INTEGRATION PACK FOR NETWORK MESSAGING

For Microsoft System Center Orchestrator

For System Center 2016 and 2019, you must use the 32-bit version of the integration pack, which has the name **Kelverion_Integration_Pack_for_Network_Messaging_2.0**

For System Center 2022 and later, you must use the 64-bit version of the integration pack, which has the name **Kelverion_IP_Network_Messaging_x64_2.0**

User Guide

Version 2.0

Kelverion Integration Pack for Network Messaging

Published: February 2023

Feedback

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

Contents

Introduction

The Kelverion Integration Pack for Network Messaging is a compliant integration for Microsoft System Center Orchestrator. This Integration Pack enables users to exchange data with other Systems and custom applications using HTTP, HTTPS, and TCP mechanisms. This module delivers a range of re-usable activities to automate IT-Functions such as:

Including custom applications into Service Desk processing enabling operator free exchange of data and improved data quality. The ability to send different data formats to other systems capable of HTTP data streams and replacing manual transfer of data between systems.

The ability to include data received via HTTP streams and initiate Orchestrator runbooks to transform and populate discovered data into systems such as CMDBs and Service Desks.

Installation and Configuration

The following section outlines how to install and configure the Kelverion Integration Pack for Network Messaging.

System Requirements

The Integration Pack for Network Messaging requires the following software to be installed and configured prior to implementing the integration. For more information about installing and configuring System Center Orchestrator refer to the respective product documentation.

Kelverion_Integration_Pack_for_Network_Messaging (32-bit)

- Microsoft System Center Orchestrator *2016, 2019*
- Microsoft .NET Framework 4.7.2

Kelverion_IP_OAuth_Network_Messaging_x64 (64-bit)

- Microsoft System Center Orchestrator 2022
- Microsoft .NET Framework 4.7.2

Registering and Deploying the Integration Pack

After you download the integration pack, you register the integration pack file with the Orchestrator management server, and then deploy it to runbook servers and computers that have the Runbook Designer installed.

IMPORTANT: Ensure that you are deploying the correct version of the Integration Pack.

- For System Center 2016 and 2019, you must use the 32-bit version of the integration pack, which has the name **Kelverion_Integration_Pack_for_Network_Messaging_2.0**
- For System Center 2022 and later, you must use the 64-bit version of the integration pack, which has the name **Kelverion_IP_Network_Messaging_x64_2.0**

To register the integration pack:

1. On the management server, copy the **.OIP** file for the integration pack to a local hard drive or network share.
2. Confirm that the file is not set to **Read Only** to prevent unregistering the integration pack later.
3. Start the **Deployment Manager**.
4. In the navigation pane of the Deployment Manager, expand **Orchestrator Management Server**, right-click **Integration Packs** to select **Register IP with the Management Server**. The **Integration Pack Registration Wizard** opens.
5. Click **Next**.
6. In the **Select Integration Packs or Hotfixes** dialog box, click **Add**.

7. Locate the **.OIP** file that you copied locally from step 1, click **Open** and then click **Next**.
8. In the **Completing the Integration Pack Wizard** dialog box, click **Finish**.
9. On the **End User Agreement** dialog box, read the Kelverion License Terms**,** and then click **Accept**.
10. The **Log Entries** pane displays a confirmation message when the integration pack is successfully registered.

To deploy the integration pack:

1. In the navigation pane of the **Deployment Manager**, right-click **Integration Packs**, click **Deploy IP to Runbook Server or Runbook Designer**.
2. Select the integration pack you want to deploy, and then click **Next**.
3. Enter the name of the runbook server or computers with the Runbook Designer installed, on which you want to deploy the integration pack, click **Add**, and then click **Next**.
4. Continue to add additional runbook servers and computers running the Runbook Designer, on which you want to deploy the integration pack. Click **Next**.
5. In the **Installation Options** dialog box, configure the following settings.
6. To choose a time to deploy the integration pack, select the **Schedule installation** check box, and then select the time and date from the **Perform installation** list.
7. Click one of the following:
    a. **Stop all running runbooks before installing the integration pack** to stop all running runbooks before deploying the integration pack.
    b. **Install the Integration Packs without stopping the running Runbooks** to install the integration pack without stopping any running runbooks.
8. Click **Next**.
9. In the **Completing Integration Pack Deployment Wizard** dialog box, Click **Finish**.
10. When the integration pack is deployed, the **Log Entries** pane displays a confirmation message.

For more information about how to install integration packs, see the [How to Install an Integration Pack](https://technet.microsoft.com/en-us/library/hh420346.aspx) (https://technet.microsoft.com/en-us/library/hh420346.aspx).

Licensing the Integration Pack

After you register and deploy the integration pack you must provide a valid Kelverion license before running any runbooks that contain activities from the integration pack.

To deploy the integration pack license file to System Center Orchestrator 2019 or earlier:

1. Copy the .KAL license file to %PROGRAMFILES(X86)%\Kelverion Automation\Licenses
2. Repeat for each Orchestrator Runbook Server and Runbook Designer host system.

To deploy the integration pack license file to System Center Orchestrator 2022 or later:

1. Copy the .KAL license file to %PROGRAMFILES%\Kelverion Automation\Licenses
2. Repeat for each Orchestrator Runbook Server and Runbook Designer host system.

Integration Pack Activities

This Integration Pack for Network Messaging adds the **KA Network Messaging** category to the **Activities** page in the Runbook Designer. This category contains the following activities:

- Monitor HTTP
- Monitor TCP
- Send HTTP
- Send TCP

Common Configuration Instructions for All Activities

The following configuration instructions apply to all activities in this integration pack.

Activity Properties

Each activity has a set of required or optional properties that define the configuration of that activity. This includes how it connects to other activity or how the activity performs its actions. You can view or modify activity properties in the Orchestrator Client.

To configure the properties for an activity:

1. Double-click the activity. Alternatively, you can right-click the activity, and then click Properties.
2. Click **Finish**.

In the activity properties dialog box, several tabs along the left side provide access to general and specific settings for the activity. Although the number of available tabs for activity properties differs from activity to activity, all activities will have a **General** tab, a **Properties** tab and/or a **Filters** tab, and a **Run Behavior** tab. Some activities may have additional tabs.

General Tab

This tab contains the **Name** and **Description** properties for the activity. By default, the **Name** of the activity is the same as its activity type, and the **Description** is blank. You can modify these properties to create more descriptive names or provide detailed descriptions of the actions of the activity.

Properties/Filters Tab

These tabs contain properties that are specific to the activity.

All activities in this integration pack have the **Configuration Name** property at the top of the **Properties** tab. This property is used to specify the connection to a Microsoft System Center.

To configure the Configuration Name property:

- Click the ellipsis **(…)** button next to the **Name** field, and then select the applicable connection name. Connections displayed in the list have been previously configured in the KA Network Messaging options located in the Runbook Designer options menu.

Filter Behavior

The Monitor and Get activities use filters to determine the values that will invoke a runbook or retrieve activities. Property values of potential candidates are compared to the values of the filters to determine if they meet the criteria. When matching against values, you select one of the available methods of comparison. An option is provided to either match or not match the filter using each method. For example, the "Does not" version of a method causes alerts that do not match the filter to trigger the runbook.

- **Equals**: the property of the object exactly matches the text or number specified in the filter.
- **Does not equal**: the property of the object does not exactly match the text or number specified in the filter.
- **Is less than**: the property of the object is less than the number specified in the filter.
- **Is less than or equal to**: the property of the object is less than or equal to the number specified in the filter.
- **Is greater than**: the property of the object is greater than the number specified in the filter.
- **Is greater than or equal to**: the property of the object is greater than or equal to the number specified in the filter.
- **Contains**: the property of the object contains the exact text specified in the filter. Unlike the Equals behavior, there can be other text surrounding the matching text.
- **Does not contain**: the property of the object does not contain the exact text specified in the filter. Unlike the Equals behavior, there can be other text surrounding the matching text.
- **Starts with**: the property of the object starts with the exact text specified in the filter. Unlike the Equals behavior, there can be other text following the matching text.
- **Ends with**: the property of the object ends with the exact text specified in the filter. Unlike the Equals behavior, there can be other text preceding the matching text.

Run Behavior Tab

This tab contains the properties that determine how the activity handles multi-value published data and what notifications will be sent if the activity fails or runs for an excessive period of time.

Multi-Value Published Data Behavior

The Get activities retrieve information from another activity or outside source, and can return one or more values in the published data. For example, when you use the Get Collection Member activity, the data output from that activity might be a list of computers that belong to the specified collection.

By default, the data from the Get activity will be passed on as multiple individual outputs. This invokes the next activity as many times as there are items in the output. Alternatively, you can provide a single output for the activity by enabling the **Flatten** option. When you enable this option, you also choose a formatting option:

- **Separate with line breaks**. Each item is on a new line. This format is useful for creating human-readable text files for the output.
- **Separate with.** One or more characters of your choice separate each item.
- **Use CSV format**. All items are in CSV (comma-separated value) format. This format is useful for importing data into spreadsheets or other applications.

The activity will produce a new set of data every time it runs. The **Flatten** feature does not flatten data across multiple instances of the same activity.

Event Notifications

Some activities are expected to take a limited amount of time to complete. If they do not complete within that time, they may be stalled or there may be another issue preventing them from completing. You can define the number of seconds to wait for completion of the action. After this period, a platform event will be sent, and the issue will be reported. You can also choose whether to generate a platform event if the activity returns a failure.

To be notified when the activity takes longer than a specified time to run or fails to run:

1. In the **Event Notifications** box, enter the **number of seconds** of run time before a notification is generated.
2. Select **Report if activity fails to run** to generate run failure notifications.

For more information about Orchestrator events, see the “Event Notifications " topics in the Runbook Properties (https://technet.microsoft.com/en-us/library/hh489610.aspx#Event Notifications).

Published Data

Returned data is the foundation of a working Runbook. It is the data produced as a result of the actions of an activity. This data is published to an internal data bus that is unique for each Runbook. Subsequent activities in the Runbook can subscribe to this data and use it in their configuration. Link conditions also use this information to add decision-making capabilities to Runbooks.

An activity can only subscribe to data from the activities that are linked before it in the Runbook. You can use published data to automatically populate the property values needed by activity.

To use published data:

1. Right-click the property value box, click **Subscribe**, and then click **Published Data**.
2. Click the Activity drop-down box and select the activity from which you want to obtain the data.
3. To view additional data elements common to all runbooks, select **Show Common Published Data**.
4. Click the published data element that you want to use and then click **OK.**

For a list of the data elements published by each activity, see the Published Data tables in the activity topic. For information about the common published data items, see the Publishhed Data (http://technet.microsoft.com/en-us/library/hh403821.aspx).

Monitor HTTP Activity

The **Monitor HTTP** activity is used to listen for and respond to HTTP requests from external systems and custom applications using HTTP.

Required Properties

You must configure the following parameters. Additional parameters will be provided based on the configuration file that is selected.

Monitor Configuration	The configuration used to define filters and published data for the activity.

Published Data

The activity publishes the following activity-specific data items. Additional data may be published depending on the configuration that is selected.

Accept Types	The MIME types that are accepted by the client
Configuration File	The path to the configuration file used configure the activity
Content Encoding	The content encoding that was sent from the client
Content Length	The length of the body in bytes
Content Type	The MIME type of the body
Entity Body	The body of the request sent from the client
Http Method	The HTTP method specified by the client
Is Authenticated	Indicates whether the client is authenticated
Is Local	Indicates whether the request was sent from the local computer
Is Secure Connection	Indicates whether the TCP connection used to send the request is using the SSL protocol
Raw URL	The raw URL (without host and port) requested by the client
Realm	The realm or resource partition
URL	The URL request by the client
URL Referrer	The URI of the resource that referred the client to the server
User Agent	The agent presented by the server
User Host Address	The server IP address and port number
User Host Name	The DNS name and port number

Configuring the Monitor HTTP Activity

The Monitor HTTP configuration lets you configure Orchestrator to listen and response to HTTP request. You can create as many configurations as you require.

To setup a Monitor HTTP configuration:

1. In the Runbook Designer, click the **Options** menu and select **KA Network Messaging**. The **KA Network Messaging** dialog box appears.
2. On the **Configurations** tab, click **Add** to begin the configuration setup. The **Add Configuration** dialog box appears.
3. In the **Name** box, enter a name for the configuration. This could be the name of the server or some descriptive name to distinguish the type of configuration.
4. Click the ellipsis button (…) next to the **Type** box and select **Monitor HTTP**.
5. In the **Configuration File** box, select the configuration file used to configure the required and optional properties, filters, published data and behaviors of your activities.
6. Enter the **URI** for the monitor to listen to.
7. If necessary, enter a **Realm** to authenticate incoming requests.
8. (Optional) In the **Security Protocols** box, provide a comma delimited list of SSL/TLS values as required. For example: Tls11, Tls12, Tls13 (This property defaults to Tls12.)
9. Click **OK** to close the configuration dialog box, and then click **Finish**.

Monitor TCP Activity

The **Monitor TCP** activity is used in a runbook to listen and respond to incoming TCP requests.
Important: The Monitor TCP activity does not provide a secure connection. It is important that confidential data be secured appropriately.

Required Properties

You must configure the following properties.

Control Character Handling	System Center Orchestrator cannot publish data that contains control characters. This option is used to remove or replace those problematic characters. Options include **Remove Characters** and **Replace Characters**. The control characters that are removed or replaced are ASCII control characters 0 to 31.
Replacement Character	The character to replace control characters with when **Control Character Handling** is set to **Replace Characters.**
Text Encoding	Method used to encode the body of the incoming text. The default is **UTF-8**.

Optional Properties

You can configure the following properties as required.

Response Text	If specified, this text will be used to respond to all incoming HTTP requests.

Published Data

The activity publishes the following activity-specific data items.

Client Address	The address of the client making the HTTP request.
Incoming Text	The body of the incoming text message.
Local Address	The local IP address that is being monitored.
Port	The local port that is being monitored.

Configuring the Monitor TCP Activity

The Monitor TCP configuration lets you configure Orchestrator to listen and respond to TCP requests. You can create as many configurations as your require.

To setup a Monitor TCP configuration:

1. In the Runbook Designer, click the **Options** menu and select **KA Network Messaging**. The **KA Network Messaging** dialog box appears.
2. On the **Configurations** tab, click **Add** to begin the configuration setup. The **Add Configuration** dialog box appears.
3. In the **Name** box, enter a name for the configuration. This could be the name of the server or some descriptive name to distinguish the type of configuration.
4. Click the ellipsis button (…) next to the **Type** box and select **Monitor TCP**.
5. In the **Local Address** box select or type the local IP address that you want to listen to.
6. In the **Port box** type the port that you want to listen to.
7. Click **OK** to close the configuration dialog box, and then click **Finish**.

Send HTTP Activity

The **Send HTTP** activity can be used in a runbook to send textual data to an HTTP server using customized input properties and formatting that you specify.

The following tables list the required properties and published data for this activity. Additional required and optional properties and published data are generated based on the configuration file that is selected when you define the activity.

Required Properties

You must configure the following parameters. Additional parameters will be provided based on the configuration file that is selected.

Send Configuration	The configuration used to define inputs the activity

Published Data

The activity publishes the following activity-specific data items. Additional data may be published depending on the configuration that is selected.

Client Certificate	The client certificate associated with the request

Configuration File	The configuration file that is associated with the activity configuration that was selected
Content Encoding	The encoding used to encode the body of the response
Content Length	The length of the content returned from the server in bytes
Content Type	The content type of the response
Domain	The domain for the username used to authenticate the request
Http Method	The HTTP method of the request
Ignore Certificate Errors	Indicates whether certificate errors should be ignored
Message Body	The message body that was sent to the server
Query	The query that was sent to the server in the URL
Response Text	The response that was returned from the server
Response Uri	The URI of the internet resource that responded to the request
Server	The name of the server that sent the response
Status Code	The status of the response
Status Description	A description of the status returned with the response
URI	The URI of the server to which the request was sent
User	The username used to authenticate the request

Configuring the Send HTTP Activity

The Send HTTP configuration establishes a link between Orchestrator and an external HTTP server. You can create as many configurations as you require.

To setup a Send HTTP configuration:

1. In the Runbook Designer, click the **Options** menu and select **KA Network Messaging**. The **KA Network Messaging** dialog box appears.
2. On the **Configurations** tab, click **Add** to begin the configuration setup. The **Add Configuration** dialog box appears.
3. In the **Name** box, enter a name for the configuration. This could be the name of the server or some descriptive name to distinguish the type of configuration.
4. Click the ellipsis button (…) next to the **Type** box and select **Send HTTP**.
5. In the **Configuration File** box, select the configuration file that you want to use to customize the required and optional properties, filters, published data and behaviors of you activities.
6. In the **URI** box, type the URL of the HTTP server that you want to connect to.
7. Optionally, in the **User**, **Domain**, and **Password** boxes, enter the credentials of the user that you want to authenticate your connection.
8. Optionally, in the **Client Certificate** box, type the path to the certificate file that will be used to authenticate your connection.
9. Optionally, in the **Ignore Certificate Errors** box, select **True** to ignore errors caused by invalid certificates. This option is usefully during debugging new connections, but should be disabled in productions systems.
10. (Optional) In the **Security Protocols** box, provide a comma delimited list of SSL/TLS values as required. For example: Tls11, Tls12, Tls13 (This property defaults to Tls12)
11. Click **OK** to close the configuration dialog box, and then click **Finish**.

Send TCP Activity

The **Send TCP** activity can be used in a runbook to send textual data to a specified server using TCP.

Important: The Monitor TCP activity does not provide a secure connection. It is important that confidential data be secured appropriately.

Required Properties

You must configure the following properties.

Text Encoding	Method used to encode the outgoing text message. The default it **UTF-8**.

Optional Properties

You can configure the following properties as required.

Fail On Timeout	Indicates whether the activity should fail with an error if the connection times out.
Wait For Response	Indicates whether to wait for a response from the HTTP server.

Published Data

The activity publishes the following activity-specific data items.

Connection Timed Out	Indicates whether the connection
Port	The port number of the remote host.
Server Address	The DNS name of the remote host
Response Text	The body of the response that was returned from the remote host.

Configuring the Send TCP Activity

The Send TCP configuration establishes a link between Orchestrator and an external TCP server. You can create as many configurations as you require.

To setup a Send TCP configuration:

1. In the Runbook Designer, click the **Options** menu and select **KA Network Messaging**. The **KA Network Messaging** dialog box appears.
2. On the **Configurations** tab, click **Add** to begin the configuration setup. The **Add Configuration** dialog box appears.
3. In the **Name** box, enter a name for the configuration. This could be the name of the server or some descriptive name to distinguish the type of configuration.
4. Click the ellipsis button (…) next to the **Type** box and select **Send TCP**.
5. In the **Port** box, type the port that you want to connect to.
6. In the **Server Address** box, type the name or IP address of the server that you want to connect to.
7. Optionally, in the **Receive Timeout** and **Send Timeout** boxes, modify the timeouts used to connect to the remote server. These settings may need to be modified depending on your network and applications response times.

Click **OK** to close the configuration dialog box, and then click **Finish**.

Configuration Files for Network Messaging

The Network Messaging XML Schema is used by runbook designers to build configuration files that can be used to customize the required and optional properties, filters, published data and behaviors of the Send and Monitor HTTP activities.

Configuring The Send HTTP Activity

Before you can use the Send HTTP Activity you must create a new network messaging configuration file and add one or more **<SendRequest>** elements to define the HTTP requests that can be sent.

Each <SendRequest> defines the optional and required properties and published data for your activity and specifies how the HTTP request will be constructed. Refer to the Integration Pack for Network Messaging XML Schema for more information.

Creating a configuration file to Send HTTP requests:

1. Create a new XML document using your favorite text editor.
2. Add a top level **<HttpApplications xmlns:ka="http://www.kelverion.com">** element.
3. Add a **<ka:SendRequest>** element to the top level **< ka:HttpApplications>** element. **Optional** – add multiple **< ka:SendRequest>** blocks to define multiple requests inside one configuration file.
    a. Add a **name** attribute and assign it value.
    b. Add a **method** attribute and assign it a value of “get”, “post”, “put” or “delete”. The default is “post”.
    c. To control the timeout used to wait for a server response add a **timeout** attribute and assign it a numeric value in milliseconds. The default is 10000 milliseconds.
    d. To include a Content-type header add a **contentType** attribute and assign it the desired value.
    e. To include a User-agent HTTP header add a **userAgent** attribute and assign it the desired value.
    f. To enable HTTP cookies, add an **enableCookies** attribute and assign a value of “yes” or “no”. Within a runbook, cookies will be shared by all Send HTTP activities that have cookies enabled.
    g. To control the level of authentication and impersonation add an **authentication** attribute and assign a value of “requested” or “required”. If the attribute is not present “requested” will be used.
    h. To control the impersonation level, add an **impersonation** attribute and assign a value of “anonymous”, “delegation”, “identification”, “impersonation” or “none”. The default is “anonymous”.

4. For each input property required by your request add an **<Input>** element. Each input will be displayed in your Send HTTP Activity's properties page and can be used to insert data into your request.
    a. Add an **id** attribute and assign it an alphanumeric value. This value will uniquely identify your input.
    b. To control how your input will appear add a **name** attribute and assign it the desired value. By default, the value assigned to the **id** attribute is used.
    c. To control whether your input is mandatory or optional add a **mandatory** attribute and assign it a value of "yes" or "no". Inputs are mandatory by default.
    d. To give your input a default value, add a **default** attribute and assign it the desired value.
    e. To control the type of data represented by the input, add a **type** attribute and assign it a value of "string", "list", "date", "file", "boolean", "computer" or "folder".
    f. To control the format of date inputs, add a **format** attribute and assign it the desired value. By default, the system's regional settings will be used. See Appendix C for more information on allowed date formats.
    g. To provide values for list inputs, add a **useValueSets** attribute and assign it a list of space delimited value set names.
5. For each custom HTTP header that you want to include with your request add a <**Header>** element.
    a. Add a **name** attribute and assign it the desired name.
    b. Add a **value** attribute and assign it the desired value. You may embed references to inputs by using the '$' character followed by the **id** of the input that you want to insert (i.e. $email).
6. For each output that would be parsed from incoming HTTP requests add an **<Output>** element.
    a. Add a **name** attribute and assign it a value.
    b. To include a description with your output, add a **description** attribute and assign it a value.
    c. To control the type of published data, add a **type** attribute and assign it a value of "integer", "date" or "string". The default value is "string".
    d. To control how date values are parsed, add a **format** attribute and assign an appropriate date format. For more information on date formats see Appendix C.
    e. Add a **parser** attribute and assign it a value of "regex" or "xpath". A value "regex" means the output will be parsed using the regular expression parser. A value of

"xpath" means the output will be parsed using the XPath parser. The XPath parser is used by default.

f. To control the regular expression parse, add a **regexOptions** attribute and assign a space delimited set of values. Allowed values include "IgnoreCase", "IgnorePatternWhitespace", "Multiline", "None", "RightToLeft" and "Singleline".

g. To control how your output is published in relation to other outputs add a **correlated** attribute and assign a value of "yes" or "no". A "yes" value means that the published data item will be published as a set with other correlated data. A "no" means the data will be published normally. Outputs are not correlated by default.

h. Add an expression that will be used to parse the output value from the body of the HTTP response.

7. For POST messages, add a **< ka:MessageTemplate**> element. This template will be used to build the body of the HTTP message included with your request.

   a. Add a **format** attribute and assign it a value of "xml", "html" or "text". The default format is XML.

   b. To include or exclude the XML declaration from XML messages add an **omitXmlDeclaration** attribute and assign it a value of "no" or "yes", respectively. XML declarations are included by default.

   c. To control xml indentation, add an **indent** attribute and assign a value of "yes" or "no". Xml messages are not indented by default.

   d. To control which XML elements will be written as CDATA sections, add a **cdataElements** attribute and assign a space delimited set of element names.

   e. To control the media type of your request, add a **mediaType** attribute and assign it the desired value. The default media type is "text/xml".

   f. To control how your message is encoded add the **encoding** attribute and assign it the desired value. The default encoding is "iso-8859-1".

8. Define the template that will be used to build your message within the content of the **< ka:MessageTemplate>** element. See Appendix B for more information on building message templates.

Configuring Inputs for HTTP GET Requests

When sending a HTTP GET requests, custom inputs can be used to build the HTTP request URL and there are two ways to go about doing this. The first method, which is typically used for requests to REST web services, uses the **<PathTemplate>** element to specify a template that will be used to dynamically construct the path fragment of the request URL. The second method, which occurs automatically when there is no <PathTemplate> present, to dynamically build the path to the resource that you want to retrieve using input values obtained for the Orchestrator data bus.

For example, consider the following hypothetical send request type, which uses a GET request to retrieve information about an incident having a specified Incident Number.

```
<?xml version="1.0" encoding="utf-8"?>
<ka:HttpApplications xmlns:ka="http://www.kelverion.com" version="1.0">
    <ka:SendRequest name="Get Incident" method="get">
        <ka:Input id="IncidentNum" name="Incident Number" />
        <ka:Output name="Urgency">//Urgency</ka:Output>
        <ka:Output name="Impact">//Impact</ka:Output>
        <ka:Output name="Description">//Description</ka:Output>
    </ka:SendRequest>
</ka:HttpApplications>
```

When the Send HTTP Activity runs, it may generate an HTTP Get Request with a query that looks something like this: *http://www.myserver.com/incidents?IncidentNum=123*

Configuring Inputs for HTTP POST and PUT Requests

When sending an HTTP POST or HTTP PUT request, custom inputs can be inserted into the message template using the <ValueOf> element.

For example, consider an HTTP POST request used to submit an incident form or an HTTP PUT requested use to update an incident. The constant text of the form is built using the <Text> element and the appropriate values are inserted into using the <ValueOf> element.

```
<?xml version="1.0" encoding="utf-8"?>
<ka:HttpApplications xmlns:ka="http://www.kelverion.com" version="1.0">
    <ka:SendRequest name="Create Incident" method="post"
contentType="application/x-www-form-urlencoded">
        <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:MessageTemplate format="xml">
            <ka:Text>impact=</ka:Text>
            <ka:ValueOf select="$Impact" />
            <ka:Text>&amp;priority=</ka:Text>
            <ka:ValueOf select="$Priority" />
            <ka:Text>&amp;urgency=</ka:Text>
            <ka:ValueOf select="$Urgency" />
            <ka:Text>&amp;description=</ka:Text>
            <ka:ValueOf select="$Description" />
            <ka:Text>&amp;opened=</ka:Text>
            <ka:ValueOf select="$Opened" />
            <ka:Text>&amp;submitter=</ka:Text>
            <ka:ValueOf select="$Submitter" />
         </ka:MessageTemplate>
    </ka:SendRequest>
</ka:HttpApplications>
```

When the Send HTTP Activity runs it may generate an HTTP POST or HTTP PUT request with a body of:

```
impact=High&priority=High&urgency=Critical&description=Air+conditi
oning+off+in+server+room&opened=2010%2D12%2D
21%3A1011%3A13&submitter=William+Sanders
```

Notice that the form data in the POST request has been encoded. The Send HTTP Activity examines the Content-type header included in the message and automatically URL encodes the data if necessary.

Configuring Inputs for HTTP Delete Requests

When sending an HTTP Delete requests, custom inputs are used to build the HTTP query URL. The query string is built using the <QueryTemplate> element.

For example, consider the following hypothetical Send HTTP, which uses a Delete request to delete an incident having a specified Incident Number.

```
<ka:HttpApplications xmlns:ka="http://www.kelverion.com">
    <ka:SendRequest name="Get Incident" method="delete">
        <ka:Input id="incidentnum" />
        <ka:QueryTemplate>
            <ka:Text>incidents?incidentnum=</ka:Text>
            <ka:ValueOf select="$incidentnum" />
        </ka:QueryTemplate>
    </ka:SendRequest>
</ka:HttpApplications>
```

When the Send HTTP Activity runs, it may generate an HTTP Delete Request with a query that looks something like this:

http://www.myserver.com/incidents?incidentnum=123

Configuring the Monitor HTTP Activity

Before you can configure the Monitor HTTP activity you must first create or modify a configuration file and define one or more request types using the **<MonitorRequest>** element.

Each <MonitorRequest> element defines the published data that will be available in your activity and how to parse values for the published data from incoming HTTP requests. Refer to the Integration Pack for Network Messaging XML Schema for more information.

Creating a Configuration to Monitor HTTP Requests:

1. Create a new XML document using your favorite text editor.

   Add an **<ka:HttpApplications xmlns:ka="http://www.kelverion.com">** element.

2. Add a **<ka:MonitorRequest>** element to the top level <HttpApplications> element. **Optional** – add multiple **<ka:MonitorRequest >** blocks to define multiple requests inside one configuration file.

   a. Add a **name** attribute and assign it a value that describes the monitor activity.

   b. Add an **authentication** attribute and assign it a value of "anonymous", "basic", "digest", "integrated", "negotiate", "none", or "ntlm".

3. Add a **<ka:Response>** element to the <ka:MonitorRequest> element. Specify the text message that will be returned by the monitor in response to incoming requests.
4. For each output you want to read from the incoming HTTP requests add an **<ka:Output>** element..

   a. Add a **name** attribute and assign it value.

   b. To include a description with your output, add a **description** attribute and assign it a value.

   c. To control the type of published data add a **type** attribute and assign it a value of "integer", "date" or "string". The default value is "string".

   d. To control how date values are parsed, add a **format** attribute and assign an appropriate date format. For more information on date formats see Appendix C.

   e. Add a **parser** attribute and assign it a value of "regex" or "xpath". A value "regex" means the output will be parsed using the regular expression parser. A value of "xpath" means the output will be parsed using the XPath parser. The XPath parser is used by default.

   f. To control the regular expression parse, add a **regexOptions** attribute and assign a space delimited set of values. Allowed values include "IgnoreCase", "IgnorePatternWhitespace", "Multiline", "None", "RightToLeft" and "Singleline". The default is "None".

   g. To control how your output is published in relation to other outputs add a **correlated** attribute and assign a value of "yes" or "no". A "yes" value means that the published data item will be published as a set with other correlated data. A "no" means the data will be published normally. By default, outputs are not correlated.

   h. To allow the output to be used to filter incoming requests, add the **filtered** attribute and assign it a value if "yes" or "no". The default is "yes".

   i. To provide a list of values for filtered outputs, add a **useValueSets** attribute and assign it a set of space delimited value set names.

   j. Add an expression that will be used to parse the output from the body of the HTTP request.

XML Schema reference

The following table lists the elements of the Kelverion Integration Pack for Network Messaging XML Schema.

Element	Description
ka:Attribute	Adds an XML attribute to the message body.
ka:AttributeSet	Defines a named set of XML attributes
ka:Choose	Used in conjunction with <When> and <Otherwise> to express multiple conditional tests.
ka:Element	Creates an XML element node in the message body
ka:Header	Defines a custom HTTP header that will be sent with the HTTP request.
ka:HttpApplications	Defines the root element of the HTTP Applications activity
ka:If	Contains a template that will be applied only if a selected condition is true.
ka:Input	Defines an input that can be used to retrieve data from the Integration Server data bus.
ka:Message	Writes a message to System Center Orchestrator.
ka:MessageTemplate	Defines the format and structure of the message body.
ka:MonitorRequest	Defines an activity that can be used to monitor HTTP requests.
ka:Otherwise	Specifies a default action for the <Choose> element.
ka:Output	Defines an output that can be used to parse data from an HTTP request/response and publish it to the System Center Orchestrator data bus.
ka:Response	Specifies the text response to be returned from a monitored HTTP request.
ka:SendRequest	Defines an activity that can be used to send HTTP requests
ka:Text	Writes literal text in the message body
ka:When	Specifies an action for the <choose> element.
ka:Value	Defines a value for an input or filter.
ka:ValueOf	Writes the value of an input variable to the message body
ka:ValueSet	Defines a named set of values.
ka:Variable	Declares a global variable

Attribute Element

The **Attribute** element is used to add an attribute to XML element.

```
<ka:Attribute
    name="attribute-name"
    namespace=”uri-reference”>
</ka:Attribute>
```

Element information

Number of occurrences	Unlimited
Parent elements	<ka:Element>, <ka:Message>, <ka:Otherwise>, <ka:Message>, <ka:Variable>, <ka:When>
Child elements	<ka:Choose>, <ka:If>, <ka:Message>, <ka:Text>, <ka:Variable>

Attributes

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

Remarks

The contents of this element specify the value of the attribute.

Examples

Add a severity attribute to an incident element and fills it with the value of the ‘severity’ input.

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

AttributeSet Element

The **AttributeSet** element creates a named set of attributes.

```
<ka:AttributeSet
    name="name"
    useAttributeSets="name-list">
</ka:AttributeSet>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<ka:MessageTemplate>
Child elements	<ka:Attribute>

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.

Remarks

The content of the **<ka:AttributeSet>** element consists of zero or more **<ka:Attribute>** elements that specify the attributes in the set. To use attribute sets, specify a **useAttributeSets** attribute on a **<ka:Elememt>** element. Specifying a useAttributeSets attribute can be an efficient way to declare multiple attributes on an element.

To accomplish the same results using **<ka:Attribute>**, you would have to use an **<ka:Attribute>** element for each attribute in each named attribute set.

Examples

Create an Attribute Set that can be applied to any element:

```
<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>
```

Choose Element

Tests multiple conditions in conjunction with the <ka:Otherwise> and <ka:When> elements.

```
<ka:Choose>
</ka:Choose>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<ka:Attribute>, <ka:Element>, <ka:If>, <ka:Message>, <ka:Otherwise>, <ka:Variable>, <ka:When>
Child elements	<ka:Otherwise>, <ka:When>

Remarks

The <ka:When> children of the <ka:Choose> element are tested, in order from top to bottom until a test attribute on one of these elements accurately describes conditions in the source data or until an <ka:Otherwise> element is reached. Once a <ka:When> or <ka:Otherwise> element is choses, the <ka:Choose> block is exited.

For simple conditional testing, use the <ka:If> element.

Examples

Use the <ka:Choose> element to map a text representation of the Impact element to a numeric value.

```
<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>
```

Element

Creates an output element with the specified name within the body of the HTTP request.

```
<ka:Element
    name="name"
   namespace="uri-reference"
   useAttribueList="qualified-name”>
</ka:Element>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<ka:MessageTemplate>, <ka:Otherwise>
Child elements	<ka:Attribute>, <ka:Choose>, <ka:If>, <ka:Text>, <ka:ValueOf>, <ka:Variable>

Attributes

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

Remarks

The <ka:Element> element allows an element to be created with a computed name. The name of the element to be created is specified by a required name attribute and an optional namespace attribute. The content of the <ka:Element> element is a template for the attributes and children of the created element.

Examples

Create an **Incident** element with **Impact**, **Urgency**, **Priority** and **Description** child elements.

```
<ka:MessageTemplate 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:MessageTemplate>
```

Header Element

The<ka:Header> element is used to define a custom HTTP header to be included in an HTTP web request.

```
<ka:Header name="name" value="string" />
```

Element Information

Number of occurrences	Unlimited
Parent elements	<ka:SendRequest>
Child elements	None

Attributes

Attribute	Description
name	Required. Specifies the header's field name.
value	Required. Defines the expression that will be used to build the header's field value.

Examples

The following example is an HTTP GET request used to retrieve incidents that occur between a user defined date/time range. The email of the user making the request is passed to the server in the custom "From" header.

```
<?xml version="1.0" encoding="UTF-8"?>
<ka:HttpApplications xmlns:ka="http:"//www.kelverion.com version="1.0">
    <ka:SendRequest name="Get Incidents" method="get">
        <ka:Input id="from" name="From Date" type="date" />
        <ka:Input id="to" name="To Date" type="date" />
        <ka:Input id="requestor" name="Requested By" />
        <ka:Header name="From" select="$requestor" />
        <ka:Output name="Incident ID"
                   correlated="yes">//Incident/ID"</ka:Output>
        <ka:Output name="Summary"
                   correlated="yes">"//Incident/Summary"</ka:Output>
    </ka:SendRequest>
</ka:HttpApplications
```

HttpApplications Element

The <HttpApplications> element is used to define the root element of an Integration Pack for Network Messaging activity.

```
<ka:HttpApplications
    version="version">
</ka:HttpApplications>
```

Element Information

Number of occurrences	One
Parent elements	No parent elements
Child elements	<SendRequest>, <MonitorRequest>

Attributes

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

Remarks

Every Integration Pack for Network Messaging configuration file must declare <HttpApplications> as its document element.

Input Element

The **<Input>** element defines an input property that can be used in a Send HTTP activity to collect information from the user so that it can be inserted into the body of an HTTP POST request, the URL of a GET request or as part of an HTTP header.

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

Element Information

Number of occurrences	Unlimited
Parent elements	<SendRequest>
Child elements	No child elements

Attributes

Attribute	Description
id	Required. Specifies the unique identifier.
name	Optional. Specifies the name used to display the input in the System Center Orchestrator client. Default is the value of the “id” attribute.
mandatory	Optional. “yes” specifies that the input is required. “no” specifies that the input is optional. Default is “yes”.
default	Optional. Specifies a default value.
type	Optional. Specifies the type of data that the data represents. Some data types will include a browser when accessed from the System Center Orchestrator client. Default is “string”.
format	Optional. Specifies how date/time values will be formatted. The default is the system’s current regional settings.
useValueSets	Optional. For list inputs, this space-delimited list of value set names is used to provide values for the list browser.

Remarks

Each <Input> property will appear as a required or optional property.

If Element

Allows simple conditional message fragments.

```
<ka:if
    test="boolean-expression">
</ka:If>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<Attribute>, <Element>, <Message>, <Otherwise>, <Variable>, <When>, output elements
Child elements	<Attribute>, <Choose>, <Element>, <If>, <Text>, <Variable>, output elements

Attributes

Attribute	Description
test	Required: Specifies the condition to be tested.

Remarks

The content is a message template. The expression is evaluated, and the resulting object is converted to **true** or **false**. If the result is **true**, the content template is instantiated; otherwise nothing is created.

Examples

The following example sends an HTTP POST request to close an Incident. The message template uses the <If> element to terminate with an error message if the input properties are empty.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<ka:HttpApplications>
    <ka:SendRequest name="Close Incident" method="post">
        <ka:Input id="closedAt" name="Closed Date" />
        <ka:Input id ="closedBy" name="Closed By" />
        <ka:Input id="closedAt" name="Closed Date" />
        <ka:Input id ="closedBy" name="Closed By" />
        <ka:MessageTemplate 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:MessageTemplate>
    </ka:SendRequest>
</ka::HttpMessaging>
```

Message Element

The **<Message>** reports an event in System Center Orchestrator.

```
<ka:Message
    terminate="yes|no">
</ka:Message>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<Attribute>, <Element>, <If>, <Message>, <Otherwise>, <Variable>, <When>, output elements
Child elements	<Attribute>, <Choose>, <Element>, <If>, <Text>, <ValueOf>, <Variable>, output elements

Attributes

Attribute	Description
terminate	Optional. “yes” terminates the activity and reports the message as an error. “no” reports the message as and continues sending the message. Default is “no”.

Examples

The following example sends an HTTP POST request to close an Incident. The message template uses the <If> element to terminate with an error message if the input properties are empty.

MessageTemplate Element

The **<MessageTemplate>** element defines the template that will be used to construct the body of an HTTP POST request.

```
<ka:MessageTemplate
    format="xml|text|html"
    cdataElements="name list"
   encoding="encoding"
   indent="yes|no"
   mediaType="string"
   omitXmlDeclaration="yes|no"
   version="1.0">
</ka:MessageTemplate>
```

Element Information

Number of occurrences	Once
Parent elements	<SendRequest>
Child elements	<Attribute>, <Choose>, <Element>, <If>, <Message>, <Text>, <ValueOf>, <Variable>, output elements

Attributes

Attribute	Description
format	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). Valid values include "xml", "text", "html".
cdataElements	Optional. A white-space separated list of elements whose text contents should be written as CDATA sections.
encoding	Optional. Sets the value of the encoding attribute in the output.
indent	Optional. "yes" indicates that the message body should be indented to its hierarchic structure. "no" indicates that the message will not be indented.
mediaType	Optional. Defines the MIME type of the message body. The default is "text/xml".
omitXmlDeclaration	Optional. "yes" specifies that the XML declaration (<?xml ...?>) should be omitted in the message body. "no" specifies that the XML declaration should be included in the message body. The default is "no".
version	Optional. Sets the W3C version number for the message body (only used with format="html" for format="xml".

MonitorRequest Element

The <MonitorRequest> element is used to configure an activity to monitor HTTP requests.

```
<ka:MonitorRequest
    name="string"
    authentication="anonymous|basic|digest|none|integrated|negotiate|ntml">
</ka:MonitorRequest>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<HttpApplications>
Child elements	<Output>, <Response>, <ValueSet>

Attributes

Name	Value	Description
name	String	Optional. Specifies a name that identifies the monitor.
authentication	none anonymous basic digest integrated negotiate ntml	Optional. Specifies the scheme used to authenticate HTTP requests. “none” specifies no authentication is allowed. “anonymous” specifies anonymous authentication. “basic” specifies basic authentication. “digest” specifies basic authentication. “integrated” specifies integrated windows authentication. “negotiate” specifies that the monitor negotiates with the client to determine the authentication scheme. “ntlm” specifies that NTLM authentication. The default is “anonymous”.

Examples

```
<ka:MonitorRequest name="Incident Created" authentication="anonymous">
    <ka:Response>Incident Handled OK</ka:Response>
    <ka:ValueSet name="Urgency">
        <ka:Value>High</ka:Value>
        <ka:Value>Medium</ka:Value>
        <ka:Value>Low</ka:Value>
    </ka:ValueSet>
    <ka:ValueSet name="Impact">
        <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="Incident ID">//Incident/IncidentID</ka:Output>
    <ka:Output name="Urgency" correlated="yes" filtered="yes"
               useValueSets="Impact">//Incident/Urgency</ka:Output>
    <ka:Output name="Priority" correlated="yes" filtered="yes"
```

```
                useValueSets="Priority">//Incident/Priority</ka:Output>
    <ka:Output name="Impact" correlated="yes" filtered="yes"
                useValueSets="Impact">//Incident/Impact</ka:Output>
    <ka:Output name="Submitter" correlated="yes"
                filtered="yes">//Incident/Submitter</ka:Output>
    <ka:Output name="Opened" type="date" correlated="yes"
                filtered="yes">//Incident/Opened</ka:Output>
    <ka:Output name="Summary">//Incident/Summary</ka:Output>
</ka:MonitorRequest>
```

Otherwise Element

Provides multiple conditional testing in conjunctions with the <Choose> and <When> elements.

```
<ka:Otherwise/>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<Choose>
Child elements	<Attribute>, <Choose>, <Element>, <If>, <Text>, <ValueOf>, <Variable>, output elements

Remarks

Provides a default condition for <Choose>. Other alternatives are indicated by <When> elements.

For simple conditional testing, use the <If> element.

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>
```

Output Element

The <Output> element is used to define the published data for a Monitor/Send HTTP activity.

```
<ka:Output
    name="string"
    description="string"
    parser="xpath|regex"
    type="string|number|date|boolean"
    correlated="yes|no"
    filtered="yes|no"
    useValueSets="name list">
</ka:Output>
```

Element Information

Number of occurrences		Unlimited
Parent elements		<MonitorRequest>, <SendRequest>
Child elements		None

Attributes

Attribute	Description
name	Required. Specifies the name uniquely identify the output.
correlated	Optional. “yes” specifies that the output should be published as a part of a correlated data set. “no” specifies the output should be published as an individual value. Default is “no”
description	Optional: Specifies a description for the output.
filtered	Optional: “yes” specifies that the output can be used to filter incoming requests for Monitor Request activities. “no” specifies that the output cannot be used for filtering. The default is “no”
parser	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. The default is “xpath”.
type	Optional: Specifies that published data type that will be displayed in the System Center Orchestrator Client. The default is “string”. Valid values include “string”, “number”, “date” and “boolean”
useValueSets	Optional. For filtered outputs, this space delimited list of value sets names is used to provide values for the filter’s list browser.

Remarks

The published data in your HTTP Applications activities 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.

Examples

The following examples uses the **correlated** attribute to group outputs used to publish employee information to the System Center Orchestrator data bus.

```
<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>
```

This request may return an XML formatted message with multiple Employee elements. For example:

```
<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>
```

Which, publishes the following correlated data to the Microsoft System Center Orchestrator data bus:

ID: 0012	ID: 0013
First Name: Kevin	First Name: Nancy
Last Name: Walters	Last Name: Haddix
Phone: 6187852341	Phone: 6187854555
Email:	Email: nancy.haddix@acme.com
Department: Customer Support	Department: Human Resources

PathTemplate Element

The **<PathTemplate>** element defines the template that will be used to construct the path portion of the URL for an HTTP web request.

```
<ka:PathTemplate>
    <!-- Content:(<ka:Text>*, <ka:ValueOf>*) -->
</ka: PathTemplate>
```

Element Information

Number of occurrences	One
Parent elements	<ka:SendRequest>
Child elements	<ka:Choose>, <ka:Element>, <ka:If>, <ka:Text>, <ka:ValueOf>, <ka:Variable>, output elements

Example

The following example sends an HTTP GET request to retrieve an issue using the Issue Key provided from the Orchestrator data bus.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<ka:HttpApplications>
    <ka:SRequest name="Get Issue" method="get">
        <ka:Input id="IssueKey" name="Issue Key" />
        <ka:PathTemplate>
            <ka:Text>issues/</ka:Text>
            <ka:ValueOf select="$IssueKey”/>
        </ka:PathTemplate>
    </ka:SendRequest>
 </ka:HttpMessaging>
```

SendRequest Element

The **<SendRequest>** element is used to define an HTTP request that can be sent using the Send HTTP Activity.

```
<ka:SendRequest
    name="name"
    method="get|post"
    timeout="number"
    enableCookies="yes|no"
    userAgent="string"
    contentType="string"
    authentication="requested|required"
    impersonation="anonymous|delegation|identification|impersonation|none"
</ka:SendRequest>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<HttpApplications>
Child elements	<Input>,<Header>, <MessageTemplate>, <Output>, output elements

Attributes

Attribute	Description
name	Required. Uniquely identifies the request.
method	Optional. “post” specifies that the request will be sent using HTTP POST. “get” specifies that the request will be sent using HTTP GET. The default is “post”
enableCookies	Optional. “yes” specifies that the request will include cookies. “no” specifies that the request will not include cookies. The default is “no”.
userAgent	Optional. The User-agent header to be included in the request. The default is none.
contentType	Optional. The Content-type header to be included in the request. The default is none.
authentication	Optional. The authentication level used to authenticate the request. “requested” specifies that both the client and server should be authenticated, but the request will not fail if the server is not authenticated. “required” specifies that both the client and server should be authenticated, and the request will fail if the server is not authenticated. The default is “requested”.
impersonation	Optional. Specifies the security impersonation level. “none” specifies that impersonation is not assigned. “anonymous” specifies that the server cannot impersonate the client. “identification” specifies that server can obtain information about the client, but that it cannot impersonate the client. “impersonation” means the server can impersonate the client on its local system. “delegation” specifies that the server can impersonate the client on remote systems. The default is “anonymous”.

Response Element

The <Response> element is used to specify the text response that the Monitor HTTP activity should return in response to an incoming HTTP request.

```
<ka:Response>
   <!-- Content: response text -->
</ka:Response>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<MonitorRequest>
Child elements	No child elements

Remarks

The content of this element specifies the response that the Monitor HTTP activity should send to each HTTP client.

Example

```
<ka:MonitorRequest name="Ping" authentication="anonymous">
    <ka:Response>Monitor HTTP Activity OK</ka:Response>
</ka:MonitorRequest>
```

Text Element

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

```
<ka:Text/>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<Attribute>, <Element>, <If>, <Otherwise>, <Message>, <Variable>, <When>, output elements
Child elements	No child elements

Remarks

In a message template, text can be generated directly with or without the <Text> element. However, with this element you can exert some control over the whitespace that is created.\\

For example, sometimes you might want to introduce a white space character to separate two date values. You can use a <Text> element to accomplish this.

Examples

```
<?xml version="1.0" encoding="utf-8"?>
<ka:HttpApplications xmlns:ka="http://www.kelverion.com" version="1.0">
    <ka:SendRequest name="Create Incident" method="post"
contentType="application/x-www-form-urlencoded">
        <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:MessageTemplate format="xml">
            <ka:Text>impact=</ka:Text>
            <ka:ValueOf select="$Impact" />
            <ka:Text>&amp;priority=</ka:Text>
            <ka:ValueOf select="$Priority" />
            <ka:Text>&amp;urgency=</ka:Text>
            <ka:ValueOf select="$Urgency" />
            <ka:Text>&amp;description=</ka:Text>
            <ka:ValueOf select="$Description" />
            <ka:Text>&amp;opened=</ka:Text>
            <ka:ValueOf select="$Opened" />
            <ka:Text>&amp;submitter=</ka:Text>
            <ka:ValueOf select="$Submitter" />
         </ka:MessageTemplate>
    </ka:SendRequest>
</ka:HttpApplications>
```

When Element

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

```
<ka:When test="boolean expression"/>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<Choose>
Child elements	<Attribute>, <Choose>, <Element>, <If>, <Text>, <ValueOf>, <Variable>, output elements.

Attributes

Attributes	Description
test	Required. Specifies a boolean expression to be tested.

Remarks

Describes one of the alternatives to be chosen by the <Choose> element. The default alternative is described by the <Otherwise> element.

For simple conditional testing, use the <If> element.

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>
```

Value Element

The **<Value>** element is used in conjunction with the <ValueSet> and <Input> elements to define the items to be displayed within the list browser associated with the input properties in an Send HTTP activity.

```
<ka:Value/>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<ValueSet>
Child elements	No child elements.

Example

```
<ka:HttpApplications xmlns:ka="http://www.kelverion.com">
    <ka:SendRequest name="Create Incident" method="post">
        <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:Input id="Impact" type="list" useValueSets="Impact" />
        <ka:Input id="Priority" type="list" useValueSets="Priority" />
        <ka:Input id="Urgency" type="list" useValueSets="Urgency" />
        <ka:Input id="Description" />
        <ka:Input id="Opened" type="date" />
        <ka:Input id="Submitter" />
        <ka:MessageTemplate format="xml">
            <Incident>
                <Impact>
                    <ka:ValueOf select="$Impact />
                </Impact>
                <Priority>
                    <ka:ValueOf select="$Priority" />
                </Priority>
                <Urgency>
                    <ka:ValueOf select="$Urgency" />
                </Urgency>
                <Description>
                    <ka:ValueOf select="$Description" />
                </Description>
```

```
                <Opened>
                    <ka:ValueOf select="$Opened" />
                </Opened>
                <Submitter>
                    <ka:ValueOf select="$Submitter" />
                </Submitter>
            </Incident>
        </ka:MessageTemplate>
    </ka:SendRequest>
</ka:HttpApplications>
```

ValueOf Element

The **<ValueOf>** element inserts the runtime value of a specified input property.

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

Element Information

Number of occurrences	Unlimited
Parent elements	<Attribute>, <Element>, <If>, <Message>, <Otherwise>, <MessageTemplate>, <Variable>, <When>, output elements
Child elements	No child elements.

Attributes

Attribute	Value	Description
select	expression	Required. The qualified name of an input property proceeded by a dollar sign '$'. For example, "$EmployeeID".

Remarks

The <ValueOf> element inserts a text string representing the runtime value of the specified input property.

Example

```
<ka:HttpApplications xmlns:ka="http://www.kelverion.com">
    <ka:SendRequest name="Create Incident" method="post">
        <ka:Input id="Impact" />
        <ka:Input id="Urgency" />
        <ka:Input id="Priority" />
        <ka:Input id="Description" />
        <ka:Input id="Submitter" />
        <ka:Output name="IncidentNum">
        count(//Incident)
        </ka:Output>
        <ka:MessageTemplate 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:MessageTemplate>
    </ka:SendRequest>
</ka:HttpApplications>
```

ValueSet Element

The <ValueSet> element is used in conjunction with the <Value> and <Input> elements to define the items to be displayed within the list browser associated with the input properties in an Send HTTP activity.

```
<ka:ValueSet
    name="string">
</ka:ValueSet>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<SendRequest>
Child elements	<Value>

Attributes

Name	Value	Description
name	String	Required. Uniquely identifies the value set.

Example

```
<ka:HttpApplications xmlns:ka="http://www.kelverion.com" version="1.0">
    <ka:SendRequest name="Create Incident" method="post">
        <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:Input id="Impact" type="list" useValueSets="Impact" />
        <ka:Input id="Priority" type="list" useValueSets="Priority" />
        <ka:Input id="Urgency" type="list" useValueSets="Urgency" />
        <ka:Input id="Description" />
        <ka:Input id="Opened" type="date" />
        <ka:Input id="Submitter" />
        <ka:MessageTemplate format="xml">
            <ka:Element name="Incident">
                <ka:Element name="Impact">
                    <ka:ValueOf select="$Impact />
                </ka:Element>
                <ka:Element name="Priority">
                    <ka:ValueOf select="$Priority" />
                </ka:Element>
```

```
                <ka:Element name="Urgency">
                    <ka:ValueOf select="$Urgency" />
                </ka:Element>
                <ka:Element name="Description">
                    <ka:ValueOf select="$Description" />
                </ka:Element>
                <ka:Element name="Opened">
                    <ka:ValueOf select="$Opened" />
                </ka:Element>
                <ka:Element name="Submitter">
                    <ka:ValueOf select="$Submitter" />
                </Element>
            </ka:Element>
        </ka:MessageTemplate>
    </ka:SendRequest>
</ka:HttpApplications>
```

Variable Element

The **<Variable>** declares a global variable that can be used to insert a constant value into the body of an HTTP request.

```
<ka:Variable
    name="qualified-name"
    select="expression">
</ka:Variable>
```

Element Information

Number of occurrences	Unlimited
Parent elements	<Attribute>, <Element>, <If>, <Otherwise>, <MessageTemplate>, <Variable>, <When>, output elements
Child elements	<Attribute>, <Choose>, <Element>, <If>, <Text>, <ValueOf>, <Variable>, output elements

Attributes

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

Example

```
<ka:Variable name=”id” select=”1234” />
...
<ka:ValueOf select=”$id” />
```

Appendix C: Date and Time Formats

The following table describes the standard date and time format strings.

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

The following table describes the custom date and time format specifiers.

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 leasing 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 '\\'.