



Companies and Intellectual
Property Commission

a member of **the dti** group

CIPCs Validation Rest API

Instruction Manual for Validation Service

Contents

Revisions Table.....	3
Introduction of Validator Rest API.....	3
How to Call Validator Rest API	3
Reading response from REST API.....	4

Revisions Table

Author	Version	Date	Comments
Ashish Singhvi	V0.1	01/03/2018	First Draft
Hennie Viljoen	V0.2	06/03/2018	Cosmetic Changes
Hennie Viljoen	V0.3	06/03/2018	Corrected Grammar Mistakes
















Introduction of Validator Rest API

The CIPC iXBRL validator is deployed as a Rest API which can be accessed on the internet. Consumer needs to call the said API and display the output in their preferred format. The validator validates the iXBRL document and returns the output in csv files. Consumption platform has to read the csv files and display the result in desired format.

Please note: this manual is subject to change since it is releases as part of the Pilot Testing Phase, and issues may be identified during testing. The purpose of the Validation Service is to assist Software Service Providers who didn't incorporate sophisticated validation engines into their own XBRL client-side software.

How to Call Validator Rest API

URL	https://validation.cipc.co.za/bushchat-api/ws1000/ws1001
Method	POST
Data Parameters	<p>relativeSource file name of iXBRL document (.xhtml)</p> <p>arguments -sp IXBRL, XBRL, Dimension, Formula -X</p> <p>input zip file name which contains the .xhtml file to be validated</p> <p>Example: relativeSource : sampledoc.xhtml arguments : -sp IXBRL, XBRL, Dimension, Formula -X input : sampledoc.zip</p>
JavaScript JQuery AJAX sample example	<pre>var form = new FormData(); form.append("relativeSource", "sampledoc.xhtml"); form.append("arguments", "-sp IXBRL, XBRL, Dimension, Formula -X"); form.append("input", "sampledoc.zip"); var settings = {</pre>

	<pre>"async": true, "crossDomain": true, "url": "https://validation.cipc.co.za/bushchat-api/ws1000/ws1001", "method": "POST", "headers": { "cache-control": "no-cache", "postman-token": "1c3276ea-a939-7b3a-ee71-b6cfd2d3b1f7" }, "processData": false, "contentType": false, "mimeType": "multipart/form-data", "data": form } \$.ajax(settings).done(function (response) { console.log(response); });</pre>						
Success Response	<p>IsSuccessStatusCode : true , if call succeed Content: returns stream as output which contains the set of csv files. Consumer needs to read the below set of csv files,</p> <table border="1" data-bbox="571 940 1416 1325"> <tr> <td data-bbox="571 940 875 1152"> FORMULAout.csv  FORMULAout.csv </td> <td data-bbox="875 940 1146 1152"> IXBRL.csv  IXBRL.csv </td> <td data-bbox="1146 940 1416 1152"> ixbrl_schema.csv  IXBRL.csv </td> </tr> <tr> <td data-bbox="571 1152 875 1325"> SCHEMA.csv  SCHEMA.csv </td> <td data-bbox="875 1152 1146 1325"> XBRL.csv  XBRL.csv </td> <td data-bbox="1146 1152 1416 1325"></td> </tr> </table>	FORMULAout.csv  FORMULAout.csv	IXBRL.csv  IXBRL.csv	ixbrl_schema.csv  IXBRL.csv	SCHEMA.csv  SCHEMA.csv	XBRL.csv  XBRL.csv	
FORMULAout.csv  FORMULAout.csv	IXBRL.csv  IXBRL.csv	ixbrl_schema.csv  IXBRL.csv					
SCHEMA.csv  SCHEMA.csv	XBRL.csv  XBRL.csv						

Reading response from REST API

There are mainly 5 different types of csv files generated from validator API. Consumer has to read each file in order to verify if the file has errors / warnings. CIPC portal allows files which have warnings. If there are any errors found in any of these output files, CIPC submission portal will reject the same. There are two different structures as defined in output files. All output files have results in a formula format.

➤ **Structure 1:** for IXBRL, IXBRL Schema, Schema and XBRL.

ERROR_CODE	ERROR_TYPE	LINE NUMBER	COLUMN NUMBER	FILE PATH	ERROR MESSAGE
------------	------------	-------------	---------------	-----------	---------------

Where,

- **ERROR_CODE:** This is a unique code used to identify an error.
- **ERROR TYPE:** The error type is either “E” or “W”. “E” stands for “Error” & “W” stands for “Warning”. If the error type appears as “E” then this means an error is found and to clear validation one needs to solve such error. If the error type is “W” then the same can be either solved or can be ignored.
- **LINE NUMBER:** This will show the line number where the error or warning is present in the document.
- **COLUMN NUMBER:** This will take you to the exact location in a line where the error or warning is situated.
- **FILE PATH:** This shows the location of the file for which error is shown.
- **ERROR MESSAGE:** This is the detailed description of the error along with references to the specifications.
- **E.g.:** -1##E##1##4440##default_output.xbrl##cvc-id.2: There are multiple occurrences of ID value 'idfactxxx'.

➤ **Structure 2:** for FORMULA

Formula output is divided into three sub category – **Header, Error & Footer.**

- 1. HEADER:** For each assertion, header and footer will be displayed. Header contains the following:

Type of formula	ID of formula	Label of formula	Expression for formula	Type of severity	File path
-----------------	---------------	------------------	------------------------	------------------	-----------

Where,

- **Type of formula** – There are two type of formula assertions used in the taxonomy - Value Assertion or Existence Assertion
- **ID of formula** – Every formula is holding a unique ID.
- **Label of formula** – Each formula has been assigned a unique label which includes a number. This column displays that label.
- **Expression for formula** – The exact formula expression is explained here.
- **Type of severity (Error or Warning)** – This displays the severity of the business rule result - ERROR or WARNING. Warnings can be ignored.
- **File Path** – The path of XML file containing the assertion.

- E.g. - Header##Value Assertion##ec_23##valueAssertion_41##Expression Text xxx##
##ERROR##Filename

2. ERROR: The formula error shows result in the following manner:

Pre-condition	Formula Result	Business rule description	Variable
---------------	----------------	---------------------------	----------

Where,

- Pre-condition - It will check if any pre condition is required to run such formula. Pre-condition are True, False or NA (if not present). Currently, there is no pre condition requirement for CIPC formula linkbase. It will always be NA.
 - Formula Result - The formula result explained whether the result is correct or incorrect. If it is defined as "True" then this means that the formula is executed properly and there is no error. If the result is "False" then this means that the value is not correct and correction is required to change this label from False to True.
 - Business rule description - Formula result is followed by the business rule description.
 - Variable - Value for each variable is defined as per the formula linkbase format to understand the formula error. Variables are already structured in the formula linkbase.
 - E.g. - NA##FALSE##"emperor: The value of ""Declaration of director's report presence"" MUST be reported for the current reporting period."##"v2[DateOfEndOfReportingPeriod2013;D2017;;;2017-12-31]"##"v1"
3. **FOOTER:** For each assertion, the footer will display the count of TRUE, FALSE and NOT EVALUATED result. So, each footer will contain 3 values in this order – TRUE results count, FALSE results count, and NOT EVALUATED results count.

E.g. Footer##0##2##0## which will result into this expression -
Footer##True##False##Not Evaluated ## (This means there are 2 count for False values)

Please Note: Sample result files for each type of errors, please refer [here](#).