IceGrid Property Set Semantics

Ice servers and clients are configured with properties. For servers deployed with IceGrid, these properties are automatically generated into a configuration file from the information contained in the application descriptor. The settings in that configuration file are passed to server via the `--Ice.Config` command-line option.

Property descriptors allow you to define property sets to efficiently manage and specify properties. Here are some of the benefits of using property sets:

- You can define sets of properties at the application or node element level and reference these properties in other property sets.
- You can specify properties for a specific server or service instance.

There are two kinds of property sets:

- Named property sets
  Named property sets are defined at the application or node level. They are useful only as the target of references from other property sets. Specifically, a named property set has no effect unless you reference it from a server descriptor.

- Unnamed property sets
  Unnamed property sets can be defined in server, service, icebox, server-instance or service-instance elements and define the properties for a server or service. Unnamed property sets can reference named property sets.

Named and unnamed property sets are defined with the same properties descriptor. The context and the attributes of a `properties` element distinguish named property sets from unnamed property sets. Here is an example that defines a named and an unnamed property set:

XML

```
<application name="App">
    <properties id="Debug">
        <property name="UseDebug" value="1"/>
    </properties>

    <node name="TheNode">
        <server id="TheServer" exe="./server">
            <properties>
                <property name="Identity" value="hello"/>
            </properties>
        </server>
    </node>
</application>
```

In this example, we define the named property set `Debug` and the unnamed property set of the server `TheServer`. The server configuration will contain only the `Identity` property because the server property set does not reference the `Debug` named property set.

The `properties` element is used to reference a named property set: if a `properties` element appears inside another `properties` element, it is a reference to another property set and it must specify the `refid` attribute. With the previous example, to reference the `Debug` property set, we would write the following:

XML

```
<application name="App">
    <properties id="Debug">
        <property name="UseDebug" value="1"/>
    </properties>

    <node name="TheNode">
        <server id="TheServer" exe="./server">
            <properties>
                <properties refid="Debug"/>
                <property name="Identity" value="hello"/>
        </properties>
        </server>
    </node>
</application>
```

Property sets, whether named or unnamed, are evaluated as follows:

1. Within a `properties` element, IceGrid locates all references to named property sets and evaluates all property settings in the referenced property sets.
2. Explicit property definitions following any named references are then evaluated and added to the property set formed in the preceding step. This means that explicit property settings override corresponding settings in any referenced property sets.

It is illegal to define a reference to a property set after setting a property value, so references to property sets must precede property definitions. For example, the following is illegal:

XML

```
<properties>
    <property name="Prop1" value="Value1"/>
    <properties refid="Ref1"/>
</properties>
```

Just as the order of the property definitions is important, the order of property set references is also important. For example, the following two property sets are not equivalent:

XML

```
<properties>
    <properties refid="Ref1"/>
    <properties refid="Ref2"/>
</properties>

<properties>
    <properties refid="Ref2"/>
    <properties refid="Ref1"/>
</properties>
```

Named property sets are evaluated at the point of definition. If you reference other property sets or use variables in a named property set definition, you must make sure that the referenced property sets or variables are defined in the same scope. For example, the following is correct:

XML

```
<application name="App">

    <variable name="level" value="1"/>

    <properties id="DebugApp">
        <property name="DebugLevel value="${level}">
    </properties>

</application>
```

However, the following example is wrong because the `${level}` variable is not defined at the `application` scope:

XML

```
<application name="App">

    <properties id="DebugApp">
        <property name="DebugLevel value="${level}">
    </properties>

    <node name="TheNode">
        <variable name="level" value="1"/>
    </node>

</application>
```

If both the `application` and the `node` define the `${level}` variable, the value of the `${level}` variable in the `DebugApp` property set will be the value of the variable defined in the application descriptor.

So far, we have seen the definition of an unnamed property set only in a server descriptor. However, it is also possible to define an unnamed property set for server or service instances. This is a good way to specify or override properties specific to a server or service instance. For example:

XML

```
<application name="TheApp">
  <server-template id="Template">

    <parameter name="instance-name"/>

    <server id="${instance-name}" exe="./server">
      <properties>
        <property name="Timeout" value="30"/>
      </properties>
    </server>
  </server-template>

  <node name="TheNode">
    <server-instance template="Template" instance-name="MyInst">
      <properties>
        <property name="Debug" value="1"/>
        <property name="Timeout" value="-1"/>
      </properties>
    </server-instance>
  </node>
</application>
```

Here, the server instance overrides the `Timeout` property and defines an additional `Debug` property.

The server or service instance properties are evaluated as follows:

1. The unnamed property set from the template server or service descriptor is evaluated.
2. The unnamed property set from the server or service instance descriptor is evaluated and the resulting properties are added to the property set formed in the preceding step. This means that property settings in a server or service instance descriptor override corresponding settings in a template server or service descriptor.

The server or service instance unnamed property set and its parameters provide two different ways to customize the properties of a server or service template instance. It might not always be obvious which method to use: is it better to use a parameter to parameterize a given property or is it better to just specify it in the server or service instance property set?

For example, in the previous descriptor, we could have used a parameter with a default value for the `Timeout` property:

XML

```
<application name="TheApp">
  <server-template id="Template">

    <parameter name="instance-name"/>
    <parameter name="timeout" default="30"/>

    <server id="${instance-name}" exe="./server">
      <properties>
        <property name="Timeout" value="${timeout}"/>
      </properties>
    </server>
  </server-template>

  <node name="TheNode">
    <server-instance template="Template" instance-name="MyInst" timeout="-1">
      <properties>
        <property name="Debug" value="1"/>
      </properties>
    </server-instance>
  </node>
</application>
```

Here are some guidelines to help you decide whether to use a parameter or a property:

- Use a parameter for a property that should always be set.
- Use a parameter if you want to make the property obvious to the reader and user of the template.
- Do not use a parameter for optional properties if you want to rely on a default value for the server.
- Do not use parameters for properties that are rarely used.

See Also

- Properties and Configuration
- Using IceGrid Deployment
- Properties Descriptor Element
- Application Descriptor Element
- Node Descriptor Element
- Server-Instance Descriptor Element
- Service-Instance Descriptor Element