How To Validate an XML Document

There are several ways to validate an XML document. Which method you use depends on what you want to do, what resources you have, and what version of isCOBOL you use. This document will review these three methods to validate an XML programmatically:

1. Validate while parsing
2. Validate using an XSD schema
3. Validate while parsing using an XSD schema

1. Validate while parsing

If you need to parse an XML of any structure and want to validate the XML while parsing, the XML PARSE statement is the one to use.

The simplest way to use XML PARSE with validation is as follows:

```
xml parse xml-document-cust
    processing procedure xml-handler
```

The XML to parse should reside in an alphanumeric or group alphanumeric item in memory.

The xml-handler paragraph is executed multiple times, once for each part of the XML being parsed and validated. The predefined item XML-EVENT contains the type of XML component currently being processed. If a validation issue occurs, XML-EVENT contains the string "EXCEPTION".

When an exception is encountered, the predefined items XML-CODE and XML-ERRMSG can be used to obtain error details. XML-CODE holds the return code in the high-order halfword and a reason code in the low-order halfword.

Attached program “XMLParse.cbl” is an example of how XML PARSE works. For more details on its syntax and capabilities, refer to the online documentation.

The xml-handler paragraph in this sample shows different types of XML-EVENTS for each type of XML component being parsed and logs those elements in the isCOBOL log file. But for the ‘EXCEPTION’ it shows the error information on the console.

To enable the log information while parsing you need to include the following properties in the runtime properties file.

```
iscobol.tracelevel=1
iscobol.logfile=isxml.log
```

Command to compile the test program: *iscc XmlParse.cbl*

Command to run the test program: *iscrun XMLPARSE*

The example above should parse and validate successfully. However, if you want to test the validation mechanism, you can introduce small errors, such as the following:
Correct version:

```
02 pic x(44) value '  <cust_balance>1050.75</cust_balance>'.
```

Incorrect version:

```
02 pic x(44) value '  <cust_balance>1050.75</cust_balan>'.
```

2. Validate using an XSD schema

If you want to validate the contents of an XML document of any structure against an XSD schema, without writing custom parsing logic, you can use the java.xml classes.

Attached is a generic COBOL program for XML validation, "XmlValidate.cbl". You can call this program from your code, passing it the XML and XSD names. A sample program is attached called "CallXmlValidate.cbl", which uses "custdoc.xml" and "custschema.xsd", also attached.

Command to compile the sample programs: *iscc XMLValidate.cbl CallXMLValidate.cbl*

Command to run the test program: *iscrun CALLXMLVALIDATE*

The example above should be parsed and validated successfully with the following output:

Validating...
xml-file: custdoc.xml
xsd-file: custschema.xsd
Validation Ok

However, if you want to test the validation mechanism, you can introduce a small error in the XML. For example:

Correct version:
<cust_balance>1050.75</cust_balance>

Incorrect version:
<cust_balance>1050.75</cust_balan>

3. Validate while parsing using an XSD schema

In 2025R1 Veryant introduced a new XML PARSE syntax, the VALIDATE phrase, that allows you to validate against a schema while parsing, like this:

```
xml parse xml-document-1 validating with file XMLSCH
      processing procedure xml-handler-paragraph
```

The XML to parse should reside in an alphanumeric or group alphanumeric item in memory.

Like method #1 above, the xml-handler paragraph is executed multiple times, once for each part of the XML being parsed and validated. The predefined item XML-EVENT contains the type of XML component currently being processed. If a validation issue occurs, XML-EVENT contains the string "EXCEPTION".

When an exception is encountered, the predefined items XML-CODE and XML-ERRMSG can be used to obtain error details. XML-CODE holds the return code in the high-order halfword and a reason code in the low-order halfword.

The attached "XMLParseValidating.cbl" is a complete and simple example of how this works. For more details on its syntax and capabilities, refer to the online documentation.

The XSD schema used for validation in this sample, "XMLSCH", is also attached.

Command to compile the test program: *iscc XMLParseValidating.cbl*

Command to run the test program: *iscrun XMLPARSEVALIDATING*

The example above will produce several validation exceptions:

Exception: 1607680: (1:2335) cvc-complex-type.2.4.a: Invalid content was found starting with element 'quantityOnHand'. One of '{itemName}' is expected.

Exception: 100: (1:23) Attribute name "itemNumber" associated with an element type "stockItem" must be followed by the '=' character.

Exception: 1607680: (1:44) cvc-complex-type.3.2.2: Attribute 'warehouse' is not allowed to appear in element 'stockItem'.

Exception: 1607680: (1:32) cvc-datatype-valid.1.2.1: '123-Ab' is not a valid value for 'integer'.

Exception: 1607680: (1:76) cvc-complex-type.2.4.b: The content of element 'stockItem' is not complete. One of '{quantityOnHand}' is expected.

Exception: 1607680: (1:112) cvc-complex-type.2.4.d: Invalid content was found starting with element 'comment'. No child element is expected at this point.

Exception: 1607680: (1:103) cvc-maxExclusive-valid: Value '100' is not facet-valid with respect to maxExclusive '100' for type '#AnonType_quantityOnHandstockItemType'.

Online URL: https://support.veryant.com/phpkb/article.php?id=352