

# Product Description for Saxon-EE-V (Enterprise Edition, Validation Package)



Version 10 released Mar 2020

Page 1/6

This document lists the features provided by Saxon 10 Enterprise Edition, Validation Package (Saxon-EE-V) - a license that enables all the facilities of Saxon Professional Edition, plus XSD schema validation; it does not include Schema Aware XSLT or XQuery.

This document does not form part of any contract unless expressly incorporated.

## Language Support

### 1. XSLT (Transformation Processing)

- |  |  |
|--|--|
| <b>1.1 XSLT 3.0 Basic Processor</b>        | Provides all mandatory features from the XSLT 3.0 specification (including try/catch, iterate, accumulators, maps, named modes, content value templates, and extended patterns).                                   |
| <b>1.2 XSLT 3.0 Serialization</b>          | Provides the serialization feature.  |
| <b>1.3 XSLT 3.0 Compatibility</b>          | Provides XSLT 1.0 compatibility mode.  |
| <b>1.4 XSLT 3.0 Dynamic Evaluation</b>     | Provides use of the XSLT 3.0 instruction <code>xsl:evaluate</code> which allows dynamic evaluation of XPath expressions.   |
| <b>1.5 XSLT 3.0 XPath 3.1 Feature</b>      | Provides full use of XPath 3.1 features, including XPath 3.1 functions, and maps and arrays.   |
| <b>1.6 XSLT 3.0 Higher-Order Functions</b> | Provides higher-order functions: specifically, the ability to use functions as values, including dynamic function calls, inline functions, partial function application, and the XPath 3.1 higher-order functions. |

Optional features not provided: XSLT 3.0 Schema Awareness, XSLT 3.0 Streaming.

For more details see: [XSLT 3.0 conformance](#).

Relevant W3C Specification: [XSLT 3.0 Recommendation \(08 June 2017\)](#).

### 2. XPath

- |                            |  |
|----------------------------|--|
| <b>2.1 XPath 3.1 Basic</b> | Provides all XPath 3.1 features which do not require schema-awareness or higher-order functions. This includes an implementation of maps and arrays, and support for JSON. |
|----------------------------|--|

# Product Description for Saxon-EE-V (Enterprise Edition, Validation Package)



Version 10 released Mar 2020

Page 2/6

**2.2 XPath 3.1 Schema Aware** Provides schema-awareness: specifically, any use of source documents with type annotations, and any use of XPath expressions that contain the names of schema components such as element declarations and types, other than the built-in types.

**2.3 XPath 3.1 Higher-Order Functions** Provides higher-order functions: specifically, the ability to use functions as values, including dynamic function calls, inline functions, partial function application, and specific higher-order functions.

For more details see: [XPath 3.1 conformance](#).

Relevant W3C Specification: [XPath 3.1 Recommendation \(21 March 2017\)](#).

## 3. XQuery

**3.1 XQuery 3.1 Minimal Conformance** Provides Minimal Conformance (including try/catch and "group-by") as defined in section 5 of the XQuery 3.1 specification.

**3.2 XQuery 3.1 Modules** Provides the Module feature.

**3.3 XQuery 3.1 Serialization** Provides the Serialization feature.

**3.4 XQuery 3.1 Higher-Order Functions** Provides the Higher-Order Function feature.

Optional features not provided: XQuery 3.1 Schema Aware, XQuery 3.1 Typed Data, XQuery 3.1 Static Typing, XQuery Update 1.0.

For more details see: [XQuery 3.1 conformance](#).

Relevant W3C Specification: [XQuery 3.1 Recommendation \(21 March 2017\)](#).

## 4. XSD (XML Schema Validation)

**4.1 XML Schema 1.0 Validation** Saxon includes a complete implementation of XML Schema 1.0. This provides the ability to process XSD 1.0 schema documents and use them to validate instance documents. Note that Saxon does not expose the full PSVI, as required by the conformance rules in the XSD 1.0 Recommendation. Also includes Saxon extension functions to provide access to a compiled schema.

# Product Description for Saxon-EE-V (Enterprise Edition, Validation Package)



Version 10 released Mar 2020

Page 3/6

For more details see: [XML Schema 1.0 conformance](#).

Relevant W3C Specification: [XML Schema 1.0 Recommendation \(28 October 2004\)](#).

## 4.2 XML Schema 1.1 Validation

Saxon includes a complete implementation of XML Schema 1.1. This provides the ability to process schema documents that use the new features of XSD 1.1, and use them to validate instance documents. More specifically, in the language of section 2.4 of the specification, it is a General-Purpose Web-Aware Validator. Also includes Saxon extension functions to provide access to a compiled schema.

For more details see: [XML Schema 1.1 conformance](#).

Relevant W3C Specification: [XML Schema 1.1 Recommendation \(05 April 2012\)](#).

## Performance Features

### 5. Byte code generation

Allows hot-spot code generation for XSLT, XQuery, XPath, and XML Schema. Available for both Java and .NET platforms, typically giving a 25% performance boost.

For more details see: [Compiling a Stylesheet](#), [Compiling Queries](#).

### 6. Import stylesheet packages

Allows the importing of stylesheet packages in compiled form. Possible with all editions provided the package only uses features available in that edition. Note that this edition does not provide the ability to create compiled stylesheet packages.

For more details see: [Compiling a Stylesheet](#).

### 7. Multi-threading (XPath)

Takes advantage of multi-core CPUs by providing automatic parallel execution of the `collection()` function.

### 8. Optimizer (Basic)

The Basic optimizer provided with all Saxon editions provides a wide range of static and dynamic optimizations including full pipelining of list operations, lazy evaluation of variables, elimination of redundant sorting operations, etc.

## 9. Reading W3C schemas and DTDs

The W3C web server now routinely rejects requests for commonly-referenced files such as the DTD for XHTML, causing parsing failures. In response to this, Saxon now includes copies of these documents within the issued JAR file, and recognizes requests for these documents, satisfying the request using the local copy.

## 10. Streaming (XPath)

Provides `saxon:stream()`, an extension function to allow large documents to be processed without holding the whole document in memory.

For more details see: [Streaming of Large Documents](#).

## Extensibility

### 11. EXSLT extension functions

A selection of EXSLT extension functions are provided (in the modules Common, Dates and Times, Math, Random, and Sets), as listed in the documentation.

For more details see: [EXSLT extensions](#).

### 12. EXPath extension functions

A selection of EXPath extension functions are provided (in the modules Archive, Binary, and File), as listed in the documentation.

For more details see: [EXPath extensions](#).

### 13. Extensibility using custom classes

Ability to write extension functions (for use in XSLT, XQuery, or XPath) by implementing a Saxon-defined interface and registering the implementation with the Saxon Configuration.

For more details see: [Extensibility](#).

### 14. Extensibility using reflexion (Java and .NET)

## Product Description for Saxon-EE-V (Enterprise Edition, Validation Package)



Version 10 released Mar 2020

Page 5/6

Ability to access existing Java or .NET methods dynamically and invoke them as extension functions by means of dynamic loading and reflexion.

For more details see: [Extensibility](#).

### 15. Saxon extension functions (Advanced)

Extension functions, as listed in the documentation, in the Saxon namespace. The Advanced level includes those that depend on streaming or schema-awareness: `saxon:schema()`, `saxon:stream()`, and `saxon:validate()`.

For more details see: [Saxon extension functions](#).

### 16. SQL extension

XSLT extension functions and instructions providing access to SQL databases. Available on the Java platform only (not .NET).

For more details see: [Saxon SQL extension](#).

### 17. XSLT element extensibility

Ability to implement XSLT extension instructions by implementing a Saxon-defined interface and registering the implementation with the Saxon Configuration.

For more details see: [Writing XSLT extension instructions](#).

## Localization

### 18. Localization (Advanced)

Run-time localization support for formatting of dates and numbers, and sorting and comparison of strings, building on the capabilities of the [ICU-J library](#). The Advanced level also includes APIs which allow additional languages to be supported.

For more details see: [Unicode collation](#), [Localizing numbers and dates](#).

## Interfaces and APIs

### 19. JAXP API

# Product Description for Saxon-EE-V (Enterprise Edition, Validation Package)



Version 10 released Mar 2020

Page 6/6

Implementations of the standard JAXP interfaces for XSLT transformation, XPath evaluation, and XML Schema validation. Applies to the Java platform only.

For more details see: [JAXP API conformance](#).

## 20. S9API API

Saxon's native interface for processing XSLT, XQuery, XPath, and XML Schema. Available in slightly different forms on the Java and .NET platforms.

## 21. Support for DOM

Ability to use a DOM (Document Object Model) for the input and output of transformations and queries. On the .NET platform this includes the System.XML DOM classes.

For more details see: [Object models](#).

## 22. Optimized support for DOM

Indexing of a supplied DOM tree to provide fast navigation (the Domino Model).

For more details see: [Domino Tree Model](#).

## 23. Support for JDOM2, AXIOM, DOM4J, and XOM

Ability to use a JDOM2, AXIOM, DOM4J, and XOM for the input or output of transformations and queries. Applies to the Java platform only. Note that the code for these interfaces is open source and can be compiled to work with Saxon-HE, but it does not come packaged with the Saxon-HE download.

For more details see: [Object models](#).

## 24. XQJ API

Implementations of the standard XQJ interfaces for XQuery processing. Applies to the Java platform only. Note that the XQJ interfaces have been removed from the standard download of Saxon-HE because the Oracle specification license is not open source, but they are available on request.

For more details see: [XQJ API conformance](#).