FreezeScript Inspection XML Reference

This page describes the XML elements comprising the FreezeScript inspection descriptors.

On this page:

<dumpdb> Descriptor Element

The top-level descriptor in a descriptor file. It requires one child descriptor, <database>, and supports any number of <dump> descriptors. This descriptor has no attributes.

<database> Descriptor Element

The attributes of this descriptor define the key and value types of the database. It supports any number of child descriptors, but at most one <record> descriptor. The <database> descriptor also creates a global scope for user-defined symbols.

The attributes supported by the <database> descriptor are described in the following table:

Name	Description
key	Specifies the Slice type of the database key.
value	Specifies the Slice type of the database value.

As an example, consider the following <database> descriptor. In this case, the Freeze map to be examined has key type int and value type ::Employee:

XML

```
<database key="int" value="::Employee">
```

<record> Descriptor Element

Commences the database traversal. Child descriptors are executed for each record in the database, but after any <dump> descriptors are executed. The <record> descriptor introduces the read-only symbols key, value and facet into a local scope. These symbols are accessible to child descriptors, but not to <dump> descriptors. The facet symbol is a string indicating the facet name of the object in the current record, and is only relevant for Freeze evictor databases.

Note that database traversal only occurs if a <record> descriptor is present.

<dump> Descriptor Element

Executed for all instances of a Slice type. Only one <dump> descriptor can be specified for a type, but a <dump> descriptor is not required for every type. The read-only symbol value is introduced into a local scope. The attributes supported by this descriptor are described in the following table:

Name	Description

`type`	Specifies the Slice type ID.
`base`	If `type` denotes a Slice class, this attribute determines whether the <dump> descriptor of the base class is invoked. If `true`, the base class descriptor is invoked after executing the child descriptors. If not specified, the default value is `true`.
`contents`	For `class` and `struct` types, this attribute determines whether descriptors are executed for members of the value. For `sequence` and `dictionary` types, this attribute determines whether descriptors are executed for elements. If not specified, the default value is `true`.

Below is an example of a <dump> descriptor that searches for certain products:

XML

```
<dump type="::Product">
    <if test="value.description.find('scanner') != -1">
        <echo message="Scanner SKU: " value="value.SKU"/>
    </if>
</dump>
```

For class types, `dumpdb` first attempts to locate a <dump> descriptor for the object's most-derived type. If no descriptor is found, `dumpdb` proceeds up the class hierarchy in an attempt to find a descriptor. The base object type, `Object`, is the root of every class hierarchy and is included in the search for descriptors. It is therefore possible to define a <dump> descriptor for type `Object`, which will be invoked for every class instance.

Note that <dump> descriptors are executed recursively. For example, consider the following Slice definitions:

Slice

```
struct Inner {
    int sum;
};
struct Outer {
    Inner i;
};
```

When `dumpdb` is interpreting a value of type `Outer`, it executes the <dump> descriptor for `Outer`, then recursively executes the <dump> descriptor for the `Inner` member, but only if the `contents` attribute of the `Outer` descriptor has the value `true`.

<iterate> Descriptor Element

Iterates over a dictionary or sequence, executing child descriptors for each element. The symbol names selected to represent the element information may conflict with existing symbols in the enclosing scope, in which case those outer symbols are not accessible to child descriptors. The attributes supported by this descriptor are described in the following table:

Name	Description
`target`	The sequence or dictionary.
`index`	The symbol name used for the sequence index. If not specified, the default symbol is `i`.
`element`	The symbol name used for the sequence element. If not specified, the default symbol is `elem`.
`key`	The symbol name used for the dictionary key. If not specified, the default symbol is `key`.
`value`	The symbol name used for the dictionary value. If not specified, the default symbol is `value`.

Shown below is an example of an <iterate> descriptor that displays the name of an employee if the employee's salary is greater than $3000.

XML

```
<iterate target="value.employeeMap" key="id" value="emp">
    <if test="emp.salary > 3000">
        <echo message="Employee: " value="emp.name"/>
    </if>
</iterate>
```

<if> Descriptor Element

Conditionally executes child descriptors. The attributes supported by this descriptor are described in the following table:

Name	Description
test	A boolean expression.

Child descriptors are executed if the expression in test evaluates to true.

<set> Descriptor Element

Modifies a value. The value and type attributes are mutually exclusive. If target denotes a dictionary element, that element must already exist (i. e., <set> cannot be used to add an element to a dictionary). The attributes supported by this descriptor are described in the following table:

Name	Description
target	An expression that must select a modifiable value.
value	An expression that must evaluate to a value compatible with the target's type.
type	The Slice type ID of a class to be instantiated. The class must be compatible with the target's type.
length	An integer expression representing the desired new length of a sequence. If the new length is less than the current size of the sequence, elements are removed from the end of the sequence. If the new length is greater than the current size, new elements are added to the end of the sequence. If value or type is also specified, it is used to initialize each new element.
convert	If true, additional type conversions are supported: between integer and floating point, and between integer and enumeration. Transformation fails immediately if a range error occurs. If not specified, the default value is false.

The <set> descriptor below modifies a member of a dictionary element:

XML

```
<set target="new.parts['P105J3'].cost" value="new.parts['P105J3'].cost * 1.05"/>
```

This <set> descriptor adds an element to a sequence and initializes its value:

XML

```
<set target="new.partsList" length="new.partsList.length + 1" value="'P105J3'"/>
```

<add> Descriptor Element

Adds a new element to a sequence or dictionary. It is legal to add an element while traversing the sequence or dictionary using <iterate>, however the traversal order after the addition is undefined. The key and index attributes are mutually exclusive, as are the value and type attributes. If neither value nor type is specified, the new element is initialized with a default value. The attributes supported by this descriptor are described in the following table:

Name	Description
target	An expression that must select a modifiable sequence or dictionary.
key	An expression that must evaluate to a value compatible with the target dictionary's key type.
index	An expression that must evaluate to an integer value representing the insertion position. The new element is inserted before index. The value must not exceed the length of the target sequence.
value	An expression that must evaluate to a value compatible with the target dictionary's value type, or the target sequence's element type.
type	The Slice type ID of a class to be instantiated. The class must be compatible with the target dictionary's value type, or the target sequence's element type.

`convert`	If `true`, additional type conversions are supported: between integer and floating point, and between integer and enumeration. Transformation fails immediately if a range error occurs. If not specified, the default value is `false`.

Below is an example of an `<add>` descriptor that adds a new dictionary element and then initializes its member:

XML

```
<add target="new.parts" key="'P105J4'"/>
<set target="new.parts['P105J4'].cost" value="3.15"/>
```

`<define>` Descriptor Element

Defines a new symbol in the current scope. The attributes supported by this descriptor are described in the following table:

Name	Description
`name`	The name of the new symbol. An error occurs if the name matches an existing symbol in the current scope.
`type`	The name of the symbol's formal Slice type.
`value`	An expression that must evaluate to a value compatible with the symbol's type.
`convert`	If `true`, additional type conversions are supported: between integer and floating point, and between integer and enumeration. Execution fails immediately if a range error occurs. If not specified, the default value is `false`.

Below are two examples of the `<define>` descriptor. The first example defines the symbol `identity` to have type `Ice::Identity`, and proceeds to initialize its members using `<set>`:

XML

```
<define name="identity" type="::Ice::Identity"/>
<set target="identity.name" value="steve"/>
<set target="identity.category" value="Admin"/>
```

The second example uses the enumeration we first saw in our discussion of custom database migration to define the symbol `manufacturer` and assign it a default value:

XML

```
<define name="manufacturer" type="::BigThree" value="::DaimlerChrysler"/>
```

`<remove>` Descriptor Element

Removes an element from a sequence or dictionary. It is legal to remove an element while traversing a sequence or dictionary using `<iterate>`, however the traversal order after removal is undefined. The attributes supported by this descriptor are described in the following table:

Name	Description
`target`	An expression that must select a modifiable sequence or dictionary.
`key`	An expression that must evaluate to a value compatible with the key type of the target dictionary.
`index`	An expression that must evaluate to an integer value representing the index of the sequence element to be removed.

`<fail>` Descriptor Element

Causes transformation to fail immediately. If `test` is specified, transformation fails only if the expression evaluates to `true`. The attributes supported by this descriptor are described in the following table:

Name	Description
`message`	A message to display upon transformation failure.
`test`	A boolean expression.

The following `<fail>` descriptor terminates the transformation if a range error is detected:

XML

```
<fail message="range error occurred in ticket count!" test="value.ticketCount > 32767"/>
```

<echo> Descriptor Element

Displays values and informational messages. If no attributes are specified, only a newline is printed. The attributes supported by this descriptor are described in the following table:

Name	Description
`message`	A message to display.
`value`	An expression. The value of the expression is displayed in a structured format.

Shown below is an `<echo>` descriptor that uses both `message` and `value` attributes:

XML

```
<if test="value.ticketCount > 32767">
    <echo message="range error occurred in ticket count: " value="value.ticketCount"/>
</if>
```

See Also

- Freeze Maps
- Freeze Evictors
- Facets and Versioning
- Custom Database Migration
- FreezeScript Descriptor Expression Language