

Errorhandler

1. Description

The Errorhandler data type represents the set of actions that should be executed if a Workflow command fails.

2. Attributes

Attribute	Description	Required
quiet	Specifies if errorhandler should operate quietly. If this is set true, then messages regarding caught build exceptions will not be logged.	No. Defaults to false.

3. Parameters specified as nested elements

command

A [command type](#) describes what command name to run. Only the command's name is a required attribute. The other attributes are ignored.

The example below shows how to specify to run a command named Recover

```
<command name="Recover" />
```

Note:

This tag is essentially a short hand for using the controller task, the specified command assumed to be in the same module

email

An email element describes what command name to run. Only the command's name is a required attribute. The other attributes are ignored.

The example below shows how to specify to email a message to admin

```
<email to="admin@domain.com" from="user@domain.com"  
subject="workflow encountered an error" />
```

prompt

A prompt element describes that the command should take input from the user before proceeding. The message's string is a required attribute.

The example below shows how to prompt a user with a message

```
<prompt message="Continue? " />
```

task

The errorhandler is also a TaskContainer and therefore, any task (or sequence) can be called.

The example below shows how to prompt a user with a message

```
<echo message="invoking defibrillator " />
```

4. Examples

Shows errorhandler data type used in the [workflow](#) element.

```
<workflow name="Restart">
  <errorhandler>
    <email to="admin@domain.com" from="user@domain.com"
      message="Restart failed"/>
    <prompt message="Continue with Recover command?"/>
    <command name="Recover"/>
  </errorhandler>
  <tasksequence>
    <echo>running Stop command</echo>
    <controller updateproperties="false">
      <execute>
        <context depot="{depot.name}"
          entityClass="{context.type}"
          entityName="{context.name}"/>
        <command name="Stop"/>
      </execute>
    </controller>
    <echo>running Start command</echo>
    <controller updateproperties="false">
      <execute>
        <context depot="{depot.name}"
          entityClass="{context.type}"
          entityName="{context.name}"/>
        <command name="Start"/>
      </execute>
    </controller>
  </tasksequence>
</workflow>
```

This example shows a workflow called Start, that first checks to see if the service is already running and if the check fails, to call the startService command.

```
<workflow name="Start">
```

Errorhandler

```
<errorhandler quiet="true">
  <command name="startService"/>
</errorhandler>
<tasksequence>
  <echo>running upService command</echo>
  <controller updateproperties="false">
    <execute>
      <context depot="{depot.name}"
              entityClass="{context.type}"
              entityName="{context.name}"/>
      <command name="upService"/>
    </execute>
  </controller>
</tasksequence>
</workflow>
```

This example shows a use of errorhandler in the [apply-macro](#) task.

```
<apply-macro name="doAction">
  <errorhandler quiet="true">
    <fail/>
  </errorhandler>
  <queryresults refid="deployments.query"/>
</apply-macro>
```