

## Chapter 9

# Water Export: Converting an Object to a String

### IN THIS CHAPTER

- ◆ Exporting a Water object to XML formats
- ◆ Exporting a Water object to CSV and plain text formats
- ◆ Creating a simple HTML report

EXPORTING DATA IS A REQUIREMENT for many applications. Water Export is a collection of methods that convert a Water object into a formatted string. Every method has a name that starts with `to_`. The list of Water Export methods are:

- ◆ `to_xml`: Water object converted to a string that is a valid XML 1.0 string
- ◆ `to_concise_xml`: Returns a string in ConciseXML format
- ◆ `to_html`: Returns an XHTML string ready for browser viewing
- ◆ `to_plain_text`: Returns a plain text string that has no markup
- ◆ `to_csv`: Returns a string in CSV format

## Converting an Object to an XML string

The following example defines a `person` class and creates a few instances in a vector. The variable `a_data` that holds the instances will be used in all the following examples.

```
<defclass person
  first_name
  last_name
  employee_number=optional
  last_updated=optional
/>
```

```

<set a_data=
  <vector
    <person first_name="Mike"
      last_name="Plusch"
      employee_number=23
      last_updated=<date 2002 10 2/>
    />
    <person first_name="Christopher"
      last_name="Fry"
      employee_number=19
    />
  />
/>

```

## Converting an Object to XML 1.0

ConciseXML supports a backward-compatible XML 1.0 format as well as the standard concise format. When formatting an object as XML 1.0, any field of a Water object containing a string will be shown as an XML 1.0 attribute. Fields containing non-string values, as well as strings that contain a linefeed, will put the fields within the `attributes` element. `employee_number`, for example, is an integer and, therefore, the field is put in the `attributes` element.

```
a_data.<to_xml/>
```

### Output in XML 1.0:

```

<vector>
  <person last_name="Plusch" first_name="Mike">
    <attributes>employee_number=23
      last_updated=<date>2002-10-2</date>
    </attributes>
  </person>
  <person last_name="Fry" first_name="Christopher">
    <attributes>employee_number=19 </attributes>
  </person>
</vector>

```

ConciseXML supports an XML 1.0 format as well as the concise format. Anything in the XML 1.0 format can be converted into the concise format and vice-versa. Both formats can be interchanged at any level without any loss of information.

A Water object can be formatted in other XML 1.0 formats as well, but only the XML 1.0 format of ConciseXML will preserve all the information in the object. The following example converts `a_data` into an XML 1.0 compatible string using the format `xml_format.attr_name`. Fields are represented using an `attr` element. The field key is the value of the `name` attribute of `attr`. The field value is put in the content argument of `attr`.

```
a_data.<to_ekdata xml_format.attr_name/>.<to_xml/>
```

**Output:**

```
<vector>
  <person>
    <attr name='name'>Mike</attr>
    <attr name='last_name'>Plusch</attr>
    <attr name='employee_number'>23</attr>
    <attr name='last_updated'><date>2002-10-2</date></attr>
  </person>
  <person>
    <attr name='name'>Christopher</attr>
    <attr name='last_name'>Fry</attr>
    <attr name='employee_number'>19</attr>
  </person>
</vector>
```

The following example shows an instance of `ejb-jar` as a `Water` object in `ConciseXML` format and the conversion to an XML 1.0 format called `xml_format.value_and_part_may_be_merged`.

```
<ejb-jar
  enterprise-beans=
    <session
      ejb-name="Hello"
      home="org.acme.HelloHome"
      remote="org.acme.HelloObject"
      ejb-class="org.acme.HelloBean"
      session-type="Stateless"
      transaction-type="Container"
    />
  assembly-descriptor=
    <container-transaction
      method=<method ejb-name="Hello" method-name="*" />
      trans-attribute="Required"
    />
/>.<to_ekdata xml_format.value_and_part_may_be_merged/>
```

**Output:**

```
<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>Hello</ejb-name>
      <home>org.acme.HelloHome</home>
      <remote>org.acme.HelloObject</remote>
      <ejb-class>org.acme.HelloBean</ejb-class>
```

```

        <session-type>Stateless</session-type>
        <transaction-type>Container</transaction-type>
    </session>
</enterprise-beans>
<assembly-descriptor>
    <container-transaction>
        <method>
            <method>
                <ejb-name>Hello</ejb-name>
                <method-name>*</method-name>
            </method>
        </method>
        <trans-attribute>Required</trans-attribute>
    </container-transaction>
</assembly-descriptor>
</ejb-jar>

```

## Converting an Object to ConciseXML

ConciseXML is the native syntax for Water. ConciseXML files can be directly executed by Water or parsed by any ConciseXML-compliant parser. By default, formatted output is shown in the ConciseXML syntax.



ConciseXML is described in detail in Chapter 2.

---

```
a_data.<to_concise_xml/>
```

### Output in ConciseXML:

```

<vector
  <person "Mike" "Plusch" 23 <datetime 2002 10 2/> />
  <person "Christopher" "Fry" 19/>
/>

```

## Converting Hypertext Object to XHTML

Hypertext objects are easily converted to XHTML by calling the `to_html` method.

```
<TABLE><TR><TD>one</></></>.<to_html/>
```

### Output in ConciseXML:

```
<TABLE><TR><TD>one</TD></TR></TABLE>
```

Some hypertext objects have special HTML formatters for printing HTML tags that are more compatible across different browsers.

## Converting an Object to a non-XML string

The majority of systems in production do not accept XML data. Water Export converts a Water object to any number of output formats. Two common formats are CSV, and plain text.

### Converting an Object to Text

If you need to display a Water object in a plain text format, use the `to_plain_text` method. This is very useful for creating a text e-mail message to send to pagers or cellular phones.

```
a_data.<to_plain_text/>
```

**Output in Plain Text:**

```
first_name: Mike
last_name: Plusch
employee: 23
last_updated: somedate
first_name: Christopher
last_name: Fry
employee: 19
```

### Comma Separated Values

The CSV format is one of the most common data formats because of its simplicity, small size, and wide adoption. Most databases, reporting tools, and spreadsheets can easily import CSV files.

```
a_data.<to_csv/>
```

**Output as a CSV-formatted string:**

```
first_name,last_name,employee,last_updated
Mike,Plusch,23,2002-10-2
Christopher,Fry,19
```

By default, `to_csv` will print a header row with the field names. Add the argument `header=false` to omit the header row.

```
a_data.<to_csv header=false/>
```

**Output as a CSV-formatted string:**

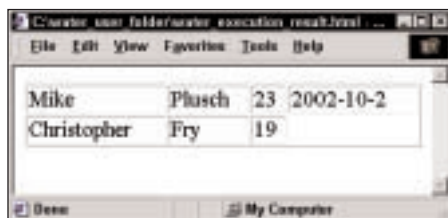
```
Mike,Plusch,23,2002-10-2  
Christopher,Fry,19
```

## Simple Reporting

Many applications require reports on either paper or online HTML reports. Water makes it easy to create an HTML-formatted report. The `to_table` method in the following code is defined on `vector`. It can be called on any vector to return an HTML object as output. The HTML will show each object as a row, and each field as a cell.

```
vector.  
<defmethod to_table fields=null>  
  <TABLE width="100%" border=1>  
    this.<for_each returns='all'>  
      <TR>  
        value.<for_each include=user_or_vector_field returns='all'>  
          <TD><do value.<to_html/> /></TD>  
        </for_each>  
      </TR>  
    </for_each>  
  </TABLE>  
</defmethod>
```

```
a_data.<to_table/>
```



The screenshot shows a web browser window with a menu bar (File, Edit, View, Favorites, Tools, Help) and a status bar (Done, My Computer). The main content area displays a table with two rows and four columns. The first row contains 'Mike', 'Plusch', '23', and '2002-10-2'. The second row contains 'Christopher', 'Fry', and '19'.

Mike	Plusch	23	2002-10-2
Christopher	Fry	19	

Figure 9-1: The HTML-formatted report

The preceding example creates a very simple report, but it could be extended to provide basic reporting features such as page breaks, categories, and headers. The rendered HTML output from the example is shown in Figure 9-1.

## Summary

Water Export converts Water objects into XML and non-XML formats that can be read by other systems. The default formatter methods are defined for all objects, but they can be overridden for every object.

Water Export can be extended with more custom formatter methods implemented in the Water language.

