

Starting with Convertigo Web Integrator

Quick Guide



TABLE OF CONTENTS

1 Getting Familiar with Convertigo Web Integrator

CWI Purpose 1-1
CWI Concepts and Objects 1-2
General Concept 1-2
CWI Objects 1-2
Definitions 1-3
Connector 1-3
Screen Class 1-4
Transaction 1-5
CWI Studio 1-7
General Presentation 1-7
View Description 1-8
Projects View 1-9
Properties View 1-10
Connector View1-14

2

My First Convertigo Web Integrator Project

Web Integration Project	2-1
Project Description	2-1
Opening the Sample Project from the Studio	. 2-4
Setting Up the Project	2-5



3

Creating a Project and Setting Connector Parameters 2-6
Defining Screen Classes 2-10
Root Screen Class 2-10
Additional Screen Classes 2-15
Case of a Country-Specific Google Search Page - Definition of the googleSearchPageFR Screen Class
Creating an Extraction Rule 2-24
Defining a Transaction
Case of a Country-Specific Google Search Page - Definition of the ongoogleSearchPageFREntry Screen Class Handler
Executing a Transaction

Exposing Projects as Web Services & Presenting Data

Exposing Projects as Web Services	3-1
Exposing a Project as a REST Web Service	. 3-1
Exposing a Project as a SOAP Web Service	. 3-2
XSL AJAX Presentation	3-18



Figure 1 - 1:	CWI exposes any Web application as SOAP or REST Web service	1-1
Figure 1 - 2:	CWI objects	1-3
Figure 1 - 3:	Connectors folder	1-4
Figure 1 - 4:	Screen classes and Inherited screen classes folders	1-4
Figure 1 - 5:	Criterion and inherited criterion	1-5
Figure 1 - 6:	Transactions folder	1-6
Figure 1 - 7:	"On entry" and "on exit" screen class handlers	1-7
Figure 1 - 8:	List of statements	1-7
Figure 1 - 9:	CWI Studio	1-8
Figure 1 - 10:	Formatting of unsaved, inherited and disabled objects in the Projects View	1-10
Figure 1 - 11:	Properties View	1-11
Figure 1 - 12:	Text property field edition	1-11
Figure 1 - 13:	Javascript expression field edition	1-12
Figure 1 - 14:	Text property edition via edition window	1-12
Figure 1 - 15:	Javascript expression window	1-12
Figure 1 - 16:	Edition window	1-13
Figure 1 - 17:	Configuration window	1-13
Figure 1 - 18:	Text property edition via drop-down menu	1-13
Figure 1 - 19:	Selecting a value in the drop-down menu	1-14
Figure 1 - 20:	Connector View (Design tab selected)	1-15
Figure 1 - 21:	Connector View (Output tab selected)	1-16



Figure 1 - 22:	DOM Tree1-17
Figure 1 - 23:	HTML element selected in the Web Browser1-18
Figure 1 - 24:	XPath Evaluator1-18
Figure 2 - 1:	Connection to Google HTTP server2-2
Figure 2 - 2:	Input of a search keyword in Google search field2-2
Figure 2 - 3:	Click on Google Search button2-3
Figure 2 - 4:	Extraction of Web page titles and URLs in XML format2-3
Figure 2 - 5:	Click on the Next button2-4
Figure 2 - 6:	Process stops on Google page with no Next button2-4
Figure 2 - 7:	Launching the New Project wizard2-5
Figure 2 - 8:	Selecting the CWI sample project2-5
Figure 2 - 9:	Sample project in Projects View2-5
Figure 2 - 10:	New Project wizard2-6
Figure 2 - 11:	Selecting a project type2-7
Figure 2 - 12:	Entering a project name2-7
Figure 2 - 13:	Entering a connector name2-8
Figure 2 - 14:	Setting HTTP or HTML connector parameters2-8
Figure 2 - 15:	New project summary window2-9
Figure 2 - 16:	sampleGoogle project appearing in the Projects View
Figure 2 - 17:	DOM Tree and Web Browser2-10
Figure 2 - 18:	Google general search page2-11
Figure 2 - 19:	Detection criterion of the googleWebPages parent screen class2-11
Figure 2 - 20:	Expanding an element to find relevant nodes and attributes in the DOM Tree 2-12
Figure 2 - 21:	Generating XPath from class and text attributes2-12
Figure 2 - 22:	XPath expression of the detection criterion of the googleWebPages screen class 2-13
Figure 2 - 23:	New Screen Class wizard2-13
Figure 2 - 24:	Giving a name to the new screen class2-14
Figure 2 - 25:	New screen class appearing in the Projects View2-14
Figure 2 - 26:	Selection of a detection criterion for the googleSearchPage screen class2-16

Figure 2 - 27:	Attribute nodes of the HTML element selected as detection criterion for the googleSearchPages screen class	2-16
Figure 2 - 28:	googleSearchPage screen class in Inherited screen classes folder	2-17
Figure 2 - 29:	googleSearchPage XPath evaluation on a Google result page	2-18
Figure 2 - 30:	Selection of a detection criterion for the googleResultsPage screen class	2-18
Figure 2 - 31:	Searching for a better detection criterion	2-19
Figure 2 - 32:	Generation of XPath for the googleResultsPage screen class	2-20
Figure 2 - 33:	googleResultsPage screen class in Inherited screen classes folder	2-20
Figure 2 - 34:	Example of country-specific non-anglophone Google search page (France)	2-21
Figure 2 - 35:	Example of country-specific anglophone Google search page (UK)	2-22
Figure 2 - 36:	Selection of a detection criterion for the country-specific screen class	2-23
Figure 2 - 37:	Expanding an element to find relevant attributes in the DOM Tree for the coust specific screen class	intry- 2-23
Figure 2 - 38:	XPath expression of country-specific redirecting link	2-24
Figure 2 - 39:	googleSearchPageFR screen class in Projects View	2-24
Figure 2 - 40:	New Extraction Rule wizard	2-26
Figure 2 - 41:	Giving a name to the new extraction rule	2-26
Figure 2 - 42:	HTML tables definition	2-27
Figure 2 - 43:	googleResults extraction rule in Extraction rules folder	2-27
Figure 2 - 44:	Extraction rule properties to be set	2-28
Figure 2 - 45:	Selection of an XPath for Google result items	2-29
Figure 2 - 46:	Root XPath expression and tree	2-29
Figure 2 - 47:	Root XPath expression and expanded tree	2-30
Figure 2 - 48:	Generating the table row XPath corresponding to Google results	2-30
Figure 2 - 49:	Google results XPath based upon root XPath	2-30
Figure 2 - 50:	Creating a new row	2-31
Figure 2 - 51:	New row	2-31
Figure 2 - 52:	Row properties	2-31
Figure 2 - 53:	Creating a new column	2-32
Figure 2 - 54:	New column	2-32



Figure 2 - 55:	Column properties2	-32
Figure 2 - 56:	Extraction rule columns in Projects Folder2	-33
Figure 2 - 57:	Expanding the LI element to find an XPath corresponding to Web page titles 2	-33
Figure 2 - 58:	H3 node corresponding to a Web page title2	-34
Figure 2 - 59:	Row XPath expression and tree2	-34
Figure 2 - 60:	Generating the XPath expression of Web page titles2	-35
Figure 2 - 61:	Web page titles XPath based upon row XPath2	-35
Figure 2 - 62:	Selection of a node for generating the XPath of URLs2	-36
Figure 2 - 63:	Row XPath expression and tree2	-36
Figure 2 - 64:	Generating the XPath expression of URLs2	-37
Figure 2 - 65:	URL XPath based upon row XPath2	-37
Figure 2 - 66:	Generation of XML document2	-38
Figure 2 - 67:	Result of the XML generation displayed under the Output tab2	-39
Figure 2 - 68:	Structure of the XML output2	-39
Figure 2 - 69:	New Transaction wizard2	-41
Figure 2 - 70:	Entering a new transaction name2	-41
Figure 2 - 71:	New transaction in Projects View2	-42
Figure 2 - 72:	Creating a new transaction variable2	-42
Figure 2 - 73:	New Variable wizard2	-43
Figure 2 - 74:	Giving a name to the new variable2	-43
Figure 2 - 75:	New variable in transaction Variables folder2	-43
Figure 2 - 76:	Variable properties2	-44
Figure 2 - 77:	New transaction handler window2	-45
Figure 2 - 78:	New entry screen class handler in the Projects View2	-46
Figure 2 - 79:	Selection of Google search field in Web Browser2	-47
Figure 2 - 80:	Expansion of the DOM tree to find an appropriate node for the search field2	-47
Figure 2 - 81:	Prevously generated XPath corresponds to one element of the Web page only 2	2-48
Figure 2 - 82:	New Statement wizard - Input HTML set value2	-48
Figure 2 - 83:	Proper statement name2	-49

Figure 2 - 84:	Improper statement name2	2-49
Figure 2 - 85:	New Input HTML set value statement2	<u>2</u> -49
Figure 2 - 86:	HTML Input set value statement properties2	2-50
Figure 2 - 87:	Edition of the Expression property2	2-50
Figure 2 - 88:	Updated statement2	2-51
Figure 2 - 89:	Selection of the Google Search button in the Web Browser2	2-51
Figure 2 - 90:	Expansion of the DOM tree to find an appropriate element for the Search button	 2-52
Figure 2 - 91:	New Statement wizard - Mouse action2	2-53
Figure 2 - 92:	New mouse click statement on entry screen class handler2	2-53
Figure 2 - 93:	Mouse click statement properties2	2-54
Figure 2 - 94:	Trigger editor window2	2-54
Figure 2 - 95:	Entry screen class handler properties2	2-56
Figure 2 - 96:	ongoogleSearchPageFR screen class handler in Projects View2	2-57
Figure 2 - 97:	Selection of a detection criterion for the googleSearchPageFR screen class2	2-58
Figure 2 - 98:	Expanding an element to find relevant attributes in the DOM Tree2	2-58
Figure 2 - 99:	Generation of XPath for the googleSearchPageFR screen class2	2-59
Figure 2 - 100:	New Statement wizard - Mouse action2	2-59
Figure 2 - 101:	New mouse click statement on entry screen class handler2	2-60
Figure 2 - 102:	clickLinkGoogleCom statement - Synchronization value2	2-60
Figure 2 - 103:	New exit screen class handler in the Projects View2	2-61
Figure 2 - 104:	Selection of the Google results Next link in the Web Browser2	2-62
Figure 2 - 105:	Expansion of the DOM tree to find an appropriate node for the Next button2	2-62
Figure 2 - 106:	New Statement wizard - Mouse action2	2-63
Figure 2 - 107:	New mouse click statement on exit scren class handler2	2-64
Figure 2 - 108:	Exit screen class handler properties2	2-64
Figure 2 - 109:	Google page with no Next link2	2-65
Figure 2 - 110:	Creation of a new screen class based on the detection of the Next link2	2-66
Figure 2 - 111:	Expansion of the DOM tree to find an appropriate node for the Next button2	2-66
Figure 2 - 112:	googleResultsPageCurrent screen class in the Projects View2	2-67



Figure 2 - 113:	ongoogleResultsPageFinalExit screen class handler in the Projects View	2-67
Figure 2 - 114:	ongoogleResultsPageFinal properties	2-68
Figure 2 - 115:	Screen class handler properties - Screen class drop-down menu	2-68
Figure 2 - 116:	XML output without accumulate data in same table property set	2-70
Figure 2 - 117:	Extraction rule properties - Accumulate data in same table	2-71
Figure 2 - 118:	XML output with accumulate data in same table property set	2-72
Figure 2 - 119:	Convertigo test platform URL	2-72
Figure 2 - 120:	Test platform	2-73
Figure 2 - 121:	XML output in the Execution result window of the test platform	2-74
Figure 3 - 1:	URL syntax for exposing transactions as REST Web services	3-2
Figure 3 - 2:	searchGoogle transaction properties	3-3
Figure 3 - 3:	Setting the Public Method property to true	3-4
Figure 3 - 4:	Update transaction schema from transaction definition	3-5
Figure 3 - 5:	Update transaction schema- Confirmation window	3-5
Figure 3 - 6:	Transaction schema types window	3-6
Figure 3 - 7:	Project folder in Project Explorer	3-7
Figure 3 - 8:	Project schema in Design tab of Editor	3-8
Figure 3 - 9:	Project schema in Source tab of Editor	3-9
Figure 3 - 10:	Transaction data types in schema	3-10
Figure 3 - 11:	URL syntax for displaying the SOAP Web service WSDL	3-10
Figure 3 - 12:	SOAP Web service WSDL	3-11
Figure 3 - 13:	Launching the Web Services Explorer	3-12
Figure 3 - 14:	Web Services Explorer	3-12
Figure 3 - 15:	Web Services Explorer	3-13
Figure 3 - 16:	URL of the SOAP service in the Web Services Explorer	3-14
Figure 3 - 17:	Web Services Explorer panes after a WSDL URL is reached	3-14
Figure 3 - 18:	WSDL Binding Details in Actions pane of the Web Services Explorer	3-15
Figure 3 - 19:	List of methods displayed in the Web Services Explorer	3-15
Figure 3 - 20:	WSDL service elements displayed in the Navigator pane	3-15

Figure 3 - 21:	Invoke a WSDL Operation view in the Actions pane	3-16
Figure 3 - 22:	Invoking a WSDL Operation	3-16
Figure 3 - 23:	Status pane after WSDL operation invocation	3-16
Figure 3 - 24:	SOAP Request envelope detail	3-17
Figure 3 - 25:	SOAP response envelope detail	3-17
Figure 3 - 26:	New Sheet wizard	3-18
Figure 3 - 27:	Entering a name for the new style sheet	3-19
Figure 3 - 28:	New style sheet object in the Projects View	3-19
Figure 3 - 29:	Project Explorer - New file	3-19
Figure 3 - 30:	New File wizard	3-20
Figure 3 - 31:	New file in Project Explorer	3-20
Figure 3 - 32:	XSL style sheet	3-21
Figure 3 - 33:	XSL style sheet templates	3-22
Figure 3 - 34:	Sheet properties	3-23
Figure 3 - 35:	Transaction properties	3-24
Figure 3 - 36:	Transaction Style Sheet property	3-24
Figure 3 - 37:	XML output after XSL transformation	3-25





Table 1 - 1:	CWI objects	1-2
Table 1 - 2:	Connector types	1-3
Table 1 - 3:	Screen class handler types	1-6
Table 1 - 4:	CWI views	1-8
Table 1 - 5:	Standard Eclipse buttons	1-9
Table 1 - 6:	Projects View tabs and icons	1-9
Table 1 - 7:	Properties View icons	1-11
Table 1 - 8:	Connector View parts	1-14
Table 1 - 9:	Connector View tabs, icons and field	1-15
Table 1 - 10:	Dom Tree icons	1-17
Table 1 - 11:	XPath Evaluator elements	1-19
Table 2 - 1:	Table extraction rule - line and column description	2-25
Table 2 - 2:	Extraction rule properties	2-28
Table 2 - 3:	New transaction handler window description	2-45
Table 2 - 4:	Synchronizer types	2-55
Table 2 - 5:	Result property values for entry screen class handlers	2-56
Table 2 - 6:	Result property values for exit screen class handlers	2-64
Table 2 - 7:	Google result page distinction	2-65
Table 3 - 1:	Web Services Explorer	
Table 3 - 2:	Style Sheet transaction property	





This chapter presents the purpose, concepts and objects of Convertigo Web Integrator (CWI), one of Convertigo Enterprise Mashup Studio's components.

It also describes the four views of the Eclipse-based CWI Studio and their elements (tabs, icons, menu).

- CWI Purpose
- **CWI** Concepts and Objects
- **CWI Studio**

1.1 **CWI Purpose**

The purpose of Convertigo Web integrator (CWI) is to expose Web sites and applications as SOAP or REST Web services, thus allowing any application to programmatically access data and business logic from the target Web site.



Figure 1 - 1: CWI exposes any Web application as SOAP or REST Web service

CWI encompasses a large number of applications:

- Reusing a company's Web applications assets to build new SOA based architectures.
- Integrating non intrusively existing enterprise Web applications into SOA.



- Creating APIs on any extranet or public Web application.
- Collecting data from Web sites to be inserted in databases.
- *Exchanging* data with any Web application such as eGoverment sites or partner sites.
- Integrating Web applications into workflow managers or complex BPM processes.

1.2 CWI Concepts and Objects

This section presents the general concept and notions of the application. It then relates notions to real CWI objects, which are defined in detail in a dedicated sub-section.

- General Concept
- CWI Objects
- Definitions

1.2.1 General Concept

CWI accesses a target HTML Web site and navigates through HTML Web *screens* (Web pages, application forms, etc.) following variable-based *transactions* while extracting information on the fly. During the extraction process, information is structured into XML format.

Navigation and extraction depend on events, actions and rules based on CWI objects, that are set as part of a Web integration project.

1.2.2 CWI Objects

The following table describes CWI objects:

Table 1 - 1: CWI objects

Object ^a	Description
Connector	Defines the type of application (HTTP, HTML, etc.) CWI connects to.
Screen class	Defines a family of HTML pages with common characteristics.
Criterion	Is a characteristic element defining a specific screen class.
Extraction rule	Is a rule defining how information is extracted from a Web screen.
Transaction	Defines the actions Convertigo must perform on HTML pages.
Screen class handler	Is a set of statements triggered on detection of a specific screen class.
Statement	Is an action or set of actions performed on HTML pages.
Variable	Is the transaction input variable.

a. Objects are indented in the column according to their indentation level in the Project tree of the Projects View (see Figure 1 - 2)

All of these objects are set when developing a Web integration project. Once the project has been created, these objects appear in the **Projects View** of the CWI Studio (*"CWI Studio"* on page 1 - 7):



Figure 1 - 2: CWI objects



To get started with your first Web integration project, see "My First Convertigo Web Integrator Project" on page 2-1

1.2.3 Definitions

CONNECTOR

A connector defines the type of application or data source CWI is able to connect to.

Two types of connectors are available in CWI:

Table	1 -	2:	Connector	types
Tuble		<u> </u>	Connicotor	typeo

Connector type	Description
HTTP Connector	Enables Convertigo to connect to any HTTP flow.
HTML connector	Enables Convertigo to connect to any HTML based application.



Connector parameters are set in the **New Project** wizard which opens when creating a Web integration project (see "Creating a Project and Setting Connector Parameters" on page 2-6).

Once a new project has been created, the corresponding connector appears in the *Connectors* folder of the project, in the **Projects View** of the CWI Studio (*"CWI Studio"* on page 1 - 7):



Figure 1 - 3: Connectors folder

SCREEN CLASS

Screen classes define families of Web screens (Web page, application form, etc.) with common characteristics (called *criteria*). Screen classes are:

- uniquely identified by at least one criterion;
- associated to extraction rules.

A single root screen class, called **Default screen class**, is created by default when creating a project The default screen class is not - but can be - associated to any criterion or extraction rule. It is parent to all other screen classes.

A screen class based on a parent screen class (defined with extra criteria and extraction rules for instance) is called *Inherited screen class*. It inherits criteria and extraction rules from its parent screen class.

Inherited screen classes appear in subfolders named *Inherited screen classes* in the **Projects View** of the CWI Studio (*"CWI Studio"* on page 1 - 7):



Figure 1 - 4: Screen classes and Inherited screen classes folders

CRITERION

A Web screen is considered belonging to a screen class when it matches all criteria defined by the user for this screen class. If no criterion matches, the Web screen is considered belonging to the default screen class (provided that the default screen class is not associated to any detection criterion).

Criteria are set using the *XPath language*, which addresses parts of an XML document; an XPath can be, for example, the path to a logo, a field, a word or any other identifying element on a Web screen. If at least one element is adressed by the XPath on the HTML page, the criterion matches.

XPaths are managed in the CWI Studio using the **XPath Evaluator**, which is part of the **Connector View** of the CWI Studio (*"CWI Studio"* on page 1 - 7).



Figure 1 - 5: Criterion and inherited criterion



For more information on XPath, see the XPath tutorial from W3Schools at http://www.w3schools.com/XPath/.

EXTRACTION RULE

Extraction rules define which data must be extracted from the target application and how. They are used to generate parts of the XML output resulting from a transaction execution.

During a *transaction* process, when a screen class is detected (i.e. all criteria defined for this screen class match), all screen class-specific extraction rules are executed, including those from its inherited screen classes.

TRANSACTION

Web page navigation is performed using HTML transactions, that is a series of actions (called *statements*) performed from one Web screen to another. These actions are triggered by events associated with the detected screen class, using *screen class handlers*.

A transaction can be summed up as a process:

- taking variables in input;
- producing an XML document in output.

When executing the transaction, transaction variables are automatically added as variables of the JavaScript scope. Transaction variables can be given a fixed (default) value, or their value can be set as input parameter prior to executing a transaction (for an example of transaction variable setting, see *"To set a transaction variable"* on page 2-42).

Once a transaction has been defined, it appears in the Transactions folder of a given



connector, in the **Projects View** of the CWI Studio ("CWI Studio" on page 1 - 7):



Figure 1 - 6: Transactions folder

SCREEN CLASS HANDLER

Basically, a screen class handler is meant to answer the following question : "Now that I have accessed this screen, what am I supposed to do?".

In other words, once a given screen class has been accessed and detected (depending on user-defined criteria), the corresponding screen class handler is triggered and performs user-defined actions.

In CWI, two types of screen class handler can be defined for a given screen class:

Table 1 - 3: Screen class handler types

Screen class handler	Description
on <screenclass>Entry</screenclass>	Triggered when CWI detects a given < ScreenClass >. Is identified by a is symbol on the left of the screen class handler name (see Figure 1 - 7).
on< <i>ScreenClass</i> >Exit	Triggered after extraction rules have been executed for a given < ScreenClass >. Is identified by a symbol on the left of the screen class handler name (see Figure 1 - 7).



Other handlers exist, but their use falls out of the scope of this guide.

Once a screen class handler has been created, it appears in the *Functions* folder of the transaction, in the **Projects View** of the CWI Studio (*"CWI Studio"* on page 1 - 7):



Figure 1 - 7: "On entry" and "on exit" screen class handlers

A screen class handler can contain one or several statements.

For example: Type in a keyword (statement #1) then click on *Next* (statement #2):



Figure 1 - 8: List of statements

STATEMENT

A statement is an action that is performed on **entry** to or on **exit** from a Web screen, provided that the screen class of this Web screen has been detected. Statements include mouse clicks, data inputs, etc.

1.3 CWI Studio

This section describes the CWI Studio, which comprises four distinct panes: the **Projects View**, the **Connector View**, the **Properties View** and the **Console**.

- General Presentation
- View Description

1.3.1 General Presentation

Based on an Eclipse platform, the CWI Studio is divided into four views that all include:

- one or several tabs, located either on the upper or on the lower part of the view;
- icons;
- three standard Eclipse buttons (Menu , Minimize) and Maximize)

The four views are illustrated and named in the figure below:





Figure 1 - 9: CWI Studio

The following table describes the views of the Convertigo Web Integrator studio:

View Name	Description		
Projects View	Displays in a tree structure all CWI projects and their related objects: connectors, screen classes (including criteria, extraction rules and inherited screen classes) and transactions (including screen class handlers and statements).		
Properties View	Displays the properties of the CWI object selected in the Projects View . Used to modify a given object properties.		
Connector View / Editor	 The Connector View is divided into three parts: the DOM Tree - displays the Document Object Model Tree (logical structure) of the Web page accessed by CWI; the Web Browser - displays the Web page accessed by CWI, depending on the connector parameters set for the project; the XPath Evaluator - displays XPath-specific information such as the XPath expression of a node. Contains shortcuts (icons) for creating new child screen classes, criteria, extraction rules, statements, etc. It is automatically updated as an Editor whenever files (in editable format such as .xml, .css or .xsl) are edited. 		
Console View	Standard Eclipse console. Displays information about running tasks, errors, etc. Different consoles are available (standard output, Engine, Studio, etc.).		

Table 1 - 4: CWI views

1.3.2 View Description

All four views contain common standard Eclipse buttons in the upper right corner of the view:

Table 1	- 5:	Standard	Eclipse	buttons
---------	------	----------	---------	---------

Button	lcon	Description
Menu		Opens a drop-down menu with a content identical to the view toolbar.
Minimize		Minimizes the current view without closing it. The view is still visible in the form of an icon that can be clicked to be enlarged again.
Maximize		Maximizes the current view to full screen. Maximizing a view can also be achieved by double clicking the upper bar of the view.

The following section describes the four panes of the CWI Studio in further detail.

PROJECTS VIEW

The **Projects View** displays all information (objects, folders and files) associated with a project. Information appear in distinct tree structures depending on the selected tab.

In addition to project tree structures and standard Eclipse buttons (see Table 1 - 5 on page 1-9), the **Projects View** includes tabs and icons:

Table 1	- 6.	Projects	View	tabs	and	icons
	υ.	1 10/000	VICVV	1003	and	100113

	View Element	Description
Tabs	Projects	Convertigo view. Contains all projects and related objects. Objects are presented in a tree structure and contained in folders.
	Project Explorer	Standard Eclipse view. Contains all projects and related files. Files are represented in a tree structure.
lcons		<i>Find</i> icon. Search project tree for a specific object
	•	<i>Decrease priority</i> icon. Available only if an extraction rule is selected. Decreases its priority, thus changing extraction rules execution order for this screen class and its children classes.
	•	<i>Increase priority</i> icon. Available only if an extraction rule is selected. Increases its priority, thus changing extraction rules execution order for this screen class and its children classes.
		Save icon. Available only when at least one object has been modified. Saves all the project. All objects will be saved, and project xml file will be updated. Note: Remember to save projects on a regular basis.
		<i>Refresh</i> icon. Available only if a project is selected. Refreshes Projects View by reloading project content from the hard drive. Identical to closing and re-opening project.
	۵	<i>Import</i> icon. Imports a new project (in CAR or XML format) in the Studio.

In the Projects View, objects are formatted depending on their status:

- plain text unchanged object;
- bold object changed and unsaved;



- red disabled object;
- orange not executable object (child of a disabled object);
- grey inherited object.

The figure below shows the different types of formatting in the **Projects View**.

🎦 Projects 🛛 🔓 Project Explorer 🛛 🔑 🔻 🖌	i 🗈 🎯 🏹 🗖 🗖
😑 🥸 sample_documentation_CWI	
🖃 🗁 Connectors	
🖹 🥞 GoogleConnector	
🖹 🗁 Screen classes	
🖻 📇 Default_Screen_class	
😑 🗁 Inherited screen classes	
🖃 📇 googleWebPages	
🗈 🗁 Criteria	Unsaved object
🖃 🧀 Inherited screen classes	
🖹 🔀 googleResultsPageFina 🗐 🗁 Criteria	
Exists node at "//8	@class="gb1" and cc
🖨 🗁 Extraction rules	Inherited object
🗉 💁 googleResults	
🗄 🗁 Inherited screen classe	s
🖻 🔠 googleSearchPage	Disabled object
🗄 🗁 Transactions	Disabled object
× [>

Figure 1 - 10: Formatting of unsaved, inherited and disabled objects in the Projects View

Once an object has been selected in the **Projects View**, its properties appear in the **Properties View**.

PROPERTIES VIEW

The Properties View displays the properties of the object selected in the Projects View.

Properties depend on the type of the selected object. They are presented by categories (*Expert, Configuration, Information, Selection, Properties*, etc. - categories can be hidden by clicking on $\boxed{1}$) that can be either expanded or collapsed by clicking respectively on 1 or 2.

Properties 🛛	[≌] 券 ▽ □ □]
Property	Value
Configuration	
Accumulate data in same table	true
Comment	
Display referer	false
Flip table orientation	false
Is active	true
Tag name	listResults
🖃 Information	
Depth	n/a
Java class	com.twinsoft.convertigo.beans.common.XMLTat
Name	googleResults
Priority	1245402035240
QName	/sample_documentation_CWI/_data/cn/QQMIK.
Туре	Table
Selection	
XPath	//DIV[@id="res"]
<	

Figure 1 - 11: Properties View

In addition to object properties and standard Eclipse buttons (see Table 1 - 5 on page 1-9), the **Properties View** includes one tab (*Properties*) and icons:

Table 1	- 7:	Pro	perties	View	icons
		1.10	portiou	1011	100110

lcon	Description			
	Show Categories icon. Displays or hides property categories. When categories are hidden, properties appear in alphabetical order.			
	Show Advanced Properties icon. Displays or hides advanced properties, if relevant. Not used for Convertigo objects properties.			
	Restore Default Value icon. Available only when a property has been modified. Restores it to default value. Not used for Convertigo objects properties.			

To edit properties

- 1 Left-click on the property value in the **Properties View**;
- 2 Depending on the property type:
 - the value is highlighted with white background; edit it by typing a new value in the field:

 Properties 	
Comment	
XPath	//DIV[@id="res"]
	S
and and	manne

Figure 1 - 12: Text property field edition

• the value is highlighted with clear blue background; edit it by typing a new



JavaScript expression in the field:



Figure 1 - 13: Javascript expression field edition

• the value is higlighted and a . symbol appears on the right of the field:



Figure 1 - 14: Text property edition via edition window

Click on . Depending on the property type:

A. A Javascript expression window or an edition window opens:

Javascript expression	
'//A[contains(text(), "Next")]'	
	~
ОК Са	ncel

Figure 1 - 15: Javascript expression window

Web Integration project	
OK Cano	el

Figure 1 - 16: Edition window

- 1 Enter the new JavaScript expression or the text comment in the window.
- 2 Click on OK.
- **B.** A configuration window opens (in this example, the **Trigger Editor** window for statement synchronization):

Trigger editor			
Type of synchronizer Timeout (ms) This synchronizer wait	Document completed 60000 for a number of document completed.		
Number of document completed 1			
	OK Cancel		

Figure 1 - 17: Configuration window

- 1 Set the parameters in the different fields and menus.
- 2 Click on OK.
- the value is higlighted and a value symbol appears on the right of the field:

Properties		!cem
Action	click 🔍	Cems
Comment	\cup	!cems
Is active	true	!cen.
XPath	'//A[contains(text(),"Next")]'	!ce
the second se		-

Figure 1 - 18: Text property edition via drop-down menu



1	Click on	A drop-down menu appears:	
---	----------	---------------------------	--

Type	mouse action	
🖃 Properties		!ce:
Action	click 💌	!cem
Comment	click 🔨	!cen
Is active	mousedown	!ce
YPath	mouseup	Icem
Argen	mouseover	
	mouseout 🔛	!ce
		!cer
	and a series of the second	1 c
	and the surface	! (! (

Figure 1 - 19: Selecting a value in the drop-down menu

- 2 Select a value in the drop-down menu.
- 3 Press Enter.

CONNECTOR VIEW

The **Connector View** displays connector-specific information. This information depends on the type of connector set for the project. In the case of HTML connectors, the view is divided into three parts.

The table below describes these parts:

Table 1 - 8: Connector View parts

Name	Description			
Dom Tree	Displays the logical structure of the Web page displayed in the Web Browser .			
Web Browser	Displays the Web page accessed by CWI.			
XPath Evaluator	Displays the XPath expression of HTML elements selected in the Web Browser and result nodes of the written XPath execution on the DOM Tree . Includes shortcuts to create new criteria, statements, screen class, etc.			

As other views, the **Connector View** also includes tabs and icons:

sample_documentation_CWI Goog	leConnector 🛛	- 6
💽 🤹 🛐 😵 🖒 🖒 Curren	t selection : HTML/BODY/DIV/DIV[1]/NOBR	
🗳 🦻 🗢 🗢	G 🗑 💸 🗙 Address http://www.google.com/webhp?hl=en	S X 2 X ≤ 2 1/1
HEAD A	Web Images ⊻ideos Maps News Shopping Gmail more ▼	iGoogle Search settings Sign in
Attributes		
IFRAME		тм
■ •• ● DIV		
● DIV		
E····● DIV E····● CENTER		Advanced Search
i∎ • BR	Google Search I'm Feeling Lucky	Language Tools
• BR		
DOM Tree 1		Web Browser
Path //B[@class="gb1"	and contains(text(),"Web")]	
Document		
B B	XPath Evalua	tor
	tabs	<u>⊻</u>

Figure 1 - 20: Connector View (Design tab selected)

The table below describes the **Connector View** icons that are available when the **Design** tab is selected (by default):

	View Element	Description
Tab	Output	Displays the XML code generated following either a transaction (according to extraction rules) or a <i>Generate XML</i> action (see Figure 1 - 21).
	Design	Displays the Dom Tree, Web Browser and XPath Evaluator.
lcon	2	<i>Toggle auto refresh domTree</i> icon. Press this icon to activate the auto refresh domTree function, which automatically refreshes the DOM Tree when a new Web page is displayed in the Web Browser.
	₹¢	Show current screen class icon. Highlights in light grey in the Projects View the screen class detected for the Web page currently displayed in the Web Browser .
	3	Generate XML icon. Generates the XML code of the Web page currently displayed in the Web Browser by executing the related extraction rules. The XML code is displayed under the Output tab of the Connector View .
	0	Stop transaction icon. Stops the current transaction (while a transaction is in process).
	⇔	<i>Learn</i> icon. Available only when a transaction is selected. Activates the <i>Learn</i> mode - any action manually performed in the HTML Connector View is recorded and HTTP statements are automatically generated in corresponding screen class entry handler. For advanced users only.

Table 1 - 9: Connector View tabs, icons and field



	Table 1	- 9:	Connector	View	tabs,	icons	and	field ((
--	---------	------	-----------	------	-------	-------	-----	---------	---

View Element		Description	
	⇔	Accumulate learning mode icon. Available only when Learn mode is on. Accumulate data on visited Web screens. Screen class exit handlers are automatically created in the transaction. For advanced users only.	
Field	Current selection	Displays the path to the node selected in the DOM Tree (left- click) or to the HTML element selected in the Web Browser (right-click).	

When the **Output** tab is selected, the view displays XML code:



Figure 1 - 21: Connector View (Output tab selected)

DOM TREE

The **DOM Tree** models an HTML (or XML) document as a tree structure called *node-tree*. It contains all nodes (parents, children and siblings) of the HTML document it models (which is displayed in the **Web Browser** in the CWI Studio):

- root nodes;
- element nodes;
- text nodes;
- attribute nodes;
- namespace nodes;
- processing instruction nodes;
- comment nodes.



Figure 1 - 22: DOM Tree

The table below describes the **Dom Tree** icons:

Table 1 - 10: Dom Tree icons

lcon	Description
\$	<i>SyncTree</i> icon. Synchronizes the DOM Tree with the HTML page currently displayed in the Web Browser .
*	Parent node icon. Selects and highlights in green the parent node of the node currently selected in the DOM Tree . Also outlines in green the corresponding HTML element in the Web Browser .
¢	<i>Previous node</i> icon. Selects and highlights in green the node appearing before the node currently selected in the DOM Tree . Also outlines in green the corresponding HTML element in the Web Browser .
\Leftrightarrow	<i>Next node</i> icon. Selects and highlights in green the node appearing after the node currently selected in the DOM Tree . Also outlines in green the corresponding HTML element in the Web Browser .

WEB BROWSER

The **Web Browser** displays the Web page accessed by CWI. Its logical structure is displayed in the DOM Tree on the left of the browser.

The **Web Browser** automatically displays the HTML page of the HTTP server defined as **Target Server** in the connector parameters (see *"Creating a Project and Setting Connector Parameters"* on page 2-6).

All HTML elements (image, text, field, etc.) of Web pages displayed in the **Web Browser** can be selected with a right-click. Once selected, they appear outlined in green in the **Web Browser**, and their corresponding element node is highlighted in green in the **DOM Tree**:





Figure 1 - 23: HTML element selected in the Web Browser

The XPath expression of selected elements can be generated either by right-clicking the required element node in the DOM Tree and selecting **Generate selection XPath (Enter)** or by selecting it with a left-click and pressing **Enter**. The XPath expression then appears in the **XPath Evaluator**.

The Web Browser includes a toolbar with:

- standard Browser icons Back, Forward, Refresh, Stop;
- an address bar;
- icons to manage Web page tabs New, Close, Next, Previous. The Allow alert box icon allows to display or block JavaScript pop-ups (blocked by default in CWI).

XPATH EVALUATOR

The purpose of the **XPath Evaluator** is to display the XPath expression of HTML elements previously selected (with a right-click) in the **Web Browser**. The **Document** field displays the results from running the XPath expression specified in the **xPath** field on the **DOM Tree**.

The **XPath Evaluator** also contains icons for creating new CWI objects (statements, criteria, screen classes, etc.) from the XPath currently displayed in the **xPath** field.

<pre>weath **////B[@class="gb1" and contains(text(),"Web")]</pre>	<
Document	

Figure 1 - 24: XPath Evaluator

The table below describes the fields and icons of the XPath Evaluator:

XPath Evaluator Elements		Description
Field	xPath	Displays the XPath expression generated from the Dom Tree . XPath expressions can also be typed in directly and evaluated on the DOM Tree
	Document	Displays the results from running the XPath expression specified in the <i>xPath</i> field on the DOM Tree .
Icon	2	<i>Create new child screen class from current XPath</i> icon. Launches the New Screen Class wizard.
	8	Create new criterion from current XPath icon. Launches the New Criterion wizard.
		Create new extraction rule from current XPath icon. Launches the New Extraction Rule wizard.
		Create new statement from current XPath icon. Launches the New Statement wizard. Available when a suitable handler or container is selected in the Projects View only.
	0	<i>Evaluate XPath</i> icon. Evaluates the XPath expression specified in the <i>xPath</i> field on the DOM Tree . The result is displayed in the <i>Document</i> field.
		<i>Backward XPath History</i> icon. Displays in the <i>xPath</i> field the previous XPath stored in the XPath history.
	٢	<i>Forward XPath History</i> icon. Displays in the <i>xPath</i> field the next XPath stored in the XPath history.
lcon	3	Set Anchor icon. Applicable after an XPath has been generated for a given node. Fixes this XPath and highlights it in yellow. Used to develop complex XPath from an <i>anchored</i> XPath.

Table 1 - 11: XPath Evaluator elements





This chapter gives instructions on how to set up a simple CWI project.

- Web Integration Project
- Setting Up the Project

2.1 Web Integration Project

This section describes the sample Web Integrating project and how to open the completed sample project from the Studio:

- Project Description
- Opening the Sample Project from the Studio

2.1.1 Project Description

The purpose of this Web integration project (called *sample_doc_CWI*) is to list in XML format the Web page titles and URLs returned by the Google search engine after sending a search request.

The process can be summed up as follows:

1 Connect to the Google Web site at www.google.com.





Figure 2 - 1: Connection to Google HTTP server



When accessing the www.google.com Website, Google servers provide Internet users with a "localized" version of the Google search engine. This adds an extra step to the project, since CWI must then be redirected to the standard Google home page (see "Case of a Country-Specific Google Search Page - Definition of the googleSearchPageFR Screen Class" on page 2-21).

2 Key in search keywords in the Google search field (for example convertigo cliplet).

Web Images Videos Maps News Shopping Gmail more ▼	<u>iGoogle Sign in</u>					
Google						
Convertigo Cliplet Convertigo Cliplet Google Search I'm Feeling Lucky Language Tools						
Newl <u>Land on the Moon</u> in Google Earth.						
Advertising Programs - Business Solutions - About Google - Go to Google France						
92009 - <u>Privaov</u>						
http://www.google.com/services/						

Figure 2 - 2: Input of a search keyword in Google search field

3 Click on the **Google Search** button.


Figure 2 - 3: Click on Google Search button

4 Extract all Web page titles and URLs (according to *extraction rules*) into an XML list of results.



Figure 2 - 4: Extraction of Web page titles and URLs in XML format

5 Click on the **Next** link.





Figure 2 - 5: Click on the Next button

6 Repeat step 4 as long as a **Next** link is present on Google result pages.

7 Stop process when accessing a Google result page with no Next link:



Figure 2 - 6: Process stops on Google page with no Next button

2.1.2 Opening the Sample Project from the Studio

The completed sample project is stored in the application installation folder.

The following procedure describes how to access it from the Studio using the **New Project** wizard.

To open the sample project from the Studio

1 In the **Studio** toolbar, click on **New > Project**.



Figure 2 - 7: Launching the New Project wizard

A New Project wizard opens.

2 Expand Convertigo Projects, then Documentation Samples, and select Web Integrator:



Figure 2 - 8: Selecting the CWI sample project

3 Click on **Next**, then **Finish**.

The sample project opens in the **Projects View**:



Figure 2 - 9: Sample project in Projects View

You can now start setting up your own CWI project, sample_doc_CWI.

2.2 Setting Up the Project

The following sections explain step by step how to set up your first Web integration project:

- Creating a Project and Setting Connector Parameters
- Defining Screen Classes



- Defining a Transaction
- Executing a Transaction
- 2.2.1 Creating a Project and Setting Connector Parameters

The first steps consist in creating a project and setting the connector parameters.

In this sample project:

- the project is called sample_doc_CWI;
- the connector is an HTML connector called GoogleConnector accessing the www.google.com Web site.

To create a project and set a connector

- 1 Launch the **Convertigo Studio**.
- 2 Select File > New > Project or click on the toolbar then select Project.

A New Project wizard opens:

New Project	
Select a wizard	
Wizards: type filter text	
General Convertigo Projects Cv5 JavaScript SvN SvN Svb Examples	
?	Next > Finish Cancel

Figure 2 - 10: New Project wizard

3 Expand Convertigo Projects, then Web Integration and select HTML Web Site:



Figure 2 - 11: Selecting a project type

4

In this example, **HTML Web Site** is selected because the application to which CWI connects is an HTML application.

Click on Next:

New Convertigo project This wizard creates a new Convertigo project
Please use a relevant project name. Avoid the use of special characters (âàéàêù) and punctuation characters as space, pound or others. We suggest you use only lowercase letters. If you use uppercase letters, be sure use the same letter case when you will call transactions using the convertigo's url interface. The project name also defines the Convertigo's physical and virtual directories: - All your project's URL will be held in the convertigo/tomcat/webapps/ <your_project_name>directory. - Your project's URL will be http://cserver_name>i<port>/projects/syour_project_name>j.cxml</port></your_project_name>
Project's name sample_doc_CWI
Image: Second

Figure 2 - 12: Entering a project name

- 5 In the **Project's name** field, type in the project name (for example sample_doc_CWI).
- 6 Click on Next:





Figure 2 - 13: Entering a connector name

The **Connector name** field is filled by default with the connector name sample_doc_CWIConnector. You can choose another name (for example GoogleConnector).

7 Click on Next:

Figure 2 - 14: Setting HTTP or HTML connector parameters

- 8 Enter the required information in the **Target Server** section:
 - in the HTTP server field, type in the target Web site URL address (for example www.google.com);
 - in the HTTP Port field, type in the HTTP port if required (left blank here);
 - check the SSL checkbox to connect to a secured HTTP server (HTTPS server).

- 9 If you're using a proxy server to access the Web, enter the required information in the Proxy Server section (Proxy Server address, Proxy Port and SSL).
- 10 Click on Next.

The New project summary window opens:

2	
New project summary This step summarizes all the configuration options	
Here are all the configuration options you chose during the projet setup. Click "finish" to create the project or "back" to review parameters. Project name : sample_doc_CWI Connector configuration Target server : http://www.google.com:80 Proxy server : http://!8080	
Image: Constraint of the sector of the se	Cancel

Figure 2 - 15: New project summary window

11 Check that all settings are correct, then click on **Finish**. The project now appears in the **Projects View**:



Figure 2 - 16: sampleGoogle project appearing in the Projects View

As we can see, one screen class and one transaction have been created by default in their respective folders (*Screen Classes* and *Transactions*).

The HTML page of the HTTP server specified as **Target Server** opens in the **Web Browser** of the **Connector View**, with its **DOM Tree** to the left:



😻 legacyCRM legacyCRM	1C 🕸 sampleGoogle2 Google 🍽 sampleGoogle GoogleC 🛛 🔭 🗖 🗖	
😰 孝 🔕 🖒 🖒 Current selection :		
🗳 Þ 🗢 🗢	Image: Second system Image: S	
HTML	Web Images Vidéo Maps Actualités Groupes Gmail plus •	
BODY	<u>iGoogle</u> <u>Connexion</u>	
GOOQIE		
	France	
	Recherche avancée Préférences	
	Recherche Google J'ai de la chance <u>Outils linguistiques</u>	
	Rechercher dans : Viveo V Pages francophones V Pages : France	
	Programmes de nublicité - Solutions d'entrenrise - À nronos de Google -	

Figure 2 - 17: DOM Tree and Web Browser

This is the starting point of the project.

The following steps will consist in defining a set of screen classes and in setting their *detection criteria* (in other words, HTML elements - text, links, logos, etc. - defining a Web screen in a unique manner).

When CWI accesses a page, all criteria are checked; if all criteria defined for a screen class match page elements, then CWI associates the page to this screen class. It is then able to trigger a number of statements (*"Defining a Transaction"* on page 2 - 40).

2.2.2 Defining Screen Classes

To define a screen class, you must first find which element of the accessed Web page can be considered sufficiently relevant to serve as a detection criterion. Using a screen element as a detection criterion for a screen class means:

- 1 Selecting the element on the screen (right-click).
- 2 *Finding* its most accurate nodes in the DOM Tree.
- 3 Generating its XPath.
- 4 *Creating* the corresponding screen class from the generated XPath expression.

ROOT SCREEN CLASS

In our example project, the first and most general screen class that we want to define as **root** screen class (called googleWebPages in this project) is the screen class representing a general Google Web page.

Web pages only (not images, videos, maps, etc.) are to be returned throughout our search. This means that the *Web* item must be present on the upper left corner of the Google page,



and that this item must be plain text (as opposed to hypertext) (see Figure 2 - 18):

Figure 2 - 18: Google general search page

This item will serve as detection criterion for the root screen class.

The following procedure explains how to:

- 1 select a detection criterion for a screen class (here, the parent screen class googleWebPages);
- 2 generate the corresponding XPath;
- 3 create the screen class from the generated XPath expression.

To find a criterion and create a screen class

1 Right-click on the word "Web" in the upper left corner of the Google search page.

The element is outlined in green in the **Web Browser** and its corresponding element is highlighted in green in the **DOM Tree**:



Figure 2 - 19: Detection criterion of the googleWebPages parent screen class



2 Expand the B element in the **Dom Tree**:



Figure 2 - 20: Expanding an element to find relevant nodes and attributes in the DOM Tree

Two nodes are associated with the B element: an attribute node (class="gbl"), which represents the element class, and a text node (Web), which represents the element content.

We will select both and generate the XPath from the selection.

- 3 While keeping the Shift button pressed, select class="gbl" and Web in the DOM Tree with a left-click.
- 4 Press Enter or right-click on the selection and select Generate selection XPath (Enter).



Figure 2 - 21: Generating XPath from class and text attributes

An XPath expression corresponding to the selected HTML element (i.e. "Web" item) and

based on the two nodes selected as detection criteria for the googleWebPages screen class appears in the **XPath Evaluator**:



Figure 2 - 22: XPath expression of the detection criterion of the googleWebPages screen class

The results from running the XPath expression on the DOM appears in the **Document** window.

We will now create, based on this XPath, the corresponding screen class called googleWebPages:

- 5 In the **Projects View**, select the screen class that we want as *parent screen class* for the new screen class (Default_Screen_Class).
- 6 In the XPath Evaluator, click on the Create new child screen class from current

XPath icon [

A New Screen Class wizard opens:

2	
New Screen Class	
Please select a screen class template.	
Screen class	
Image: Second se	Cancel

Figure 2 - 23: New Screen Class wizard

7 Click on Screen class, then on Next:



a	
Informations Please enter a name for object.	
Name: googleWebPages	
Image: Constraint of the sector of the se	Cancel

Figure 2 - 24: Giving a name to the new screen class

- 8 Type in the name of the new screen class (for example googleWebPages).
- 9 Click on Finish.

The screen class now appears in the **Projects View** as an *inherited* screen class of the Default_Screen_class:



Figure 2 - 25: New screen class appearing in the Projects View

The criterion previously defined for the screen class is automatically generated and appears in the related Criteria folder.

10 Save your project by clicking on 🤖 or by pressing Ctrl + S.



Remember to save your project on a regular basis.

To check if detection criteria are correct for this screen class

1 Assuming that the **Web Browser** currently displays a Google search page, click on the

Show current screen class icon 🔁 in the Connector View toolbar.

In the **Projects View**, the googleWebPages screen class is automatically highlighted, showing that detection criteria identifying this screen class do all match on the accessed Web page.

2 Type in a keyword (for example convertigo cliplet).

3 Click on the **Google Search** button.

Results are displayed in a Google result page.

In the Connector View toolbar, click on the Show current screen class icon 👌.

As in the first step, the googleWebPages screen class is automatically highlighted in the **Projects View**.

We can infere from this test that the detection criterion identifying Google Web pages is well set. Indeed, each type of Google Web page (*result* and *search* pages) which will be accessed throughout the transaction is detected by CWI as belonging to the googleWebPages screen class.

ADDITIONAL SCREEN CLASSES

The following step consists in creating two additional scren classes:

- one corresponding to the Google search page, called googleSearchPage;
- one corresponding to Google result pages, called googleResultsPage.

Those two additional screen classes are **children** screen classes of the googleWebPages: they **are** Google Web pages, but they also present specificities (one is a *result* page, the other is a *search* page).

They automatically inherits the googleWebPages detection criteria. Additional detection criteria are then added to their definition to differentiate them.



For projects set in countries for which a country-specific Google homepage exists, a third screen class must be defined : googleSearchPageFR (see "Case of a Country-Specific Google Search Page - Definition of the googleSearchPageFR Screen Class" on page 2-21). This screen class is detected using the "Google.com in English" link of country-specific Google homepages. It is used to redirect CWI back to the English Google search page.

To create additional screen classes, the procedure is identical to the one used to create the googleWebPage screen class:

- 1 Selection of a detection criterion.
- 2 Generation of XPath.
- 3 Creation of a new child screen class.

GOOGLESEARCHPAGE SCREEN CLASS

To create the googleSearchPage screen class

1 Right-click on the Google logo in the Google search page displayed in the **Web Browser**:





Figure 2 - 26: Selection of a detection criterion for the googleSearchPage screen class

This detection criterion has been selected because the Google search page does include a Google logo, whereas result pages do not.

2 Expand the DOM Tree to select the most relevant node for this HTML element:



Figure 2 - 27: Attribute nodes of the HTML element selected as detection criterion for the googleSearchPages screen class

An id attribute node appears in the tree. This attribute always define an HTML element in a unique manner, so this is the best candidate for generating the XPath corresponding to the IMG element.



When present, the NAME attribute is also a good candidate. On the contrary, avoid using attribute nodes such as SRC and HREF, which are never good candidates for generating XPath corresponding to detection criteria (their value often changes).

3 Right-click the id="logo" attribute node and select Generate selection XPath (Enter) or select the attribute node with a left-click and press Enter.



You can also right-click the parent IMG element and select **Generate** selection XPath (Enter) or select the IMG element with a left-click and press Enter: the XPath will intelligently be generated with the id attribute, since it exists.

The XPath expression of the selected node appears in the XPath Evaluator.

- 4 Select the parent screen class, googleWebPages.
- 5 In the XPath Evaluator, click on the Create new child screen class from current

XPath icon [

The New Screen Class wizard opens.

- 6 Click on Screen class, then on **Next**.
- 7 Type in the name of the new screen class (for example googleSearchPage).
- 8 Click on Finish.

The screen class now appears in the **Projects View** as an *inherited* screen class of the googleWebPages screen class:



Figure 2 - 28: googleSearchPage screen class in Inherited screen classes folder

The criterion previously defined for this screen class is automatically generated and appears in the related Criteria folder. Criterion inherited from the parent screen class appears greyed.

9 Save your project by clicking on is pressing Ctrl + S.



The next step consists in creating the googleResultsPage screen class identifying Google result pages.



GOOGLERESULTSPAGE SCREEN CLASS

We will now launch a Google search with the keywords convertigo cliplet and define a screen class for result pages.

To check that there will not be any conflict between the detection criteria of the different screen classes, we can launch the search in the **Web Browser** and evaluate the XPath previously defined using the **XPath Evaluator**.

To check that the criterion is a unique identifier for a screen class

- 1 Type in convertigo cliplet in the Google search field of the Web Browser.
- 2 Once results have been displayed in the browser, type in the previous XPath (// IMG[@id="logo"]) in the **xPath** field of the **XPath Evaluator**.
- 3 Click on 😧 in the **XPath Evaluator**.

The result of the evaluation is displayed in the **Document** field of the **XPath Evaluator**:

	×Path //IMG[@id="logo"]	<u>^</u>
	Document	
	····· • root	
۷٤		
	<	 ✓

Figure 2 - 29: googleSearchPage XPath evaluation on a Google result page

No IMG element with an id="logo" attribute was found in the document tree model of the page currently displayed in the **Web Browser**. The criterion selected for the googleSearchPage screen class is therefore a unique differentiator for this screen class.

To create the googleResultsPage screen class

1 In the Web Browser, right-click on the right of the first result:



Figure 2 - 30: Selection of a detection criterion for the googleResultsPage screen class



Searching for a detection criteria is a sampling process that requires selecting different elements on the page to select the best suited for detecting a screen class.

The LI container is highlighted in the **DOM Tree**. This container contains all nodes modeling the first result returned by the Google search engine. It does not include any attribute that could possibly serve as strong differentiator for the screen class. We must now look in parent nodes to find one.

2 To do so, click on 🦙 to select the parent node.

An OL container is outlined in green.

This element is useless for detection because it does not contain any attribute or text content that could serve to generate an accurate XPath for the selected HTML element.

3 Click on 🤧.

A DIV container is outlined in green. For the same reasons as previously mentioned, this node cannot be used to generate an accurate XPath.

Click on s.

Another DIV container is highlighted in green:



Figure 2 - 31: Searching for a better detection criterion

This node contains an id attribute, which can be used as unique attribute for generating an element's XPath, as mentioned in the definition of the previous screen class.

5 Right-click the id="res" attribute node and select Generate selection XPath (Enter) or select the attribute node with a left-click and press Enter:





Figure 2 - 32: Generation of XPath for the googleResultsPage screen class

The XPath expression of the selected node appears in the **XPath Evaluator**.



You can also right-click the parent DIV element and select **Generate** selection XPath (Enter) or select the IMG element with a left-click and press Enter: the XPath will intelligently be generated with the id attribute, since it exists.

- 6 Select the parent screen class, googleWebPages.
- 7 In the XPath Evaluator, click on the Create new child screen class from current

XPath icon [🔠 .

The New Screen Class wizard opens.

- 8 Click on Screen class, then on **Next**.
- 9 Type in the name of the new screen class (for example googleResultsPage).
- 10 Click on Finish.

The screen class now appears in the **Projects View** as an *inherited* screen class of the googleWebPages screen class:



Figure 2 - 33: googleResultsPage screen class in Inherited screen classes folder

The criterion previously defined for the screen class is automatically generated and appears in the related Criteria folder. Criterion inherited from the parent screen class appears greyed.

The main screen classes have been created.



As we develop transactions, additional screen classes will be needed in the project.

As mentioned before, Google servers automatically redirect Internet users to country-specific Google search pages. In this case, you need to create a country-specific screen class, which is detected when accessing the country-specific Google search page.

The following sub-section describes how to create such screen class.



The following extra screen class is needed only if you access the www.google.com Web site from outside the USA and need CWI to be redirected from your country-specific Google page to the Google.com page in English.

```
CASE OF A COUNTRY-SPECIFIC GOOGLE SEARCH PAGE - DEFINITION OF THE googleSearchPageFR SCREEN CLASS
```

When CWI connects to the www.google.com Web site, if a Google country-specific page exists, it automatically opens in the browser.

Country-specific Google homepages can be defined as standard Google search pages with a *redirecting* hypertext link in the lower right corner of pages. This link can be:

a Google.com in English link in non-anglophone country specific pages:



Figure 2 - 34: Example of country-specific non-anglophone Google search page (France)

a Go to Google.com link in anglophone country-specific pages:





Figure 2 - 35: Example of country-specific anglophone Google search page (UK)

To detect country-specific pages, we will define a country-specific screen class (called googleSearchPageFR in our project) having as detection criteria the redirecting link as well as criteria already defined for the googleSearchPage screen class.

The country-specific screen class can be defined as follows:

- it is a child screen class of the googleSearchPage screen class and, as such, inherits its criteria:
 - "Web" link in upper left corner of the Google page;
 - Google logo.
- it is characterized by the presence of a redirecting link (Google.com in English or Go to Google.com) in the lower right corner of the page.

To create a country-specific (googleSearchPageFR) screen class

- 1 In the country-specific Google search page displayed in the **Web Browser**, right-click on the redirecting link (**Google.com in English** or **Go to Google.com**).
 - The link is outlined in green in the **Web Browser** and its corresponding element is highlighted in green in the **DOM Tree**:



Figure 2 - 36: Selection of a detection criterion for the country-specific screen class

We will expand the A element node to see if it contains interesting attributes to generate the XPath corresponding to the link.

2 Expand the A element in the **DOM Tree** by clicking on \blacksquare :



Figure 2 - 37: Expanding an element to find relevant attributes in the DOM Tree for the country-specific screen class

Two nodes are associated with the A element: an href attribute node, which contains the URL address of the hypertext link, and a text node (Google.com in English or Go to Google.com), which represents the element text content.

The href attribute is not interesting as regards XPath generation because its value may often change. We will select only the text content and generate the XPath from it.

3 Right-click the text node To Google.com in English or To Google.com and select

Generate selection XPath (Enter) or select it with a left-click and press Enter.

The XPath expression of the selected node appears in the XPath Evaluator:





Figure 2 - 38: XPath expression of country-specific redirecting link

- **4 Select the parent screen class**, googleSearchPage.
- 5 In the XPath Evaluator, click on the Create new child screen class from current

XPath icon [

The New Screen Class opens.

- 6 Click on Screen class, then on Next.
- 7 Type in the name of the new screen class (for example googleSearchPageFR).
- 8 Click on **Finish**.

The screen class now appears in the **Projects View** as an *inherited* screen class of the googleSearchPage screen class:



Figure 2 - 39: googleSearchPageFR screen class in Projects View

The criterion previously defined for the screen class is automatically generated and appears in the related **Criteria** folder. Criteria inherited from the parent screen class appear greyed.

9 Save your project by clicking on is or by pressing Ctrl + S.

The next steps now consist in:

- 1 *Creating* the *extraction rule* associated with the googleResultsPage screen class.
- 2 Defining a transaction.

2.2.3 Creating an Extraction Rule

This step shows how to create a new extraction rule associated with a given screen class.

The logical process for creating an extraction rule is the following:

- 1 Generate the root XPath of the extraction rule in the **XPath Evaluator** HTML elements (data) to be extracted will be localized using relative XPaths based on this root XPath. We will choose as root XPath the googleResultsPage criterion's XPath, (//DIV[@id="res"]), which addresses the results container in the Web page.
- 2 Create a new extraction rule by launching the New Extraction Rule wizard and choose an extraction rule template. In this sample project, the extraction rule template is *Table*. This is the best-suited form for creating a list (called listResults) of Web page titles and URLs. This table can be defined as follows:

Table element	Name	Description	
Line	resultItem	Each table row corresponds to a Google result.	
Column	title and url	Two columns per row - corresponding to Web page titles and URLs	

Table 2 - 1: Table extraction rule - line and column description

- set extraction rule properties in the Properties View.In this project:
 - set row name and row XPath (Description property in the Properties View).
 - set column names and column XPaths (Description property in the Properties View).



1

You can also choose to start by launching the **New Extraction Rule** wizard after having right-clicked on the relevant screen class. In this case, you will have to set the root XPath property in the **Properties View**.

To create an extraction rule

Type in the root XPath (//DIV[@id="res"]) in the **xPath** field of the **XPath Evaluator**



You can also generate the root XPath from the **DOM Tree** ("To create the googleResultsPage screen class" on page 2 - 18, steps 1 to 4).

2 Click on the Create new extraction rule from current XPath icon in the XPath Evaluator.

A New Extraction Rule wizard opens:





Figure 2 - 40: New Extraction Rule wizard

- 3 Click on Table.
- 4 Click on Next:

2	
Informations Please enter a name for object.	
New_Table	
⑦ < <u>Back</u> <u>Next</u> > Einish	Cancel

Figure 2 - 41: Giving a name to the new extraction rule

- 5 In the Name field, enter a name (for example googleResults).
- 6 Click on Next:

4	
New Table Please configure rows and lines	
XPath : //DIV(@id="res"] Headers ✓ Headers at row ① + -	
⑦ < <u>Back</u> Next > Einish	Cancel

Figure 2 - 42: HTML tables definition

This window is a help to setting properties. It recalls the root XPath that has previously been set and displays parameters of HTML tables. It is used only if the XPath specified prior to launching the **New Extraction Rule** wizard corresponds to a HTML table. This is not the case in our project.

7 Click on **Finish**.

The new extraction rule appears in the related Extraction rules folder:



Figure 2 - 43: googleResults extraction rule in Extraction rules folder

8 Select the extraction rule, its properties are displayed in the **Properties View**:



Properties 🛛		19 2 - 2 🛛 🗘			
Property	Value				
Configuration					
Accumulate data in same table	false				
Comment					
Display referer	false				
Flip table orientation	false	VML to a posting			
Is active	true				
Tag name 🖌 🛶 🛶 🛶 🛶	XMLTable	transaction			
🚊 Information		results in the			
Depth	n/a	XML output			
Java class	com.twinsoft.convertigo.be	апьтсопшонтлиствое			
Name	googleResults				
Priority	1261149135308				
QName	/sample_documentation_CWI/_data/cn/QQMIK				
Туре	Table				
Selection					
XPath <	//DIV[@id="res"]	Root XPath			

Figure 2 - 44: Extraction rule properties to be set

We must now set the following extraction rule parameters:

Table 2 - 2: Extraction rule properties

Property	Description		
Tag name	Name of the XML table structure in the XML output. Defines the tag name that will be created in the XML output after extraction. Value in this project: listResults.		
Description	Description of the properties of the XML table rows and columns: name, XPath, application of the rule on children elements.		
XPath	XPath corresponding to the HTML elements (Web page title, URL in this project) to be extracted.		

9 Set the **Tag name** property to listResults.

10 If the **XPath** property is empty, set it to //DIV[@id="res"].



This is the case if the **New Extraction Rule** wizard has been launched by right-clicking on the relevant screen class and selecting **New** > **Extraction rule**.

We will now set the table properties. The first step is to specify the *row* properties: name (resultItem) and XPath, knowing that the XPath must correspond to Google results (one row = one Google result).

To set the row properties of the table extraction rule

1 When in a Google result page, right-click on the right of the first result.



Figure 2 - 45: Selection of an XPath for Google result items

An LI element is highlighted in green in the DOM Tree.

This element contains the nodes of Google result items, and consequently the nodes of Web page titles and URLs that we want to extract.

Therefore, the LI node is a good candidate for generating the XPath corresponding to Google result items (i.e. for generating the XPath that must be set as the table row XPath).

So we will now:

- evaluate the root XPath;
- browse the resulting **Document** tree structure to find the element corresponding to Google results (LI element);
- generate its XPath.
- 2 Type in the root XPath //DIV[@id="res"] in the xPath field of the XPath Evaluator.
- 3 Press Enter or click on the Evaluate icon .

The resulting **Document** tree displays the corresponding tree structure:

	xPath //DIV[@id="res"]	× ×
<u>S</u>	Document	
[]	🖃 🗣 root	
۲		
		\sim

Figure 2 - 46: Root XPath expression and tree

4 Expand the **Document** tree to find an LI element:



×Path //DIV[@id="res"]		~ ~
Document ■ root ■ o DIV ■ • ● DIV ■ • ● OL ■ • ● OL ■ • ● OL ■ • ● UI ■ • ● UI	Convertigo - C-EMS Web ClipperLes "cliplets" sont créées par les développeurs qui utilisent le Studio de développement Convertigo basé sur Eclipse. Le studio Convertigo dispose d'une www.convertigo. fr/fr//convertigo- web-clipper_2.html - En cache	

Figure 2 - 47: Root XPath expression and expanded tree



To check that LI elements do correspond to Google result items, rightclick any LI node in the **Document** field of the **XPath Evaluator** and select **Show in browser**. The corresponding Google result is outlined in green in the **Web Browser**.

5 Right-click on an LI element and select Generate selection XPath (Enter) or select the element with a left-click and press Enter:



Figure 2 - 48: Generating the table row XPath corresponding to Google results

The xPath field is updated and displays the relative XPath expression of Google results:

×Path	//DIV[@id="res"]//LI
5	and the second statement of the

Figure 2 - 49: Google results XPath based upon root XPath

We will now set this XPath using the extraction rule description sub-object.

6 In the **Projects View**, right-click on the extraction rule description.

A contextual menu appears:



Figure 2 - 50: Creating a new row

- 7 Select New > Row.
- 8 A new row sub-object appears as child of the description sub-object:



Figure 2 - 51: New row

9 Select the row sub-object with a left-click.

The Properties View is automatically updated:

Properties 🛛		ŧ.		\bigtriangledown	
Property	Value				
🔲 Configuration					
Row tag name	row				
Selection					
Row XPath	.//TR				
<					>

Figure 2 - 52: Row properties

10 In the **Properties View**, click in the **Value** column of the **Row tag name** property.

The default value (row) is highlighted.

- 11 Enter the required row tag name (resultItem).
- 12 Press Enter.

In the Projects View, the row sub-object is automatically renamed.

- In the Properties View, click in the Value column of the Row XPath property.The default value (. //TR) is highlighted.
- 14 Enter the required XPath (. //LI).



This is the *relative* XPath corresponding to Google result items: the "dot" appearing before the double *slash* stands for the root XPath specified in the **XPath** property of the extraction rule. This means: "all LI nodes included in the parent node identified by the root XPath".

- 15 Press Enter.
- **16** Save your project by clicking on 🤖 or by pressing Ctrl + S.

The resulItem row is properly set.

We will now set the columns, title and url.

17 In the **Projects View**, right-click on the resulItem sub-object.

A contextual menu appears:



Figure 2 - 53: Creating a new column

18 Select *New > Column*.

A new column sub-object appears as child of the resultItem sub-object:



Figure 2 - 54: New column

19 Select the column sub-object with a left-click.

The Properties View is automatically updated:

Properties 🛛		[1] → 💀 🗸 🗖
Property	Value	
📃 Configuration		
Column tag name	column	
Selection		
Column XPath	./TD	
Extract children	true	

Figure 2 - 55: Column properties

20 In the **Properties View**, click in the Value column of the Column tag name property.

The default value (row) is highlighted.

- 21 Enter the required row tag name (title).
- 22 Press Enter.

In the **Projects View**, the column sub-object is automatically renamed.

23 Repeat points 17 to 22 to create the second column: url.

Table columns are now created. They appear in the **Projects View**:



Figure 2 - 56: Extraction rule columns in Projects Folder

24 Save your project by clicking on 🤖 or by pressing Ctrl + S.

Their XPaths (**Column XPath** property) are set by default to ./TD, which only applies to HTML table extraction. It is not an appropriate XPath in our project.

The next steps consist in finding appropriate XPaths for Web page titles and URLs and in setting them as values of the **Column XPath** property for the two previously created columns..

To set the column properties of the table extraction rule

- 1 In the **DOM Tree**, select a LI node corresponding to a Google result item.
- 2 Expand the node by clicking on \square .



Figure 2 - 57: Expanding the LI element to find an XPath corresponding to Web page titles

We are looking for an XPath corresponding to Web page titles. The LI node contains an H3 element (standard Heading3 HTML node). This is a relevant element for



generating the required XPath.

3 Select H3 in the **Dom Tree**.

As we can see in the **Web Browser**, the title of a Google result item is outlined in green:



Figure 2 - 58: H3 node corresponding to a Web page title

The H3 element is the container of the text we want to extract (Web page titles).

We will now:

- evaluate the row XPath;
- browse the resulting **Document** tree structure to find the element corresponding to Web page titles (H3 element);
- generate its XPath.
- 4 Type in the row XPath in the **xPath** field of the **XPath Evaluator**.
- 5 Press Enter or click on the Evaluate icon

The resulting **Document** tree displays the corresponding tree structure:

	xPath //DIV[@id="res"]//LI	
		<u>M</u>
<u>S</u>	Document	· · · · · · · · · · · · · · · · · · ·
🕑 🔘	😑 🌻 root	
	i ⊕ u	
l 🔊 🕐		
	<u>∎</u> ● LI	
	<u>∎</u> ● LI	
	<u>⊞</u> ● LI	
		<u> </u>

Figure 2 - 59: Row XPath expression and tree

6 Expand the Document tree to find an H3 element



To check that H3 elements do correspond to Web page titles, right-click any H3 node in the **Document** field of the **XPath Evaluator** and select **Show in browser**. The corresponding Web page title is outlined in green in the **Web Browser**. All Web page titles are selected by this XPath.

7 Right-click on an H3 element and select Generate selection XPath (Enter) or select the element with a left-click and press Enter:



Figure 2 - 60: Generating the XPath expression of Web page titles

The **xPath** field is updated and displays the relative XPath expression of Web page titles:

xPath	//DIV[@id="res"]//LI//H3	
Ln A	manthe all the last and the second	~

Figure 2 - 61: Web page titles XPath based upon row XPath

We will now set this XPath as value of the **Column XPath** property for the title column.

8 In the **Projects View**, select the title column with a left-click.

The **Properties View** is automatically updated.

9 In the **Properties View**, click in the **Value** column of the **Column XPath** property.

The default value (./TD) is highlighted.

- **10** Type in the generated XPath by prefixing it with a "dot": . / /H3.
- 11 Press Enter.
- 12 Save your project by clicking on 🤖 or by pressing Ctrl + S.

The title column is now properly set.

We will now repeat the same procedure to find an appropriate XPath for URLs.

13 In the Web Browser, right-click on the URL of the first Google result:





Figure 2 - 62: Selection of a node for generating the XPath of URLs

A CITE element is highlighted in green in the **DOM Tree**. It is the container of the text we want to extract (URL).

We will now:

- evaluate the row XPath.
- browse the resulting **Document** tree structure to find the element corresponding to URLs (CITE element);
- generate its XPath.
- 14 Type in the row XPath in the **xPath** field of the **XPath Evaluator**.
- 15 Press Enter or click on the Evaluate icon

The resulting **Document** tree displays the corresponding tree structure:

	xPath //DIV[@id="res"]//LI	< >
<u>SO</u>	Document	<u> </u>
[]	🖃 🌒 root	
S		
	EI II	
	∎ ● LI	
	E • ● LI	
	<	

Figure 2 - 63: Row XPath expression and tree

16 Expand the **Document** tree to find a CITE element.



To check that CITE elements do correspond to URLs, right-click any CITE node in the **Document** field of the **XPath Evaluator** and select **Show in browser**. The corresponding URL is outlined in green in the **Web Browser**. All URLs are selected by this XPath.

17 Right-click on a CITE element and select Generate selection XPath (Enter) or select the element with a left-click and press Enter:



Figure 2 - 64: Generating the XPath expression of URLs

The **xPath** field is updated and displays the relative XPath expression of URLs:



Figure 2 - 65: URL XPath based upon row XPath

We will now set this XPath as value of the **Column XPath** property for the url column.

18 In the **Projects View**, select the url column with a left-click.

The **Properties View** is automatically updated.

19 In the Properties View, click in the Value column of the Column XPath property.

The default value (./TD) is highlighted.

- 20 Type in the generated XPath by prefixing it with a "dot": . //CITE.
- 21 Press Enter.
- 22 Save your project by clicking on 🤖 or by pressing Ctrl + S.

The googleResults extraction rule is now properly set.

You can now check that the newly created extraction rule is correct and creates a list of results in XML format with the required XML tag names.

To check that your extraction rule is correct

Assuming that the **Web Browser** displays a Google results page, click on the **Generate XML** icon





Figure 2 - 66: Generation of XML document

2 Click on the **Output** tab at the bottom of the **Connector View**.

Extracted data appears in XML format:


Figure 2 - 67: Result of the XML generation displayed under the Output tab

All of the XML tag names previously set when creating the extraction rule appear in the XML document:

XML	Browser			-
xml ve</th <th>ersion="1.</th> <th>0" encoding="ISO-8859-1"?></th> <th></th> <th></th>	ersion="1.	0" encoding="ISO-8859-1"?>		
		-ten "Carala Caralastan" arabanta "c		
listR	esults > <	ttor= GoogleConnector context= :	Extraction rule tag	name
<re< th=""><th>sultItem></th><td>, ι</td><td></td><td>,</td></re<>	sultItem>	, ι		,
<	(title>Con	vertigo Web Clipper		
A 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	url>www	.convertigo.com/pdf/datasheets/con		i
	esuititem: suititem a		Row tag name	
<	title>	MS Web Clipper		
<	url>www	.twinsoft.com/index2.php?option=co	om_content <mark>&</mark> amp;do_pdf=:	1 🌖
<th>esultItem:</th> <th>> </th> <th>Column to a name</th> <th></th>	esultItem:	> 	Column to a name	
re	्यद	The second s	Column tag name	s 🗸

Figure 2 - 68: Structure of the XML output

Now that screen classes and extraction rule(s) have been defined, we will create a *transaction* (the process that automates navigation and extraction throughout Web pages) based on:

- screen class handlers components of the transaction triggered on detection of screen classes;
- statements actions performed on the Web page when a screen class handler is triggered.



The next section shows how to define transactions, screen class handlers and statements.

2.2.4 Defining a Transaction

As mentioned in *"Transaction"* on page 1 - 5, a transaction can be summed up as a process:

- taking variables in input;
- producing an XML document in output.

Only one variable is defined in the transaction of this sample project: the Google search keyword.

The XML output is produced on execution of the googleResults extraction rule on several pages. It is quite similar to the one produced by the **Generate XML** command (see Figure 2 - 67).

The first steps are the following:

- 1 *Create* the transaction.
- 2 Create a screen class handler triggered on detection of the googleSearchPage screen class.

The first screen class handler to be created is the one that is triggered on entry to the first page CWI connects to (depending on URL parameters set for the connector): the Google search page. That is why the first screen class handler is associated to the googleSearchPage screen class.

- 3 Associate two statements with this screen class handler:
 - first statement = "input keyword";
 - second statement = "click on the Google Search button".

To create a transaction

- 1 In the **Projects View**, right-click on the connector (GoogleConnector in our project) or on the **Transactions** folder.
- 2 Select *New > Transaction*.

A New Transaction wizard opens:

2	
New Transaction Please select a transaction template.	
() HTML transaction	
Image: Constraint of the second se	Cancel

Figure 2 - 69: New Transaction wizard

3 Select HTML transaction.

4 Click on Next:

~			
Inform	nations		
Please	enter a name for object.		
Name:	New_HTML_transaction		
?	< Back	Next > Finish	Cancel

Figure 2 - 70: Entering a new transaction name

- 5 Enter the transaction name (for example searchGoogle).
- 6 Click on Finish.

The new transaction appears in the Transactions folder of the Projects View:





Figure 2 - 71: New transaction in Projects View

To set a transaction variable

We will now set the transaction *variable*, that is the transaction input data (i.e. search keyword).



Any transaction variable is added to the executed transaction JavaScript scope.

1 In the **Projects View**, right-click on the transaction (searchGoogle in this project)...

A contextual menu appears.





2 Select New > Variable.

A New Variable wizard is automatically launched.

3 Select Variable.

New Variable Please select a variable template.
Defines a requestable variable
Cancel

Figure 2 - 73: New Variable wizard

- 4 Click on Next;
- 5 In the **Name** field that appears, type in the name of the variable (for example keyword).

4	
Informations Please enter a name for object.	
Name: keyword	7

Figure 2 - 74: Giving a name to the new variable

6 Click on Finish.

The new variable appears in the transaction Variables folder in the Projects View:



Figure 2 - 75: New variable in transaction Variables folder

7 Save your project by clicking on 🤖 or by pressing Ctrl + S.



The transaction variable is now created and saved. We can now set its properties.



Only the **Description** property is set here as an example. For example, you can set a **Default value**. A default value is used as default search keyword if none is given in the transaction. This can be useful for setting parameters such as languages for example, which are supposed to be fixed.

8 Select the keyword variable in the **Projects View**.

The **Properties View** is automatically updated:

Properties 🛛		Ē	 ⇒i	Ŗ	\bigtriangledown		3
Property	Value						
📃 Expert							
Cached key	true						
Personalizable	false						
Schema type	xsd:string						
WSDL exposed	true						
🖃 Information							
Depth	n/a						
Java class	com.twinsoft.convertigo.be	ans.	varia	bles.	Req	uestat	əli
Name	keyword						
Priority	1258546409444						
QName	/sample_doc_CWI/_data/cn	/QQ	MIKU	b/tr/	MM-	-3ilN/vl	bį
Туре	Variable						
Properties							
Comment							
Default value							
Description	new variable						
HTTP method	POST						
HTTP name							
Multivaluated	false						
6 - 18							
<						>	Ł

Figure 2 - 76: Variable properties

9 Click in the Value column of the **Description** property.

The property default value (new variable) is highlighted.

- **10** Type in a description for the variable (for example keyword to be searched).
- 11 Press Enter.

The value property is updated.



In the context of this Quick Guide project, all other values are left as default values.

The transaction variable is now created and properly set.

12 Save your project by clicking on 🤖 or by pressing Ctrl + S.

The previously defined variable is a Google search keyword to be entered in the search field

of the Google homepage.

We will now create the first screen class handler corresponding to the googleSearchPage screen class and add the required statements.

To create a screen class handler

- 1 Right-click on the transaction name (in our project, searchGoogle).
- 2 Select **New > Handler** in the contextual menu.

A New transaction handler window opens:



Figure 2 - 77: New transaction handler window

This window is divided into:

- a first part dedicated to screen class handlers (with three checkboxes and a dropdown menu);
- a second part dedicated to other types of handlers (with four checkboxes).

The table below describes these elements:

Table 2 - 3: New transaction handler window description

Window Element		Description
Screen classScreen classhandler settingstransactionCheckboxeshandler		Check this box if you want to create a screen class handler (as opposed to other available handlers) that wil be triggered on screen class detection.
	Entry handler	Check this box if screen class handler must be triggered when accessing the screen class. An entry screen class handler handles the following actions: access Web page, execute statements, exit to next Web page. Available only if Screen class transaction handler checkbox is checked.



Window Element		Description		
	Exit handler	Check this box if the screen class handler must be triggered when exiting the screen class. An exit screen class handler handles the following actions: access Web page, extract data according to extraction rules, execute statements, exit to next Web page. Available only if Screen class transaction handler checkbox is checked.		
Drop-down menu	Screen class	Displays the list of screen classes available in the project.		
Other handlers settings <i>Checkboxes</i>	Start of transaction	Check this box if you want to create a handler that will be triggered when starting the transaction.		
	XML generation	Check this box if you want to create a handler that will be triggered at the end of the transaction (once the XML output has been generated).		
	Default transaction entry handler	Check this box to create a default transaction entry handler.		
	Default transaction exit handler	Check this box to create a default transaction exit handler.		

Table 2 - 3: New transaction handler window description (...)

The screen class handler we are about to create is triggered when accessing the home Google search page, so its parameters are the following:

- this is a screen class transaction handler, meaning that all the screen class handler settings of the New transaction handler window (see Figure 2 - 77) must be set;
- this is an *entry* handler, because no data has to be extracted from this page;
- it is triggered on detection of the googleSearchPage screen class.

We will now set the scren class handler parameters:

- 3 Check the Screen class transaction handler checkbox.
- 4 In the ScreenClass drop-down menu, select googleSearchPage.
- 5 Check the Entry handler checkbox.
- 6 Click on OK.

The screen class handler ongoogleSearchPageEntry appears in the Functions folder of the searchGoogle transaction of the Projects View:

Q Default_HTML_Transaction			
🚊 🕜 searchGoogle			
🖻 🗁 Functions			

Figure 2 - 78: New entry screen class handler in the Projects View

Note the left to right yellow arrow \Rightarrow identifying *entry* screen class handlers.

The following step consists in creating the statement that will enter the keyword in the search field. This is done by generating first the XPath corresponding to the Google

search field, for Convertigo to know where the search keyword is to be entered.

To create a statement

1 Assuming that the **Web Browser** displays the Google search page, right-click on the Google search field.

The search field is outlined in green in the **Web Browser** and the corresponding element is highlighted in green in the **DOM Tree**:



Figure 2 - 79: Selection of Google search field in Web Browser



2 Expand the DOM Tree to find an appropriate node:

Figure 2 - 80: Expansion of the DOM tree to find an appropriate node for the search field

As mentioned before, the name HTML attribute is a good candidate for generating XPaths (as well as id attributes).

3 Right-click on name="q" and select **Generate selection XPath (Enter)** or select name="q" with a left-click and press **Enter**.

The XPath corresponding to the Google search field is displayed in the **xPath** field of the **XPath Evaluator**.

As we can see in the **Document** field, this Xpath corresponds to one element only:





Figure 2 - 81: Prevously generated XPath corresponds to one element of the Web page only

- 4 In the **Projects View**, select the ongoogleSearchPageEntry screen class handler, because we want to create the statement in this handler.
- 5 In the XPath Evaluator, click on the Generate new statement from current XPath

icon **o**, which is active because a suitable handler or container is selected in the **Projects View**).

A New Statement wizard opens.

6 Click on **Input HTML set value** statement, which sets a value into an HTML input field:

~		
New Statement Please select a statement template.		
🚺 Break	Browser Property Change	Container statement
Context Add Text Node	Context Get	Context Set
Cookies Add	Cookies Get	💋 Create event
A Credentials Statement	S Default entry Handler	🕵 Default exit Handler
🏹 DoWhile	K Entry Handler	K Exception Statement
🕵 Exit Handler	K Function	🕵 Get Attachement
Set nodes	🕵 Get url	🕖 HTTPStatement
🕵 Handler	? If	 C Input HTML set checked
10 Input HTML set selected	txt] Inpact ML set value	🦻 Invoke Browser JS
Key action	kog 🗸	Mouse action
Mouse action advance	Navigation Bar	1 Return Statement
Simple Statement	Tab Management	⊾) While
0	< Back Next >	Finish Cancel

Figure 2 - 82: New Statement wizard - Input HTML set value

- 7 Click on Next.
- 8 Leave the statement name untouched or enter a new statement name in the Name field (for example set_keyword_value):

Informations	
Please enter a name for object.	
Name: set_keyword_value	

Figure 2 - 83: Proper statement name



If the statement name is several words long, separate each word with an underscore instead of a blank space. Otherwise, an error message will appear in the Informations part of the window (see Figure 2 - 84).

Inform	nations	5
🔇 Nan Don'	ne must be normalized. 't start with number and don't use non ASCII caracters.	1
Name:	set keyword value	
		<
	the second second second second second second second	

Figure 2 - 84: Improper statement name

9 Click on **Finish**. The new statement appears in the **Projects View**:

earchGoogle
😑 🗁 Functions
🖃 🔿 ongoogleSearchPageEntry
txi set value of //todo

The variable whose value must be set has still to be specified (corresponding to the / /todo comment).

10 Select the newly created statement with a left-click.

The **Properties View** is automatically updated:



Figure 2 - 85: New Input HTML set value statement

Properties 🛛	[변화 🖾 🗸 🗖
Property	Value
🖃 Expert	
Synchronisation	Wait time timeout:0
Information	
Depth	n/a
Java class	com.twinsoft.convertigo.beans
Name	setValue_keyword
Priority	1245404432999
QName	/sampleGoogle/_data/cn/QQMI
Туре	Input HTML set value
Properties	
Commentaire	
Est active	true
Evénement d'interface utilisateur	false
Expression	//todo
XPath	'//INPUT[@name="q"]'

Figure 2 - 86: HTML Input set value statement properties

- 11 Click on the value of the **Expression** property (//todo).
- 12 Click on . A JavaScript expression window opens:

Javascript expression	
//todo	
	OK Cancel

Figure 2 - 87: Edition of the Expression property

This form contains JavaScript expressions used as input for transactions.

- 13 Delete //todo.
- **14** Type in the name of your variable: keyword.



Any kind of JavaScript expression can be entered in this form. For example, keyword + "toto".

The statement is updated in the **Projects View**:



Figure 2 - 88: Updated statement

15 Save your project by clicking on **[**] or by pressing Ctrl + S.

We will now repeat the process (steps 1 to 9 of this procedure) to create the second statement of the ongoogleSearchPage screen class handler, which can be summed up as "click on the **Google Search** button".

16 In the **Web Browser**, right-click on the Google **Search** button.

The **Google Search** button is outlined in green in the **Web Browser** while its HTML container is highlighted in green in the **DOM Tree**:



Figure 2 - 89: Selection of the Google Search button in the Web Browser

17 Expand the DOM Tree to find an appropriate element:





Figure 2 - 90: Expansion of the DOM tree to find an appropriate element for the Search button

As mentioned before, the name HTML attribute is a good candidate for generating XPaths (as well as the id attribute). To be sure there will be no conflict with any other HTML element (with same attribute name), we can add the type attribute (type="submit", which is characteristic of HTML form validating buttons).

18 While keeping the Shift key pressed, select name="btnG" and type="submit" with a left-click and press Enter or right-click and select Generate selection XPath (Enter).

The XPath corresponding to the **Google Search** button is displayed in the **xPath** field of the **XPath Evaluator**.



Check in the **Document** field of the **XPath Evaluator** that the XPath corresponds to one element only (see step 3 of this procedure).

- 19 In the **Projects View**, select the ongoogleSearchPageEntry screen class handler, because we want to add a statement to this handler.
- 20 In the XPath Evaluator, click on the Generate new statement from current XPath icon .

A New Statement wizard opens.

21 Click on **Mouse action** statement, which emulates mouse clicks:

-		
New Statement Please select a statement template.		
📔 Break	Browser Property Change	Container statement
Context Add Text Node	Context Get	Context Set
Cookies Add	Cookies Get	Create event
A Credentials Statement	K Default entry Handler	🕵 Default exit Handler
🏹 DoWhile	K Entry Handler	K Exception Statement
Kit Handler	K Function	🕵 Get Attachement
Foet nodes	🕵 Get url	E HTTPStatement
Kandler	? If	 C Input HTML set checked
10 💌 Input HTML set selected	$txt\tilde{I}$ Input HTML set value	🦻 Invoke Browser JS
Key action	🖉 Log	Moraction
Mouse action advance	Navigation Bar	Return Statement
Simple Statement	Tab Management	⊾) While
0	< Back Next >	Finish Cancel

Figure 2 - 91: New Statement wizard - Mouse action

- 22 Click on Next.
- 23 Leave the statement name untouched or enter a new name in the **Name** field (see *Warning* on step 8, page 2-48): for example clickSearchButton.
- 24 Click on Finish. The new statement appears in the Projects View:



Figure 2 - 92: New mouse click statement on entry screen class handler

25 Select the newly created statement with a left-click.

The Properties View is automatically updated:



Properties 🛛	■ 🐉 🖪 🗸 🗖 🗖
Property	Value
🖃 Expert	
Synchronisation	Document Completed:1 timeout:60000
Information	
Depth	n/a
Java class	com.twinsoft.convertigo.beans.statements.MouseS
Name	clickSearchButton
Priority	1245404727749
QName	/sampleGoogle/_data/cn/QQMIKUb/tr/MM-3ilN/st/nY
Туре	Mouse action
Properties	
Action	click
Commentaire	
Est active	true
XPath	'//INPUT[@name="btnG" and @type="submit"]'
	>

Figure 2 - 93: Mouse click statement properties

One property must still to be set: the **Synchronisation** property.

This property answers the following question: "Once the click has been performed, when can the transaction resume (for example once the result page have has been loaded), and what is the allowed maximum waiting time?".

- 26 In the **Properties View**, click on the value of the **Synchronisation** property.
- 27 Click on .

A Trigger Editor window opens:

Trigger editor	
Type of synchronizer Timeout (ms) This synchronizer wait (Document completed 60000 for a number of document completed.
Number of document of	completed 1
	OK Cancel

Figure 2 - 94: Trigger editor window

Three types of synchronizer are available:

Synchronizer type	Description
Document completed	Transaction continues running once a number of document have been successfully loaded. The number of documents must be set in the Number of document completed field of the window: This synchronizer wait for a number of document completed. Number of document completed 1
Xpath	Transaction continues running once an Xpath has been found. The XPath must be set in the Waiting for XPath field: This synchronizer wait while the xpath Waiting for Xpath
Wait time	Transaction continues running after the specified wait time has been reached. With this parameter, you can also decide to monitor any DOM change while waiting: Check this to monitor any DOM changes while waiting. (Can cause heavy CPU load)
Screen class	Transaction continues running once one of the specified screen classes has been reached. The waited screen classes must be selected in the Waiting for Screen Classes list: This synchronizer waits for one of the selected ScreenClasses defined here to be detected. You can select multiple screen classes by holding the Ctrl key while selecting the screen class with the mouse Waiting for Screen Classes GenericWebPages Default_Screen_class GenericWebPage GoogleResultsPage GoogleSearchPage GoogleSearchPageUK

Table 2 - 4: Synchronizer types

- 28 Leave default values unchanged.
- 29 Click on OK.
- 30 Save your project by clicking on 🤖 or by pressing Ctrl + S.

The actions to be performed on detection of the googleSearchPage screen class are now defined.

We must inform the transaction that once all the screen class handler statements have been executed, a new criterion match detection must be carried out on the next accessed Web page without extracting data.

This is done through the Result entry screen class property, which can take the



following values:

Table 2 - 5: Result property values for entry screen class handlers

The value	Means that the application will
 blank>	stop the transaction and extract data from last detected screen class using extraction rules
continue	stop the transaction and extract data from last detected screen class using extraction rules.
redetect	redetect a new screen class without extracting data. This is the default value, which corresponds to what the transaction should do in our case.
skip	stop the transaction without extracting data from last detected screen class.

31 Select the ongoogleSearchPageEntry screen class handler in the Projects View.

Properties 🛛	(변) 🐉 🖾 🗸 🗖 🗖
Property	Value
💻 Information	
Depth	n/a
Java class	com.twinsoft.convertigo.beans.stat
Name	ongoogleSearchPageEntry
Priority	0
QName	/sampleGoogle2/_data/cn/QQMIKUb
Туре	Entry Handler
Properties	
Commentaire	
Est active	true
Result	redetect
Screen class	googleSearchPage

The Properties View is automatically updated:

Figure 2 - 95: Entry screen class handler properties

The first screen class handler, which is intended to input a search keyword and click on the **Google Search** button, is now properly set.



The second screen class handler we are about to create is needed only if you access the www.google.com Web site from outside the USA and need CWI to be redirected from your country-specific Google page to the Google.com page in English.

CASE OF A COUNTRY-SPECIFIC GOOGLE SEARCH PAGE - DEFINITION OF THE ongoogleSearchPageFREntry SCREEN CLASS HANDLER

This screen class handler will be triggered on detection of the googleSearchPageFR screen class, when accessing a country-specific Google page (for example, the French version). Its role will be to handle, through a dedicated statement, the mouse click on the **Google.com in English** link.

This screen class handler can be defined as follows:

• this is a screen class transaction handler,

- this is an *entry* handler, because no data needs to be extracted from the screen class;
- it is triggered on detection of the googleSearchPageFR screen class.

After it has been created, this screen class handler will be associated with the statement allowing to switch back to the English Google search page. This statement is "click the **Google.com in English** link".

To create the ongoogleSearchPageFR screen class handler

- 1 Right-click on the transaction name (in our project, searchGoogle).
- 2 Select New > Handler in the contextual menu. A New transaction handler window opens.



For more information on this window, see Table "New transaction handler window description" on page 2-45.

- 3 Check the Screen class transaction handler checkbox.
- 4 In the ScreenClass drop-down menu, select googleSearchPageFR.
- 5 Check the Entry handler checkbox.
- 6 Click on OK.

The screen class handler <code>ongoogleSearchPageFREntry</code> appears in the Functions folder of the <code>searchGoogle</code> transaction in the **Projects** View:

🖹 🕐 search(Soogle
📮 🗁 Fur	octions
🗄 ·· 🔿	ongoogleSearchPageEntry
	ongoogleSearchPageFREntry

Figure 2 - 96: ongoogleSearchPageFR screen class handler in Projects View

We now need to define the statement that will perform, on detection of the googleSearchPageFR screen class, the required action: "click on the **Google.com in English** link".

To this end, we will use the same XPath as the one used to create the googleSearchPageFR screen class, corresponding to the mentioned link.

7 Assuming that the **Web Browser** displays a country-specific Google search page, right-click on the **Google.com in English** link.

The **Google.com** in **English** link is outlined in green in the Web Browser and its corresponding element is highlighted in green in the **DOM Tree**:





Figure 2 - 97: Selection of a detection criterion for the googleSearchPageFR screen class

We will expand the A element node to see if it contains interesting attributes to generate the XPath corresponding to the link.

8 Expand the A element in the DOM Tree by clicking on **...**:



Figure 2 - 98: Expanding an element to find relevant attributes in the DOM Tree

As we can notice, two nodes are associated with the A element: a href attribute node, which contains the URL address of the hypertext link, and a text node (Google.com in English), which represents the element content.

The href attribute is not so interesting as regards XPath generation because its value may often change. We will select the text content only and generate the XPath from it.

Right-click the text node Google.com in English and select Generate selection XPath (Enter) or select it with a left-click and press Enter:



Figure 2 - 99: Generation of XPath for the googleSearchPageFR screen class

The XPath expression of the selected node appears in the XPath Evaluator.

- **10** Select the ongoogleSearchPageFREntry handler object in the **Projects view**.
- 11 In the XPath Evaluator, click on the Create new statement from current XPath icon

Projects View.

A New Statement wizard opens.

12 Click on **Mouse action** statement, which emulates mouse clicks:



Figure 2 - 100: New Statement wizard - Mouse action

13 Click on Next.



- 14 Leave the statement name untouched or enter a new name in the **Name** field (see *Warning* on step 8, page 2-48): for example clickLinkGoogleCom.
- 15 Click on Finish.

The new statement appears in the Projects View:



Figure 2 - 101: New mouse click statement on entry screen class handler

The statement's **Synchronisation** property is set by default to the following value:



Figure 2 - 102: clickLinkGoogleCom statement - Synchronization value

You can leave the default value.



For more information on the **Synchronisation** property, see Table 2 - 4 on page 2-55.

16 Save your project by clicking on is pressing Ctrl + S.

The actions to be performed on detection of the googleSearchPageFR screen class are now defined.

The ongoogleSearchPageFR **Result** property is set by default to redetect. This informs the transaction that once all the screen class handler statements have been executed for this screen class, a new criterion match detection must be carried out on the next accessed Web page without extracting data.

You can leave the default value.



For more information on the **Result** property for **entry** screen class handlers, see Table 2 - 5 on page 2-56.

The ongoogleSearchPageFR screen class handler is now properly set.

The next step is to define a third (or second if you have not defined any country-specific screen class and corresponding screen class handler) screen class handler associated with the googleResultsPage. Its role is to handle data extraction (i.e. *XMLisation* of the result page) and exit the result page by navigating to the next page.

This screen class handler can be defined as follows:

this is a screen class transaction handler,

- this is an exit handler, because data needs to be extracted from the screen class;
- it is triggered on detection of the googleResultsPage screen class.

To create the ongoogleResultsPageExit screen class handler

- 1 Right-click on the transaction name (in our project, searchGoogle).
- 2 Select New > Handler in the contextual menu. A New transaction handler window opens.



For more information on this window, see Table "New transaction handler window description" on page 2-45.

- 3 Check the Screen class transaction handler checkbox.
- 4 In the **ScreenClass** drop-down menu, select googleResultsPage.
- 5 Uncheck the Entry handler checkbox and check the Exit handler checkbox.
- 6 Click on **OK**.

The screen class handler ongoogleResultsPageExit appears in the Functions folder of the searchGoogle transaction in the **Projects View**:



Figure 2 - 103: New exit screen class handler in the Projects View

Note the right to left blue arrow *identifying exit screen class handlers*.

We will now create the statement associated to this screen class handler. This statement can be summed up as follows: "in the Google result page, click on the **Next** link".

The following step thus consists in creating the statement that will do so. First, we have to generate the XPath corresponding to the **Next** link of Google result pages.

7 Assuming that the Web Browser displays a Google results page, right-click on the Google Next link.

The **Next** link is outlined in green in the **Web Browser** and the corresponding HTML node is highlighted in green in the **DOM Tree**:





Figure 2 - 104: Selection of the Google results Next link in the Web Browser



8 Expand the **DOM Tree** to find an appropriate node:

Figure 2 - 105: Expansion of the DOM tree to find an appropriate node for the Next button

No other interesting attribute being present in the **DOM Tree**, the Next text node seems the best candidate for generating the XPath corresponding to the **Next** link.

9 Right-click on Twext and select Generate selection XPath (Enter) or select

T Next with a left-click and press Enter.

The XPath corresponding to the Google **Next** link is displayed in the **xPath** field of the **XPath Evaluator**.



Check in the **Document** field of the **XPath Evaluator** that the XPath corresponds to one element only (see step 3 of this procedure).

- 10 In the **Projects View**, select the ongoogleResultsPageExit screen class handler, because we want to add a statement to this handler.
- 11 In the **XPath Evaluator**, click on the **Generate new statement from current XPath** icon .

A New Statement wizard opens.

12 Click on **Mouse action** statement, which emulates mouse clicks:



Figure 2 - 106: New Statement wizard - Mouse action

- 13 Click on Next.
- 14 Leave the statement name untouched or enter a new name in the **Name** field (see *Warning* on step 8, page 2-48): for example clickNext.
- 15 Click on Finish.

The new statement appears in the **Projects View**:





Figure 2 - 107: New mouse click statement on exit scren class handler

16 Save your project by clicking on **[**] or by pressing Ctrl + S.

To complete the setting of the ongoogleResultsPageExit screen class handler, one last property needs to be set: the **Result** property.

In the context of *exit* screen class handlers, this property specifies what results from exiting the HTML page.

The Result property can take the following values:

Table 2 - 6: Result property v	lues for <i>exit</i> screen class handlers
--------------------------------	--

The value	Means that the application will
 blank>	end the transaction.
accumulate	accumulate extracted data (data is extracted from last detected screen class then added to any other extracted data) and redetect a new screen class. This is the default value, which corresponds to what the transaction should do in our case.



For more information on the use of the **Result** property in the context of **entry** screen class handlers, see Table 2 - 5 on page 2-56.

17 Select the ongoogleResultsPageExit screen class handler in the Projects View.

The Properties View is automatically updated:

Properties 🛛	🔁 🍄 💀 🗸 🗖 🗖				
Property	Value				
Information					
Depth	n/a				
Java class	com.twinsoft.convertigo.beans.statements.ScExi				
Name	ongoogleResultsPageExit				
Priority	0				
QName	/sample_doc_CWI/_data/cn/QQMIKUb/tr/MM-3ilN				
Туре	Exit Handler				
Properties					
Comment					
Is active	true				
Result	accumulate				
Screen class	googleResultsPage				

Figure 2 - 108: Exit screen class handler properties

The second screen class handler is now created together with its statement.

An issue is emerging - for any given search, the last Google result pages has no **Next** link:



Figure 2 - 109: Google page with no Next link

Executing the "click Next link" statement therefore returns an error.

Moreover, we need to find an ending condition for the transaction - the absence of a **Next** link is a suitable stopping criteria.

A distinction needs to be made between:

- current result pages with Next link;
- the *final* result page without **Next** link.

The problem can be summed up as follows:

Table 2 - 7: Google result page distinction

A Google result page with	ls a	And its screen class is
a results DIV element and a Next link	current Google page	googleResultsPageCurrent
a results DIV element and NO Next link	final Google page	googleResultsPageFinal

The googleResultsPageCurrent screen class is basically defined as a googleResultsPage screen class, to which an extra criterion must be added: the **Next** link.

- 18 In the **Projects View**, select googleResultsPage with a left-click.
- 19 In the **Web Browser**, right-click on the **Next** link:





Figure 2 - 110: Creation of a new screen class based on the detection of the Next link



20 Expand the **DOM Tree** to find an appropriate node:

Figure 2 - 111: Expansion of the DOM tree to find an appropriate node for the Next button

21 Right-click on The Next and select Generate selection XPath (Enter) or select

TNext with a left-click and press Enter.

The XPath corresponding to the Google **Next** link is displayed in the **xPath** field of the **XPath Evaluator**.

22 In the XPath Evaluator, click on the Create new child screen class from current

XPath icon 🔠

A New Screen Class wizard opens.

- 23 Click on Screen class.
- 24 Click on Next.

- 25 In the Name field, enter googleResultsPageCurrent.
- 26 Click on **Finish**.

The new screen class appears in the **Projects View** as an *inherited* screen class of the googleResultsPage screen class. Its inherited criteria and extraction rule are greyed:

🚊 ا 🔤 googleResultsPage
🕀 🗁 Criteria
🕀 🗁 Extraction rules
🖻 🗁 Inherited screen classes
🖻 🔠 googleResultsPageCurrent
🖃 🗁 Criteria
Exists node at "//A[contains(text(),"Next")]"
🖻 🗁 Extraction rules
🖻 🛞 googleResults
🖮 😐 description
🖮 😐 resultItem
···· • title
ur ur

Figure 2 - 112: googleResultsPageCurrent screen class in the Projects View

We will now create the googleResultsPageFinal screen class.

The existing googleResultsPage screen class can simply be assimilated with the googleResultsPageFinal screen class, since it is based on detection of the results DIV element. If no Next link is detected, the screen class is not assimilated with the googleResultsPageCurrent screen class.

We just need to rename the parent screen class:

- 27 Right-click on the googleResultsPage screen class in the Projects View.
- 28 Select **Rename**. The screen class name is highlighted.
- **29** Type in googleResultsPageFinal and press Enter.



When renaming a screen class, all screen class handlers are automatically renamed and the values of their screen class properties are changed.

The screen class handler ongoogleResultsPageExit becomes ongoogleResultsPageFinalExit.



Figure 2 - 113: ongoogleResultsPageFinalExit screen class handler in the Projects View

This poses a problem in our project because the "click on Next" statement applies to



current Google result pages, not to *final* result page: we therefore need to associate the screen class handler with the proper screen class.

30 Select the ongoogleResultsPageFinalExit in the **Projects View**.

The Properties View is automatically updated:

🔲 Properties 🛛		Ë	⇒i ⇒i	R.	\bigtriangledown	
Property	Value					
Information						
Depth	n/a					
Java class	com.twinsoft.convertigo.beans.stat					
Name	ongoogleResultsPageFinalExit					
Priority	0					
QName	/sampleGoogle/_data/cn/QQMIKUb/					
Туре	Exit Handler					
Properties						
Commentaire						
Est active	true					
Result	accumulate					
Screen class	googleResultsPageFinal					

Figure 2 - 114: ongoogleResultsPageFinal properties

31 Click on the value of the **Screen class** property. The associated screen class is highlighted and a drop-down menu appears:

🔲 Properties 🗙		Ē] ≱¦⊳	4	\bigtriangledown	
Property	Value					
Information						
Depth	n/a					
Java class	com.twinsoft.convertigo.bea	ns.sl	tat			
Name	ongoogleResultsPageFinalExit					
Priority	0					
QName	/sampleGoogle/_data/cn/QQI	MIKL	ιь/			
Туре	Exit Handler			_		
Properties	googleResultsPageCurrent		^			
Commentaire	googleResultsPageFinal					
Est active	looogleSearchPageFR					
Result	googleWebPages					
Screen class	googleResultsPageFinal		~			
				_		

Figure 2 - 115: Screen class handler properties - Screen class drop-down menu

- **32** Select googleResultsPageCurrent.
- **33** Save the project by clicking on 🕞 or by pressing Ctrl + S.



No screen class handler is needed in the transaction for the googleResultsPageFinal screen class because no action needs to be performed on this page. The transaction will just stop when accessing this page (after execution of extraction rules).

2.2.5 Executing a Transaction

The Web integration project is now set up.

You can test the transaction either:

- by executing it from the Studio using the Execute command;
- by calling a properly formatted URL from a Web browser to access a test platform included in the project.

To execute the transaction from the Studio, the input variable defined for the transaction (keyword, see "To set a transaction variable" on page 2-42) must be given a default value.

To execute the transaction from the Studio

1 In the **Projects View**, left-click on the transaction variable (keyword).

The Properties View is automatically updated.

- 2 Click in the Value column of the Default Value property.
- 3 Enter a search keyword (for example convertigo cliplet).
- 4 Press Enter.
- 5 Save the project by clicking on 🔖 or by pressing Ctrl + S.

The transaction variable is now set as a fixed variable with a default value.

- 6 In the **Projects View**, right-click on the transaction searchGoogle.
- 7 Select Execute.

The transaction starts. The **Connector Editor** of the CWI Studio shows the successive steps of the transaction being executed in real time.

Results appear in the **Output** tab of the **Connector View**:





Figure 2 - 116: XML output without accumulate data in same table property set

The XML output is structured into several distinct XML tables (two in our project), each starting and ending with a listResults tag name (the extraction rule name). These tables correspond to the Google result pages successively detected and from which Web page titles and URLs have been extracted.

8 To accumulate data in a single XML table, select the extraction rule googleResults with a left-click in the **Projects View**.

The **Properties View** is automatically updated.

9 Set the extraction rule Accumulate data in same table property to true:

Properties 🛛	[≌] \$\$ ₪ ▽ □ □]
Property	Value
Configuration	
Accumulate data in same table	true
Comment	
Display referer	false
Flip table orientation	false
Is active	true
Tag name	listResults
Information	
Depth	n/a
Java class	com.twinsoft.convertigo.beans.common.XMLTable
Name	googleResults
Priority	1245402035240
QName	/sample_documentation_CWI/_data/cn/QQMIK
Туре	Table
Selection	
XPath	//DIV[@id="res"]

Figure 2 - 117: Extraction rule properties - Accumulate data in same table

- 10 Save the project by clicking on 🙀 or by pressing Ctrl + S.
- **11** Relaunch the execution of the transaction using the **Execute** command (steps 6 and 7 of this procedure).

Results are now accumulated in a single XML table:





Figure 2 - 118: XML output with accumulate data in same table property set

To test the transaction using the test platform

- Launch a standard Web browser.
- 2 In the URL address field, type in the URL in the following format:



For example, in our project:

- HostName = localhost;
- ConvertigoPortNumber = 18080;
- ProjectName = sampleGoogle;

For example, in a Mozilla Firefox URL address bar:

http://localhost:18080/convertigo/projects/sample_doc_CWI/

Figure 2 - 119: Convertigo test platform URL

The test platform opens:

🖉 Index - Windows Internet Explorer				
😋 💽 👻 http://localhost:18080/conv	vertigo/projects/sample_doc_CWI/	v (+)	× Google	P -
🚖 🕸 💌 Index		1	🟠 🔹 🔝 🔹 🌐 🔹 🔂 Page 🗸	💮 Outils 👻 🎽
CONVERTIGO ENTERPRISE MASHUP Sample_doc_CWI project Web Integration project GaogleConnector connector	SERVER			
Cliplet name	Add to	Parameters	Comment	
Default HTML Transaction	🛨 Google 💽 netvibes			
searchGoogle_	Coogle entribes	keyword convertigo cliplet new variable		
			😏 Intranet local	💐 100% 🔻 📑

Figure 2 - 120: Test platform

It displays:

- in a first table:
 - the project name (in our example, sampleGoogle);
 - the **project type** (a Web integration project type);
- in a second table:
 - the connector defined for the project (in our example, GoogleConnector);
 - all transactions defined for the connector (in our example, Default_HTML_Transaction, searchGoogle) and their variables (keyword for the searchGoogle transaction).
- 3 Enter a search keyword in the **keyword** field (Parameters column of the searchGoogle transaction): for example convertigo cliplet.
- 4 Click on the searchGoogle link.

The transaction starts. The **Connector Editor** of the CWI Studio shows the successive steps of the transaction being executed in real time.

5 Once the transaction has been completed, the list of results is displayed in an **Execution result** window in the test platform:





Figure 2 - 121: XML output in the Execution result window of the test platform


The previous chapter describes how to extract data into a simple XML output document.

The first section of this chapter introduces Web services and gives instructions on how to expose projects either as REST or SOAP Web services.

The second section is dedicated to XSL transformation. It describes how to transform XML data into an HTML document for display purpose using XSL style sheets.

- Exposing Projects as Web Services
- XSL AJAX Presentation



Examples of this chapter are based on the Web integration project described in chapter "My First Convertigo Web Integrator Project" on page 2 - 1.

3.1 Exposing Projects as Web Services

Convertigo Web Integrator allows to expose projects either as REST or as SOAP Web services.

Exposing projects as REST Web services is a straightforward process: it requires calling the transaction from any Web Browser via a properly formatted URL.

Exposing projects as SOAP Web services requires setting transactions as public methods and selecting data types before generating SOAP request and response envelopes.

This section describes how to expose projects as Web services in both architectures:

- Exposing a Project as a REST Web Service
- Exposing a Project as a SOAP Web Service

3.1.1 Exposing a Project as a REST Web Service

CWI transactions are by design exposed as methods of a REST Web service: each transaction can be called via an URL; each transaction is a REST method providing data.

Transactions result in the generation of an XML document including data represented in XML format.



The URL syntax used for the external call is standard URL syntax.

To invoke a transaction as a REST Web service method

- 1 Launch your default Web Browser.
- 2 In the URL address bar, type in the URL in the following format:

http://<HostName>:<ConvertigoPortNumber>/convertigo/projects/
<ProjectName>/.xml?__connector=<ConnectorName>&__transaction=
<TransactionName>&keyword=<KeywordName>

For example, in our project:

- HostName = localhost;
- ConvertigoPortNumber = 18080;
- ProjectName = sample_doc_CWI;
- ConnectorName = googleConnector;
- *TransactionName* = searchGoogle;
- KeywordName = convertigo cliplet;

For example, in a Mozilla Firefox URL address bar:

http://localhost:18080/convertigo/projects/sample_doc_CWI/.xml?__connector=GoogleConnector&__transaction=searchGoogle&keyword=convertigo cliplet

Figure 3 - 1: URL syntax for exposing transactions as REST Web services

3 Press Enter.

The transaction is launched in the CWI Engine.

4 Switch to the **CWI Studio**.

In the **Web Browser**, you can see Convertigo accessing the Google homepage, then inputing the keyword (convertigo cliplet in our project) and navigating through Web pages.

5 Click on the **Output** tab of the **Connector View**.

The view displays the result list in XML format.

6 Switch to the Web Browser launched in step 1.

CWI has sent the response data in XML format to its client.

3.1.2 Exposing a Project as a SOAP Web Service

A CWI project can be seen as a SOAP service and CWI transactions as SOAP methods.

Exposing projects (resp. transactions) as SOAP Web services (resp. Web service methods) requires:

1 *Informing* Convertigo that **one or several** transactions will be exposed as SOAP Web service methods; this is done by setting one or several transactions as *public methods* through the **Public Method** transaction property.

2 Describing all methods and request/response SOAP envelopes for these methods in WSDL (*Web Services Description Language*) format.



For more information on WSDL, see the WSDL tutorial from W3Schools available at <u>http://www.w3schools.com/wsdl/</u>.

The following procedure gives instructions on how to to expose transactions as SOAP Web service methods.

To expose transactions as SOAP Web service methods

1 In the **Projects View**, click on the transaction (searchGoogle in our project).

The Properties View is automatically updated:

🔲 Properties 🛛	🖼 🍰 🖾 🗸 🗖 🗖
Property	Value
🖃 Expert	
Call the biller	false
Character set	ISO-8859-1
Client cache for response	false
Handles cookies	true
HTTP parameters	[[Content-Type, application/x-www-form-urlencoded]]
Include certificate group	false
Maintains connector state	false
Public method	false
Request template	
Response life time	
Synchronizer	Document Completed:1 timeout:60000
Information	
Depth	n/a
Java class	$com.twinsoft.convertigo.beans.transactions.Html {\it Transaction}$
Name	searchGoogle
Priority	0
QName	/sample_doc_CWI/_data/cn/QQMIKUb/tr/MM-3ilN/transact
Туре	HTML transaction
Properties	
Comment	
Response delay	60
Style sheet	None
Sub path	
XML attributes to include	[Z@e91265

Figure 3 - 2: searchGoogle transaction properties

2 Set the **Public Method** property to true and press **Enter**.



Include certificate group	Taise
Maintains connector state	false
Public method	false 🛛 🛃
Request template	true
Response life time	false
Synchronizer	Document Completed:1 timeout:60000
Information	
epth	n/a

Figure 3 - 3: Setting the Public Method property to true

3 Save the project by clicking on 🙀 or by pressing Ctrl + S.

When the the **Public Method** property is set to true, the transaction is exposed in the SOAP WSDL as a Web service method. The searchGoogle transaction created in our sample project will therefore be exposed as a Web service method, and the sample_doc_CWI project as a Web service.

We now need to generate the schema (in XML format) of:

- the project,
- each transaction (one in our case).

The overall schema structure for the whole project (for the sample_doc_CWI Web service) is automatically generated by Convertigo. Schema parts corresponding to each exposed transaction are automatically generated as well, but additionnal data must be set prior to generating them.

To sum up, for a transaction:

- The request schema is automatically generated by CWI.
- The *response schema* is generated by CWI based on manually selected types among types automatically proposed by CWI depending on the transaction definition.



The WSDL is automatically generated in both "RPC" and "document/ literal" modes.

- 4 In the **Projects View**, right-click on the transaction.
- 5 Select Update schema from transaction's definition:

🖻 🗁 Transactions		
Default_HTM	4L_Transaction	- (-
Err SearchGor	Mashup composer 🕨	
🖻 🗢 or	New •	2
0 or <=	Cvs 🕨	1
- D	Edit handlers	1
⊡	Execute	12
51	Update schema from transaction's definition	
🖃 🥭 Variat	Update schema from current generated XML	<
KE	Default transaction	L

Figure 3 - 4: Update transaction schema from transaction definition

A confirmation window opens:

Do you really want to extract the schema? Warning: the previous one will be replaced.
Oui Non Annuler

Figure 3 - 5: Update transaction schema- Confirmation window

6 Click on **Yes**.



Clicking on Yes overwrites previously stored schema.

A Transaction schema types window opens:





Figure 3 - 6: Transaction schema types window

The window contains data types corresponding to extraction rules defined on project screen classes.

In our project sample_doc_CWI, only one extraction rule is defined: listResults, the table extraction rule associated with the googleResultsPageCurrent screen class. The data type corresponding to this extraction rule appears in the **Transaction** schema types window as a listResultsTableType type (see Figure 3 - 6).

One type is always present: ConvertigoError.

This standard type corresponds to an exception error XML that can appear in the output XML of a transaction when the Convertigo engine stops in error.



The ConvertigoError data type should always be selected.

- 7 While keeping the Shift button pressed, select both types.
- 8 Click on OK.

The project schema is updated: the **transaction response schema** is generated, the searchGoogle transaction appears bold in the **Projects View**.

9 Save the project by clicking on in by pressing Ctrl + S.

Data types used to describe the transaction and all other types generated by CWI for this project are fully described in the sample_doc_CWI.xsd file. It can be opened in an **Editor**.

To edit the sample_doc_CWI.xsd file

1 In the **Projects View**, click on the **Project Explorer** tab.

2 Expand the project folder.



Figure 3 - 7: Project folder in Project Explorer

3 Doucle-click on the sample_doc_CWI.xsd file to open it.

The sample_doc_CWI.xsd file appears in the Editor:

• in the **Design** tab of the **Editor**, the schema appears as follows:



Light Elements Types GoogleConnector _searchGoogle : GoogleConnector _searchGoogleResponse GoogleConnector _searchGoogleWithLimit : GoogleConnector _ SearchGoogleWithLimit : GoogleConnector _ Default HTML _TransactionReguestData GoogleConnector _searchGoogleWithLimit : GoogleConnector _ searchGoogleReguestData GoogleConnector _ searchGoogleWithLimitResponse GoogleConnector _ searchGoogleWithLimitResponse GoogleConnector _ searchGoogleReguestData GoogleConnector _ searchGoogleReguestData GoogleConnector _ searchGoogleWithLimitResponseDat GoogleConnector _ searchGoogleWithLimitResponseData GoogleConnector _ searchGoogleWithLimitResponseData GoogleConnector _ searchGoogleWithLimitResponseData GoogleConnector _ searchGoogleWithLimitResponseData
Attributes GoogleConnectorTypes

Figure 3 - 8: Project schema in Design tab of Editor

• in the **Source** tab of the **Editor**:

	sample_doc_CWI GoogleConnector	S sample_doc_CWI.xsd 🛛	
	xml version="1.0" enco</td <td>ling="UTF-8"?></td> <td>~</td>	ling="UTF-8"?>	~
	<pre>xsd:schema xmlns:xsd="h</pre>		
	<pre><xsd:complextype name="C</pre></td><td>onvertigoError"></xsd:complextype></pre>		
	<xsd:sequence></xsd:sequence>		
	<pre><xsd:element <="" name="contex" pre=""></xsd:element></pre>	st" type="sample doc CWI ns:ConvertigoErrorContext"/>	
	<pre><xsd:element name="excep</pre></th><th>tion" type="xsd:string"></xsd:element></pre>		
	<pre><xsd:element <="" name="messa" pre=""></xsd:element></pre>	ge" type="xsd:string"/>	
	<pre><xsd:element <="" name="stack" pre=""></xsd:element></pre>	trace" type="xsd:string"/>	
	<pre><xsd:complextype name="C</pre></th><th>onvertigoErrorContextVariable"></xsd:complextype></pre>		
	<xsd:attribute name="nam</th><th>e" type="xsd:string"></xsd:attribute>		
	<pre><xsd:attribute name="val</pre></th><th><pre>ze" type="xsd:string"></xsd:attribute></pre>		
	<pre><xsd:complextype name="C</pre></th><th>onvertigoErrorContext"></xsd:complextype></pre>		
	<xsd:sequence></xsd:sequence>		
	<pre><xsd:element "<="" maxoccurs="" pre=""></xsd:element></pre>	unbounded" minOccurs="0" name="variable" type="sample doc CWI name="variable" type="sample doc"	5
	<xsd:element googl"<="" name="Googl</th><th>aConnector searchGoogleWithLimit" pre="" type="sample doc CWI ns:Goog</th><th>2</th></tr><tr><th></th><th><pre><xsd:element name="></xsd:element>	<pre>econnector searchGoogleWithLimitResponse"></pre>	
	<xsd:complextype></xsd:complextype>		
	<xsd:sequence></xsd:sequence>		
	<xsd:element g<="" name="respo</th><th>nse" th="" type="sample doc CWI ns:GoogleConnector _searchGoogleWithL:</th><th>i</th></tr><tr><th></th><th></xsd:sequence></th><th></th><th></th></tr><tr><th></th><th></xsd:complexType></th><th></th><th></th></tr><tr><th></th><th></xsd:element></th><th></th><th></th></tr><tr><th></th><th><xsd:complexType name="><th><pre>pogleConnector searchGoogleWithLimitRequestData"></pre></th><th></th></xsd:element>	<pre>pogleConnector searchGoogleWithLimitRequestData"></pre>	
	<xsd:sequence></xsd:sequence>		
	<xsd:element name="keywo.</th><th>rd" type="xsd:string"></xsd:element>	-	
	<pre><xsd:element name="maxPa</pre></th><th>ges" type="xsd:string"></xsd:element></pre>		
	<xsd:complextype name="G</th><th><pre>>ogleConnectorsearchGoogleWithLimitResponseData"></xsd:complextype>		
	<xsd:sequence></xsd:sequence>		
	<pre><xsd:element minoccurs="</pre></th><th><pre>)" name="error" type="sample_doc_CWI_ns:ConvertigoError"></xsd:element></pre>		
	<pre><xsd:element "<="" maxoccurs="" pre=""></xsd:element></pre>	<pre>unbounded" minOccurs="0" name="listResults" type="sample_doc_CW.</pre>	1
	<pre><xsd:element googl"<="" name="Googl</pre></th><th><pre>%ConnectorsearchGoogle" pre="" type="sample_doc_CWI_ns:GoogleConnect(</pre></th><th>2</th></tr><tr><th></th><th><pre><xsd:element name="></xsd:element></pre>	<pre>>ConnectorsearchGoogleResponse"></pre>	
	<xsd:complextype></xsd:complextype>		
	<xsd:sequence></xsd:sequence>		
	<pre><xsd:element)<="" name="response;" pre=""></xsd:element></pre>	<pre>ise" type="sample_doc_CWI_ns:GoogleConnectorsearchGoogleRespon")</pre>	٤ -
	<xsd:complextype name="G</th><th><pre>>ogleConnectorsearchGoogleRequestData"></xsd:complextype>		
	<xsd:sequence></xsd:sequence>		
	<pre><xsd:element name="keywo.</pre></th><th>d" type="xsd:string"></xsd:element></pre>	~	
	<	<u>></u>	
Des	ign Source		

Figure 3 - 9: Project schema in Source tab of Editor

Data types selected in step 7 of previous process appear in the schema:



🕾 sample_doc_CWI GoogleConnector 🛛 🛐 sample_doc_CWI.xsd 🛛	(
<pre><?xml version="1.0" encoding="UTF-8"?></pre>	/
<pre><xsd:schema pre="" xm<="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"></xsd:schema></pre>	lns:sample_doc_CWI_1
<pre><xsd:complextype name="ConvertigoError">*</xsd:complextype></pre>	
<xsd:sequence></xsd:sequence>	
<pre><xsd:element name="context" type="sample_doc_CWI_ns:Convert</pre></td><td>igoErrorContext"></xsd:element></pre>	
<pre><xsd:element name="exception" type="xsd:string"></xsd:element></pre>	
<pre><xsd:element name="message" type="xsd:string"></xsd:element></pre>	
<pre><xsd:element name="stacktrace" type="xsd:string"></xsd:element></pre>	
	Selected
	data types
<pre><xsd:complextype name="ConvertigoErrorContextVariable"></xsd:complextype></pre>	uata types
<pre><xsd:attribute name="name" type="xsd:string"></xsd:attribute></pre>	
and a set of the set of the set	
a series server	
, Adur Stightman	and and
xsd: complex lype	
<pre><xsd:complex1ype name="listkesultsTableType" pre="" }<=""></xsd:complex1ype></pre>	
<pre><xsd:sequence></xsd:sequence></pre>	
<pre><xsd:element <="" minoccurs="0" name="resultitem" td="" type="sample_o"><td>loc_CW1_ns:resultitem_</td></xsd:element></pre>	loc_CW1_ns:resultitem_
<pre><xsd:attribute <="" boxec:string"="" name="referer" pre="" type="xsd:string" use="option"></xsd:attribute></pre>	1a1"/>
	(
Design Source	

Figure 3 - 10: Transaction data types in schema

Editing this schema can be usefull when, for example, specific restrictions are to be set on some elements.

The searchGoogle transaction is now exposed by CWI as a SOAP Web service method of a SOAP Web service (the project).

To display the SOAP Web service WSDL

- 1 Launch your default Web Browser;
- 2 In the URL address bar, type in the URL in the following format:

http://<HostName>:<ConvertigoPortNumber>/convertigo/projects/
<ProjectName>/<SOAPmode>?wsdl

For example, in our project:

- HostName = localhost;
- ConvertigoPortNumber = 18080;
- ProjectName = sample_doc_CWI;
- SOAPmode = .ws for RPC mode, SOAPmode = .wsl for document/litteral mode.

For example, in a browser URL address bar:

ktp://localhost:18080/convertigo/projects/sample_doc_CWI/.wsl?wsdl

Figure 3 - 11: URL syntax for displaying the SOAP Web service WSDL

3 Press Enter.

The Web browser displays the SOAP Web service WSDL with:

- the list of methods (in our project, the searchGoogle method);
- the method input variable;
- the output format description.

🌈 http://localhost:18080/convertigo/projects/sample_doc_CWI/.wsl?wsdl - Windows Internet Explorer 📃 🗖
🚱 🕤 🔻 🔣 http://localhost:18080/convertigo/projects/sample_doc_CWI/.wsl?wsdl 🔍 🐓 🗙 Google
😭 🕸 🔣 http://localhost:18080/convertigo/projects/sample_d 🔄 🖄 🔹 🗟 🔹 🔂 🔹 🔂 🖓 Page 🔹 🎡 Outils 🤟
<pre><?xml version="1.0" encoding="ISO-8859-1" ?> - <wsdl:definitions name="sample_doc_CWI" targetnamespace="http://www.convertigo.com/convertigo/projects/sample_doc_CWI" xmlns:sample_doc_cwi_ns="http://www.convertigo.com/convertigo/projects/sample_doc_CWI" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> </wsdl:definitions></pre> - <wsdl:types> - <wsdl:types> - <xsdl:schema targetnamespace="http://www.convertigo.com/convertigo/projects/sample_doc_CWI"></xsdl:schema></wsdl:types></wsdl:types>
 - <wsd!:message name="GoogleConnectorsearchGoogleResponse_RPC"> <wsd!:part <br="" name="response">type="sample_doc_CWI_ns:GoogleConnectorsearchGoogleResponseData" /> </wsd!:part></wsd!:message>
<pre>- <wsdl:message name="GoogleConnectorsearchGoogleRequest_RPC"></wsdl:message></pre>
<pre></pre>
<pre><wsdl:input message="sample_doc_CWI_ns:GoogleConnectorsearchGoogleRequest_RPC"></wsdl:input></pre>
<pre>- <wsd:operation name="sample_doc_CWI_ns:Google"></wsd:operation></pre>
<pre>- <wsdl:binding name="sample_doc_CWISOAPBinding_RPC" type="sample_doc_CWI_ns:sample_doc_CWIPortType_RPC"></wsdl:binding></pre>
Terminé SIntranet local 🔍 100% 🕶

Figure 3 - 12: SOAP Web service WSDL

To invoke a transaction as a SOAP Web service method

Eclipse provides a tool named **Web Services Explorer** which is used, among other features, to invoke SOAP Web services without writing SOAP envelopes. We will launch it to invoke our newly created SOAP Web service.

1 Click on *Run > Launch the Web Services Explorer* or click on the *i* icon in the studio toolbar:





Figure 3 - 13: Launching the Web Services Explorer

The Web Services Explorer view opens in front of the Connector View:



Figure 3 - 14: Web Services Explorer

The **Web Services Explorer** connects to the service WSDL and displays *WSDL binding details* such as:

- methods (called Operations in the interface);
- endpoints.

The interface contains three panes and six icons:

Table 3 - 1: Web Services Explorer

Exp	lorer elements	Description
Pane	Navigator	Contains the WSDL tree structure: service URL, methods, WSDL Binding details, etc.
	Actions	Contains, in the form of links and buttons, actions to be performed on methods, endpoints, services, etc
	Status	Displays error messages and other information.

Exp	lorer elements	Description
lcons	\Diamond	<i>Back</i> icon. Navigates back one page in the Explorer.
	4	<i>Forward</i> icon. Navigates forward one page in the Explorer.
Icons		<i>UDDI Page</i> icon. Displays the UDDI Main tree in the Navigator and related actions ion UDDI registries n the Actions pane.
		WSIL Page icon. Displays the WSIL Main tree in the Navigator and related actions on WSILs in the Actions pane.
	R	WSDL Page icon. Displays the UDDI Main tree in the Navigator and related actions on WSDL services in the Actions pane.
	*	<i>Favorites</i> icon. Displays favorites UDDI registries, WSILs and WSDL services.

- 2 Click the WSDL Page icon in the Web Services Explorer toolbar.
- 3 In the **Navigator** pane of the **Web Services Explorer**, click on **WSDL Main** link.

The Actions pane of the Web Services Explorer is automatically updated:

🕮 sample_doc_CWI GoogleConnector 🛛 🚱	Web Services Explorer 🛛	
Web Services Explorer	수 수 🗊 🖪 🧏	*
পদ্র- Navigator 🔗 🖉	Actions	Â
	Open WSDL Enter the URL of a WSDL document and click Go to explore. WSDL URL Browse Go Reset	
		_
		-

Figure 3 - 15: Web Services Explorer

4 In the **WSDL URL** of this pane, enter the URL of the SOAP service in the following format:

http://<HostName>:<ConvertigoPortNumber>/convertigo/projects/
<ProjectName>/<SOAPmode>?wsdl



For example, in our project:

- HostName = localhost;
- ConvertigoPortNumber = 18080;
- ProjectName = sample_doc_CWI
- *SOAPmode* = .ws for RPC mode, *SOAPmode* = .wsl for document/litteral mode.

[Actions
	🖉 Open WSDL
	Enter the URL of a WSDL document and click Go to explore.
	WSDL URL Browse
	http://localhost:18080/convertigo/projects/sample_doc_CWI/.wsl?wsdl
	Go Reset
	man man man

Figure 3 - 16: URL of the SOAP service in the Web Services Explorer

5 Click on **Go**. The three panes are automatically updated:

🕮 sample_doc_CWI GoogleConnector 🛛 🚱 Web Services Explorer 🛛 🗖 🗖				
Web Services Explorer $\diamond \Rightarrow \square$ 🖪 🗟 🏖 \star				
ୟିନ- Navigator 🔗 🖉	Actions			7
B. WSDL Main D: ⊉ http://localhost: 18080/convertigo/pro	A WSDL Service Details			
⊕- [*] ² / ₂ sample_doc_CWI	Shown below are the details for this Bindings	<service> ele</service>	ment. Click on a binding to see its operations.	
	Name	Туре	Documentation	
	sample doc CWISOAPBinding RPC	SOAP		
	sample_doc_CWISOAPBinding	SOAP		
	i Status			٦
	IWAB0381I http://localhost:18080/c	onvertigo/pro	jects/sample_doc_CWI/.wsl?wsdl was successfully opened.	
			<u>•</u>	

Figure 3 - 17: Web Services Explorer panes after a WSDL URL is reached

6 Expand sample_doc_CWI in the Navigator pane, then click on sample_doc_CWISOAPBinding link. The Actions panes is updated:

🕾 sample_doc_CWI GoogleConnector 🛛 🎯 Web Services Explorer 🔀 📃 🗖		
Web Services Explorer $\Diamond \Rightarrow 0$ 🗓 🗟 🏂		
Pa- Navigator 🔗 🖉	Actions	
WSDL Main	A WSDL Binding Details	
Sample_doc_CWI Sample_doc_CWISOAPBinding_ Sample_doc_CWISOAPBinding_ Sample_doc_CWISOAPBinding	Shown below are the details for this SOAP binding> element. Click on an operation to fill in its parameters and invoke it or specify additional endpoints.	
	▼ Operations	
	Name Documentation	
	GoogleConnector_searchGoogle	
	▼ Endpoints <u>Add</u> <u>Remove</u>	
	Endpoints	
	http://localhost:18080/convertigo/projects/sample_doc_CWI/.wsl	
	Go Reset	
	i Status 🖉	
	IWAB0381I http://localhost: 18080/convertigo/projects/sample_doc_CWI/.wsl?wsdl was successfully opened.	
	_	

Figure 3 - 18: WSDL Binding Details in Actions pane of the Web Services Explorer

The **Operations** table of the **Actions** pane displays all available *methods* (corresponding to transactions) for the invoked service (sampleGoogle):

 Operations 		5
Name	Documentation	Method
GoogleConnector searchGoogle	=	(=Transaction)
A Marine	and the second s	_

Figure 3 - 19: List of methods displayed in the Web Services Explorer

The Navigator pane displays the elements of the WSDL service in a tree structure:

95- Navigator	\$° []
器 WSDL Main ⊡- ⊉ http://localhost: 18080/convertigo/projects/sample_doc_CW	/I/.wsl?wsdl
sample_doc_CWI Image: sample_doc_CWISOAPBinding_RPC	WSDL Service
sample_doc_CWISOAPBinding	
GoogleConnector_searchGoogle	Method
and a man and a series of the	

Figure 3 - 20: WSDL service elements displayed in the Navigator pane

7 Click on GoogleConnector__searchGoogle in the Operations table. The Actions pane is automatically updated:



Actions		Q
🕙 Invoke a WSDL Operation	Source	
Enter the parameters of this WSDL operation and click Go to invoke.		
http://localhost:18080/convertigo/projects/sample_doc_CWI/.wsl		
- Body		
▼ <u>GoogleConnector searchGoogle</u>		
<u>keyword</u> string		
Go Reset		

Figure 3 - 21: Invoke a WSDL Operation view in the Actions pane

The method input variable as well as its type are displayed (here, keyword, variable of string type).

8 Type in a search keyword in the keyword field: for example convertigo cliplet.

▼ GoogleConnector	searchGoogle
keyword string	
convertigo cliplet	
Gorean	

9 Click on **Go**:

The **Status** pane is automatically updated. A **source** link appears on the upper right corner of the pane.

10 Click on the **source** link:

i Status	_
The SOAP response failed schema validation. Please switch to the source view for the SOAP response in US XML format.	

Figure 3 - 23: Status pane after WSDL operation invocation

The Status pane diplays :

• the SOAP request envelope searchGoogleRequest, which was generated using

Figure 3 - 22: Invoking a WSDL Operation

the input variable entered as keyword:



Figure 3 - 24: SOAP Request envelope detail

 the SOAP response envelope searchGoogleResponse, which corresponds to the WSDL description previously generated. The list of results included in the response envelope is similar to the list obtained when invoking a REST service. The difference is that it is embedded in a SOAP envelope:



Figure 3 - 25: SOAP response envelope detail



3.2 XSL AJAX Presentation

Convertigo features a *test platform* for calling projects, visualizing project-related transactions and testing them.

This platform is accessed using any standard Web browser.



To see how to launch the test platform, see "To test the transaction using the test platform" on page 2-72.

The test platform includes an AJAX library (JavaScript) for calling transactions and presenting them using XSL (for *eXtensible Stylesheet Language*) style sheets.

XSL style sheets process XML nodes of extracted XML data to transform them into an HTML document. The XSL *transformation* is performed by the AJAX library included in the project.

The basic process is the following:

- 1 *Creating* an XSL style sheet that processes our transaction XML output. This is done in two steps: first creation of a CWI **Sheet** object, then creation and edition of the related .xsl file;
- **2** Associating the previously created style sheet with the transaction for CWI to know that, on call of a transaction, this XSL style sheet must be returned together with associated XML data.

We will now create the XSL style sheet and associate it to our transaction.

To set up an XSL transformation

- 1 In the **Projects View**, right-click on the transaction (searchGoogle).
- 2 Select New > Sheet.

A New Sheet wizard opens:

New Sheet Please select a sheet template.	
XSL style sheet	
C < Back Next > Finish	Cancel

Figure 3 - 26: New Sheet wizard

3 Click on XSL style sheet.

4 Click on Next:



Figure 3 - 27: Entering a name for the new style sheet

- 5 Enter a name in the **Name** field (for example results_list).
- 6 Click on Finish.

The new style sheet object is created in the **Sheets** folder of the transaction:



Figure 3 - 28: New style sheet object in the Projects View

Now that the Convertigo sheet object has been created, we will associate it with a physical file, results.xsl, that will be edited for presenting data as required.

- 7 In the **Projects View**, click on the **Project Explorer** tab.
- 8 Right-click on the project (for example sample_doc_CWI).
- 9 Select New > File:



Figure 3 - 29: Project Explorer - New file

A New File wizard opens:



New File		
File		
Create a new file resource.		
Enter or select the parent folder:		
sample_doc_CWI		
⊥ 🗁 sample_doc_CWI		
File name:		
Advanced		
Havanced >>		
0	Finish Cancel	
Ÿ	Cancor	

Figure 3 - 30: New File wizard

10 In the **File name** field, enter a filename (results.xsl for example).

11 Click on Finish.

The results.xsl file is:

• created in the Project Explorer tree structure:



Figure 3 - 31: New file in Project Explorer

displayed in the Editor (Source tab selected):

🕮 sampleGoogle GoogleConnector 🛛 🕅 results.xsl 🛛	- 8
	~
	~
	>
Design Source	

Figure 3 - 32: XSL style sheet

After edition, the XSL style sheet is divided into *template rules* that match parts of the XML document to be processed: the document (document element) and the list of results (listResults element).

In our case, it is edited to obtain a document defined as follows:

- the title of the Web page is "Google search results" (template rule matching on / document node and HTML title identified by <h2> tag name):
- following the title, the Web content is set in an HTML table divided into two columns called Title and Url (template rule matching on listResults nodes of the XML output generated by the transaction). For each resultItem of the XML output, the title and url values are displayed in previously defined columns.





Figure 3 - 33: XSL style sheet templates



For more information on XSL, see the XSLT tutorial from W3Schools at http://www.w3schools.com/xsl/.

Now that the XSL style sheet has been created, we must associate it to the sheet previously created in the transaction, by setting the **URL** sheet property to results.xsl.

12 In the **Projects View**, click on the sheet (results_list).

The Properties View is automatically updated:

Properties 🛛	
Property	Value
Information	
Depth	n/a
Java class	com.twinsoft.convertigo.beans.core.Sheet
Name	results_list
Priority	0
QName	/sample_doc_CWI/_data/cn/QQMIKUb/tr/MM-3i
Туре	XSL style sheet
Properties	
Browser	*
Comment	
URL	

Figure 3 - 34: Sheet properties

- 13 Click on the value of the **URL** property. The value is highlighted.
- 14 Type in the name of the previously created XSL style sheet file: results.xsl for example, and press Enter.
- **15** Save the project by clicking on **i** or by pressing Ctrl + S.

The sheet object is now created. The next step consists in associating the sheet with the transaction execution, by setting a transaction property: **Style Sheet**.

This property specifies whether or not and which style sheet must be used to perform the XSL transformation of the transaction execution. The property can take on three values:

Table 3 - 2: Style Sheet transaction property

Style Sheet value	Description
None	Does not process with XSLT. Usually set for Web services (SOAP or REST), as plain XML data is then to be returned.
From transaction	Uses the XSL style sheet attached to this transaction.
From last detected screen class	Uses XSL style sheet attached to the last detected screen class.

16 In the **Projects View**, click on the transaction (searchGoogle).

The Properties View is automatically updated:



Properties	[≒] 券 ▽ □ □
Property	Value
🖃 Expert	
Call the biller	false
Character set	ISO-8859-1
Client cache for response	false
Handles cookies	true
HTTP parameters	[[Content-Type, application/x-www-form-urle
Include certificate group	false
Maintains connector state	false
Public method	true
Request template	
Response life time	
Synchronizer	Document Completed:1 timeout:60000
Information	
Depth	n/a
Java class	com.twinsoft.convertigo.beans.transactions
Name	searchGoogle
Priority	0
QName	/sample_doc_CWI/_data/cn/QQMIKUb/tr/MM
Туре	HTML transaction
Properties	
Comment	
Response delay	60
Style sheet	None
Sub path	
XML attributes to include	[Z@15b0bc6

Figure 3 - 35: Transaction properties

17 Click on the value of the Style Sheet property. A drop-down menu appears:

Comment		:cems
Response delay	60	!cem
Style sheet	None 💌	!cer
Sub path	None	¹ cems
XML attributes to include	From transaction From last detected screen class	!cem
Land and the second	- man	icer.

Figure 3 - 36: Transaction Style Sheet property

18 Select From transaction and press Enter.

19 Save the project by clicking on 🤖 or by pressing Ctrl + S.

We will switch back to the test platform and invoke the transaction to see how XML data was transformed.

20 Follow instructions of procedure "To test the transaction using the test platform" on page 2 - 72.

The XML output generated by the transaction (see Figure 2 - 121) is processed by the defined XSL style sheet. It appears in the **Execution result** window as an HTML document:

Chapter "Exposing Projects as Web Services & Presenting Data" XSL AJAX Presentation

🕹 Index - Mozilla Firefox			
Eichier Édition Affichage Historique	Marque-pages Yahoo! Outils	2	()
C × \color \box	http://localhost:18080/convertigo/pr	rojects/sampleGoogle/# 🏠 🔹 Google	P
	Execution result		8
convertico	Résultats Google	correspondants à la recherche	^
comple@coade project	Title	Url	
Project type : web integration	Convertigo - C-EMS Web Clipper	www.convertigo.com/en//convertigo-web-clipper_2.html -	
GoogleConnector connecto	Convertigo - C-EMS Web Clipper	www.convertigo.com/en//convertigo-web-clipper_3.html -	
Cliplet name Default HTML Transaction	Convertigo Web Clipper (CWC)	www.twinsoft.fr/intl/fr//convertigo-ems-web-clipper.htm -	
searchGoogle	Microsoft Solution Finder - View solution	https://solutionfinder.microsoft.com//SolutionDetailsView.aspx?	
<u>testLearnMode</u>	IBM - Convertigo Web Clipper (CWC)	www.developer.ibm.com/gsdod/solutiondetails.do?solution	
	IBM - Convertigo Web Clipper (CWC)	www.developer.ibm.com/gsdod/solutiondetails.do?solution	
	Convertigo - Rechercher	www.twinsoft.com/component/option.com_flang.frl -	~

Figure 3 - 37: XML output after XSL transformation

