Intel Cloud Integrity Technology 3.0

**Localization**

# Background

To prepare for international use, and also to facilitate easy changes to text that may appear in the user interface, the product needs a framework for separating localizable text from the code.

# Principles

## Programming Constants vs Localizable Text

A programming constant can be an ENUM value, a field name, or other identifier that is part of the programming interface and is not localizable.

User input is not localizable. It is stored and retrieved as it was provided.

All other text is generally localizable. APIs tend to have little or no localizable text at all. User interfaces tend to have a lot of localizable text.

For example, in the following data structure:

{ "id": 1234, "status": "ERROR", "description": "a new widget", "faults": [ { "@class": "com.intel.mtwilson.widgets.ColorFault", "color": "red" } ] }

* The keys “id”, “status”, “description”, “faults”, “@class”, and “color” are programming constants and are not localizable.
* The values “ERROR” and “com.intel.mtwilson.widgets.ColorFault” are programming constants that can be used to select localizable text to display, but are not localizable themselves.
* The value 1234 is generated by the server and is not localizable.
* The value “a new widget” and “red” are user input and are not localizable.

For example, in the following HTML section:

<table>

<caption>widget details</caption>

<tr><th>id</th><td>1234</td></tr>

<tr><th>status</th><td>ERROR</td></tr>

<tr><th>description</th><td>a new widget</td></tr>

<tr><th>faults</th><td>1</td></tr>

</table>

<table>

<caption>widget errors</caption>

<tr><th>type</th><td>com.intel.mtwilson.widgets.ColorFault</td></tr>

<tr><th>color</th><td>red</td></tr>

</table>

* The table headings “id”, “status”, “description”, “faults”, “type”, and “color” are localizable.
* The table captions “widget details” and “widget errors” are localizable.
* The values “ERROR”, “com.intel.mtwilson.widgets.ColorFault” are programming constants that can be used to select localizable text to display.
* The value “1234” is generated by the server and is not localizable.
* The values “a new widget” and “red” are user input and is not localizable.

# Implementation

## Localizing Text in HTML

We use the i18next javascript library to localize text in the HTML user interface. The i18next library is included in mtwilson-core-html5.

Continuing the user interface example from the principles section, the following code demonstrates how to prepare the HTML for localization using i18next by adding “data-i18n” attributes to localizable HTML elements.

<table>

<caption data-i18n="label.widget\_details">widget details</caption>

<tr><th data-i18n="label.id">id</th><td>1234</td></tr>

<tr><th data-i18n="label.status">status</th><td data-i18n="status.ERROR">ERROR</td></tr>

<tr><th data-i18n="label.description">description</th><td>a new widget</td></tr>

<tr><th data-i18n="label.faults">faults</th><td>1</td></tr>

</table>

<table>

<caption data-i18n="label.widget\_errors">widget errors</caption>

<tr><th data-i18n="label.type">type</th><td data-i18n="faults.com.intel.mtwilson.widgets.ColorFault">com.intel.mtwilson.widgets.ColorFault</td></tr>

<tr><th data-i18n="label.color">color</th><td>red</td></tr>

</table>

The following translation file would be used with the above code to localize the text:

{

"label": {

"widget\_details": "Widget Details",

"widget\_errors": "Widget Errors",

"id": "Record Number",

"status": "Status",

"description": "Description",

"faults": "Errors",

"color": "Color"

},

"status": {

"ERROR": "Error",

"ACTIVE”: "Active”,

"PENDING": "Pending"

},

"faults": {

"com": {

"intel": {

"mtwilson": {

"widgets": {

"ColorFault": "Invalid Color"

}

}

}

}

}

}

## Translation Files for HTML

The main translation file for mtwilson-core-html5 has the prefix “translation”.

Other features can include their own files, such as “quickstart”, with the localized text they require. This helps to avoid naming conflicts for text across features.

For example:

src/main/html5/public/mtwilson-core-html5/locales/translation.en-US.json

src/main/html5/public/mtwilson-core-html5/locales/quickstart.en-US.json

Example content of translation.en-US.json:

{

"nav": { "title": "Cloud Integrity Technology" }

}

Example content of quickstart.en-US.json:

{

"packages": {

"attestation\_service": "Attestation Service",

"trustagent\_ubuntu": "Trust Agent",

"policy\_agent": "Policy Agent",

"director": "Trust Director",

"key\_broker": "Key Broker",

"key\_broker\_proxy": "Key Broker Proxy",

"openstack\_extensions": "OpenStack Extensions",

"openstack": "OpenStack",

"openstack\_barbican": "OpenStack Barbican",

"kmip4j": "KMIP"

},

"tasks": {

"com\_intel\_mtwilson\_deployment\_task\_RemoteInstall": "Installing $t(quickstart:packages.{{package\_name}}) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_FileTransfer": "Copying {{filename\_csv}} to {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_PreconfigureAttestationService": "Configuring $t(quickstart:packages.attestation\_service) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_CreateTrustAgentUserInAttestationService": "Creating $t(quickstart:packages.trustagent\_ubuntu) credential in $t(quickstart:packages.attestation\_service) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_CreateTrustDirectorUserInAttestationService": "Creating $t(quickstart:packages.director) credential in $t(quickstart:packages.attestation\_service) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_CreateTrustDirectorUserInKeyBroker": "Creating $t(quickstart:packages.director) credential in $t(quickstart:packages.key\_broker) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_PostconfigureAttestationService": "Finishing $t(quickstart:packages.attestation\_service) configuration on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_PostconfigureKeyBroker": "Finishing $t(quickstart:packages.key\_broker) configuration on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_PreconfigureKeyBroker": "Configuring $t(quickstart:packages.key\_broker) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_PreconfigureKeyBrokerProxy": "Configuring $t(quickstart:packages.key\_broker\_proxy) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_PreconfigureOpenstackExtensions": "Configuring $t(quickstart:packages.openstack\_extensions) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_PreconfigureTrustAgent": "Configuring $t(quickstart:packages.trustagent\_ubuntu) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_PreconfigureTrustDirector": "Configuring $t(quickstart:packages.director) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_SynchronizeSoftwarePackageTargets": "Synchronizing $t(quickstart:packages.{{package\_name}})",

"com\_intel\_mtwilson\_deployment\_task\_ImportAttestationServiceCertificatesToKeyBroker": "Importing $t(quickstart:packages.attestation\_service) certificates to $t(quickstart:packages.key\_broker)",

"com\_intel\_mtwilson\_deployment\_task\_PostconfigureOpenstack": "Finishing $t(quickstart:packages.openstack) configuration on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_CreateTrustDirectorUserInOpenstack": "Creating $t(quickstart:packages.director) credential in $t(quickstart:packages.openstack) on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_PostconfigureTrustDirector": "Finishing $t(quickstart:packages.director) configuration on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_RetrieveLinuxOperatingSystemVersion": "Detecting operating system name and version on {{host}}",

"com\_intel\_mtwilson\_deployment\_task\_RetrieveLinuxKernelVersion": "Detecting Linux kernel version on {{host}}"

},

"faults": {

"com\_intel\_mtwilson\_deployment\_jaxrs\_faults\_Connection": "Connection failed"

}

}

## Integration of i18next

The i18next.js library is included by index.html in the mtwilson-core-html5 project. A second script, “i18n\_util.js”, is also included in mtwilson-core-html5 and it defines how i18next is activated on page load. Note that when loading dynamic content, i18next is not automatically invoked (yet) on that content, so an explicit call must be made to localize dynamically loaded content.

# References

Documentation for the i18next javascript library:

<http://i18next.com/>