

KELVERION AUTOMATION

INTEGRATION MODULE FOR DATA MANIPULATION

*FOR KELVERION RUNBOOK STUDIO AND
AZURE AUTOMATION*

User's Guide

Version 1.1

Kelverion Integration Module for Data Manipulation

Copyright © Kelverion Automation Limited. All rights reserved.

Published: October 2019

FEEDBACK

Send suggestions and comments about this document to support@kelverion.com

CONTENTS

Installation and Configuration	6
System Requirements.....	6
Installing the Integration Module from PowerShell Gallery.....	6
Installing the Integration Module Manually.....	7
Licensing the Integration Module.....	8
Data Manipulation Smart Connections	9
Azure Global Connection Assets.....	9
Data Manipulation Activities.....	10
Common Configuration Instructions for All Activities.....	10
Activity Properties	10
Activity.....	10
Discovery	10
Parameters	10
Filters.....	10
Retry Behaviour	11
Apply-Xslt.....	11
Required Parameters.....	11
Outputs.....	11
PowerShell Command Reference.....	11
Compose-Text.....	11
Discovery Parameters	12
Required Parameters.....	12
Optional Parameters	12
Outputs.....	12
PowerShell Command Reference.....	12
Configuring Inputs for Compose-Text	12
Convert-JsonToXml.....	13
Required Parameters.....	13
Optional Parameters	13
Outputs.....	13
PowerShell Command Reference.....	14
JSON to XML using Convert-JsonToXml	14
Converting to Valid XML.....	14
XML Attributes	14
XML Declaration	14
Convert-XmlToJson.....	14
Required Parameters.....	15
Optional Parameters	15
Outputs.....	15
PowerShell Command Reference.....	15
Parse-Text.....	15
Discovery Parameters	15
Required Parameters.....	15

Filters.....	15
Outputs.....	15
PowerShell Command Reference.....	16
Parsing and Correlated Data for Parse-Text.....	16
Custom Date Time Formats.....	18
Standard Date and Time Format Specifications	18
Custom Date and Time Format Strings	18
Element Reference.....	20
XML Specification for Parse-Text and Compose-Text.....	21
<ka:DataManipulation>	21
Definition and Usage.....	21
Syntax.....	21
Attributes	21
<ka:ComposeText>	21
Definition and Usage.....	21
Syntax.....	21
Attributes	21
<ka:ParseText>	21
Definition and Usage.....	21
Syntax.....	21
Attributes	22
<ka:Attribute>	22
Definition and Usage.....	22
Syntax.....	22
Attributes	22
Example.....	22
<ka:AttributeSet>	22
Definition and Usage.....	22
Syntax.....	22
Attributes	22
Example.....	23
<ka:Choose>	23
Definition and Usage.....	23
Syntax.....	23
Example.....	23
<ka:Element>	23
Definition and Usage.....	23
Syntax.....	23
Attributes	24
Example.....	24
<ka:Input>	24
Definition and Usage.....	24
Syntax.....	24
Attributes	24

<ka:If>	25
Definition and Usage	25
Syntax	25
Attributes	25
Example	25
<ka:Message>	26
Definition and Usage	26
Syntax	26
Attributes	26
Example	26
<ka:TextTemplate>	26
Definition and Usage	27
Syntax	27
Attributes	27
<ka:Otherwise>	27
Definition and Usage	27
Syntax	27
Example	27
<ka:Output>	28
Definition and Usage	28
Syntax	28
Attributes	28
Example	29
<ka:Text>	29
Definition and Usage	29
Syntax	29
Example	29
<ka:When>	30
Definition and Usage	30
Syntax	30
Attributes	30
Example	30
<ka:Value>	30
Definition and Usage	30
Syntax	31
Example	31
<ka:ValueOf>	31
Definition and Usage	31
Syntax	31
Attributes	31
Example	31
<ka:Variable>	32
Definition and Usage	32
Syntax	32
Attributes	32

INSTALLATION AND CONFIGURATION

The following sections outline how to deploy and configure the Kolverion Integration Module for Data Manipulation.

SYSTEM REQUIREMENTS

The Integration Module for Data Manipulation requires the following software to be installed and configured prior to implementing the integration.

- Kolverion Runbook Studio 3.4
- Microsoft .NET Framework 4.6.2
- Azure Automation account

The Integration Module for Data Manipulation supports Azure Automation and Service Management Automation (SMA) platforms:

- Azure Automation requirements:
 - Azure Automation account
 - Hybrid Worker Group with at least one Hybrid Worker (Only required to access on-premises resources.)
- SMA requirements:
 - System Center 2016 SMA
 - At least one Automation Worker

INSTALLING THE INTEGRATION MODULE FROM POWERSHELL GALLERY

You can install the Integration module directly from the PowerShell Gallery to your Azure Automation Account and Hybrid Workers. If you want to build runbooks in Kolverion Runbook Studio using the integration module you must also install it on your Runbook Studio host system.

To install the Integration Module for Runbook Studio:

1. Confirm you have the latest PowerShellGet module. See online documentation for more information: <https://www.powershellgallery.com/>
2. Start a PowerShell window as Administrator and run the command:
`Install-Module -Name Kolverion.DataManipulation`

To install the Integration Module in Azure Automation:

1. Go to the PowerShell Gallery URL:
<https://www.powershellgallery.com/packages/Kolverion.DataManipulation>
2. Click the button “**Deploy to Azure Automation**” and you will be redirected to the Microsoft Azure portal.
3. Select your Automation account and click **OK** to upload the module.

To install the Integration Module on a Hybrid Worker:

1. Confirm you have the latest PowerShellGet module on your Hybrid Worker computer. See online documentation for more information: <https://www.powershellgallery.com/>
2. Start a PowerShell window as Administrator and run the command:
`Install-Module -Name Keverion.DataManipulation`

INSTALLING THE INTEGRATION MODULE MANUALLY

After download the integration module, you must upload it to your Portal and Worker hosts. If you want to build runbooks in Keverion Runbook Studio using the integration module you must also install it on your Runbook Studio host system.

To install the Integration Module for Runbook Studio:

1. Copy the **Keverion.DataManipulation.zip** to your computer.
2. Right click on the file and select properties. Unblock the file if it is blocked.
3. Unzip the **Keverion.DataManipulation.zip** file.
4. Copy the **Keverion.DataManipulation** folder to a location in the %PsModulePath% path.

To install the Integration Module in Azure Automation Portal:

1. Go to your automation account and select **Assets**.
2. In **Assets** select **Modules**.
3. In **Modules** select **Add a module**.
4. In **Upload File** field, select the **Keverion.DataManipulation.zip**.
5. Select **OK** to upload the module.

To install the Integration Module on a Hybrid Worker:

1. In the Keverion Runbook Studio or Microsoft Azure portal:
 - a. Create an Runbook with the following PowerShell: `$Env:PSModulePath`.
 - b. Test this runbook on the Hybrid Worker and record the output for use in Step 5.
2. Copy the **Keverion.DataManipulation.zip** to your Hybrid Worker computer.
3. Right click on the file and select properties. Unblock the file if it is blocked.
4. Unzip the **Keverion.DataManipulation.zip** file.
5. Copy the **Keverion.DataManipulation** folder to a location in the path returned in Step 1b.
Note: The Hybrid Worker uses its own unique value for %PSModulePath%, and will differ from that of the system.

To install the Integration Module in SMA 2016:

1. Go to your Service Management Portal and select **Automation**.
2. In Automation select **Assets**, and then at the bottom select **Import Module**.

3. In the Import Integration Module dialog, select **Browse for file...** and select the **Kelverion.DataManipulation.zip**.
4. Select the Complete **checkmark** to upload the module.

To install the Integration Module on an SMA Automation Worker:

1. In the Service Management Portal:
 - a. Create an Runbook with the following PowerShell: **\$Env:PSModulePath**.
 - b. Test this runbook on an Automation Worker and record the output for use in Step 5.
2. Copy the **Kelverion.DataManipulation.zip** to your Automation Worker computer.
3. Right click on the file and select properties. Unblock the file if it is blocked.
4. Unzip the **Kelverion. DataManipulation.zip** file.
5. Copy the **Kelverion.DataManipulation** folder to a location in the path returned in Step 1b.
Note: The Automation Worker uses its own unique value for %PSModulePath%, and may differ from that of the system.

LICENSING THE INTEGRATION MODULE

Licenses for Kelverion Integration Modules are managed and deployed using Kelverion Runbook Studio.

Register an Integration Module license with Runbook Studio:

1. Open Kelverion Runbook Studio
2. In the **File** tab, click **About**.
3. Click **License Information**.
4. Click the **Integration Modules** tab.
5. Click **Add License**.
6. Select the integration module license file (*.kaml) and click **Open**.
7. You should see your entitlements displayed in the list.
8. Click **OK**.

Note: Entitlements will not display until after the Integration Module has been installed on the Runbook Studio computer.

License keys are distributed to Azure using connection assets. To create a Connection Asset with a license key and upload it to Azure, do the following:

1. In the **Home** tab, click **Sign In**.
2. Select the Azure Automation Account that will contain the connection.
3. Click **New Asset** and click **Connection**.
4. In the **Name** field, enter a name to identify the connection.
5. In the **Connection Type** field, select the desired connection type.
6. Enter the appropriate connection information in the provided fields.
7. Click **OK**.


To update all Connection Assets to use the latest license key and upload it to Azure, do the following:

1. In the **Home** tab, click **Sign In**.
2. Click **Azure (Online)**.
3. Right-click the Azure Automation Account that contains the connection assets you would like to update.
4. Click **Update License Keys**.
5. A summary dialog is displayed, listing all updated connection assets.

DATA MANIPULATION SMART CONNECTIONS

In Kelterion Runbook Studio, you can configure one or more Smart Connections in order to establish reusable links between Data Manipulation specification files. You can create as many Smart Connections as you require, specifying links to multiple files.

Adding a Smart Connection to Kelterion Runbook Studio:

1. Click **Smart Connections** , on the **Quick Access Toolbar**, or press CTRL+SHIFT+C.
2. In the **Smart Connections** dialog, click **New**.
3. In the **Name** box, enter a name for the configuration. This could be the name of the instance or a descriptive name to distinguish the type of configuration.
4. In the optional **Description** box, enter a description of the Smart Connection.
5. From the **Connection type** menu, select *Kelterion.DataManipulation*
6. In the **Specification File** box, type the file path to the specification file.
7. Add additional connections if applicable.
8. Click **OK** to close the configuration dialog box, and then click **Ok**.

AZURE GLOBAL CONNECTION ASSETS

The activities in the Kelterion Integration Module for Data Manipulation require connection information in order to operate in Azure.

The recommended way to pass connection information to your activities in your runbooks is to use Global Connection Assets. Global connection assets let you securely define connection information in Azure which can then be retrieved on demand using either the *Get-AutomationConnection* cmdlet or Connection Asset Data Source.

Adding a global connection asset to your Azure Automation Account:

1. In Kelterion Runbook Studio, click the **Azure** panel
2. Select your Azure subscription and Automation Account
3. Select **New Asset** in the main toolbar and select **Connection**
4. In the **Name** box, enter a name for the connection asset
5. In the optional **Description** box, enter a brief description describing the connection.
6. For the **Connection Type** select *Kelterion.DataManipulation*.

7. Click **OK** to close the New Connection dialog box.

DATA MANIPULATION ACTIVITIES

This integration module adds the **Kelverion.DataManipulation** module to the **Cmdlets** item in the On-Premises panel in the Runbook Studio. This module contains the following Cmdlets:

- Apply-Xslt
- Compose-Text
- Convert-XmlToJson
- Convert-JsonToXml
- Parse-Text

COMMON CONFIGURATION INSTRUCTIONS FOR ALL ACTIVITIES

The following configuration instructions apply to all activities in this integration module. Links to this section are included in the configuration instructions for each activity.

ACTIVITY PROPERTIES

ACTIVITY

1. Specify a **Label**, a short name of the activity.
2. Specify a **Description**, a short description of the activity.
3. Specify **Checkpoint**, configures the activity to checkpoint the runbook after the activity execution.

DISCOVERY

Specify a **Connection**. Selecting a Smart Connection will start the discovery process and present different options based on the Smart Connection selected.

PARAMETERS

1. Configure any mandatory parameters required by the activity.
2. Configure any optional parameters available.

FILTERS

Not all activities provide filters. If filters are available a **Filters** panel will be listed under **Activity Properties**.

Specify **Filters**. Filters are used to determine the values the activity will be performed on. This can be **Get** type activities that return the records matching the filters. Or could be **Update** or **Delete** type activities that perform the update or delete on the matching records. Filter options vary depending on the target system.

Possible Filters options:

- Equals
- Does not equal
- Is less than
- Is less than or equal to
- Is greater than
- Is greater than or equal to
- Contains
- Does not contain
- Matches
- Does not match
- Starts with
- Ends with

RETRY BEHAVIOUR

1. Specify **Enable retry**, configures an activity to retry execution until some condition is met.
2. Specify **Delay before each retry attempt**, the delay between retry attempts.
3. Specify **Retry until this condition is true**, a PowerShell condition expression that evaluates to TRUE or FALSE. See Azure documentation on details.

APPLY-XSLT

The **Apply-Xslt** activity is used in a runbook to apply an XSL transformation on input XML.

REQUIRED PARAMETERS

Element	Description	Valid Values
Connection	Azure connection asset name	Azure connection asset name
XML Input	The XML text to apply the XSLT transformation to	String
XSLT	The XSLT to apply to the XML Input	String

OUTPUTS

The XML output of the transformation.

POWERSHELL COMMAND REFERENCE

`Apply-Xslt -Connection <Hashtable> -Xslt <string> -XmlInput <string>`

For PowerShell usage type “*get-help Apply-Xslt -full*” in PowerShell

COMPOSE-TEXT

The **Compose-Text** activity is used in a runbook to compose text (xml, html or text) from the specified inputs.

DISCOVERY PARAMETERS

Element	Description	Valid Values
Compose Configuration	The name of a compose configuration in the specification file.	The name of a compose configuration in the specification file.

REQUIRED PARAMETERS

Available required parameters are determined by your specification file and the selected Compose Configuration.

Element	Description	Valid Values
Connection	Azure connection asset name	Azure connection asset name

OPTIONAL PARAMETERS

Available optional parameters are determined by your specification file and the selected Compose Configuration.

OUTPUTS

The composed text.

POWERSHELL COMMAND REFERENCE

```
Compose-Text -Connection <Hashtable> -ConfigurationXml <string> -Values  
<Hashtable> [-ConfigurationName <string>]]
```

For PowerShell usage type “*get-help Compose-Text -full*” in PowerShell

CONFIGURING INPUTS FOR COMPOSE-TEXT

When composing text, custom inputs can be inserted into the text using the **<ka:ValueOf>** element.

For example, consider a Compose-Text used to create a text block representing an incident form. The constant text of the form is built using the <ka:Text> element and the appropriate values are inserted into using the <ka:ValueOf> element.

```
<?xml version="1.0" encoding="utf-8"?>  
<ka:DataManipulation xmlns:ka="http://www.kelverion.com">  
  <ka:ComposeText name="Compose Ticket">  
    <ka:Input id="Impact"/>  
    <ka:Input id="Priority"/>  
    <ka:Input id="Urgency"/>  
    <ka:Input id="Description"/>
```

```

<ka:Input id="Opened" type="date"/>
<ka:Input id="Submitter"/>
<ka:TextTemplate format="text">
  <ka:Text>impact=</ka:Text>
    <ka:ValueOf select="$Impact"/>
    <ka:Text>&priority=</ka:Text>
    <ka:ValueOf select="$Priority"/>
    <ka:Text>&urgency=</ka:Text>
    <ka:ValueOf select="$Urgency"/>
    <ka:Text>&description=</ka:Text>
    <ka:ValueOf select="$Description"/>
    <ka:Text>&opened=</ka:Text>
    <ka:ValueOf select="$Opened"/>
    <ka:Text>&submitter=</ka:Text>
    <ka:ValueOf select="$Submitter"/>
  </ka:TextTemplate>
</ka:ComposeText>
</ka:DataManipulation>

```

When the Compose Text object runs it will generate the text using the input values and generate text similar to this:

```

impact=High&priority=High&urgency=Critical&description=Air conditioning off in
server room&opened=Wed, 26 Sep 2018 12:10:11 GMT&submitter=William Sanders

```

CONVERT-JSONTOXML

The **Convert-JsonToXml** activity is used in a runbook to convert JSON formatted text into XML. For more details about JSON specification please refer to <http://www.json.org/>.

REQUIRED PARAMETERS

Element	Description	Valid Values
Connection	Azure connection asset name	Azure connection asset name
JSON Text	The JSON text to convert to XML	String

OPTIONAL PARAMETERS

Element	Description	Valid Values
Array Element Name	Specify to insert an array-level element name in the XML	String
Root Element Name	Specify to insert a top-level element name in the XML	String

OUTPUTS

The converted XML output.

POWERSHELL COMMAND REFERENCE

```
Convert-JsonToXml -Connection <Hashtable> -JsonText <string> [-ArrayElementName  
<string>] [-RootElementName <string>]
```

For PowerShell usage type “get-help Convert-JsonToXml –full ” in PowerShell

JSON TO XML USING CONVERT-JSONTOXML

CONVERTING TO VALID XML

Since valid XML must have a single root node, the specified JSON must also have a single property in the root JSON object. If the JSON text has multiple root-level properties, the activity will fail. You can either edit the JSON text so that it results in proper XML, or you can use the **Root Element Name** optional property to introduce a top-level XML node.

```
{"person":{"id":"1","name":"John"},"car":{"make":"Ford"}}
```

➔ Invalid XML

```
{"info":{"person":{"id":"1","name":"John"},"car":{"make":"Ford"}}}
```

➔ Valid XML

XML ATTRIBUTES

In order to convert JSON properties into XML attributes, the properties must be prefixed with ‘@’ and they should be at start of the JSON object:

```
"person":{"@id":"1","name":"John"}
```

➔ `<person id="1"><name>John</name></person>`

XML DECLARATION

The XML declaration and processing attributes must be prefixed with ‘?’:

```
{  
  "?xml":{"@version":"1.0","@encoding":"utf-8"},  
  "info":{"person":{"@id":"1","name":"John"}}  
}  
➔  
<?xml version="1.0" encoding="utf-8"?>  
  
<info><person id="1"><name>John</name></person></info>
```

CONVERT-XMLTOJSON

The **Convert-XmlToJson** activity is used in a runbook to convert XML text into JSON text. For more details about JSON specification please refer to <http://www.json.org/>.

REQUIRED PARAMETERS

Element	Description	Valid Values
Connection	Azure connection asset name	Azure connection asset name
XML Text	The XML text to convert to JSON	String

OPTIONAL PARAMETERS

Element	Description	Valid Values
Skip Root	Specify to ignore the root XML node	Boolean

OUTPUTS

The converted JSON output.

POWERSHELL COMMAND REFERENCE

```
Convert-XmlToJson -Connection <Hashtable> -Xslt -XmlText <string> [-SkipRoot  
<SwitchParameter>]
```

For PowerShell usage type “*get-help Convert-XmlToJson -full*” in PowerShell.

PARSE-TEXT

The **Parse-Text** activity is used in a runbook to parse a text (xml, html or text).

DISCOVERY PARAMETERS

Element	Description	Valid Values
Parse Configuration	The name of a parse configuration in the specification file.	The name of a parse configuration in the specification file.

REQUIRED PARAMETERS

Element	Description	Valid Values
Connection	Azure connection asset name	Azure connection asset name
Text	The input text to parse.	String

FILTERS

Filters are determined by your specification file and the selected **Parse Configuration**.

OUTPUTS

Outputs are determined by your specification file and the selected **Parse Configuration**.

Element	Description	Valid Values
Correlated Data	Correlated data set	PSObject

POWERSHELL COMMAND REFERENCE

Parse-Text -Connection <Hashtable> -Text <string>

For PowerShell usage type “*get-help Parse-Text -full*” in PowerShell

PARSING AND CORRELATED DATA FOR PARSE-TEXT

The output data in your Parse-Text command will frequently have mutual relationships or connections with each other. When this occurs, you will want to set the value of the **correlated** attribute to “yes” to ensure that the data is published within a dataset with other correlated outputs.

For example, consider a Parse-Text configured to parse employee information. It may contain the following custom outputs, which have the values of the correlated attribute, set to “yes” to indicate that they should be published in data sets.

```
<ka:Output name="Employee Count" correlated="no">count(//Employee)</ka:Output>
<ka:Output name="ID" correlated="yes">//Employee/ID</ka:Output>
<ka:Output name="First Name" correlated="yes">//Employee/FirstName</ka:Output>
<ka:Output name="Last Name" correlated="yes">//Employee/LastName</ka:Output>
<ka:Output name="Phone" correlated="yes">//Employee/Phone</ka:Output>
<ka:Output name="Email" correlated="yes">//Employee/Email</ka:Output>
<ka:Output name="Department" correlated="yes">//Employee/Department</ka:Output>
```

Provided this sample data:

```
<Employee>
  <EmployeeID>012</EmployeeID>
  <FirstName>Kevin</FirstName>
  <LastName>Walters</LastName>
  <Phone>6187852341</Phone>
  <Email>kevin.walters@acme.com</Email>
  <Department>Customer Support</Department>
</Employee>
<Employee>
  <EmployeeID>013</EmployeeID>
  <FirstName>Nancy</FirstName>
  <LastName>Haddix</LastName>
  <Phone>6187854555</Phone>
  <Email>nancy.haddix@acme.com</Email>
  <Department>Human Resources</Department>
</Employee>
```

This publishes the following correlated as output:

ID: 0012

First Name: Kevin

Last Name: Walters

Phone: 6187852341

Email: keven.walters@acme.com

Department: Customer Support

ID: 0013

First Name: Nancy

Last Name: Haddix

Phone: 6187854555

Email: nancy.haddix@acme.com

Department: Human Resources

CUSTOM DATE TIME FORMATS

STANDARD DATE AND TIME FORMAT SPECIFICATIONS

Format	Description
d	Short date pattern
D	Long date pattern
f	Long date and short time
F	Long date and long time
g	Short date and short time
G	Short date and long time.
M, m	Month day pattern
O, o	Round-trip date/time pattern.
R, r	RFC1123 date/time pattern. Always ddd, dd MMM yyyy HH:mm:ss GMT
s	ISO 8601 standard. Always yyyy-MM-ddTHH:mm:ss
t	Short time pattern
T	Long time format
u	Universal time display. Always yyyy-MM-dd HH:mm:ssZ
U	Long date and long time using universal time

CUSTOM DATE AND TIME FORMAT STRINGS

Format	Description
d, %d	The day of the month. Single-digit days do not have a leading zero. Use %d if the format pattern is not combined with any other format specifiers.
dd	The day of the month. Single-digit days have a leading zero.
ddd	The abbreviated name of the day of the week.
dddd	The full name of the day of the week.
gg	The period or era.
h, %h	The hour in a 12 hour clock. Single digit hours do not have a leading zero. Use %h if the format pattern is not combined with other format patterns.
hh	The hour in a 12-hour clock. Single-digit hour have a leading zero.
H, %H	The hour in a 24-hour clock. Single-digit hours do not have a leading zero. Use %H if the format pattern is not combined with other format patterns.
HH	The hour in a 24-hour clock. Single-digit hours have a leading zero.
K	Local, UTC or unspecified
m, %m	The minute. Single-digit minutes do not have a leading zero. Use %m if the format pattern is not combined with other format patterns.
mm	The minute. Single-digit minutes have a leading zero.
M, %M	The numeric month. Single-digit months do not have a leading zero. Use %M if the format pattern is not combined with other format patterns.
MMMM	The full name of the month.
s, %s	The second. Single-digit seconds do not have a leading zero. Use %s if the format pattern is not combined with other format patterns.
ss	The second. Single-digit seconds have a leading zero.
t, %t	The first character in the AM/PM designator. Use %t if the format pattern is not combined with other format patterns.

tt	The AM/PM designator.
y, %y	The year without a century. If the year without a century is less than 10, the year is displayed without a leading zero. Use %y if the format pattern is not combined with other format patterns.
yy	The year without a century. If the year without the century is less than ten, the year is displayed with a leading zero.
yyy	The year in three digits. If the year is less than 100, the year is displayed with a leading zero.
yyyy	The year in four or five digits (depending on the calendar being used), including the century.
yyyyy	The year in five digits. Pads with leading zeros to get five digits.
yyyyyy	The year in six digits. Pads with leading zeroes to get six digits.
z, %z	The time zone offset. Single digit hours do not have a leading zero. Use %z if the format pattern is not combined with other format patterns.
zz	The full time zone offset. Single-digit hours have a leading zero.
:	The default time separator
/	The default date separator
\c	Where c is any character. Displays the character literally. To display the backslash character use '\\.

ELEMENT REFERENCE

Element	Description
Attribute	Adds an XML attribute to the message body.
AttributeSet	Defines a named set of XML attributes
ComposeText	Used to define the root element of a Compose Text object.
Choose	Used in conjunction with <When> and <Otherwise> to express multiple conditional tests.
Element	Creates an XML element node in the message body
DataManipulation	Defines the root element of the Text Specification
If	Contains a template that will be applied only if a selected condition is true.
Input	Defines an input that can be used to retrieve data from the Integration Server data bus.
Message	Used to report errors.
TextTemplate	Defines the format and structure of the text body.
Otherwise	Specifies a default action for the <Choose> element.
Output	Defines an output that can be used to parse data from text and publish as output.
Text	Writes literal text in the text body
When	Specifies an action for the <choose> element.
Value	Defines a value for an input or filter.
ValueOf	Writes the value of an input variable to the text body
ValueSet	Defines a named set of values.
Variable	Declares a global variable

XML SPECIFICATION FOR PARSE-TEXT AND COMPOSE-TEXT

<ka:DataManipulation>

Used by: Compose Text and Parse Text

DEFINITION AND USAGE

The <ka:DataManipulation> element is used to define the root element of XML specification.

SYNTAX

```
<ka:DataManipulation xmlns:ka=http://www.kelverion.com version="version">
  <!-- Content-->
</ka:DataManipulation>
```

ATTRIBUTES

Attribute	Value	Description
version	1.0	Optional. The product version. Default is "1.0"

<ka:ComposeText>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:ComposeText> element is used to define the root element of a Compose Text object.

SYNTAX

```
<ka:ComposeText name="name">
  <!-- Content-->
</ka:ComposeText>
```

ATTRIBUTES

Attribute	Value	Description
name	name	Required. Specifies the name of this ComposeText.

<ka:ParseText>

Used by: Parse Text

DEFINITION AND USAGE

The <ka:ParseText> element is used to define the root element of a Parse Text object.

SYNTAX

```
<ka:ParseText name="name">
  <!-- Content-->
</ka:ParseText>
```

ATTRIBUTES

Attribute	Value	Description
name	Name	Required. Specifies the name of this ParseText.

<ka:Attribute>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:Attribute> element is used to add attributes to XML elements.

SYNTAX

```
<ka:Attribute name="attributename" namespace="uri">
  <!-- Content: template -->
</ka:Attribute>
```

ATTRIBUTES

Attribute	Value	Description
name	name	Required. Specifies the name of the attribute
namespace	URI	Optional. Defines the namespace URI for the attribute.

EXAMPLE

```
<ka:Element name="Incident">
  <ka:Attribute name="severity">
    <ka:ValueOf select="$Severity"/>
  </ka:Attribute>
</ka:Element>
```

<ka:AttributeSet>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:AttributeSet> element creates a named set of attributes. The AttributeSet can be applied as a whole to the text body document.

SYNTAX

```
<ka:AttributeSet name="name" useAttributeSets="name list">
  <!-- Content: ka:Attribute* -->
</ka:AttributeSet>
```

ATTRIBUTES

Attribute	Value	Description
name	name	Required. Specifies the name of the attribute set
useAttributeSets	name list	Optional. Defines the namespace URI for the attribute.

EXAMPLE

```
<ka:AttributeSet name="Critical">
  <ka:Attribute name="impact">Critical</ka:Attribute>
  <ka:Attribute name="urgency">High</ka:Attribute>
  <ka:Attribute name="priority">High</ka:Attribute>
</ka:AttributeSet>
```

<ka:Choose>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:Choose> element is used in conjunction with <ka:When> and <ka:Otherwise> to express multiple conditional tests. If no <ka:When> is true, the content of <ka:Otherwise> is processed. If no <ka:When> is true and no <ka:Otherwise> element is present then nothing is created. Tip: For simple conditional testing use the <ka:If> element.

SYNTAX

```
<ka:Choose>
  <!-- Content: (ka:When+, ka:Otherwise?) -->
</ka:Choose>
```

EXAMPLE

```
<Impact>
  <ka:Choose>
    <ka:When test="$Impact='Critical'">
      <ka:Text>1</ka:Text>
    </ka:When>
    <ka:When test="$Impact='High'">
      <ka:Text>2</ka:Text>
    </ka:When>
    <ka:When test="$Impact='Medium'">
      <ka:Text>3</ka:Text>
    </ka:When>
    <ka:When test="$Impact='Low'">
      <ka:Text>4</ka:Text>
    </ka:When>
  </ka:Choose>
</Impact>
```

<ka:Element>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:Element> element is used to create an element in the text body.

SYNTAX

```
<ka:Element name="name" namespace="URI" useAttributeList="namelist">
  <!-- Content: template -->
</ka:Element>
```

ATTRIBUTES

Attribute	Value	Description
name	name	Required. Specifies the name of the element to be created. The value of the name attribute can be set to an expression that is computed at run-time, like this:<ka:Element name="{ \$ElementName}" />
namespace	URI	Optional. Specifies the namespace URI of the element. The value of the namespace attribute can be set to an expression that is computed at run-time, like this:<ka:Element name="{ \$ElementName}" namespace="{ \$Namespace}" />
useAttributeSets	namelist	Optional. A white space separated list of Attribute Sets containing attributes to be added to the element.

EXAMPLE

```
<ka:TextTemplate format="xml">
  <ka:Element name="Incident">
    <ka:Element name="Impact">
      <ka:ValueOf select="$Impact"/>
    </ka:Element>
    <ka:Element name="Urgency">
      <ka:ValueOf select="$Urgency"/>
    </ka:Element>
    <ka:Element name="Priority">
      <ka:ValueOf select="$Priority"/>
    </ka:Element>
    <ka:Element name="Description">
      <ka:ValueOf select="$Description"/>
    </ka:Element>
  </ka:Element>
</ka:TextTemplate>
```

<ka:Input>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:Input> element defines an input that can be used to collect information from the user so that it can be formatted into the output text.

SYNTAX

```
<ka:Input id="id" name="name" mandatory="yes|no" default="string"
format="string" useValueSets="name list"
type="boolean|date|computer|file|folder|list|string"/>
```

ATTRIBUTES

Attribute	Value	Description
id	id	Required. Specifies the unique identifier.
name	name	Optional. Specifies the name used to display the input in the Runbook Designer. Default is the value of the "id" attribute.

mandatory	yes no	Optional. "yes" specifies that the input is required. "no" specifies that the input is optional. Default is "yes".
default	string	Optional. Specifies a default value.
type	boolean date computer file folder list string	Optional. Specifies the type of data that the data represents. Some data types will include a browser when accessed from the Runbook Designer. Default is "string".
format	date format	Optional. Specifies how date/time values will be formatted. The default is the system's current regional settings.
useValueSets	name list	Optional. For list inputs, this space-delimited list of value set names is used to provide values for the list browser.

<ka:If>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:If> element contains a template that will be applied only if a specified condition is true.

SYNTAX

```
<ka:if test="expression">
  <!--Content: template -->
</ka:if>
```

ATTRIBUTES

Attribute	Value	Description
test	expression	Required. Specifies the condition to be tested.

EXAMPLE

```
<ka:DataManipulation xmlns:ka="http://www.kelverion.com" version="1.0">
  <ka:ComposeText name="Build Closed Message">
    <ka:Input id="closedAt" name="Closed Date"/>
    <ka:Input id="closedBy" name="Closed By"/>
    <ka:TextTemplate format="xml">
      <CloseIncident>
        <ka:If test="$closedAt=''">
          <ka:Message terminate="yes">The 'Closed At' field must not be
            empty</ka:Message>
        </ka:If>
        <ClosedAt>
          <ka:ValueOf select="$closedAt"/>
        </ClosedAt>
        <ka:If test="$closedBy=''">
          <ka:Message terminate="yes">The 'Closed By' field must not be
            empty </ka:Message>
        </ka:If>
      </CloseIncident>
    </ka:TextTemplate>
  </ka:ComposeText>
</ka:DataManipulation>
```

```

        </ka:If>
      </CloseIncident>
    </ka:TextTemplate>
  </ka:ComposeText>
</ka:DataManipulation>

```

<ka:Message>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:Message> element writes a message to the text body. This element is used to report errors.

SYNTAX

```

<ka:Message terminate="yes|no">
  <!-- Content:template -->
</ka:Message>

```

ATTRIBUTES

Attribute	Value	Description
terminate	yes no	Optional. "yes" terminates the object and reports the message as an error. "no" reports the message as and continues sending the message. Default is "no".

EXAMPLE

```

<ka:DataManipulation xmlns:ka="http://www.kelverion.com" version="1.0">
  <ka:ComposeText name="Create Closed Message">
    <ka:Input id="closedAt" name="Closed Date"/>
    <ka:Input id="closedBy" name="Closed By"/>
    <ka:TextTemplate format="xml">
      <CloseIncident>
        <ka:If test="'$closedAt=''">
          <ka:Message terminate="yes">The 'Closed By' field must not be
            empty</ka:Message>
        </ka:If>
        <ClosedAt>
          <ka:ValueOf select="'$closedAt'"/>
        </ClosedAt>
        <ka:If test="'$closedBy=''">
          <ka:Message terminate="yes">The 'Closed By' field must not be
            empty</ka:Message>
        </ka:If>
        <ClosedBy>
          <ka:ValueOf select="'$closedBy'"/>
        </ClosedBy>
      </CloseIncident>
    </ka:TextTemplate>
  </ka:ComposeText>
</ka:DataManipulation>

```

<ka:TextTemplate>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:TextTemplate> element defines the template that will be used to construct the output text.

SYNTAX

```
<ka:TextTemplate format="xml|text|html" cdataElements="name list"
indent="yes|no" omitXmlDeclaration="yes|no" version="1.0">
  <!-- Content:(<ka:Element>-*- , <ka:Text>-*- , template) -->
</ka:TextTemplate>
```

ATTRIBUTES

Attribute	Value	Description
format	xml text html	Optional. Defines the output format. The default is XML (but if the first child of the root node is <html> and there are no preceding text nodes, then the default is HTML).
cdataElements	name list	Optional. A white-space separated list of elements whose text contents should be written as CDATA sections.
omitXmlDeclaration	yes no	Optional. "yes" specifies that the XML declaration (<?xml ...?>) should be omitted in the text body. "no" specifies that the XML declaration should be included in the text body. The default is "no".

<ka:Otherwise>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:Otherwise> element specifies a default action for the <ka:Choose> element. The action will take place when none of the <ka:When> conditions apply.

SYNTAX

```
<ka:Otherwise>
  <!-- Content: template -->
</ka:Otherwise>
```

EXAMPLE

```
<Impact>
  <ka:Choose>
    <ka:When test="$Impact='Critical'">
      <ka:Text>1</ka:Text>
    </ka:When>
    <ka:When test="$Impact='High'">
      <ka:Text>2</ka:Text>
    </ka:When>
    <ka:When test="$Impact='Medium'">
```

```

    <ka:Text>3</ka:Text>
  </ka:When>
  <ka:When test="$Impact='Low'">
    <ka:Text>4</ka:Text>
  </ka:When>
  <ka:Otherwise>
    <ka:Message terminate="yes">Invalid impact value provided.
  </ka:Message>
  </ka:Otherwise>
</ka:Choose>
</Impact>

```

<ka:Output>

Used by: Parse Text

DEFINITION AND USAGE

The <ka:Output> element defines how data is parsed from input text and published as output.

SYNTAX

```

<ka:Output name="string" description="string" parser="xpath|regex|split"
type="string|number|date" correlated="yes|no" filtered="yes|no"
useValueSets="name list">
  <!--Content: Expression -->
</ka:Output>

```

ATTRIBUTES

Attribute	Value	Description
name	name	Required. Specifies the name uniquely identify the output.
correlated	yes no	Optional. "yes" specifies that the output should be published as a part of a correlated data set. "no" specifies the output is should be published as an individual value. Default is "no"
description	string	Optional. Specifies a description for the output.
filtered	yes no	Optional. "yes" specifies that the output can be used to filter incoming requests for Monitor Request objects. "no" specifies that the output cannot be used for filtering. The default is "yes"
format	date format	Optional. Used for parsing date values. Dates should be in the specified format to be parsed into output. The default format is "R".
parser	xpath regex split	Optional. "xpath" specifies that the XPath parser will be used to interpret the select expression. "regex" specifies that the regular expressions will be used to interpret the parse expression. "split" specifies a parser used for delimited values. The default is "xpath".
type	string number date	Optional. Specifies that published data type that will be displayed in the Runbook Studio. The default is "string".

EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<ka:DataManipulation xmlns:ka="http://www.kelverion.com" version="1.0">
  <ka:ParseText name="Parse User Info">
    <ka:ValueSet name="provinces">
      <ka:Value>Ont</ka:Value>
      <ka:Value>BC</ka:Value>
      <ka:Value>QB</ka:Value>
    </ka:ValueSet>
    <ka:Output name='Name' description='User name' type='string'
parser='xpath' correlated='yes' filtered='yes'>//User/Name</ka:Output>
    <ka:Output name='Phone' description='User phone' type='string'
parser='xpath' correlated='yes' filtered='yes'>//User/Phone</ka:Output>
    <ka:Output name='Provnicie' type='string' parser='xpath' correlated='yes'
filtered='yes' useValueSets='provinces'>//User/Provnicie</ka:Output>
  </ka:ParseText>
</ka:DataManipulation>
```

<ka:Text>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:Text> element is sued to write literal text to the text body.

SYNTAX

```
<ka:Text>
  <!-- Content:#PCDATA -->
</ka:Text>
```

EXAMPLE

```
<ka:DataManipulation xmlns:ka="http://www.kelverion.com" version="1.0">
  <ka:ComposeText name="Compose Ticket">
    <ka:Input id="Impact"/>
    <ka:Input id="Priority"/>
    <ka:Input id="Urgency"/>
    <ka:Input id="Description"/>
    <ka:Input id="Opened" type="date"/>
    <ka:Input id="Submitter"/>
    <ka:TextTemplate format="xml">
      <ka:Text>impact=</ka:Text>
      <ka:ValueOf select="$Impact"/>
      <ka:Text>&priority=</ka:Text>
      <ka:ValueOf select="$Priority"/>
      <ka:Text>&urgency=</ka:Text>
      <ka:ValueOf select="$Urgency"/>
      <ka:Text>&description=</ka:Text>
      <ka:ValueOf select="$Description"/>
      <ka:Text>&opened=</ka:Text>
      <ka:ValueOf select="$Opened"/>
      <ka:Text>&submitter=</ka:Text>
      <ka:ValueOf select="$Submitter"/>
    </ka:TextTemplate>
  </ka:ComposeText>
</ka:DataManipulation>
```

```

    </ka:TextTemplate>
  </ka:ComposeText>
</ka:DataManipulation>

```

<ka:When>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:When> element is used to specify an action for the ka:Choose element. The <ka:When> element evaluates an expression and if it returns true then an action is performed.

Note: The <ka:When> element is used in conjunction with <ka:Choose> and <ka:Otherwise> to express multiple conditional tests.

SYNTAX

```

<ka:When test="boolean expression">
  <!-- Content: template -->
</ka:When>

```

ATTRIBUTES

Attribute	Value	Description
test	boolean expression	Required. Specifies a boolean expression to be tested.

EXAMPLE

```

<Impact>
  <ka:Choose>
    <ka:When test="$Impact='Critical'">
      <ka:Text>1</ka:Text>
    </ka:When>
    <ka:When test="$Impact='High'">
      <ka:Text>2</ka:Text>
    </ka:When>
    <ka:When test="$Impact='Medium'">
      <ka:Text>3</ka:Text>
    </ka:When>
    <ka:When test="$Impact='Low'">
      <ka:Text>4</ka:Text>
    </ka:When>
  </ka:Choose>
</Impact>

```

<ka:Value>

Used by: Compose Text and Parse Text

DEFINITION AND USAGE

Defines a list browser value for an input for filter.

SYNTAX

```
<ka:Value>
  <!--Content:string-->
</ka:Value>
```

EXAMPLE

```
<ka:DataManipulation xmlns:ka="http://www.kelverion.com" version="1.0">
  <ka:ParseText name="Parse Ticket">
    <ka:ValueSet name="Impact">
      <ka:Value>High</ka:Value>
      <ka:Value>Medium</ka:Value>
      <ka:Value>Low</ka:Value>
    </ka:ValueSet>
    <ka:ValueSet name="Urgency">
      <ka:Value>High</ka:Value>
      <ka:Value>Medium</ka:Value>
      <ka:Value>Low</ka:Value>
    </ka:ValueSet>
    <ka:ValueSet name="Priority">
      <ka:Value>Critical</ka:Value>
      <ka:Value>High</ka:Value>
      <ka:Value>Moderate</ka:Value>
      <ka:Value>Low</ka:Value>
    </ka:ValueSet>
    <ka:Output name='Priority' type='string' parser='xpath' correlated='yes'
filtered='yes' useValueSets='Priority'>//Ticket/Priority</ka:Output>
  </ka:ParseText>
</ka:DataManipulation>
```

<ka:ValueOf>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:ValueOf> element inserts the value of a specified input.

SYNTAX

```
<ka:ValueOf select="expression"/>
```

ATTRIBUTES

Attribute	Value	Description
select	expression	Required. An XPath expression that input to insert.

EXAMPLE

```
<ka:DataManipulation xmlns:ka="http://www.kelverion.com" version="1.0">
  <ka:ComposeText name="Compose Ticket">
    <ka:Input id="Impact"/>
  </ka:ComposeText>
</ka:DataManipulation>
```

```

<ka:Input id="Urgency"/>
<ka:Input id="Priority"/>
<ka:Input id="Description"/>
<ka:Input id="Submitter"/>
<ka:TextTemplate format="xml">
  <Incident>
    <Impact>
      <ka:ValueOf select="$Impact"/>
    </Impact>
    <Urgency>
      <ka:ValueOf select="$Urgency"/>
    </Urgency>
    <Priority>
      <ka:ValueOf select="$Priority"/>
    </Priority>
    <Description>
      <ka:ValueOf select="$Description"/>
    </Description>
    <Submitter>
      <ka:ValueOf select="$Submitter"/>
    </Submitter>
  </Incident>
</ka:TextTemplate>
</ka:ComposeText>
</ka:DataManipulation>

```

<ka:Variable>

Used by: Compose Text

DEFINITION AND USAGE

The <ka:Variable> declares a global variable that can be used to insert a constant value into the text body.

SYNTAX

```

<ka:Variable name="name" select="expression">
  <!-- Content: template -->
</ka:Variable>

```

ATTRIBUTES

Attribute	Value	Description
name	Name	Required. Specifies the name of the variable.
select	expression	Optional. Defines the value of the variable.