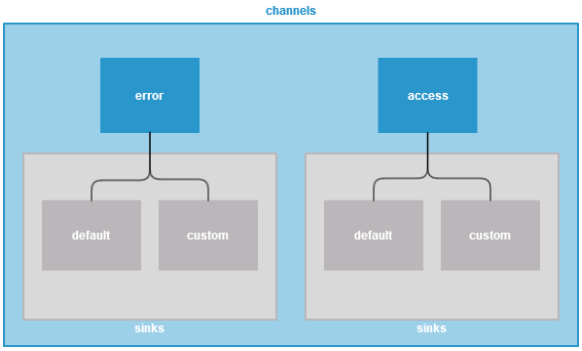
Defining a Custom Logger Configuration

This page explains the **Logger adapter** in Bridge context. If you were looking for the same information regarding the PAS Designer, refer to Logger Adapter in the Designer guide.

Runtime 2020.8 If you have special logging needs, you do not need to stick with the standard logging configuration (xUML Service Standard Log and xUML Service Transaction Log): You can define your own logger configuration based on the concepts of channels and sinks:



Element	Description	Allowed Values / Examples	
channel	A channel is an object that describes the data that will be written to a log file. It is identified by a channel name. The following channel names are reserved for internal use of the xUML Runtime: • error • access • channels starting with "xUML" in any casing	`error`	Write service logging data (bridgeserver log).
		`access`	Write transaction logging data.
sink	A channel can contain an arbitrary number of sinks. Sinks define the logging output and how it is written: • log file name and path patterns • log file format • logged content Sink names are not important but you need them to access the logging configurations via the xUML Runtime API. Do not rename the sinks of the access and error channels.		

For more information about the concept in general, refer to xUML Runtime Logger Configuration.

On this Page:

Related Pages:

- Log Adapter Reference
- logTo() Function

Related Documentation:

- BRIDGE Integration Platform User's Guide
  - xUML Service Standard Log
  - xUML Service Transaction Log
  - xUML Runtime Logger Configuration

Modeling a Custom Logger Configuration

Example File (Builder project E2E Action Language/Operating):

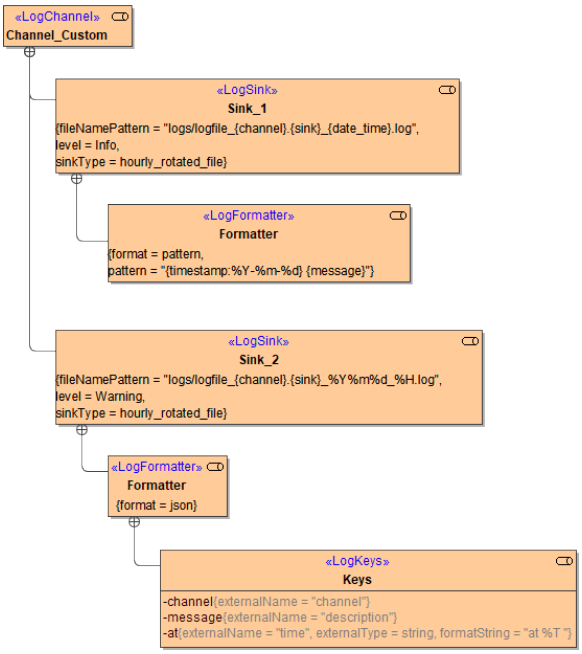
<your example path>\E2E Action Language\Operating\uml\logger_channels.xml

In your UML, models, you can define your own custom channel and sinks to log to.This is done by class diagram containing a set of classes with dedicated stereotypes:

- <<LogChannel>> for channels
- <<LogSink>> for sinks
- <<LogFormatter>> for logfile content formats

- <<LogKeys>> for defining your own JSON format on base of the standard JSON

Define the corresponding classes and relate them with containment relations as depicted below:



The above class diagram defines the following configuration:

Channel	Sink		
Channel_Custom	**Sink_1**	Filename / Path	`logs/logfile_Channel_Custom.Sink_1_2020-06-25-16.log`
		Content	Log all messages with log level **Info** or higher to this sink.
		Type	Rotate log file hourly.
		Format	Output logs to a text file using a dedicated pattern.
	Sink_2	Filename / Path	`logs/logfile_Channel_Custom.Sink_2_20200625_16.log`
		Content	Log all messages with log level **Warning** or higher to this sink.
		Type	Rotate log file hourly.
		Format	Output logs in JSON format.
		JSON attributes	• Add **channel** to the logged content • Overwrite attribute name **message** with name **description**. • Change name and format of the timestamp (**at**).

For a detailed description of all tagged values and their allowed values refer to Log Adapter Reference.

Defining the Filename of the Log File

Sinks can be configured to log to a file by using a **sinkType** that is **daily_rotated_file** or **hourly_rotated_file**. In these cases, you can specify a filename pattern for the log file to be generated using a <<LogFormatter>> class.

The following variables are available:

Runtime 2020.8 Builder 7.10.1

Variable	Description
`{channel}`	Channel name from the <<LogChannel>> class.
`{sink}`	Sink name from **name**.
`{date_time}`	Timestamp of format `%Y-%m-%d` or `%Y-%m-%d-%H`, depending on the **sinkType** .

`{extension}`	Value depends on the formatting specified in the <<LogFormatter>> class: • **json**: If **json** is selected as log format. • **log**: If **pattern** is selected as log format.

Refer to Log Adapter Reference for more formatting options.

> You can only define one <<LogFormatter>> per sink.

Defining the Format of the Logged Content

As per default, the Logger logs to a JSON file if a custom configuration is specified. You have several options to change the log file content, though.

Logging to Text Files

Sinks can be configured to log to a text file by setting **format** of the related <<LogFormatter>> class to **pattern**.

To format the content of text files, you can use the following variables in tagged value **pattern**:

Runtime 2020.8 Builder 7.10.1

Variable	Description
`{channel}`	Log channel.
`{timestamp}`	Timestamp.
`{level}`	Log level.
`{message}`	Log message.
`{compositeName}`	Name of the service composite (see Frontend Components).
`{trxId}`	Transaction id (see Contents of the Transaction Log).
`{correlationId}`	Correlation id (see Contents of the Transaction Log).

Refer to Log Adapter Reference for more formatting options.

Logging to JSON Files

Sinks can be configured to log to a JSON file by setting **format** of the related <<LogFormatter>> class to **json**.

For JSON log files, the following JSON attributes are available:

Runtime 2020.8 Builder 7.10.1

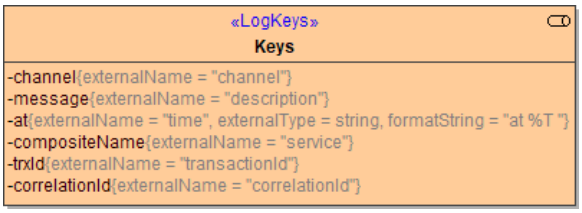
JSON Attribute	Description
`{channel}`	Log channel.
`{at}`	Timestamp.
`{message}`	Log message.
`{details}`	Serialized content of the details object.
`{compositeName}`	Name of the service composite (see Frontend Components).
`{trxId}`	Transaction id (see Contents of the Transaction Log).
`{correlationId}`	Correlation id (see Contents of the Transaction Log).

- If nothing is specified but **format = json**, the default configuration will be used and the log file will look like

```
{"at":"<a timestamp>","message":"<a log message>"}
```

- You can change the appearance of the JSON file using a <<LogKeys>> class to specify your changes.

  This is done by adding an attribute to the <<LogKeys>> class whose name corresponds to the name of the attribute you want to change.



  Then, set the tagged values as follows:

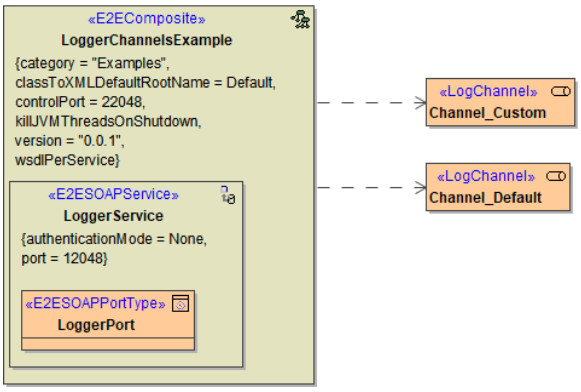
Topic	Tagged Value	Allowed Values	
add attribute	externalName	You can add all attributes listed in the table above. Attributes are only visible if an external name has been added. The external name, however, may be the same name as the name of the attribute.	
remove attribute	Empty all tagged values.		
change attribute name	externalName	Specify a name for the JSON attribute to be used in the log file. This name must conform to the rules for JSON attribute names.	
change type of attribute "at"	externalType	integer	Convert to **Integer**.
		string	Convert to **String**.
change format of attribute "at"	formatString	Specify how to format the content when serializing it to the specified **externalType**. Refer to Log Adapter Reference for all available formatting options. Note, that the format string must match the specified **externalType**. The xUML Compiler cannot check the correctness of this format string.	

- The Compiler will show an error if you try to add an unknown attribute to the <<LogKeys>> class.

Using a Custom Logger Configuration

Components

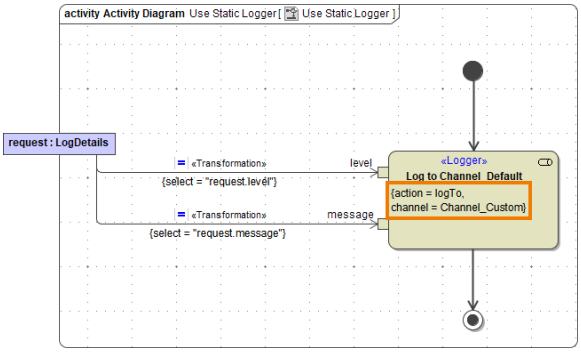
In the component diagram of the xUML service, draw a dependency from the composite to the log channels you want to use:



Activity Diagram

With the <<Logger>> adapter you can write into the log file of the service. Use action **logTo** in this case to use your custom logger configuration.

The following picture shows the usage of the <<Logger>> adapter:



Tag **action** of the logger adapter needs to be set to **logTo**. Additionally, provide the **channel** you want to log to.
The following input can be provided:

Runtime 2020.8 Builder 7.10.1

Name	Type	Direction	Description	Allowed Values / Example
channel	String	in	Specify the channel you want to log to.	
level	String	in	Specify a log level. Allowed error log levels are described in the log level guidelines on Bridge Server Log Levels of an xUML Service.	One of **Fatal**, **Error**, **Warning**, **Info**, and **Debug**.
message	String	in	Specify a description for the log entry.	`Item ID could not be found.`
details	Any	in	Specify an object of complex type (class or array) that contains additional details. If provided, the contents of this object will be flattened and appended to the **description** for text files. In JSON files, **details** have their own key.	any detail object

As an alternative - e.g. to set the channel dynamically - you can use Action Script and the logTo() Function.

Example

Info (Sink_1)

Pin	Value
channel	`Channel_Custom`
level	`Info`
message	`Synchronized.`

The output after running the example can be found in custom logfile `logs/logfile_Channel_Custom.Sink_1_2020-06-25-16:35:46.log`.

```
2020-06-25 Synchronized.
```

Warning (Sink_1 and Sink_2)

Pin	Value
channel	`Channel_Custom`
level	`Warning`
message	`Item not available.`

The output after running the example can be found in both logfiles:

```
 logs/logfile_Channel_Custom.Sink_1_2020-06-25-16.log

2020-06-25 Item not available.

logs/logfile_Channel_Custom.Sink_2_2020-06-25-16.log

{"time":"at 10:22:03","channel":"Channel_Custom","description":"Item not
available."}
```