

# The type-v1.0 DTD

## Table of contents

1 types.....	3
2 type.....	3
3 doc.....	4
4 supertype.....	4
5 typereference.....	4
6 command-settings.....	4
7 dependency-view.....	5
8 attributes.....	5
9 attribute.....	6
10 attribute-default.....	6
11 constraints.....	6
12 allowedvalue-constraint.....	7
13 allowedvalue.....	7
14 dependency-constraint.....	7
15 allowedtypes.....	8
16 singletontypes.....	8
17 commands.....	8
18 command.....	8
19 execution-string.....	9
20 argument-string.....	9
21 implementation.....	9
22 script.....	10
23 workflow.....	10
24 dispatch-command.....	11

24.1 include.....	12
24.2 select-dependencies.....	12
24.3 select-deployments.....	12
24.4 select-objects.....	13
25 error-handler.....	13
26 opts.....	14
27 opt.....	14
28 Base type RDF properties.....	15

**Note:**

This is a demonstration document using all possible elements in the current ControlTier type-v10.dtd.

## 1. types

The root (aka "top-level") element of the type.xml file. Contains a set of [type](#) elements. Any kind of type can be specified in any order.

```
<types>
  <type ...>
</types>
```

## 2. type

The type element defines a Type and command module. If the "order" attribute is "Setting" or "Assembly", then the elements required to define the command Module are not required.

attribute	description	values
name	Name of the type	
role	Specifies the role of the type. Concrete means it can be instantiated as objects. Abstract specifies it cannot be instantiated.	"concrete" or "abstract"
order	Name of a well known base type.	Setting, Assembly, Deployment, Service, Mediator
uniqueInstances	A boolean signifying if each instance of the type must have a unique name. "false" is default.	true or false

**Table 1: attributes**

element	description
description	Text content describing the type
<a href="#">supertype</a>	Reference to the parent type
<a href="#">command-settings</a>	Properties of the command module

**Table 2: elements**

```

<type
  name="TypeName"
  role="concrete"
  order="TypeName"
  uniqueInstances="true">

  <description>describe the type</description>

  <supertype>
    <typereference name="TypeName" />
  </supertype>

  ...
</type>

```

### 3. doc

A documentation block. Can contain any valid [Apache Forrest xdoc](#) tag. The doc tags can be used just about anywhere in the type.xml. When [ProjectBuilder's generate-forrest-docs](#) command is run, the doc tags are expanded into the generated forrest xdocs.

### 4. supertype

A sub-element of [type](#), supertype specifies the parent type of this type. A `typereference` tag is used to specify the supertype by name.

element	description
<a href="#">typereference</a>	Text content describing the type

**Table 1: elements**

### 5. typereference

Specifies a type.

attribute	description	values
name	Name of the type	Any existing type.

**Table 1: attributes**

### 6. command-settings

Properties of the command module

elements	description	values
notification	Advisory setting stating if notifications are enabled. Specify the <code>notify</code> attribute with a boolean value.	"true" or "false"
template-directory	path to template directory	Any valid path. By convention it is <code>\${module.dir}/templates</code>
<a href="#">dependency-view</a>	Parameters to the Get-Properties command	
logger	Name used by log calls	By convention, the type name

**Table 1: elements**

```
<command-settings>
  <notification notify="false" />
  <template-directory></template-directory>
  <dependency-view parents="false" children="true" proximity="1" />
  <logger name="TypeName" />
</command-settings>
```

## 7. dependency-view

Parameters used as defaults for the Get-Properties command options

attribute	description	values
parents	Include parent dependency info	"true" or "false"
children	Include children dependency info	"true" or "false"
proximity	Number of levels away from the source object	1, 2, 3, 4

**Table 1: attributes**

## 8. attributes

define attributes of the Type here. type-property values are the names of properties in the depo: namespace

elements	description
<a href="#">attribute</a>	

<a href="#">attribute-default</a>	These will be generated into type.properties
-----------------------------------	--

**Table 1: elements**

```
<attributes>
  <attribute name="attr_name" type-property="property-name" />
  <attribute-default name="attr_name" value="VALUE" />
</attributes>
```

## 9. attribute

attribute	description	values
name	Attribute's name	
type-property	Associated <a href="#">RDF property</a>	

**Table 1: attributes**

## 10. attribute-default

attribute	description	values
name	Attribute's name	
value	Attribute's default value.	

**Table 1: attributes**

## 11. constraints

A sub-element of [type](#). Type constraints are used to control the allowed kinds of dependencies (both parent and child). Also controls the allowed values for its RDF properties.

elements	description
<a href="#">allowedvalue-constraint</a>	Contains any number of <a href="#">allowedvalue</a> elements
<a href="#">dependency-constraint</a>	Specifies <a href="#">allowedtypes</a> and <a href="#">singletontypes</a>

**Table 1: elements**

```
<constraints>
  <allowedvalue-constraint
    enforced="true/false"
    type-property="depo-property-name">
    <allowedvalue value="<SOME VALUE>" default="true/false" />
    <allowedvalue value="<SOME VALUE>" default="true/false" />
  </allowedvalue-constraint>
```

```

    <dependency-constraint kind="parent/child"
enforced="true/false">
      <allowedtypes>
        <typereference name="<TYPE NAME>" />
        <typereference name="<TYPE NAME>" />
      </allowedtypes>
      <singletontypes>
        <typereference name="<TYPE NAME>" />
        <typereference name="<TYPE NAME>" />
      </singletontypes>
    </dependency-constraint>
  </constraints>

```

## 12. allowedvalue-constraint

Contains any number of allowed values.

attribute	description	values
enforced	Specifies if the constraint should be enforced	"true" or "false"
type-property	Name of RDF property.	

**Table 1: attributes**

elements	description
<a href="#">allowedvalue</a>	

**Table 2: elements**

## 13. allowedvalue

An allowed value.

attribute	description	values
value	An allowed value	Any string
default	Specifies if it should be the default value.	"true" or "false"

**Table 1: attributes**

## 14. dependency-constraint

attribute	description	values
kind	Kind of dependency.	"parent" or "child"

enforced	Specifies if the constraint should be enforced	"true" or "false"
----------	--	-------------------

**Table 1: attributes**

## 15. allowedtypes

elements	description
<a href="#">typereference</a>	

**Table 1: elements**

## 16. singletontypes

elements	description
<a href="#">typereference</a>	

**Table 1: elements**

## 17. commands

A sub-element of [type](#), the `commands` element contains the command definitions for this type.

```

<commands>
  <command name="<ANT COMMAND NAME>"
    description="<ANT COMMAND DESCRIPTION>"
command-type=" { Command | AntCommand | BsfCommand | WorkflowCommand } ">
    <opts>
      <opt parameter="<OPT NAME>"
        description="<OPT DESCRIPTION>"
        required="{true|false}"
        property="opts.<OPT NAME>"
        type="{string|boolean}"
        default="<DEFAULT STRING VALUE>" />
    </opts>
  </command>
</commands>

```

## 18. command

Defines a command for the type.

attribute	description	values
name	The	Any string

	command's name				
description	Briefly describe the command purpose.	Any string			
command-type	The type of command implementation	"Command", "BsfCommand", "AntCommand", "WorkflowCommand"			
is-static	A static command is one that can run outside of an object context	"true" or "false"	daemonized	Should the executable be launched in daemon mode?	"true" or "false"

**Table 1: attributes**

elements	description
<a href="#">opts</a>	Specifies the commands options
<a href="#">execution-string</a>	Specifies the file to execute
<a href="#">argument-string</a>	Specifies the arguments
<a href="#">implementation</a>	Specifies the AntCommand implementation
<a href="#">script</a>	Specifies the BsfCommand script
<a href="#">workflow</a>	Specifies command sequence

**Table 2: elements**

## 19. execution-string

File to execute

## 20. argument-string

Arguments to pass to executable. If the execution-string specifies an interpreter that can read script from stdin, the value of argument-string can be script code.

## 21. implementation

Ant tasks to execute.

## 22. script

BSF script to execute.

## 23. workflow

Command sequence to execute. Takes one more more command elements.

attribute	description	values
threadcount	Number of threads to execute the workflow	Any integer greater than 0

**Table 1: attributes**

elements	description
command	Name of the command. Command must be in the same module. <code>&lt;command name=" &lt;COMMAND NAME&gt; " /&gt;</code>

**Table 2: elements**

```

<command name="<WORKFLOW COMMAND NAME>"
  description="<WORKFLOW COMMAND DESCRIPTION>"
  command-type="WorkflowCommand"
  error-handler-type="handler/ignore/fail"
>
  <error-handler quiet="true/false">
    <prompt>PROMPT MESSAGE</prompt>
    <command name="<ERROR HANDLER COMMAND NAME>" />
    <notify email="NAME@HOST">
      <subject>SUBJECT</subject>
      <message-file>PATH</message-file>
    </notify>
    <report>REPORT MESSAGE</report>
  </error-handler>
  <success-handler notify-email="NAME@HOST">
    <notify-subject>SUBJECT</notify-subject>
    <notify-message-file>PATH</notify-message-file>
  </success-handler>
  <workflow threadcount="<INTEGER>">
    <command name="<COMMAND NAME>" />
    <command name="<COMMAND NAME>" />
  </workflow>
</command>

```

## 24. dispatch-command

Dispatch commands to the selected contexts. Takes one more more command elements.

attribute	description	values
name	Workflow name	defaults to command.name
requirematch	Requires one or more matches	true (default)
strategy	Execution strategy	"localdispatch" or "nodedispatch"
threadcount	Number of threads to execute the workflow	Any integer greater than 0

**Table 1: attributes**

elements	description
<a href="#">arg</a>	The argument line.
<a href="#">command</a>	The command to dispatch.
<a href="#">contexts</a>	The object contexts to execute the command.

**Table 2: elements**

```

<command name="name"
  description="description"
  command-type="DispatchCommand"
  error-handler-type="HANDLER"
  >
  <error-handler quiet="false">
    ...
  </error-handler>

  <dispatch-command threadcount="1"
    requirematch=true"
  >
    <command name="dispatchedCommand"/>
    <arg line="argLine"/>
    <contexts>
      <select-deployments sortkey="name" sortorder="ascending"
depot="{context.depot}">
        <include name="*" type="[^\.]*/>
      </select-deployments>
    </contexts>
  </dispatch-command>
</command>

```

## 24.1. include

Object include filter

attribute	description	values
name	Object name	regex
type	Object type	regex

**Table 1: attributes**

## 24.2. select-dependencies

Object include filter based on object dependencies declared in properties

attribute	description	values
relationtype	Object's base type	regex
sortkey	Object property used as sort key	
sortorder	Sort order	"descending" or "ascending" (default)
source	Data source to read property data	"file" or "context" (default)

**Table 1: attributes**

```
<select-dependencies sortkey="startup-rank"
                    sortorder="descending"
                    relationtype="deployment" source="context">
  <include name="me[12]" type="Managed-.*"/>
</select-dependencies>
```

## 24.3. select-deployments

Looks up dispatch targets from the deployments.properties file

attribute	description	values
depot	project depot name	an existing project
sortkey	Object property used as sort key	
sortorder	Sort order	"descending" or "ascending"

		(default)
--	--	-----------

**Table 1: attributes**

```
<select-deployments sortkey="startup-rank"
  sortorder="descending"
  depot="{context.depot}">
  <include name="me[12]" type="Managed-.*"/>
</select-deployments>
```

## 24.4. select-objects

Looks up dispatch targets from the specified depot

attribute	description	values
depot	project depot name	an existing project
sortkey	Object property used as sort key	
sortorder	Sort order	"descending" or "ascending" (default)
source	Data source to read property data	"file" or "context" (default)

**Table 1: attributes**

```
<select-objects sortkey="startup-rank"
  sortorder="descending"
  depot="{context.depot}">
  <include name="me[12]" type="Managed-.*"/>
</select-objects>
```

## 25. error-handler

Handles an error if one occurs

attribute	description	values
quiet	Log a message if an error is caught?	"true" or "false"
errorproperty	Name of property to set error message	string

**Table 1: attributes**

elements	description
prompt	Prompt user if an error occurs
command	Run the specified command. Must be in the same module.
notify	Send an email.
report	Call the <code>report</code> task with the error message.

**Table 2: elements****Example**

```
<error-handler quiet="true/false">
  <prompt>PROMPT MESSAGE</prompt>
  <command name="<ERROR HANDLER COMMAND NAME>" />
  <notify email="NAME@HOST">
    <subject>SUBJECT</subject>
    <message-file>PATH</message-file>
  </notify>
  <report>REPORT MESSAGE</report>
</error-handler>
```

**26. opts**

Specifies the commands options

elements	description
<a href="#">opt</a>	

**Table 1: elements**

```
<opts>
  <opt parameter="<OPT NAME>"
    description="<OPT DESCRIPTION>"
    required="{true|false}"
    property="opts.<OPT NAME>"
    type="{string|boolean}"
    default="<DEFAULT STRING VALUE>"
    defaultproperty="<PROPERTY NAME>"
  />
</opts>
```

**27. opt**

Specifies one command option.

attribute	description	values
parameter	The option's name	Any string
description	Briefly describe the option's purpose.	Any string
required	Briefly describe the option's purpose.	"true" or "false"
property	Name to set the option value	By convention it is <i>opts.parameter</i>
type	Type of parameter. A string option is one that takes an argument. A boolean option does not.	"string" or "boolean"
default	Literal value to use if the option is not specified.	Any string
defaultproperty	Property name to use if the option is not specified.	Any property that may exist at execution time.

**Table 1: attributes**

## 28. Base type RDF properties

Attribute constraints can control the allowed values and defaults for the RDF properties of base types. The following is a listing of ones that can have constraints applied.

name	description	domain
deployment-basedir	Deployment base directory	Deployment
deployment-install-root	Deployment installation root directory	Deployment
deployment-startup-rank	Deployment startup rank	Deployment
os-arch	Operating system architecture	Node
os-family	Operating system family (windows, unix)	Node
os-name	Operating system name	Node
os-version	Operating system version	Node
package-arch	Package architecture	Package

package-base	Package base name	Package
package-buildtime	Package build time	Package
package-filename	Package file name	Package
package-filetype	Package file type	Package
package-install-rank	Package install rank	Package
package-install-root	Package install directory	Package
package-release	Package release identifier	Package
package-release-tag	Package release tag	Package
package-repo-url	URL to the package in the repo	Package
package-restart	Boolean flag specifying if a restart is required (true,false)	Package
package-version	Package version	Package
settingValue	Holds the setting value	Setting
settingType	User defined type definition of the setting value	Setting