Product Description for SaxonC-EE (Enterprise Edition)

Version 13 released May 2026

This document lists the features provided by SaxonC 13 Enterprise Edition (Saxon-EE), for C/C++, PHP and Python.

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

Language Support

1. XSLT (Transformation Processing)

1.1 XSLT 3.0 Basic Processor	Provides all mandatory features from the XSLT 3.0 specification (including try/catch, iterate, accumulators, maps, named modes, content value templates, and extended patterns, as well as features retained from XSLT 2.0).
1.2 XSLT 3.0 Schema Awareness	Provides a schema-aware XSLT 3.0 processor: specifically, the ability to import an XSD schema when compiling a stylesheet, to use this for type-checking, and to process and create schema-validated instance documents.
1.3 XSLT 3.0 Serialization	Provides the serialization feature: specifically, the ability to convert the result trees produced as output of an XSLT transformation to lexical XML, or other formats including HTML, JSON, and plain text, under the control of serialization parameters defined in the stylesheet or via an external API.
1.4 XSLT 3.0 Compatibility	Provides XSLT 1.0 compatibility mode as defined in the XSLT 3.0 specification. If a stylesheet specifies `version="1.0"`, this causes certain constructs to behave in a way that retains the XSLT 1.0 behavior.
1.5 XSLT 3.0 Streaming	Includes additional features of the XSLT 3.0 specification that enable streaming (processing of documents that are too large to fit in memory).
1.6 XSLT 3.0 Dynamic Evaluation	Provides use of the XSLT 3.0 instruction `xsl:evaluate`, which allows dynamic evaluation of XPath expressions.
1.7 XSLT 3.0 XPath 3.1 Feature	Provides full use of XPath 3.1 features, including XPath 3.1 functions, and maps and arrays. Note: Saxon supports XPath 3.1 unconditionally in XSLT 3.0 stylesheets, it does not offer a processing mode in which XPath is restricted to version 3.0.

 © Saxonica Ltd. 2026

1.8 XSLT 3.0 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 the standard higher-order functions defined in the XPath 3.1 function library.

1.9 XSLT 4.0 Extensions

Provides many features from the draft XSLT 4.0 specification. Generally, Saxonica aims to avoid making incompatible changes except in major releases. But since the 4.0 features are not yet entirely stable, this policy does not apply: the aim instead will be to respond rapidly to changes in the developing specification. Maintenance releases in the 13.x series may therefore reflect changes in the draft specifications.

For more details see: XSLT 3.0 conformance.

Relevant W3C Specification: XSLT 3.0 Recommendation (08 June 2017).

2. XPath

2.1 XPath 3.1 Basic

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, as well as language constructs retained from earlier XPath versions.

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 a library of built-in higher-order functions.

2.4 XPath 4.0 Extensions

Provides many features from the draft XPath 4.0 specification. Generally, Saxonica aims to avoid making incompatible changes except in major releases. But since the 4.0 features are not yet entirely stable, this policy does not apply: the aim instead will be to respond rapidly to changes in the developing specification. Maintenance releases in the 13.x series may therefore reflect changes in the draft specifications.

For more details see: XPath 3.1 conformance.

Relevant W3C Specification: XPath 3.1 Recommendation (21 March 2017).

© Saxonica Ltd. 2026

3. XQuery

3.1 XQuery 3.1 Minimal Conformance

Provides Minimal Conformance (including try/catch and "group-by", as well as language features retained from earlier XQuery versions) as defined in section 5 of the XQuery 3.1 specification.

3.2 XQuery 3.1 Schema Aware

Provides the Schema Aware feature. This enables a query to import an XSD schema and use it for type checking, and to validate instance documents.

3.3 XQuery 3.1 Typed Data

Provides the Typed Data feature. This enables a query to accept input data that has been validated against a schema.

3.4 XQuery 3.1 Modules

Provides the Module feature, which allows a query to be made up of multiple modules.

3.5 XQuery 3.1 Serialization

Provides the Serialization feature. This allows the output of a query to be serialized as lexical XML, or in other formats including HTML, JSON, and plain text, under the control of serialization parameters contained either in the query itself, or supplied externally.

3.6 XQuery 3.1 Higher-Order Functions

Provides the Higher-Order Function feature. This provides the ability to use functions as values, including dynamic function calls, inline functions, partial function application, and a library of built-in higher-order functions.

3.7 XQuery Update 1.0

Saxon provides all the features defined in the XQuery Update 1.0 specification. The implementation allows XQuery Update 1.0 syntax to be mixed with XQuery 3.1 syntax.

XQuery Update in SaxonCS is offered "as is": the feature is present, but has not been extensively tested.

For more details see: XQuery Update 1.0 conformance.

Relevant W3C Specification: XQuery Update 1.0 Recommendation (17 March 2011).

3.8 XQuery 4.0 Extensions

Provides many features from the draft XQuery 4.0 specification. Generally, Saxonica aims to avoid making incompatible changes except in major releases. But since the 4.0 features are not yet entirely stable, this policy does not apply: the aim instead will be to respond rapidly to changes in

© Saxonica Ltd. 2026

the developing specification. Maintenance releases in the 13.x series may therefore reflect changes in the draft specifications.

Optional features not provided: XQuery 3.1 Static Typing.

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.

From Saxon 10, the Saxon schema processor accepts XSD 1.1 syntax unconditionally. XSD 1.0 schemas are processed according to the rules of the XSD 1.1 specification. This offers a very high level of backwards compatibility, except in a few areas where the XSD 1.0 rules were unclear and have been clarified in the XSD 1.1 recommendation.

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

 © Saxonica Ltd. 2026

5. Export stylesheet packages

XSLT 3.0 packaging allows stylesheet modules to be independently compiled and distributed, and provides much more "software engineering" control over public and private interfaces, and the like. The ability to save packages in compiled form ("stylesheet export file", SEF) gives much faster loading of frequently used stylesheets, and also enables in-browser execution using SaxonJS.

For more details see: Compiling a stylesheet.

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.

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. Multi-threading (XSLT)

Takes advantage of multi-core CPUs by providing automatic parallel execution of the `xsl:result-document` instruction; plus an extension attribute `saxon:threads` to allow multi-threaded execution of `xsl:for-each` instructions under the control of the stylesheet author.

9. Optimizer (Advanced)

The Advanced optimizer provides the wide range of static and dynamic optimizations featured in the Basic optimizer - including full pipelining of list operations, lazy evaluation of variables, elimination of redundant sorting operations, etc. - and additionally provides join optimization, inlining of variables and functions, just-in-time compilation of template rules, and optimized searching of large sets of template rules, where feasible.

10. Reading W3C schemas and DTDs

The W3C web server 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/DLL file, and recognizes requests for these documents, satisfying the request using the local copy.

11. Streaming (XPath and XQuery)

© Saxonica Ltd. 2026

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 XML documents.

12. Streaming (XSLT)

Allows large documents to be processed without holding the whole document in memory. Provides the streaming features of the XSLT 3.0 specification.

For more details see: Streaming XML documents.

Extensibility

13. 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.

14. 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.

15. User-defined extension functions (C++, PHP and Python)

Ability to access existing C++, PHP and Python functions dynamically.

For more details see: .

16. 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.

© Saxonica Ltd. 2026

Localization

17. Formatting numbers in English

Numbers can be formatted as words in English.

For more details see: Formatting numbers.

18. Formatting dates in English

Dates can be formatted as words in English.

For more details see: Formatting dates.

19. Sorting and collations using platform locales

Support for sorting and comparison of strings using the Unicode Collation Algorithm uses the collation facilities available directly from the JDK.

For more details see: Sorting and collations.

Interfaces and APIs

20. C++ API

SaxonC's interface for processing XSLT, XQuery, XPath, and (with Saxon-EE) XML Schema, on C++.

21. PHP extension

Interface module that allows SaxonC to be invoked from PHP applications.

22. Python extension

Interface module that allows SaxonC to be invoked from Python applications.

© Saxonica Ltd. 2026