

Code Style Guide

<customer\_name>

v1.0, 20xx-xx-xx

# Overview

Every major project has its own style guide: a set of conventions about how to write code for that project. A large codebase is much easier to understand when all the code in it is in a consistent style.

# General

* Unused flows/sub-flows must be removed before committing to version-control.
  + Do not comment flows, sub-flows, global-declarations, transformers or components without a dated note explaining why it has not been deleted.
* All flows, sub-flows, components, transformers, scopes (of any kind, including transactions) must be given brief, descriptive names.
  + incorrect: for each (uninformative)
  + correct: for each product (immediately obvious).
    - Coupled with nearby operations or connectors that refer to a product database or API, concise names promote comprehension.
* Do not use custom acronyms - use only industry-standard acronyms unless the custom acronyms are well-known and well-understood across the business.

|  |  |
| --- | --- |
| **WARNING:** | Do not leave connectors, scopes etc. with default names (eg. Bean, Database, Logger, SMTP, Variable, Choice). The goal is to allow maximum comprehension of flow logic while minimizing the amount of drill-down needed. |

## 

## 

# Naming Conventions

## Projects

* Project names are lowercase, hyphen-separated.
* When possible, project names should include the name of the business group or owner developing the project.
* When possible, identify the type of API (experience, data, process or system) in the project name.
* If the application is not an API then drop the api designation in the name.
  + *correct*: crewmanagement-legalities-process-api, reservations-shares-system-api
  + *incorrect*: loyalty-api, pricing.project, integrationProj

## Application XML Files

* Application XML file names are lowercase, hyphen-separated.
  + *correct*: process-inbound-files.xml
  + *incorrect*: ProcessInboundFiles.xml

## Flows, Sub-Flows, & Scopes

* Flow names are lower camel-case and hyphen-separated.
* If possible, append “Flow” or “Batch” to provide clearer context.
* When flows in multiple XML files are cross-referenced, name of the flow should be prepended with the name of the flow-file containing the referenced flow.
  + *correct*: process-CSVFlow, convert-CSV2ObjectFlow, csv-to-SFDC-batch
  + *incorrect*: ProcessCSVFlow, Process-CSV

## Flow Variables

* Flow variables are lower camel-case.
  + *correct*: customerID
  + *incorrect*: OrderID, order-id

## Properties Files

* Properties files are lowercase and hyphen-separated.
* Properties files must include the environment name and .properties extension.
  + *correct*: workday-dev.properties, ns-sfdc-prod.properties
  + incorrect: MyProps, dev.properties

## Application Deployments

This is the name of the actual application project; the artifact created and edited in Anypoint Studio as well as the project name in the GIT repository. The artifact name is independent from any environment (dev, qa and prod) and should be named accordingly.

1) App name will be broken in various sections &quot;kcc&quot;-&lt;business object&gt;-&lt;function&gt; 2) All characters should be lower case

3) Use dashes to separate out the various portions of the name

4) First part of name should begin with &quot;kcc&quot;

5) Second part of name, &lt;business object&gt;, should be the business object as defined in SMART 360, this would only be the high-level business objects, it would not include any subordinates, use

dashes to separate out the words if the business object contains multiple words

6) Third part of the name, &lt;function&gt;, should be descriptive of the function provided by the API

7) Best practice would be to NOT include the application name in the &lt;function&gt;, thus the backend applications could be swapped out without concern for the API name

8) The name is to 42 characters. See https://docs.mulesoft.com/runtime-manager/deploying- to-

cloudhub

So it would look something like this:

kcc-customer-orders-api-dev

kcc-customer-orders-api-qa kcc-customer-orders-api-uat kcc-customer-orders-api (prod)

\* If the application is not an api then drop the "api".

# Application Source Files

## Encoding

* All XML files must be UTF-8 encoded and always use the following XML declaration:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> |

* Always use UNIX-style line endings.

## Formatting

The guidelines below result in optimum readability. However most IDEs have

different standards, configurability and formatting engines, so it is unlikely that

such formatting will survive regeneration, auto-formatting or auto-triggered

formatting events.

### Indentation

* XML indentation is 2 whitespaces. Ensure the IDE is configured to convert tabs to spaces. All nested tags and code within CDATA blocks must use the same indentation.
* Example:

|  |
| --- |
| <foreach>   <expression-component>  <![CDATA[ flowVars['AccountClientMap'].put(message.payload[‘ClientID’],message.payload[‘Account’]);  ]]>  </expression-component>  </foreach> |

### 

### 

### Vertical Spacing

* Separate unrelated parts of the flow vertically by one blank line to enhance readability.
  + Example:

|  |
| --- |
| <flow name="contract.GenerateContract>   <flow-ref name="util.EnrichCustomerData"/>  <!-- one blank line -->  <set-variable variableName="homeownerId" value="#[message.payload[‘SunEdCustId’]]"/>  <set-variable variableName="filename" value="#[message.payload[‘PricingQuoteId’] + '\_contract.pdf']" /> </flow> |

### Line Wrapping

* Break long tags with a single newline character and indent the second line of attributes so that it is positioned right under the first attribute of the XML element on the line above.
  + *Correct:*

|  |
| --- |
| <x12-edi:config name="GenericEDI" invalidCharacterInValueFail="false"   unknownsSegmentFail="false"  valueLengthErrorFail="false"   wrongSegmentsRepeatsFail="false"   wrongValuesRepeatsFail="false"   stringCharacterSet="UNRESTRICTED" doc:name="X12 EDI">  <x12-edi:schemas>  <x12-edi:schema>/x12/005010/834.esl</x12-edi:schema>  </x12-edi:schemas> </x12-edi:config> |

* + *Incorrect:*

|  |
| --- |
| <x12-edi:config name="GenericEDI" invalidCharacterInValueFail="false" unknownsSegmentFail="false" valueLengthErrorFail="false" wrongSegmentsRepeatsFail="false" wrongValuesRepeatsFail="false" stringCharacterSet="UNRESTRICTED" doc:name="X12 EDI">  <x12-edi:schemas>  <x12-edi:schema>/x12/005010/834.esl</x12-edi:schema>  </x12-edi:schemas> </x12-edi:config> |

## Comments

Use doc:name and doc:description attributes to clearly and concisely

document functionality.

* Example:

|  |
| --- |
| <choice doc:name="Has Correlation ID been set?"  doc:description="Determines if correlation ID is in payload."> |

## Structure

### Top-Level ‘Mule’ Tag

* The top <mule> tag contains all namespace and schema-location declarations.
* The namespaces must be declared before schema locations.
  + Each namespace declaration must start with new line and proper indentation.
  + Correct:

<mule xmlns="http://www.mulesoft.org/schema/mule/core"  
 xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc"  
 xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"   
 xmlns:spring="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="  
 http://www.mulesoft.org/schema/mule/sfdc http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd  
 http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-current.xsd  
 http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/current/mule.xsd">

* + Incorrect:

<mule xmlns="<http://www.mulesoft.org/schema/mule/core>" xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc"  
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"   
xmlns:spring="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="  
http://www.mulesoft.org/schema/mule/sfdc http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd  
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-current.xsd  
http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/current/mule.xsd">

### Global Declaration

* Global declarations include, but are not limited to:
  + connector configs
  + spring beans
  + properties placeholders
  + processing strategies
* All global declarations are to be kept in a dedicated ‘global’ configuration file.
  + Examples:
    - pricing-api-global.xml
    - sfdc-sync-global.xml
* Separate unrelated global declarations vertically by one blank line.
  + Example:

|  |
| --- |
| <secure-property-placeholder:config  name="Secure\_Property\_Placeholder"  encryptionAlgorithm="Blowfish" key="${lookup.key}"  location="processenrollments.${mule.env}.properties"/>  <!-- one blank line -->  <sfdc:config name="Salesforce\_\_Config"  username="${Salesforce\_User}"  password="${Salesforce\_Pwd}"  securityToken="${Salesforce\_SecurityToken}"  url="${Salesforce\_Url}"   disableSessionInvalidation="true">  <reconnect-forever frequency="5000"/>  </sfdc:config> |

## 

## 

## Syntax and Best Practices

### MEL Expressions for flow variables, properties, lists and maps

* Access Map or List element by the key using square brackets [ ].
  + Recommended:

|  |
| --- |
| flowVars['counter']  message.payload['userId']  message.inboundProperties['params'][0] |

* + Not recommended:

|  |
| --- |
| flowVars.counter payload.'userId'  message.inboundProperties.'params'(0) |

### Message Properties

* Use <set-property> when setting one message property.

|  |
| --- |
| <set-property propertyName="header" value="foobar"/> |

* Use <message-properties-transformer> when setting multiple message properties:

|  |
| --- |
| <message-properties-transformer>  <add-message-property key="header" value="foobar"/>  <add-message-property key="apple" value="pie"/> </message-properties-transformer> |

## Session Properties

|  |  |
| --- | --- |
| **NOTE:** | "Session" in Mule is ***not the same*** *as "session" for web-applications*. |

* Use flow variables to pass values between flows.
* When crossing transport-boundaries, pass essential data using the header supported by the transport in question, or in the payload.

|  |  |
| --- | --- |
| **WARNING:** | Do not use session properties! Session properties are serialized and added to message headers when dispatched over some transport boundaries, increasing the size of the message and consuming bandwidth. |