Namespaces

SAX2 adds XML Namespace support, which is required for higher-level standards like XPath (used in XSLT, XPointer, XLink, and more), XML Schemas, RDF, and more. Every implementation of the SAX2 XMLReader interface is required to support Namespace processing in its default state. Additionally, many XML readers allow Namespace processing to be modified or disabled.

Namespace processing changes only element and attribute naming, although it places restrictions on some other names. With familiar XML 1.0 names, each XML element and attribute has a single name called the *qName* (which may contain colons). With Namespaces, elements and attributes have two-part name, sometimes called the "Universal" or "Expanded" name, which consists of a *URI* (signifying something analagous to a Java or Perl package name) and a *localName* (which never contains a colon).

SAX2 is capable of supporting either of these views or both simultaneously. Similarly, documents may use both views simultaneously. SAX2 XMLReader implementations are required to report the Namespaces style names when documents use them.

Namespaces in SAX Events

Namespace support affects the ContentHandler and Attributes interfaces. You will pay the most attention to this support in *startElement()* callbacks.

Element names

In SAX2, the startElement and endElement callbacks in a content handler look like this:

```
public void startElement (String uri, String localName,
                          String qName, Attributes atts)
  throws SAXException;

public void endElement (String uri, String localName, String qName)
  throws SAXException;
```

By default, an XML reader will report a Namespace URI and a localName for every element that belongs in a namespace, in both the start and end handler. Consider the following example:

```
<html:hr xmlns:html="http://www.w3.org/1999/xhtml"/>
```

With the default SAX2 Namespace processing, the XML reader would report a start and end element event with the Namespace URI *http://www.w3.org/1999/xhtml* and the localName *hr*. Most XMLReader implementations also report the original qName *html:hr*, but that parameter might simply be an empty string (except for elements that aren't in a namespace).

Attribute Names

For attributes, you can look up the value of a named attribute using the getValue method, and you can look up the Namespace URI or localName of an attribute by its index using the getURI and getLocalName methods (usually when you're iterating through the entire attribute list):

```
String rdfns = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
String resource = atts.getValue(rdfns, "resource");

for (int i = 0; i < atts.getLength(); i++) {
  String uri = atts.getURI(i);
  String localName = atts.getLocalName(i);
  String value = atts.getValue(i);
  /* ... */
}
```

Prefix Mappings

In addition to those events, SAX2 reports the scope of Namespace declarations using the startPrefixMapping and endPrefixMapping methods, so that applications can resolve prefixes in attribute values or character data if necessary. The callbacks look like this:

```
public void startPrefixMapping (String prefix, String uri)
  throws SAXException;
public void endPrefixMapping (String prefix)
  throws SAXException;
```

For the above example, the XML reader would make the following callback *immediately before* the start-element

event:

```
startPrefixMapping("html", "http://www.w3.org/1999/xhtml")
```

The XML reader would make the following callback *immediately after* the end-element event:

```
endPrefixMapping("html")
```

For elements with multiple namespace declarations, the *startPrefixMapping()* calls won't necessarily nest with the *endPrefixMapping()* because those *endPrefixMapping()* calls may be made in any order.

Configuration

This section applies to SAX2 applications with requirements for preserving prefixes and their declarations. Applications which write XML text often fall into this category, including many XML structure editors.

Configuring Namespace Support

The *http://xml.org/sax/features/namespaces* feature controls general Namespace processing. When this feature is true (the default), any applicable Namespace URIs and localNames (for elements in namespaces) must be available through the *startElement* and *endElement* callbacks in the ContentHandler interface, and through the various methods in the Attributes interface, and start/endPrefixMapping events must be reported. For elements and attributes outside of namespaces, the associated namespace URIs will be empty strings and the *qName* parameter is guaranteed to be provided as a non-empty string.

The *http://xml.org/sax/features/namespace-prefixes* feature controls the reporting of qNames and Namespace declarations (xmlns* attributes). Unless the value of this feature flag is changed to true (from its default of false), qNames may optionally (!) be reported as empty strings for elements and attributes that have an associated namespace URI, and xmlns* attributes will not be reported. When set to true, that information will always be available.

The following table summarizes the interaction of these two features. (For general information on using features, see the Features and Properties link.)

namespaces	namespace-prefixes	Namespace names	start/end PrefixMapping	qNames	xmlns* attributes
true	false	YES	YES	*unknown*	NO
true	true	YES	YES	YES	YES
false	false	**ILLEGAL MODE**			
false	true	*unknown*	*unknown*	YES	YES

The SAX2 default is the first entry in that table. Note that for documents where every element and attribute name is in a namespace, *qName* values (including namespace prefixes) *might not* be reported, and *xmlns** attributes (declaring them) **will not** be reported, unless the *namespace-prefixes* feature is changed from its default setting. Also, that the only way to avoid processor-specific behavior is to make that change! (For names that aren't in namespaces, only *qName* values are provided by all parsers; there are no processor-specific behaviors for those names.) Also, note that if you can't disable namespace processing just by changing the *namespaces* feature, since that places the processor into an illegal (nonsensical) mode.

Configuration Example

Consider the following simple sample document:

```
<?xml version="1.0"?>
<h:hello xmlns:h="http://www.greeting.com/ns/"
        id="a1" h:person="David"/>
```

If *namespaces* is true and *namespace-prefixes* is false (the default), then a SAX2 XML reader will report the following:

- an element with the Namespace URI "http://www.greeting.com/ns/" and the localName "hello";
- an attribute with no Namespace URI (empty string) and the qName (and usually localName) "id"; and
- an attribute with the Namespace URI "http://www.greeting.com/ns/" and the localName "person".

If *namespaces* is true and *namespace-prefixes* is true, then a SAX2 XML reader will report the following:

- an element with the Namespace URI "http://www.greeting.com/ns/", the localName "hello", and the qName "h:hello";
- an attribute with no Namespace URI (empty string), no localName (empty string), and the qName

"xmlns:h";
- an attribute with no Namespace URI (empty string), and the qName (and usually localName) "id"; and
- an attribute with the Namespace URI "http://www.greeting.com/ns/", the localName "person", and the qName "h:person".

If *namespaces* is false and *namespace-prefixes* is true, then a SAX2 XML reader will report the following:

- an element with the qName "h:hello";
- an attribute with the qName "xmlns:h";
- an attribute with the qName "id"; and
- an attribute with the qName "h:person".

Note that when SAX2 reports namespace declaration attributes, like "xmlns:h", it conforms to the *Namespaces in XML* recommendation from W3C: they aren't in any namespace. Some other W3C specifications violate that specification, and put such declarations into a `http://www.w3.org/2000/xmlns/` namespace. If you're working with namespace declarations, you may need to be aware of that issue.