In Situ Substitution Expressions (ISSEs)

CloudTest version

This document applies to CloudTest build 5623.56 and later.

In Situ Substitution Expressions (ISSEs)

In Situ Substitution Expressions (ISSEs), are text expressions that can be entered anywhere within Message text to be sent. They can also be used in various places within Validations, in HTTP headers, and in Property Sets ("PropSets"). The ISSEs are replaced with their corresponding value before the text in which they reside is used (for example, before the Message is sent or the Validation is evaluated).

ISSEs can be entered in either the Form or the XML view of the Message Editor. They can appear anywhere within the Message text. There is no relationship between ISSEs and XML (if the Message being sent is XML) – a pure text replacement is done. Therefore, ISSEs can cause XML to be inserted into the Message. There is currently no way to specify that the text produced by an ISSE is to be escaped for XML purposes – if such escaping is necessary, you must arrange for that to happen yourself.

ISSE's can be interspersed within partial field values to cause the results of the ISSEs to be concatenated to or inserted into partial constant values.

There are four major types of ISSEs: System and Custom Property references, Global Property references, expressions, and random values.

References to System and Custom Properties

This form of ISSE refers to a System or Custom Property within one of the items in the Composition. The value of the specified property is substituted in place of the ISSE.

The syntax of this form of ISSE is:

`{%% property-type : path-type : property-path %%}`

Spaces are not significant except for cases in which the names of properties specified in the property-path contain spaces.

Property-type field

This field specifies what type of property is being referenced. It can be any of the following values to denote a Custom Property (case is not significant):

- Empty string (e.g. "`{%% : path-type : property-path %%}`")
- "`prop`"
- "`custom-prop`"
- "`custom-property`"

It can be any of the following values to denote a System Property (case is not significant):

- "`sys-prop`"
- "`sys-property`"
- "`system-property`"

Path-type field

This field specifies the "starting point" of the path, as well as sometimes some other information. The "starting point" of the path is the item within the Composition hierarchy that the path is relative to. It can be any of the following values (case is not significant):

- "`Composition`"

  The path is relative to the Composition as a whole. (Therefore, the path must start with the name of a Band.)

- "`Band`"

  The path is relative to the current Band.

- "`Track`"

  The path is relative to the current Track.

- "`MessageClip`" or "`Clip`" (either one is accepted)

  The path is relative to the current Clip. If there are nested Clips, this refers to the immediate (closest) parent Clip in the hierarchy..

- "`Transaction`"

  The path is relative to the current Transaction. In the case of nesting, if there is more than one parent Transaction in the Message's container hierarchy, this refers to the immediate (closest) parent Transaction in the hierarchy.

- "`Group`"

  The path is relative to the current Group. In the case of nesting, if there is more than one parent Group in the Message's container hierarchy, this refers to the immediate (closest) parent Group in the hierarchy.

- "`Chain`"

  The path is relative to the current Chain. In the case of nesting, if there is more than one parent Transaction in the Message's container hierarchy, this refers to the immediate (closest) parent Transaction in the hierarchy.

- "`If`"

  The path is relative to the current "If" object. In the case of nesting, if there is more than one parent "If" in the Message's container hierarchy, this refers to the immediate (closest) parent "If" in the hierarchy.

- "`Switch`"

  The path is relative to the current Switch object. In the case of nesting, if there is more than one parent Switch in the Message's container hierarchy, this refers to the immediate (closest) parent Switch in the hierarchy.

- "`Chain`"

  The path is relative to the current Chain. In the case of nesting, if there is more than one parent Transaction in the Message's container hierarchy, this refers to the immediate (closest) parent Transaction in the hierarchy.

- "`Page`"

  The path is relative to the current Page. In the case of nesting, if there is more than one parent Page in the Message's container hierarchy, this refers to the immediate (closest) parent Page in the hierarchy.

- "`Message`"

  The path is relative to the current Message (the Message containing the ISSE).

- "`Browser Action`"

  The path is relative to the current Browser Action (the Browser Action containing the ISSE).

- "`Destination`"

  The path is relative to the current Message or Browser Action. Once the item is found, the item's Target is used.

Property-path field

This field specifies the path to the item and property that is to be accessed.

The path can be a simple as only a property name, in which case the property will be taken from the "current" item as specified in the path-type field. For example, the current Message, its "destination", or an immediate parent object.

In order to refer to the properties of other objects, a "path" to the property can be given, using the names of the containers of the property, separated by slashes.

For example, the following path:

Path type: `Composition`

Path: `Band2/Track3/Clip2/Message5/MyProperty`

Refers to the property named "`MyProperty`" in Message "`Message5`", which is contained in Clip "`Clip2`", in Track "`Track3`" of Band "`Band2`".

If any of the object or property names contain spaces, they must appear at their appropriate places, for example if the names in the above example contained spaces, the path would be:

Path type: `Composition`

Path: `Band 2/Track 3/Clip 2/Message 5/My Property`

"Left" portions of the path may be omitted to provide "relative" paths which reference "current" item(s) and parent(s) according to the specified path-type and the context in which the ISSE is used. For example, if path-type "Track" were used with the following path in the above example for a Message property:

Path type: `Track`

Path: `Clip 2/Message 5/My Property`

The ISSE would be referencing the specified Clip "`Clip 2`" and Message "`Message 5`" contained within the current Track (since the Band and Track portions of the path were omitted), since the path-type (starting point of the path) was specified as "Track".

For path-type "`Destination`", the path to a Message or Browser Action is given, and this specifies the Target being used by that Message or Browser Action. For example:

Path type: `Destination`

Path: `Band5/Track6/Clip2/Message27/TargetPropertyName`

A path to any arbitrary Target (by name) in any arbitrary Message Clip can also be specified. For example:

Path type: `Composition`

Path: `Band5/Track6/Clip2/Target7/TargetPropertyName`

Since a Clip's Target cannot have the same name as any individual item within the Clip, there is no ambiguity between a Clip's Targets and it's individual items.

Note that for Clips, Scripts and Targets, the "local name" (as defined in the Composition or Clip) must be used, not the name of the Repository item that is being linked to. (Since a Clip, Script or Target can be used multiple times, the Repository Name is not necessarily unique.)

If the property's value is an array, a subscript may optionally appear after the property name in order to reference a particular element in the array. For example:

```
Clip 2/Message 5/My Property[5]
```

There must be no spaces between the property name and the subscript.

"**..**" may be used within a path to express the parent of an item. For example, if the path-type is "Clip", the path could be:

```
../Clip 2/Message 5/My Property[5]
```

to specify another Clip in the same Track as the current Clip.

"**.**" may be used within a path to express the current item in the path. For example, the following two paths are equivalent:

```
Clip 2/Message 5/My Property[5]
./Clip 2/Message 5/My Property[5]
```

References to Global Custom Properties

This form of ISSE refers to a Global Custom Property from a Global Custom Property List. The value of the specified Global Custom Property is substituted in place of the ISSE. Global Custom Property Lists are defined in Central and, as the name implies, are globally accessible from all Compositions.

The syntax of this form of ISSE is:

```
{%% property-type : property-path %%}
```

Spaces are not significant except for cases in which the names of lists or properties specified in the property-path contain spaces.

Property-type field

This field must be any of the following values to denote a Global Custom Property (case is not significant):

- "`global-prop`"
- "`global-property`"
- "`global-custom-prop`"
- "`global-custom-property`"

Property-path field

This field specifies which property is to be accessed.

It can be as simple as only a property name, in which case the property will be taken from the default Global Custom Property List (named "Default").

In order to refer to properties in other lists aside from the Default, a "path" to the property can be given, using the name of the Global Custom Property List, a slash, and then the name of the Global Property in that list.

For example, the following path:

```
My list/Some property
```

Refers to the property named "`Some property`" in the Global Custom Property List named "`My list`".

The following paths are equivalent:

```
Default/Property 6
Property 6
```

Both of the above paths refer to the property named "`Property 6`" in the default Global Custom Property List.

Counters

Global Custom Properties that are defined to be of type "counter" contain integer numbers that are automatically incremented every time they are referenced. The ISSE reference syntax is the same regardless of whether the property being referenced is a counter.

Item-Level Counters

Global Custom Properties that are defined to be of type “item-level counter” behave the same as regular “counters”, except that they are incremented only the first time they are referenced within each item. Thus if there is more than one ISSE in the same item referencing the same Global Custom Property of type “item-level counter”, that property will only be incremented once for that item and all of those ISSE substitutions within that item will have the same value substituted.

The following item-level counters are supported:

- Message level counter
- Browser Action counter
- Group level counter
- Chain level counter
- Clip level counter
- Track level counter
- Band level counter
- Composition level counter

Execution of an expression (script)

This form of ISSE causes a scripting expression to be executed. The value of the expression is substituted in place of the ISSE.

The syntax of this form of ISSE is:

```
{%% expression-indicator: expression-text %%}
```

Spaces are not significant. There is currently no provision for allowing the sequence "`%%}`" to appear in the script. (This should not be a problem, since that sequence has no meaning in JavaScript. If it were desired to use that value in a string within the script, something like "`'%%' + '}'`" could be used within the script.)

Expression-indicator field

This field must be one of the following values (case is not significant):

- "`expr`"
- "`expression`"

Expression-text field

This is the expression to be executed.

It can be a simple expression, such as "`10 + 20`". The reserved variable name "`result`" is the result of the expression, thus "`result = 10 + 20;`" would have the same effect.

The script can be a complete program if desired. For example, the following would yield the same result as the prior examples:

```
var a=10; var b=20; result=a+b;
```

The expression script has full access to any of the capabilities of a Script object. It can even modify properties or other portions of the Composition, as allowed in a Script. Obviously this has the potential to be confusing if taken too far.

The result of the expression that is substituted is the value of either the variable named "result" or the last expression (without a left side) or function call executed.

See the separate documentation on scripting for a full description of what can be done within a script.

Insertion of a random value

This form of ISSE causes a random string value to be substituted in place of the ISSE.

The syntax of this form of ISSE is:

```
{%% random : string-type, string-length %%}
```

or

```
{%% randomvalue : string-type, string-length %%}
```

Spaces are not significant. The "random" and "randomvalue" keywords may be in any mixture of upper and/or lower case.

String-type field

This fields specifies the type of random string to be generated (the set of characters from which the string is to be generated). It must be one of the following values, in any combination of upper and/or lower case:

- "`alphanumeric`"

  A random alphanumeric string is to be substituted, consisting of the characters A-Z, a-z, and 0-9.

- "`alpha`"

  A random alphabetic string is to be substituted, consisting of the characters A-Z and a-z.

- "`decimaldigits`"

  A random string of decimal digits is to be substituted, consisting of the characters 0-9.

- "`hexdigits`"

  A random string of hex digits is to be substituted, consisting of the characters 0-9 and A-F.

String-length field

This field specifies the length of the random string to be generated (the number of characters). It must be an integer value greater than zero.

Examples

```
{%% random : alphanumeric, 100 %%}
{%% randomvalue: hexdigits,5 %%}
```

Encoding of ISSE substitution values

The value that is substituted in place of the ISSE can be encoded by specifying an encoding type as follows:

```
{%% property-type, encoding=encoding-type : path-type :
property-path %%}
```

Where "encoding-type" may be one of the following (case is not significant):

- `urlencode`

  The result of the ISSE is URL-encoded before substitution.

- `urldecode`

  The result of the ISSE is assumed to be URL-encoded and is URL-decoded before substitution.

- `base64`

  The result of the ISSE is Base64 encoded before substitution.

Examples:

```
{%% prop, encoding=urlencode : clip : My clip property %%}
{%% random, encoding=Base64: hexdigits, 8 %%}
```

Some examples

```
{%% prop : composition : CompositionProp1 %%}
{%%Prop:Band:BandProp2%%}
{%% Custom-prop: Track: TrackProp2 %%}
{%% custom-property: Clip: ClipProp1 %%}
{%% custom-property: MessageClip: ClipProp1 %%}
{%%Custom-property:destination:TargetProp1%%}
{%%Custom-property:desTination:TargetProp2%%}
{%%Custom-Property:message:Message1Prop1%%}
{%% sys-prop : destination : HostName %%}
{%%sys-Prop:Destination:ServicePath%%}
{%% Sys-property: desTination: Port %%}
{%% sys-property : destination : UseSSL %%}
{%%sys-Property:Destination:UserName%%}
{%% system-property : destination : URL %%}
{%% system-property : clip : Delay 5/Duration %%}
{%% eXpr: "Hello, world!" %%}
{%% eXpression: 10 + 20 + 30 %%}
{%%prop:composition:Band 1/BandProp2%%}
{%%prop:band:../Band 2/Band2Prop1%%}
{%%prop:band:Track 1/TrackProp2%%}
{%%prop:track:Clip 021/ClipProp2%%}
{%%prop:track:../../Band 1/Track 1/TrackProp1%%}
{%%prop:track:../Track 2/Track2Prop2%%}
{%%prop:clip:Message 2/Message2Prop2%%}
{%%prop:clip:../Track 1/Clip 021/ClipProp2%%}
{%%prop:clip:../../../Band 1/Track 1/Clip 021/ClipProp1%%}
{%%prop:message:../Message 2/Message2Prop2%%}
{%%prop:message:../../Clip 021/Message 2/Message2Prop2%%}
{%%prop:message:../../../Track 1/Clip 021/Message 2/Message2Prop2%%}
{%%prop:destination:../Message 2/TargetProp2%%}
{%%prop:destination:../../Clip 021/Message 2/TargetProp2%%}
{%%prop:destination:../../../Track 1/Clip 021/Message 2/TargetProp2%%}
{%%prop:clip:Target 021 local name/TargetProp2%%}
{%%prop:track:Clip 021/Target 021-2 local name/TargetProp2%%}
{%%prop:band:Track 1/Clip 021/Target 021 local name/TargetProp2%%}
{%%prop:composition:Band 1/Track 1/Clip 021/Target 021 local
name/TargetProp1%%}
{%% global-prop: Property A %%}
{%%global-property: Default/Counter 1%%}
{%%global-system-property: My list 3/My Counter 2%%}
{%% random, encoding=base64: alphanumeric,100 %%}
{%% RandomValue: DecimalDigits, 15 %%}
```