257, 260, 261, 268
 Email Notification 140
 Error Handling 138, 204
 Error Handling In An Execute Script Command 205
 Error Logs 207
 Error logs and notifications 204
 Error Notifications 207
 Error Retry Count 138
 Events 138
 Excel 48
 exec(JavaScript) 157
 executable file 328
 Execute Script 181, 297, 404, 414
 Execute Script Commands 205
 Exit Codes 330
 Exit Command 138, 205, 297
 Export Agent 321
 Export Converted Document 128
 Export Converted Image 123, 132, 134
 export data 224
 Export last data segment only 224
 Export Rows in Parent Columns 248
 Export Rows to Parent Columns 248
 Export Target 140, 247
 Export URL 123
 Exporting and Importing Template Libraries 364
 Exporting and Importing User Libraries 198
 Exporting Data 243
 Exporting Data to MS Access 243
 Exporting Data to Oracle 243
 Exporting Data with Scripts 256
 Exporting Downloaded Images and Files 246
 Extension Methods 346
 Extension scripts 337
 external assemblies 345
 External Data 223, 224
 Externals 167
 Extracting Content From a Converted Document 216
 Extracting Data From Complex Websites 13

Extracting Data From Non-HTML Content 13
 Extracting Data From Websites Using Deterrents 13
 Extracting Huge Amounts of Data 13
 Extracting URLs 120

- F -

File Capture 123
 File Download 167, 195
 File Download Action 195
 File Name 123, 126, 128, 132, 134
 File Name Attribute 123
 File Name Transformation Script 123
 File URL 123, 126, 128
 Fire Event 152
 Fire Events 153
 Fixed File Extension 123
 Flash 126
 focus 157
 Follow Pagination command 38
 Frame Completed Timeout 138
 Frames 167
 Free Private Proxy Switch Account 304
 FTP Delivery 257, 260, 261, 268
 FTP Distribution 257, 260, 261, 268
 Full Dynamic Browser 155, 293

- G -

Gateway 304
 Generate Transformation 45
 Group 176, 177
 Group Command Properties 177
 Group Commands 176
 Group Commands in Page Area 178
 Group in Page Area 176, 178
 GUI window 237
 GUID 246
 Guid GetGuidValue(int columnIndex) 346

- H -

Handle Web Browser Dialog 195
 Handle Web Browser Dialogs 149
 How to Configure Proxy Servers 304
 HTML 120
 HTML Attribute 120, 123, 132, 134

HTML Capture 132, 134
 HTML Column 207
 HTML Content 17
 HTML Parser 155, 195
 html_decode 338
 HTTP proxy servers 293, 304

- I -

ICommand 346
 ICommand GetNewCommand() 346
 IConnection 346
 IDataReader GetDataReader() 346
 IExportData 371
 IExportReader 371
 If 410
 If Selection Exists 297
 If-Modified-Since header 138
 iframes 167
 Ignore Command when Data is Missing 175
 Image Conversion Script 123, 132, 134
 Image OCR Scripts 393
 imdb 248
 Import Proxies 304
 Importing Proxy Servers 304
 Improving Agent Performance and Reliability 310
 Initialization Script 140
 Inner HTML 120
 Input data 223
 Input Parameters 140, 237
 Input Parameters with a GUI Window 237
 Insert Template from Library 198
 Installing Content Grabber 28
 int ClearAllBrowserCookies() 346
 int ClearBrowserCookies(string url) 346
 int GetIntValue(int columnIndex) 346
 Interactive 167
 Interactive mode 167
 internal data 223, 224
 Internal Database 140, 310
 Internet Explorer 9 420
 Introduction 8
 IP blocking 297
 IP Blocking & Proxy Servers 304
 IReader 346
 IReader ExecuteReader(string sql, params object[] pars) 346
 IReader GetNewReader(string sql) 346
 IsParentCommandActionError 138, 205

- J -

JQuery 138

- K -

key steps for building a web scraping agent 35

keydown 157

keypress 157

keyup 157

- L -

Limit Number of Scrolls 138

line_breaks 338

List 174

list command 117

List Commands 174

List Count 174

List of Numeric Links 145

List of Text Links 145

List selection mode 38

List Selections 109

List Start Index 174

List/sequence of numeric links 145

List/sequence of text links 145

Lists 109

Load URL 153

Loading 167

log_html 330

log_level 330

log_to_file 330

long ExecuteCount() 346

long ExecuteCount(string sql) 346

long ExecuteCount(string sql, params object[] pars)
346

Low page number threshold 207

- M -

Main Window 34

manual and automatic data extraction 297

Manual CAPTCHA 297

Manual CAPTCHA Configuration 297

Max Inner Prospects 140

Max Memory Usage 140

Max Tree Expansions 140

Max. Page Number Command 145

Max. Repeats 141

Maximum Number of Scrolls 138

Microsoft Excel 243

Microsoft Excel 2003 243

Microsoft Excel 2007 243

Microsoft Word 216

mousedown 157

mouseup 157

Movie List 248

mp4 126

Multiple Columns 248

My Templates 198

MySQL 48, 230, 243, 247

MySQL Character Encoding 247

- N -

Name Command 248

Navigate Directly to Page 145

Navigate Link 195, 383

Navigate Link command 141

Navigate Pagination Command 145

Navigate URL 142, 241

Navigate URL Command 142

New 155

New Command 120

Next Page 145

Next Page link/button 145

No Action 152, 153

No Error Handling 205

no_ui 330

None 243, 248

- O -

object ExecuteScalar() 346

object ExecuteScalar(string sql) 346

object ExecuteScalar(string sql, params object[] pars)
346

object GetConnection() 346

object GetFieldValue(int columnIndex) 346

OCR 123, 132, 134

OCR script 297

OCR scripts 297

OCR service 241

ogg 126

OleDB 48, 230, 243

Only Execute Action on Value Change 149
 Optimized Dynamic Browser 155
 Optimizing Agent Commands 313
 Optimizing Browser Activities 313
 Optional Data 175
 Oracle 48, 230, 243
 Oracle OleDb 243
 Original File Name 123
 Other Commands 181

- P -

Page Attribute 136
 Page Completed Timeout 138
 Page Count 145
 Page Interactive Timeout 138
 Page Load 167
 Page Load Timeout 138
 Page Loading 167
 Page Number Attribute 145
 Page Number Transformation Script 145
 Page Refresh 167
 Page URL 136
 PageNumberAttributeName 145
 Pages 304
 Pagination 38
 Pagination Command Link 145
 Pagination Type 145
 Parent 155
 Password 195
 Pause Agent 297
 Pause Before Restarting 140
 PDF 216
 PDF To HTML 216
 PDF-to-HTML document converter 216
 Polipo 304
 Post-completion events 167
 Posting Data and Custom Header 142
 Premium Edition 8, 334
 Private Proxy Switch 304
 Process Restart Condition 140
 Professional Edition 8
 Programming Interface 420
 promotional message 321
 Proxy 140
 proxy assembly 438
 Proxy Configuration 140
 Proxy Functions 438
 Proxy Gateway 304

Proxy List 304
 Proxy source 304
 Proxy switch 304
 Proxy Verification 304
 Public Provider 327

- R -

Refresh Document 181
 Regular Expressions 20, 338
 Regular Expressions Reference Guide 338
 Remove Internal Session Data Immediately 140, 485
 Remove skipped proxies 304
 Repeat Link Action 141, 157
 Required 138
 Restart Agent and Continue 205
 Restart and Continue Agent 138
 Restart On Memory Usage 140
 Restart On Time Interval 140
 Restart Time Interval 140
 Retry Command 138, 205, 297
 Retry Count 205
 Return 157
 Return key 157
 Reuse Browser Tab 138
 Rotation interval 304
 Run 53
 Run Agent 53
 Run Interactively 438
 Run Your Agent 53
 RunAgent.exe 330
 Running Agents from the Command-Line 330
 Running an Agent in a Session 485
 Runtime 8
 runtime data 358
 Runtime Input Parameters 237

- S -

Schedule 53
 Scheduling 53, 140
 Script 230, 410
 Script Data Provider 230
 Script Languages 338
 Script Library 343
 Script Template Library 364
 Script type 404, 410, 414

- Script Utilities 346
 - Scripting 140, 337
 - Scripting Data Distribution 274
 - Scroll to End of Page 157
 - Scroll Until End of Page 138
 - scroll() 157
 - scroll(percentage) 157
 - Select the Content to Capture 38
 - Selecting an Export Target 243
 - Selection 120, 123, 230
 - Selection Data Provider 230
 - Selection Techniques 19
 - self-contained agent 328
 - Self-Contained Agent Files 321
 - Self-Contained Agents 140
 - Send notification on critical errors 207
 - Send notification on low number of page loads 207
 - sendkey(keycode) 157
 - sendtext() 157
 - sendtext(text) 157
 - Separate 248
 - Separate Export Method 248
 - Separator 248
 - Server Edition 8
 - Session Data Cleanup 485
 - session_id 330
 - session_timeout 330
 - Sessions 140, 485
 - Set Form Field 118, 149, 174, 195, 241
 - Set Form Field Command 149
 - Set Select Box Text 149
 - Set Value 149
 - setinputtext() 157
 - setinputtext(text) 157
 - shared script library 343
 - Simple 230
 - Simple Data Provider 230, 327
 - Single Column 248
 - Single Next Link 145
 - Single Next Page Link 145
 - Skip unavailable proxies 304
 - Specifying Input Parameters on the Command Line 237
 - SQL Server 48, 230, 243
 - SQL Server Express 310
 - SQLite 223
 - SQLite database 224
 - src 120
 - Start 330
 - Start command 330
 - Start URL 34, 37
 - Static Browser 155
 - static DataTable ToDataTable(this List<string> dataRows, string columnName) 346
 - static DataTable ToDataTable(this List<string> dataRows, string columnName, string stringFormat) 346
 - static DataTable ToDataTable(this string stringValue, string columnName) 346
 - static DataTable ToDataTable(this string[] dataRows, string columnName) 346
 - static DataTable ToDataTable(this string[] dataRows, string columnName, string stringFormat) 346
 - Static Parser 311
 - Stop Agent 138, 205
 - string GetStringValue(int columnIndex) 346
 - strip_html 338
 - Sub-Container Export Method 248
 - Sub-container Export Rules 248
 - Sub-Container Merge Method 248
 - Submit 195
 - Submit button 195
 - Support Sessions 140, 485
- ## - T -
- Tag Text 120
 - Target Browser 138, 155
 - Templates 324
 - Test Your Agent 50
 - Text 120
 - The Internal Database 224
 - third-party CAPTCHA recognition service 297
 - Time 304
 - Timeout 138, 167
 - Timeouts 138
 - Timing 167
 - TNS 243
 - to_lower 338
 - to_upper 338
 - ToDataTable 388
 - TOR 304
 - Transform Page 181
 - transformation 45
 - Transformation Script 45
 - Tree Selections 140
 - trim 338
 - Type GetFieldType(int columnIndex) 346

- U -

- unbind(Event) 157
- Unique Database Names on Your Network 223
- Unique ID 120
- Upgrading a Self-Contained Agent 321, 328
- Uploading File 195
- URL 152
- URL Column 207
- URL Match 138
- URL Parameters 142
- url_decode 338
- Use default events 157
- Use Default Input 140
- Use Max. Page Number Command 145
- Use Next Pagination Set 145
- Use Original File Name 123
- Use Page Count 145
- User Agent 140
- User Interface 140
- Username 195
- Using a Self-Contained Agent 328
- Using Data Input 230
- Using Input Data 327
- Using Input Parameters 237
- Using Input Parameters in a Script 237
- Using MS SQL Server as Internal Database 310
- Using the Content Grabber Agent Service 438
- Using the Content Grabber runtime 334
- Using the Free TOR Proxy Network 304
- Using the Static Parser Browser 311
- UTF-8 247

- V -

- Value Command 248
- VB.NET 337, 338, 367
- Verification timeout 304
- Verify proxy before use 304
- Vidalia 304
- Video URL 126
- View Export Data 50
- View Log 207
- view_browser 330
- Visual Studio 343, 434
- Visual Studio Configuration 434
- Visual Studio Solution Template 343

- Visual Web Ripper 20
- void AddParameter(string pName, CaptureDataType type) 346
- void AddParameterWithValue(string pName, object pValue) 346
- void Close() 346
- void CloseDatabase() 346
- void DropTable(string tableName) 346
- void ExecuteCommandLine(string programFilePath, string arguments) 346
- void ExecuteCommandLine(string programFilePath, string inputFilePath, string outputFilePath, string options) 346
- void ExecuteNonQuery() 346
- void ExecuteNonQuery(string sql) 346
- void ExecuteNonQuery(string sql, params object[] pars) 346
- void Lock() 346
- void Notify(bool alwaysNotify = true) 207
- void Notify(string message, bool alwaysNotify = false) 207
- void OpenDatabase() 346
- void Release() 346
- void ResetBrowserSession() 346
- void SetParameterValue(string pName, object pValue) 346
- void SetSql(string sql) 346
- void TruncateTable(string tableName) 346

- W -

- Wait XPath 138
- web applications 437
- Web Element Content 120
- Web Element List 174, 175
- Web Element Selection 167
- Web Forms 195
- Web requests 438, 450
- Web Scraping Limitations 13
- Web Selection Properties 178
- Web-Scraping Techniques 16
- Website Login Forms 195
- What is Web Scraping? 11
- Windows service 438, 450
- Windows Task Scheduler 53
- windowscroll 157
- windowscroll() 157
- Worker Threads 141, 142

- X -

XML 48, 243

XPath 19, 230

XPath Factory 140

 Content Grabber

