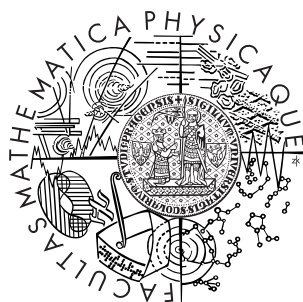


Charles University in Prague
Faculty of Mathematics and Physics

SOFTWARE PROJECT



«XRouter»

3. Users's Guide

Soběslav Benda
Miroslav Cicko
Tomáš Kroupa
Petr Sobotka
Bohumír Zámečník

Supervisor: Mgr. Martin Nečaský, Ph.D.

2011

XRouter – User's guide

Intended audience: users (typically system integrators) who configure the system

Prerequisite reading: [XRouter – Definition](#)

[XRouter – User's guide](#)

[XRouter GUI](#)

[Running XRouter GUI](#)

[Main Window](#)

[Function panel](#)

[Module panel](#)

[Configuration module](#)

[Gateway component](#)

[Directory I/O adapter](#)

[Email adapter](#)

[Web client adapter](#)

[Web service adapter](#)

[Processor component](#)

[XML Resources](#)

[Trace log and Event log](#)

[An example of the Event log viewer](#)

[Message flow editor](#)

[Workspace](#)

[Properties side panel](#)

[Import/exporting buttons](#)

[Nodes](#)

[Entry node](#)

[Action node](#)

[Content-based router \(CBR\) node](#)

[Terminator node](#)

[Demo projects](#)

[Restaurant](#)

[How to run this demo](#)

[How to create this demo](#)

[Support for SOA](#)

[Order Record keeping](#)

[Doing cancellation](#)

[Order Closing](#)

[Provider.exe](#)

[Requestor.exe](#)

This documents describes the XRouter GUI for configuring and monitoring the XRouter Service and some demo projects illustrating some of the ways which XRouter can be used.

XRouter GUI

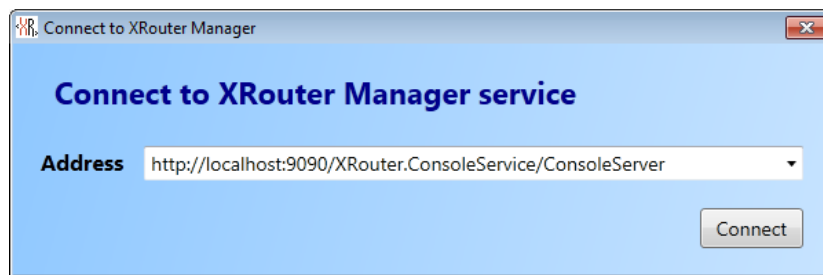
XRouter GUI is a desktop application intended for user interaction with the XRouter system. Its primary value is in viewing and editing application configuration, monitoring and administering the XRouter Service. The main features are the following:

- viewing and editing application configuration

- message flow graph
- adapters and their configuration
- adapter and message flow action types
- viewing tokens
- viewing XRouter Service event logs and trace logs
 - filtering by date and level, paging
- connecting to a XRouter Manager
- viewing XRouter Service status
- starting and stopping XRouter Service

Running XRouter GUI

The application can be started using the `XRouter.Gui.exe` program located in the XRouter installation directory or by the *XRouter Configuration Manager GUI* link in the Start Menu. First it is needed to connect to a running instance of the XRouter Manager service. Since it can run remotely its address has to be specified – in a dialog appearing when the application starts. After filling in the address click the *Connect* button.



A dialog of connection to the XRouter Manager service

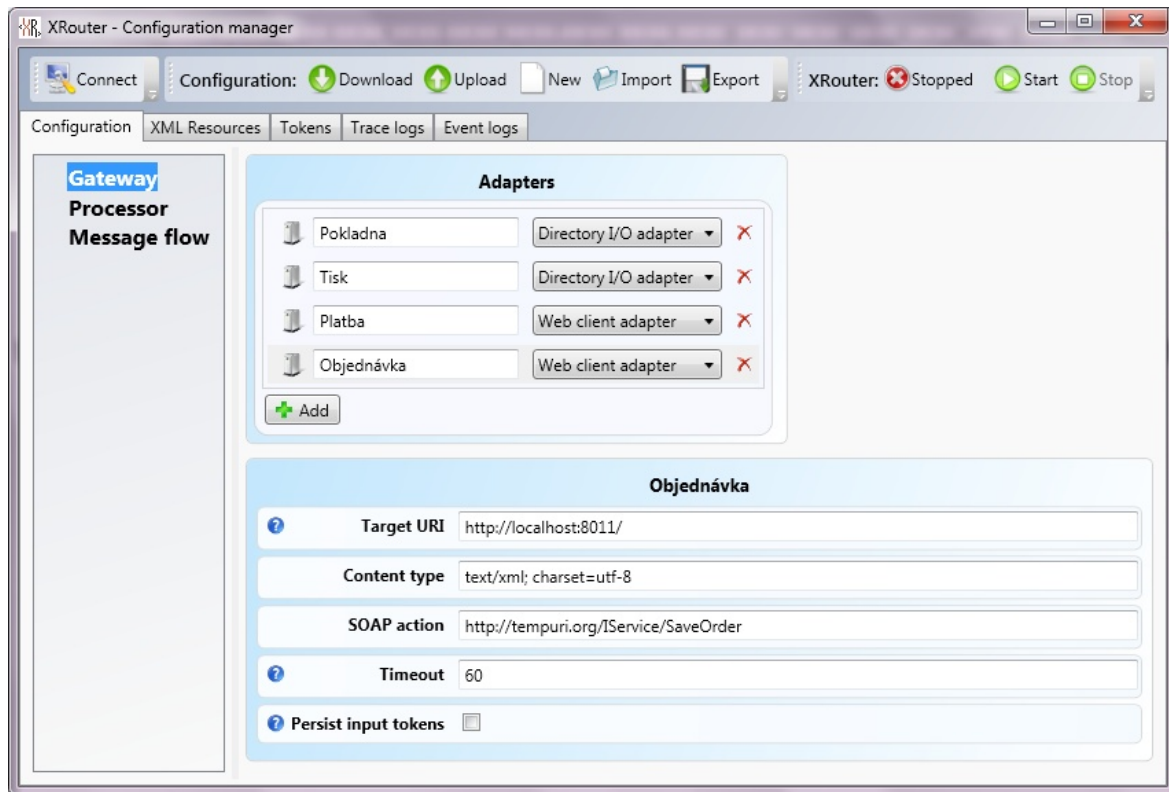
If connecting to the XRouter Manager fails an error is shown and the connection dialog is shown again. You can use the main application window only after a successful connection.

Main Window

The window may differ significantly according to the current configuration. It is divided into two main parts:

- Function panel (the toolbar-like panel on top)
- Module panel (the main panel)

The function panel is useful for performing the fundamental commands, whereas the module panel enables to configure the components and to see the information about the XRouter Service state. The modules may be switched using tabs. Each module will be described in more detail further.



The main window of XRouter Manager

Function panel

The function panel contains these commands:

- **Connect:** Shows the XRouter Manager service connection dialog where one can change to which one is the application connected. This is useful especially when managing more than one instance of XRouter.
- **Configuration:**
 - **Download:** Downloads the current configuration from the service and enables its viewing and editing. When first connected to a new instance the configuration is downloaded automatically.
 - **Upload:** Uploads the configuration from the editor to the XRouter Manager (including all changes made). It replaces the previous version (which is still archived in the database, however). The XRouter Service need to be restarted in order to take effect of the new configuration.
 - **New:** Creates a new empty configuration.
 - **Import:** Loads a configuration from a file into the editor.
 - **Export:** Saves the configuration from the editor (including all changes) into a file.
- **XRouter Service**
 - **Status:** Displays the current status of XRouter Service - running, stopped or error (if something failed). By moving the cursor on it, the details of the error are shown. The status is checked periodically, click this button to refresh it immediately.
 - **Start:** Starts the XRouter Service associated with the connected XRouter Manager.
 - **Stop:** Stops the XRouter Service.

Module panel

The application contains two types of modules:

- Configuration: Changes the behavior of the XRouter Service.
- Informative: Displays some information about the XRouter state.

The module panel contains these modules:

- Configuration
- XML resources
- Tokens
- Trace logs
- Event logs

Configuration module

The configuration Module enables to configure the components of the XRouter Service. It consist of two parts:

- Component tree
- Configuration module of the selected component

By selecting a node in the tree a particular configuration module is loaded. There is possible to change its behaviour. Configuration modules are available for the following components:

- Gateway
- Processor
- Message flow

Gateway component

In the Gateway configuration module you can configure the gateway itself and its adapters. A new adapter can be added by the *Add* button and later deleted by the red cross button. By selecting an adapter its configuration is shown. There one can set the name by which it can be identified in XRouter and choose its type (which specifies its later behavior). In the window bellow is the detail configuration that is specific to each type of adapter.

The gateway configuration module may look like this:

The screenshot displays the XRouter configuration interface. The top section, titled 'Adapters', contains a table with four entries: 'Pokladna' (Directory I/O adapter), 'Tisk' (Directory I/O adapter), 'Platba' (Web client adapter), and 'Objednávka' (Web client adapter). Each entry has a red 'X' button for deletion. Below the table is a '+ Add' button. The bottom section, titled 'Objednávka', shows the configuration for the selected adapter. It includes fields for 'Target URI' (http://localhost:8011/), 'Content type' (text/xml; charset=utf-8), 'SOAP action' (http://tempuri.org/IService/SaveOrder), 'Timeout' (60), and a checkbox for 'Persist input tokens' which is currently unchecked.

Adapters		
	Pokladna	Directory I/O adapter ✖
	Tisk	Directory I/O adapter ✖
	Platba	Web client adapter ✖
	Objednávka	Web client adapter ✖

+ Add

Objednávka

? Target URI

Content type

SOAP action

? Timeout

? Persist input tokens ☐

In case of invalid values being passed into the configuration, the adapter will not be loaded. One must be especially careful with validity of web addresses and resources, existence of directories on disk and permissions to them or with correctness of e-mail addresses.

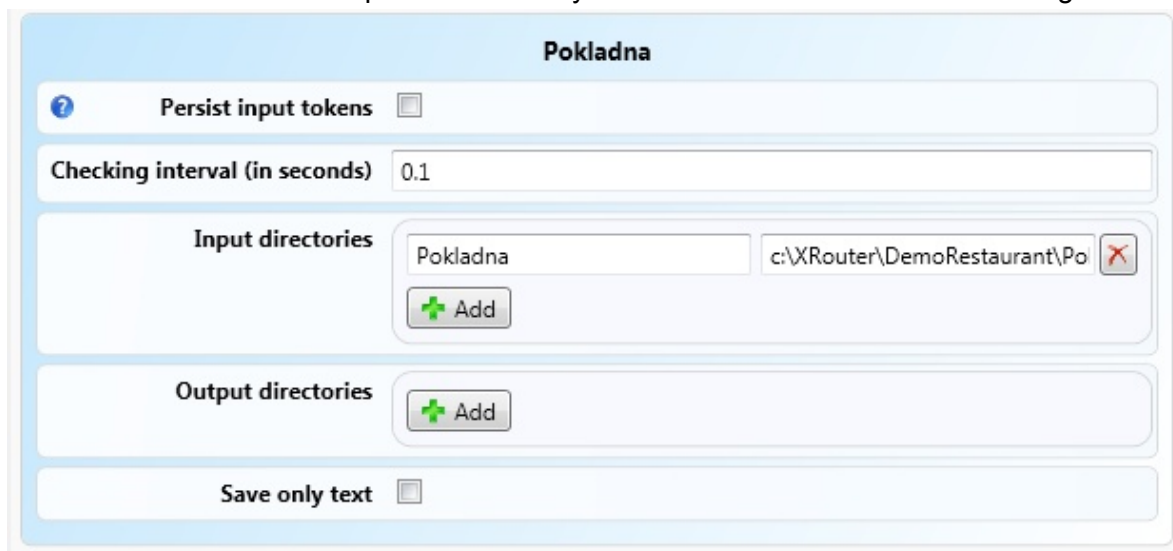
Settings common to all adapters:

- *Persist input tokens*: If checked the message flow state of each token originated from this adapter will be persistently stored (ie. in a database) after performing each step in the message flow processing.

Bellow are descriptions of configurations of various adapters present in XRouter.

Directory I/O adapter

Directory adapter provides file input/output in a shared file system directory. It periodically polls for new files in directories under its control. Any successfully loaded input file is received for further processing and removed from the file system. Also outgoing messages can be saved into files in a specified directory. Unreadable files or directories are ignored.



The screenshot shows the configuration window for the 'Pokladna' Directory I/O Adapter. It features several settings: a 'Persist input tokens' checkbox (unchecked), a 'Checking interval (in seconds)' text field with the value '0.1', an 'Input directories' section with a text field containing 'Pokladna' and a file path 'c:\XRouter\DemoRestaurant\Po', an 'Add' button, and a red 'X' button; an 'Output directories' section with an 'Add' button; and a 'Save only text' checkbox (unchecked).

A sample configuration of the Directory I/O Adapter

The following settings are available:

- *Checking interval*: The period (in seconds) in which the input directories are checked for changes. The exact interval is not guaranteed. Too small a period may cause degradation of responsiveness.
- *Input directories*: The list of directory paths which will be periodically controlled by XRouter. The paths may be added using the *Add* button and later removed using the red cross button. Each directory path represents an endpoint which then creates a part of the endpoint address, an endpoint identification unique within an XRouter Service instance (used in the message flow). The name has to be unique within a single adapter.
- *Output directories*: The list of directory paths which will be used for saving the outputs files. Otherwise similar to the *Input directories* parameter.
- *Save only text*: If checked, outputs are expected as text files, XML files otherwise.

It is not mandatory to set both input and output directories in a single adapter configuration. It depends on the desired behavior. A reasonable design might be even to create separate input and output adapters.

Email adapter

E-mail client adapter provides asynchronous sending of outgoing messages inside e-mails via SMTP. It does NOT provide receiving messages (eg. via POP3, IMAP). It can be used eg. for e-mail notifications where a XML message is stored in the attachment of an e-mail. One adapter instance represents a single e-mail template which can be sent to multiple e-mail addresses.

The screenshot shows a configuration form for an EmailAdapter. It includes fields for SMTP host, port, sender information, subject, body, and recipients. The Recipients field has an 'Add' button and a red 'X' icon. A 'Persist input tokens' checkbox is at the bottom.

A sample configuration of the Email sender adapter

The following settings are available:

- **SMTP host:** Host name or IP address of the SMTP server.
- **SMTP port:** Port where the SMTP server listens.
- **Sender address:** Source address of the e-mail message sender.
- **Sender name:** Display name of the e-mail message sender.
- **Subject:** The subject line of the e-mail message.
- **Body:** The message body.
- **Recipients:** A list of e-mail addresses of recipients of this e-mail message.

Web client adapter

HTTP client adapter provides a simple RPC-style client which can send messages to remote Web services synchronously. The content can be an arbitrary XML document (typically SOAP). The client sends the XML content to a service specified by its target URI using a specified SOAP action. The request can be terminated after given timeout. After sending the request a response from the web service is returned.

Platba	
Target URI	http://localhost:8011/
Content type	text/xml; charset=utf-8
SOAP action	http://tempuri.org/IService/GetReceipt
Timeout	60
Persist input tokens	<input type="checkbox"/>

A sample configuration of the Web client adapter

The following settings are available:

- **Target URI:** URI of the web service. Eg. `http://www.example.com:8080/path/`.
- **Content type:** Type of the content - a HTTP header.
- **SOAP action:** The method property of SOAP.
- **Timeout:** Timeout in seconds.

Web service adapter

HTTP service adapter provides a simple Web service listener (server). It can receive XML messages in requests from remote clients and respond to them in the RPC style. The XML content can be arbitrary (typically SOAP).

Web service adapter	
Listener URI prefix	http://localhost:8080/
Persist input tokens	<input type="checkbox"/>

A sample configuration of the Web service adapter

The following settings are available:

- **Listener URI prefix:** It is composed of a scheme (http), host name, (optional) port, and (optional) path. Eg. `http://www.example.com:8080/path/`.

Processor component

In the Processor module one may configure the number of threads used by the Processor component. The number has to be a positive integer greater than or equal to one. The number of threads can considerably influence the speed of message flow. A higher number enables better parallelization and more effective processing of smaller and bigger messages together (smaller messages do not have to wait so much for the bigger ones). But too high a number means a greater overhead (switching of the threads).

Concurrent processing threads	4
-------------------------------	---

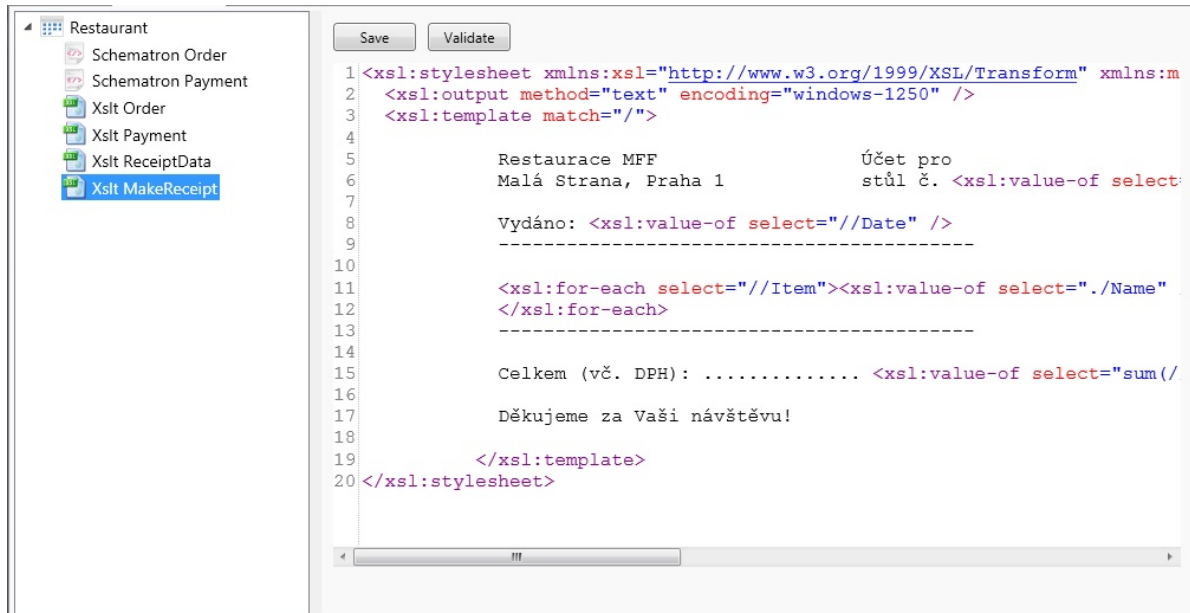
A sample configuration of the Processor component

XML Resources

The XML Resources module manages XML code used in XRouter. There are three types of resources:

- General XML
- Schematron
- XSLT

The resources are organized into groups. The module consists of a resource tree and a XML editor. By selecting a resource in the resource tree it gets opened in the XML editor. Adding or removing resources and groups is done by right-clicking on the resource tree and using the context menu.



A sample of the XML Resources module

The selected XML resource can be validated via the *Validate* button and then safely saved with the *Save* button.

Trace log and Event log

Trace log and Event log are modules that enable the user to explore the archived logs. Event log displays log entries about the XRouter Service itself, whereas Trace log displays the important steps done during the processing of messages and possible errors. Entities are divided into three levels:

- Info
- Warning
- Error

Entries may be filtered by date and time and by log level and then paged.

From	1.1.2000	To	1.1.2100	<input checked="" type="checkbox"/> Info <input checked="" type="checkbox"/> Warning <input checked="" type="checkbox"/> Error	Apply filter
Created	LogType	Message			
11.8.2011 20:46:18	Info	The 'xrouter' service is being started...			
11.8.2011 20:46:19	Error	An unexpected error occurred while starting the service: Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:46:19	Error	An unexpected error occurred while running service 'xrouter': Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:46:40	Info	The 'xrouter' service is being started...			
11.8.2011 20:46:41	Error	An unexpected error occurred while starting the service: Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:46:41	Error	An unexpected error occurred while running service 'xrouter': Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:46:53	Info	The 'xrouter' service is being started...			
11.8.2011 20:46:53	Error	An unexpected error occurred while starting the service: Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:46:53	Error	An unexpected error occurred while running service 'xrouter': Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:47:55	Info	The 'xrouter' service is being started...			
11.8.2011 20:47:55	Error	An unexpected error occurred while starting the service: Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:47:55	Error	An unexpected error occurred while running service 'xrouter': Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:50:16	Info	The 'xrouter' service is being started...			
11.8.2011 20:50:16	Error	An unexpected error occurred while starting the service: Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:50:16	Error	An unexpected error occurred while running service 'xrouter': Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:56:33	Info	The 'xrouter' service is being started...			
11.8.2011 20:59:10	Info	The 'xrouter' service is being started...			
11.8.2011 20:59:10	Info	The 'xrouter' service has been started successfully!			
11.8.2011 20:59:16	Info	The service is being stopped...			
11.8.2011 20:59:16	Info	The service has been stopped successfully!			
11.8.2011 20:59:26	Info	The 'xrouter' service is being started...			

An example of the Event log viewer

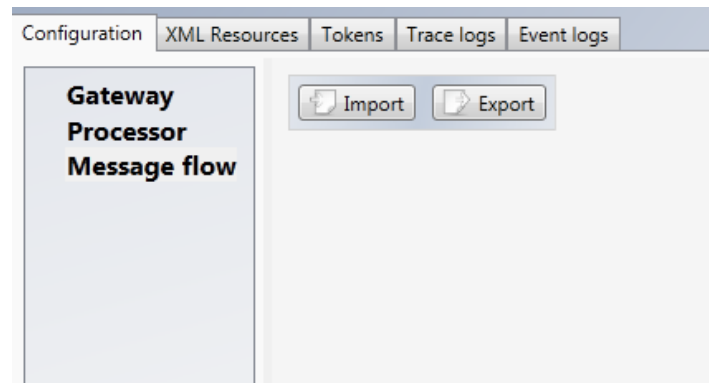
From	1.1.2000	To	1.1.2100	<input checked="" type="checkbox"/> Info <input checked="" type="checkbox"/> Warning <input checked="" type="checkbox"/> Error	Apply filter
Created	LogType	Message			
11.8.2011 20:46:19	Error	<exception type="System.Runtime.Serialization.SerializationException"> <message>Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:46:41	Error	<exception type="System.Runtime.Serialization.SerializationException"> <message>Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:46:53	Error	<exception type="System.Runtime.Serialization.SerializationException"> <message>Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:47:55	Error	<exception type="System.Runtime.Serialization.SerializationException"> <message>Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
11.8.2011 20:50:16	Error	<exception type="System.Runtime.Serialization.SerializationException"> <message>Expecting element 'AdapterConfiguration' from namespace 'http://schemas.datacontract.org/2004/07/System.ServiceModel.DataContract'...			
21.8.2011 14:16:15	Error	<exception type="System.ArgumentException"> <message>Schema must contain root node.</message> <stack-trace> at SchemaTron.ValidateSchema (C:\Program Files\Microsoft Dynamics CRM\bin\Microsoft.Dynamics.CRM.SchemaTron.dll) ...			
21.8.2011 14:17:29	Error	<exception type="System.ArgumentException"> <message>Schema must contain root node.</message> <stack-trace> at SchemaTron.ValidateSchema (C:\Program Files\Microsoft Dynamics CRM\bin\Microsoft.Dynamics.CRM.SchemaTron.dll) ...			
21.8.2011 14:17:52	Error	<exception type="System.ArgumentException"> <message>Schema must contain root node.</message> <stack-trace> at SchemaTron.ValidateSchema (C:\Program Files\Microsoft Dynamics CRM\bin\Microsoft.Dynamics.CRM.SchemaTron.dll) ...			
21.8.2011 14:18:58	Error	<exception type="System.ArgumentException"> <message>Schema must contain root node.</message> <stack-trace> at SchemaTron.ValidateSchema (C:\Program Files\Microsoft Dynamics CRM\bin\Microsoft.Dynamics.CRM.SchemaTron.dll) ...			
21.8.2011 14:37:52	Error	<exception type="System.NullReferenceException"> <message>Object reference not set to an instance of an object.</message> <stack-trace> at Microsoft.Xrm.Tooling.CrmMessageProcessor.Processor.ProcessMessage (C:\Program Files\Microsoft Dynamics CRM\bin\Microsoft.Xrm.Tooling.CrmMessageProcessor.dll) ...			
21.8.2011 16:21:17	Error	<exception type="System.ArgumentException"> <message>Cannot find component named 'gateway1'. Parameter name: componentName</message> <stack-trace> at Microsoft.Xrm.Tooling.CrmMessageProcessor.Processor.ProcessMessage (C:\Program Files\Microsoft Dynamics CRM\bin\Microsoft.Xrm.Tooling.CrmMessageProcessor.dll) ...			
21.8.2011 16:30:49	Info	Found input file Order00001.xml			
21.8.2011 16:31:10	Info	Created token with GUID b815cbd5-0c6c-4351-895e-4c42eb6ef4a7			
21.8.2011 16:33:04	Info	Dispatcher assigning token 'b815cbd5-0c6c-4351-895e-4c42eb6ef4a7' to processor 'processor'			
21.8.2011 16:33:22	Info	Processor received a token with GUID b815cbd5-0c6c-4351-895e-4c42eb6ef4a7			
21.8.2011 16:35:29	Info	Entering CBR: Typ			
21.8.2011 16:39:26	Info	Found input file Order00001.xml			
21.8.2011 16:39:26	Info	Created token with GUID 73b36dab-ea3e-4d5b-9543-da76a81eb36d			
21.8.2011 16:39:26	Info	Dispatcher assigning token '73b36dab-ea3e-4d5b-9543-da76a81eb36d' to processor 'processor'			
21.8.2011 16:39:26	Info	Processor received a token with GUID 73b36dab-ea3e-4d5b-9543-da76a81eb36d			
21.8.2011 16:39:32	Info	Entering CBR: Typ			

An example of the Trace log viewer

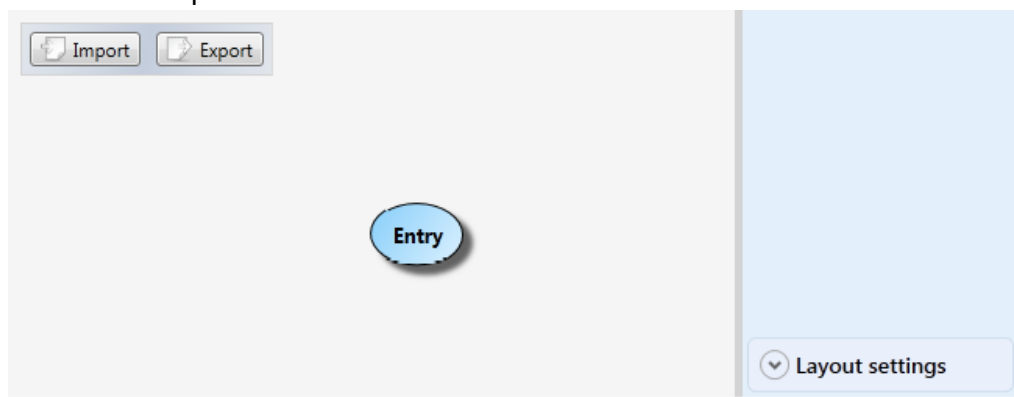
NOTE: When setting the date/time boundaries bear in mind that the From date is an inclusive lower bound and the To date is an exclusive upper bound. Specifying only the date means the time is 0 AM which excludes the rest of the day. In order to list a whole particular day one must specify the 12 PM on that day, or 0 AM on the next day.

Message flow editor

Message flow (configuration of the processor component of the XRouter Service) can be edited using *Message flow editor* located as the last entry in the left side panel of the *Configuration* tab:

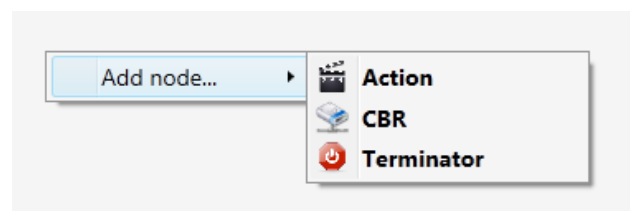


It consists of several parts:



Workspace

The largest part of Message Flow Editor, occupying the left side, is the workspace. It enables creating, moving and removing nodes (at the beginning there is a single *Entry* node like on the screenshot). Nodes can be added using mouse right-click context menu of the workspace:

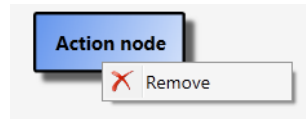


There are four types of nodes:

- *Entry* (not included in *Add node...* menu)
- *Action*
- *Content-based router (CBR)*
- *Terminator*

These types of nodes will be explained later.

A node can be removed using mouse right-click context menu (note that *Entry* node cannot be removed):



Every node can be moved around the workspace by dragging and dropping it. When a node is clicked, it is selected (which is denoted with slightly bigger size and darker color of the node). When a node is selected, its properties are shown in the properties side panel located on the right side of workspace.

Properties side panel

Properties side panel is located to the right side of the workspace. The largest portion of its area is occupied with properties of the selected node (if any node is selected). At the bottom, there is an expandable area with header *Layout settings*:

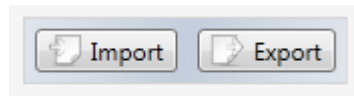


It allows changing the behaviour of the automatical node layout algorithm:

- *Apply repulsion force* – indicates whether nodes should be repulsed one from another (like reversely charged particles).
- *Apply attraction force* – indicates whether connected nodes should be attracted to each other (like with springs).
- *Length of the attraction spring* – how much should the connected nodes be attracted.

Import/exporting buttons

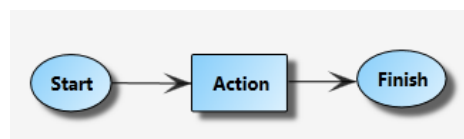
At the top-left corner, there are buttons for exporting a message flow into a file or importing a message flow from an exported file:



Nodes

The message flow is an oriented graph of nodes. Each node has a name which is displayed at the center of the node. The name is editable (using the *Name* property in the properties side panel) to better describe business it holds. NOTE: Every node must have a unique name, so that there cannot be two distinct nodes with the same name!

The processing flow among nodes is depicted with arrows, always going from the source node to the target node:



The properties side panel allows to specify the next node followed by selected node or target node of CBR branch using a target node selector button:



This is a special button. When a mouse hovers over it a target node is highlighted, so that all other nodes are semitransparent, thus showing which node is the target one. When the button is clicked, it stays pushed until a new target node is chosen by clicking. Choosing the target node can be cancelled by clicking on the button again which causes the button being no longer pushed.

The following sections describe four different types of nodes supported in the message flow.

Entry node



Every message flow has exactly one *Entry* node and it is not possible to add a new one or remove the automatically created one. This node represents an entry point and does no actions on its own. Beside *Name* it has the *Next node* property specifying to the next node in the processing flow:

A light blue rectangular panel with a white border. It contains three sections: a top section with the text "Entry node" in bold; a middle section with the label "Name:" followed by a text input field containing the word "Start"; and a bottom section with the label "Next node:" followed by a "Select" button with a green arrow icon.

Action node



Action node executes one or more actions (in parallel) performed with a token. Besides the *Name* and *Next node* (pointing to a node where the flow should continue when its actions finish) properties, it contains an editable list of actions:

A light blue rectangular panel with a white border. It contains a section titled "Actions" in bold. Inside this section is a list of actions, currently showing "Message sender" with a dropdown arrow and a red cross button to its right. Below the list is a green plus button with the text "Add".

Actions can be added using the *Add* button or removed using the red cross button on the right side of each item. There are two built-in types of actions (however more can be added as plugins):

- Message sender
- XSLT transformer

When an action is selected, the *Action configuration* panel is shown below the *Actions* list.

Message sender action

A sample configuration of the Message sender action

The message sender action takes a specific part of the token and sends it to an output endpoint on an adapter. It has the following parameters:

- **Target gateway** – the gateway name of target endpoint. Currently it should be always named gateway since the current implementation supports only one gateway.
- **Target adapter** – the name of the adapter which the message will be send to.
- **Target endpoint** – the name of the endpoint on the adapter which the message will be send to.
- **Message** – an XPath expression specifying which part of token will be send. If it starts with a dollar sign (\$), it is automatically substituted with an XPath pointing to a message named like the string following this dollar sign.
- **Metadata** – optional metadata useful in an adapter for sending the message. Meaning of metadata depends on the type of adapter.
- **Result message name** – Some adapters might return a response message. In this case this result message will be stored in the token with the name specified in the *Result message name* property.
- **Timeout** – how long the action waits until a message is received by the adapter.

XSLT transformer action

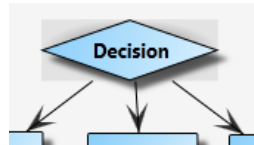
A sample configuration of the XSLT transformer action

This action takes a part of the token XML, transforms it using a specified XSLT and stores the result as a new message into the token.

- **XSLT** – specifies an XPath into the XML Resource Management (XRM) storage which selects XSLT to be used for transformation.
- **Input message** – an XPath expression specifying which part of token will be transformed. If it starts with a dollar sign (\$) it is automatically substituted with an XPath pointing to a message named like the string following this dollar sign.

- *Output message name* – The result message will be stored with the name specified in this property.
- *Is XSLT trusted* – indicates that the XSLT is allowed to contain scripting. Otherwise it is prohibited for security reasons.

Content-based router (CBR) node



A CBR node determines which Schematron schema is valid in the selected part of the token and based on this it decides which node will follow in the processing.

A sample configuration of the CBR node

- *Tested message* – an XPath expression specifying which part of token will be validated against Schematron schemas. If it starts with a dollar sign (\$) it is automatically substituted with an XPath pointing to a message name after the \$.
- *Default target* – points to a node which will be followed if no Schematron passes.
- *Branches* – CBR contains an editable collection of branches. Each branch has two parts: a schematron for validation and a target node which will be followed if the validation passes. Schematron is selected using an xpath into the Xml Resource Management (XRM) storage.

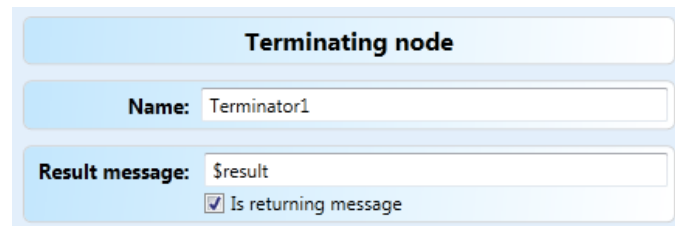
Evaluation of a CBR can be also tested in the properties side panel using *Testing panel*:

The *Evaluate* button causes the entered XML to be evaluated against Schematron schemas in the CBR's branches and the name of the next node in the processing will be displayed.

Terminator node



Message flow must be always terminated with a Terminator node. This node can optionally send a message to the original adapter (in which the token was created) as a result of the message flow processing.



Terminating node	
Name:	Terminator1
Result message:	\$result
<input checked="" type="checkbox"/> Is returning message	

A sample configuration of the Terminator node

- *Is returning message* – indicates that a reply should be sent to the original adapter.
- *Result message* – contains an XPath expression specifying which part of token will be returned as a message flow result. If it starts with a dollar sign (\$) it is automatically substituted with an xpath pointing to the message named like the string following this dollar sign.

Demo projects

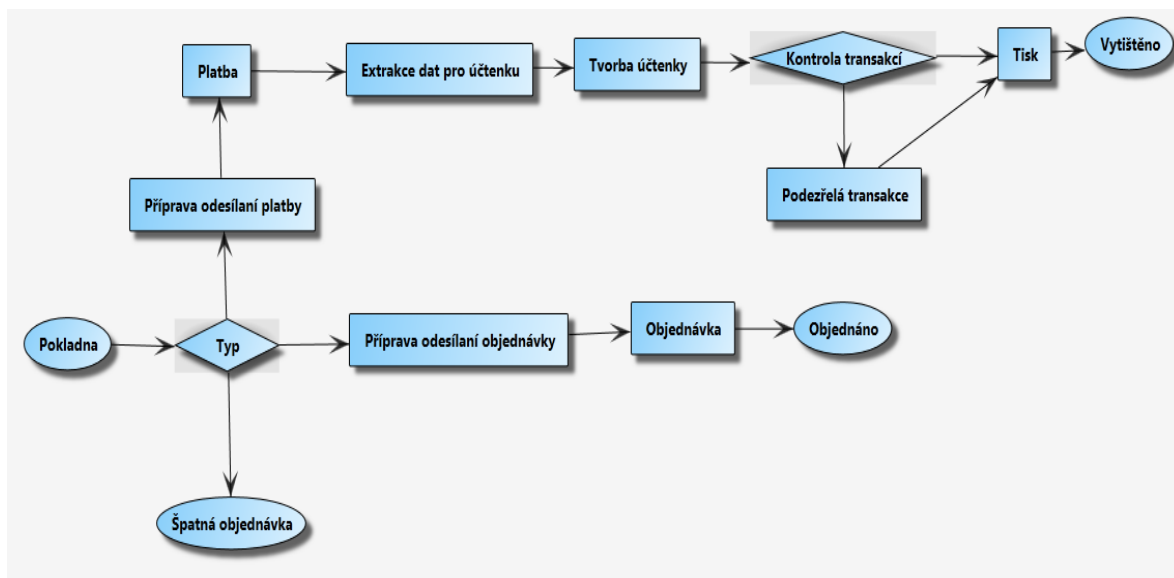
Restaurant

In order to better show the capabilities of XRouter, we prepared a simple demonstration. Its purpose is to utilize XSLT, sending e-mails, use directory adapters, Web service adapters and content-based routing in a quite real-life situation, while trying to keep it simple.

Imagine a larger restaurant where each table can order various food and drinks (for simplicity orders are made per table and not for each person). Orders can be made on one of several terminals which send the orders to the kitchens and to a central accounting system. When a table is ready to pay, the register again notifies the central system and gets back a receipt which is then printed by the terminal. The receipt also goes through a control that looks for suspicious transactions - for the sake of simplicity just sends notifications via mail if a receipt's final amount is too high.

The central system has prices and names of each food stored, and so it is easy to maintain and there should not be any synchronization problems as all the information and processes are run centrally. The central system might in fact be a simple web service that has a database with all the needed data. XRouter takes care of integrating terminals with the central system by providing a communication channel and transforming data between the forms required by each particular system.

Of course, in a real life scenario there might be more than one restaurant and each might have its own instance of XRouter (thus ensuring it could run even when being temporarily off-line). There would be a Web service to take local orders and payment request but its data might also be synchronized using a Web service connected to the central XRouter instance. On this central system there might be processes for repricing, managing menus, supplies etc.



MessageFlow of the Restaurant demo

And if we went even further, using the central XRouter might be even some sort of central data storage, central management system, accounting system, OLAP cube, systems of main suppliers and so on. A lot of these would need a specialized adapter implemented. But back to the simple demo where there is just one restaurant and all used system are just simple mock objects.

In the simulation we have some generated orders and payment requests that are entered into the systems as XML files in the directory 'Pokladna'. From there they are taken by the directory adapter and sent into a Content-based router which determines whether the given file is an order or a payment. An order is run through a XSLT to be in a correct format to pass to a Web Service. Using the Web service method Save, the order is stored into the central system. If the given file was a payment request, it requests the data for receipt from the central system using the Web service method Get (both the entered and returned data run through a simple XSLT to be in a correct format). The receipt data are then run through another Content Based Router and if the final amount is larger than 500Kč the mail notification about it are sent with the receipt as an attachment.

The receipt is generated using a slightly more complex XSLT as a text file which is then put into a directory (using a Directory adapter this time for writing) 'Tisk' which symbolizes printing the receipt. The simulation of the central system is just a Web service which at the start-up reads a menu file with the names and prices of items and remembers what each table has ordered since its last payment.

How to run this demo

- Demo files installed and generated by the installer, though you can choose not to generate them. in that case you may have to run the installation again and select adding them, as they are essential for this demo.
- You need to have the XRouter service running with the configuration of this demo. For example you may import the config file from:
`<InstallDir>\XRouter\Examples\Restaurant\Data\app-config.xml`
- Mail notification also connects to a SMTP server that of course has to be running and configured too.
- To launch the service simulating the central system by running (as admin):
`<InstallDir>\XRouter\Examples\Restaurant\Data\RestaurantService.exe`

- The input data are generated in C:\XRouter\DemoRestaurant\InputData\
 - Order<Number>.xml is a random order.
 <?xml version="1.0" encoding="utf-8"?>
 <order>
 <table>3</table>
 <item>food0007</item>
 </order>
 - Whereas Payment<Number>.xml is a request to pay for the table with the same number.
 <?xml version="1.0" encoding="utf-8"?>
 <payment>
 <table>4</table>
 </payment>
- You may generate new input data by
 <InstallDir>\XRouter\Examples\Restaurant\generate-demo-data.bat
- You can simulate incoming orders by copying the input files into the directory representing the terminal output in: C:\XRouter\DemoRestaurant\Pokladna

After each entered Payment file a receipt containing all the orders for the given table is generated into the directory representing the printer input in C:\XRouter\DemoRestaurant\Tisk:

```

Restaurace MFF                      Účet pro
Malá Strana, Praha 1                stůl č. 5

Uydáno: 27.8.2011 2:06:22
-----

Utopenec      (1X) ..... 40,00 Kč
Guláš         (3X) ..... 120,00 Kč
-----

Celkem (vč. DPH): ..... 160,00 Kč

Děkujeme za Vaši návštěvu!

```

How to create this demo

- You need to have XRouter Manager service (and XRouter database) running.
- Run XRouter Configuration Manager GUI and connect it to the XRouter Manager.
- Create new configuration by clicking *Configuration/New*.
- In the Gateway you have to create adapters by clicking *Add* in the Adapters window:
 - Type its name as 'Pokladna' and in the select list pick Directory I/O adapter. In the lower window (detail of the new adapter) add new Input directory (by clicking Input directories/Add), name it 'Pokladna' (left textbox) and set the path 'c:\XRouter\DemoRestaurant\Pokladna' (in the right textbox).
 - The second one is also Directory I/O adapter, but with name 'Tisk' and the Output directory (added the same way as in Pokladna) 'Tisk' with path 'c:\XRouter\DemoRestaurant\Tisk'.
 - 'Platba', pick type Web client adapter. Set Target URli as 'http://localhost:8011/' and SOAP action as 'http://tempuri.org/IService/GetReceipt'.
 - 'Objednávka', pick type Web client adapter. Set Target URI as 'http://localhost:8011/' and SOAP action as 'http://tempuri.org/IService/SaveOrder'.
 - 'Email', type 'E-mail sender adapter', add the email address of the recipient and fill the rest for example this way:

Email	
SMTP host	192.168.10.1
SMTP port	25
Sender address	xrouter-service@xrouter.dyndns.info
Sender name	XRouter
Subject	Kontrola podezřelých transakcí - příliš vysoká částka
Body	Tento mail byl vygenerován v rámci kontroly podezřelých transakcí: došlo k překročení hlídaného limitu celkové částky na jedné účtence.

- In the **XML Resources** tab in the left panel
 - Create a new Group: right-click, **Add/Group**, name it 'Restaurant'.
 - Add the following XML resources by right-clicking the newly created group and clicking **Add** from the content of the following files in the directory `<InstallDirectory>\XRouter\Examples\Restaurant\Data\`:
 - Xslt ReceiptData.xml (add as XSLT Xslt ReceiptData)
 - Xslt Payment.xml
 - Xslt Order.xml
 - Xslt MakeReceipt.xml
 - Schematron Payment.xml (as Schematron Schematron Payment)
 - Schematron Order.xml
 - Schematron Mail.xml

Then in the Message flow:

- Click **Entry** node and name it 'Pokladna'
- By right-clicking, **Add node/Action** or **CBR** or **Terminator** you can add various new nodes
- After selecting a node you may change its properties in the right panel
- Create Terminators, set their names as 'Špatná objednávka', 'Objednáno', 'Vyčištěno'
- Create 3 new nodes and add them all an **Action Message Sender** and set them **Name**, **Target adapter**, **Target endpoint**, **Message** and **Result message name** as follows:
 - Platba, Platba, '', \$paymentToSend, receiptData
 - Objednávka, Objednávka, '', \$orderToSend, ''
 - Tisk, Tisk, Tisk, \$receipt, ''
- Create four new nodes and add them all an **Action XSLT transformer** and set them **Name**, **Input message**, **Output message name** and select a resource as follows:
 - Příprava odesílání objednávky, //order, orderToSend, Xslt Order
 - Příprava odesílání platby, //payment, paymentToSend, Xslt Payment
 - Extrakce dat pro účtenku, \$receiptData, receiptDataExtracted, Xslt ReceiptData
 - Tvorba účtenky, \$receiptDataExtracted, receipt, Xslt MakeReceipt
- Create a CBR named Typ and click Default target's select and pick 'Špatná objednávka' in the message flow and add two branches where you choose schematron from XML Resources and use select to pick a target as follows:
 - Schematron Order, Příprava odesílání objednávky
 - Schematron Payment, Příprava odesílání platby
- Create a CBR named Kontrola transakcí and click Default target's select and pick 'Tisk' in the message flow and add one branch where you choose schematron from XML Resources and use select to pick a target as follows:
 - Schematron Mail, Podezřelá transakce

- By using *Select* in Next node create the graph of Message flow as was in the picture earlier.
- After you are ready, upload it to the server by clicking *Configuration/Upload* or export it into a file by clicking *Configuration/Export*.
- Restart the XRouter Service with the given file as configuration or let it automatically download it from the server.

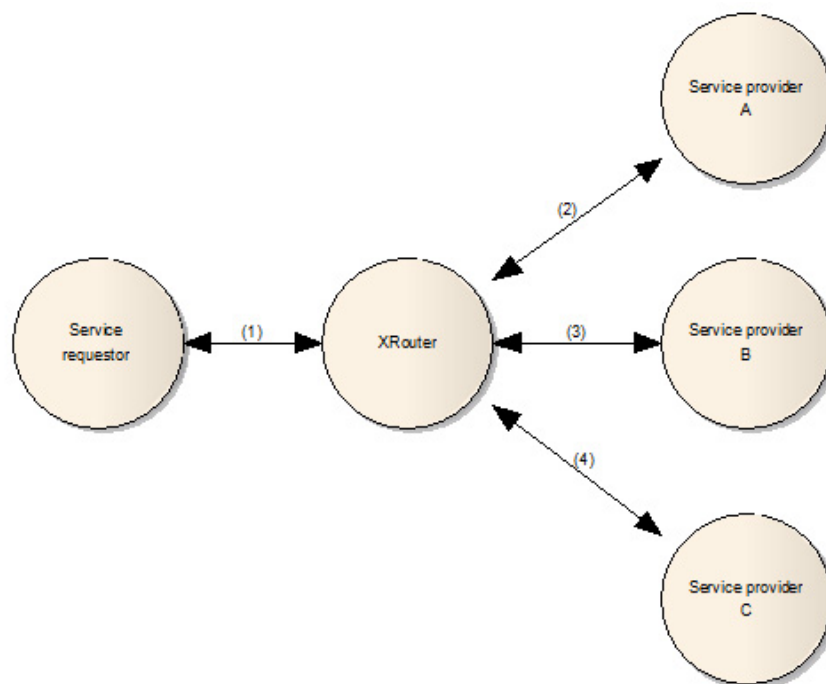
Support for SOA

This example demonstrates the technical support for SOA creation. In the background runs an instance of XRouter and protocol SOAP with WS-Addressing extension which is the crucial for creating the more complex architectures.

The instance of XRouter is used both for routing according to the SOAP headre, for modifying it (in this case it just clears its content, because the next node is the final receiver and so it is not needed anymore), and as a place containing the loose binding of service identifiers and their physical placement.

There are four external components on the picture and the XRouter Service which provides the exchange of messages among them. The service requestor needs to use the service providers (A, B, C, or potentially others). Assume the service requestor, a Web client, is a part of an application for managing orders, keeping records, doing cancellations and closing the orders. These functionalities will be implemented as Web services — service providers A, B, C.

The distribution logic of the service requestor will be moved into the XRouter service (1) and the decision about the target service provider (2),(3) or (4) will be made based on the content of messages that are produced by the service requestor (specifically their header metadata – SOAP WS-Addressing). The arrows in the picture are in fact two-way as the communication is of the request-response style.



The service requestor produces SOAP requests like the following, where only an identifier of the target receiver is put into the <To> element with WS-Addressing. In other words it

determines which operation to use. To keep it simple, the application message contains only an <order> element which contains an identifier of the order used by the application for managing orders.

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:To>A</wsa:To>
  </S:Header>
  <S:Body>
    <order id="1"/>
  </S:Body>
</S:Envelope>
```

As a response the service requestor will receive the following SOAP response from the service provider. The application message contains only a <response> element with the OK status in case the operation was succesfull (according to the application logic) or ERROR otherwise.

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Body>
    <response status="OK"/>
  </S:Body>
</S:Envelope>
```

The XRouter MessageFlow has to contain a single CBR node able to route incoming requests according to the values of the <To> element. After determining the route direction the following sequence of steps is be done:

- Action: clear the SOAP header (XSLT)
- Action: call the target service provider (with the transformed message)
- Terminator: return the response from the target service provider to the calling service requestor

For clearing the SOAP header the following XSLT may be used:

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:S="http://www.w3.org/2003/05/soap-envelope">
  <xsl:template match="node()|@"*>
    <xsl:copy>
      <xsl:apply-templates select="node()|@"*/>
    </xsl:copy>
  </xsl:template>
  <xsl:template match="S:Header"/>
</xsl:stylesheet>
```

Order Record keeping

The service requestor generates a SOAP request whose <To> element contains the identifier A and sends it to the XRouter. In the CBR the following predicate will be created:

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <ns prefix="soap" uri="http://www.w3.org/2003/05/soap-envelope"/>
  <ns prefix="wsa" uri="http://www.w3.org/2005/08/addressing"/>
```

```

<pattern id="soapEnvelope">
  <rule context="/">
    <assert test="soap:Envelope"/>
  </rule>
</pattern>
<pattern id="soapHeader">
  <rule context="/soap:Envelope">
    <assert test="soap:Header"/>
  </rule>
</pattern>
<pattern id="soapTo">
  <rule context="/soap:Envelope/soap:Header">
    <assert test="wsa:To"/>
  </rule>
</pattern>
<pattern id="soapToValue">
  <rule context="/soap:Envelope/soap:Header/wsa:To">
    <assert test="text()='A'"/>
  </rule>
</pattern>
</schema>

```

Doing cancellation

The service requestor generates a SOAP request with the identifier B in the <To/> element and sends it to the XRouter.

A Schematron schema for CBR predicate look like the one for the Order Record Keeping (which offers us the chance to use the schema composing in the XML Resources with the help of element <include>), only the pattern id="soapToValue" will have a different shape:

```

<pattern id="soapToValue">
  <rule context="/soap:Envelope/soap:Header/wsa:To">
    <assert test="text()='B'"/>
  </rule>
</pattern>

```

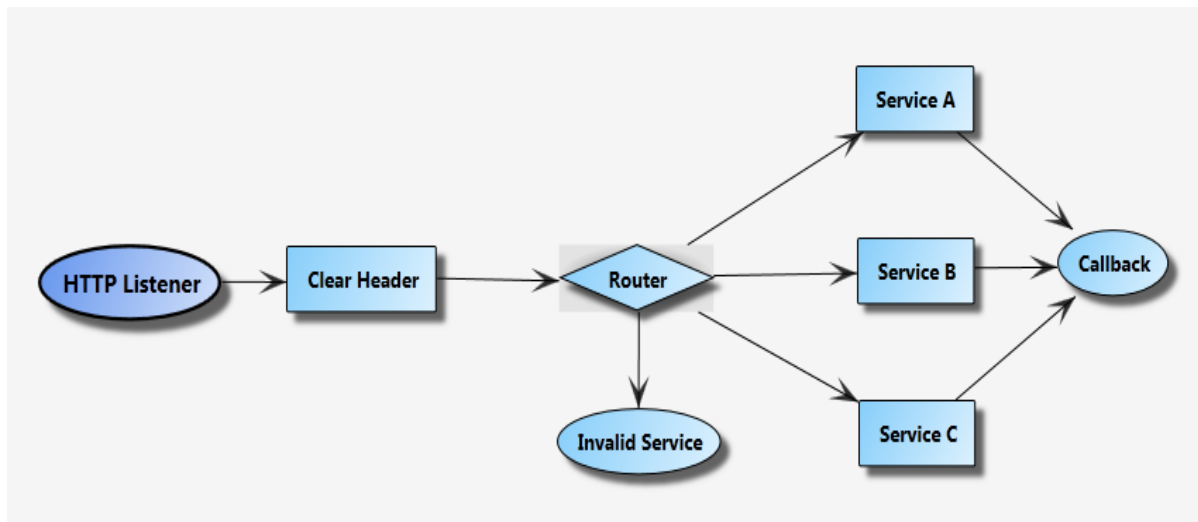
Order Closing

The service requestor generates a SOAP request with the identifier C in the <To/> element and sends it to the XRouter. A Schematron schema for the CBR predicate might again look similar to the previous one, only the pattern id="soapToValue" will have a different shape:

```

<pattern id="soapToValue">
  <rule context="/soap:Envelope/soap:Header/wsa:To">
    <assert test="text()='C'"/>
  </rule>
</pattern>

```



Message flow for the SOA demo

Note: Clearing the header can be done beforehand, because in the token both states of message are kept and so the routing may be done on the original input message while the cleared state can be used as an input to the correct service.

For the example a software support is created (simulations of the integrated components). See folder <InstallDir>\Examples\SOASupport\. Here is its documentation:

Provider.exe

Provider is represented by a native Web service run from a console application. There are three of them in this example (service types A, B, C) and with the help of Provider.exe we can run them with URI (where it is going to listen) and service type (A, B or C) as the parameters.

Example:

```

Provider.exe http://localhost:8080/ A
Provider.exe http://localhost:8081/ B
Provider.exe http://localhost:8082/ C
  
```

Requestor.exe

Requestor.exe implements a native Web client in a console application. Sending a message can be done by running the application with these parameters:

- uri - URI where XRouter listens
- requestFilename - path to the input file containing the XML request
- responseFilename - path to the output file that will contain the response

All the other resources needed for the configuration of MessageFlow (schemas, XSLTs) are mentioned above. In the XRouter we need to use one HTTPService node and three HTTPClient nodes that will call the providers. As for the HTTPClient the SOAPAction may be left empty.