Naming Conventions and Containment Tree Organisation

It makes sense to apply the same naming conventions and containment tree organization to all your Builder projects. This makes reading a model much easier. The naming conventions and containment tree organizations as described below have been practice-approved by our developers.

On this Page:

Related Pages:

- Project Organization
- Naming Conventions and Containment Tree Organisation
- Model Documentation
- Settings
- Mappings
- Sub-activities
- Logging
- Error Handling

Naming Conventions

Agree on one language for names and documentation. This can be your native language or English for international projects.

Suggestion for a naming convention:

	Convention		Example
General	upper or lower camel case		`MyClass`, `myObject`
	for **signal**, **exception**s, **send** signal action, scheduler, event observer, ports, RFC ports: Use element name in the name		`Approval`**`Signal`**`, My`**`Scheduler`**`, My`**`Port`**`, `**`MyRFCPort`**
Classes	upper camel case		`MyClass`
Attributes	lower camel case		`myAttribute`
Operations	lower camel case		`myOperation`
Services	Don't use **service** in the name, except for base services, because the composite name is displayed on the Bridge.		`BaseService_Salesforce_01`
	Use a project specific pattern.		`Intelligix_Outbound_01`
Aliases	upper camel case		`SQLAlias`
Model File Names	upper camel case		`MyModel.xml`
	libraries:	prefixes with lib	`libFTP.xml`
	BPMN:	upper camel case with BPMN information	`MyBPMN`**`Specification`**`.xml` or `My`**`BPMN`**`.xml`
	BPMN:	corresponding implementation model	`My.xml`

Containment Tree Organization

Suggestion for a containment tree organization:

How to organize ...	
modules	Use a dedicated package **Modules**. Use sub packages for XSD, XSLTs, WSDLs, ...
libraries	Use a dedicated package **Libraries.**
scheduler, SOAP ports, other frontend interfaces	Use a dedicated package **Service**.
class diagrams containing mappings	Use a dedicated package **Mappings**.
class diagrams for flat files	Use a dedicated package **Flat Files**.
manually created persistent state objects	Use a dedicated package **Persistency**.

Hint concerning the BPMN importer

The BPMN importer derives the module name in the implementation model from the name of the package where the BPMN diagram resides in the BPMN specification model. Thus, create the BPMN diagram in a package whose name is unique throughout the whole Builder project. Otherwise packages having the same name might get overwritten on BPMN import.