

Standard Business Reporting

Australian Taxation Office

ATO Validation and Rule Expression Guide

Date: 6th August 2020

Status: Final – suitable for use

* This document and its attachments are **Unclassified.**

 For further information or questions, contact the SBR Service Desk at [SBRServiceDesk@sbr.gov.au](http://sbr.gov.au/agreement/Gateway/1.0/Push/PKI) or call 1300 488 231.

International callers may use +61-2-6216 5577

VERSION CONTROL

| **Version** | **Date** | **Description of changes** |
| --- | --- | --- |
| 1.4 | 06.08.2020 | Section 2.2 updated to include principles for Validation Rule implementation across schema, channel and interactive error handling. |

Copyright

© Commonwealth of Australia 2020 (see exceptions below).
This work is copyright. Use of this Information and Material is subject to the terms and conditions in the “SBR Disclaimer and Conditions of Use” which is available at [http://www.sbr.gov.au](http://www.sbr.gov.au/software-developers/enabling-sbr-in-my-application/sbr-taxonomy/view-taxonomy). You must ensure that you comply with those terms and conditions. In particular, those terms and conditions include disclaimers and limitations on the liability of the Commonwealth and an indemnity from you to the Commonwealth and its personnel, the SBR Agencies and their personnel.

You must include this copyright notice in all copies of this Information and Material which you create.  If you modify, adapt or prepare derivative works of the Information and Material, the notice must still be included but you must add your own copyright statement to your modification, adaptation or derivative work which makes clear the nature of your modification, adaptation or derivative work and you must include an acknowledgement that the adaptation, modification or derivative work is based on Commonwealth or SBR Agency owned Information and Material. Copyright in SBR Agency specific aspects of the SBR Reporting Taxonomy is owned by the relevant SBR Agency.

Table of contents

[1 Introduction 4](#_Toc46999402)

[1.1 Purpose 4](#_Toc46999403)

[1.2 Audience 4](#_Toc46999404)

[1.3 Document context 4](#_Toc46999405)

[1 Response messages 5](#_Toc46999406)

[1.1 Error messages 5](#_Toc46999407)

[2 Data formats 6](#_Toc46999408)

[2.1 XBRL instances 6](#_Toc46999409)

[2.2 Validation 6](#_Toc46999410)

[3 Rule expression 10](#_Toc46999411)

[3.1 Form prefix labels 10](#_Toc46999412)

[3.2 Context instance labels 10](#_Toc46999413)

[3.3 Absent form or context labels 10](#_Toc46999414)

[3.4 Use of xx.xx in fact names 10](#_Toc46999415)

[3.5 Use of aliases 10](#_Toc46999416)

[3.6 Interpretation of NULL 11](#_Toc46999417)

[3.7 Boolean checks 11](#_Toc46999418)

[3.8 Use of domain in rules 11](#_Toc46999419)

[3.9 Case sensitivity 11](#_Toc46999420)

[3.10 Tuples and context 11](#_Toc46999421)

[3.11 Common modules 11](#_Toc46999422)

[4 ATO structured english 13](#_Toc46999423)

[5 Validation rules spreadsheets 24](#_Toc46999424)

[6 Previous Version Control 26](#_Toc46999425)

# **Introduction**

Purpose

The purpose of this document is to provide information that will assist software developers in the implementation of calls to the web services offered by the Australian Taxation Office (ATO) through the Standard Business Reporting (SBR) platforms (SBR Core Services and ebMS3).

Audience

The audience for this document is any organisation that will be building any ATO SBR services into their products. Typically this will be software application developers.

Document context

This document describes the validations performed by, and the rule expressions used by the ATO.

# **Response messages**

## Error messages

Business rules that are expected to be implemented by software developers are described in each of the product-specific validation rules spreadsheets. Each rule is associated with a response message code. Where a submission does not comply with a given rule, the associated message code will be returned.

Message codes returned by the ATO have the following format:

**{Jurisdiction}.{Agency}.{Function}.{Id}**

where:

**Jurisdiction** = CMN (Commonwealth)

**Agency** = ATO

**Function** = GEN (General – can apply to many functions/forms); or
Form specific, such as FBT, CTR, PTR, AS, SMSFAR, etc.

**Id** = function specific identifier.

For example: CMN.ATO.CTR.123456

### Successful requests

In the event of a successful request, for most (and all 2015 onwards) services, the following information message shall be returned (in addition to any warning messages):

| **Message Code** | **Severity Code** | **Short Description** |
| --- | --- | --- |
| CMN.ATO.GEN.OK | Information | Message Accepted |

Note that a number of older SBR Core services: AS.0001, FBT.0001, FBT.0002 (2011-2014), PS.0001, PS.0002 and TFND.0001, use the following information response message:

| **Message Code** | **Severity Code** | **Short Description** |
| --- | --- | --- |
| SBR.GEN.GEN.OK | Information | Message Received Successfully |

# **Data formats**

## XBRL instances

### Monetary Units

The XBRL 2.1 specification requires that the Qnames used in unit definitions for monetary values MUST use ISO4217 currency designations for the local part, and MUST use a namespace of “http://www.xbrl.org/2003/iso4217”. Unit definitions for monetary currencies in XBRL instances within SBR MUST conform to these requirements. In particular, amounts representing Australian dollars MUST be associated with a unit definition that uses a currency designation of “AUD”.

Unless otherwise stated in the MST, all monetary amounts in XBRL instances must be expressed in Australian dollars.

### Non-monetary Units

Unless otherwise stated in the product-specific MST, all non-monetary numeric values requiring units in XBRL instances must be expressed as xbrli:pure.

### Measurement Accuracy

The XBRL 2.1 specification requires that each numeric item (apart from those whose value is a fraction) carry either a precision or decimals attribute allowing the creator of an XBRL instance to provide a statement of the accuracy of the provided value.

Unless otherwise stated in the relevant product-specific MST, when producing XBRL instances within SBR

* 1. non-financial numeric values, such as counts, SHOULD be provided with a value of ”0” for the decimals attribute.
	2. financial amounts accurate to the dollar SHOULD be provided with a value of “0” for the decimals attribute.
	3. financial amounts accurate to the cent SHOULD be provided with a value of “2” for the decimals attribute.

When consuming XBRL instances within SBR

1. any digits considered to be insignificant SHOULD be replaced with zeros.

## Validation

### Payload validation

Each Payload received by SBR undergoes validation that is specific to the data format of the Payload and the service action that is sought to be invoked for that Payload.

To support performance, reduce duplication and maintenance of validations for SBR enabled services, service specific validation rules are implemented across the ATO technical ecosystem by use of schema, in-channel (e.g. C#) or in real time by ATO processing systems known as Interactive Error Handling (IEH).

The ATOs best practise is to maintain, where possible, validation rules as per the below guidance however in certain scenarios, business requirements, service design and/or service capacities may inhibit the ability to implement validation as per this guidance.

#### Contract / Schema validation

Used for data types, data formats, lengths, enumerations and presence of mandatory fields. SBR definitional data types may be further restricted via the specific schema to reduce the need for in-channel validation (e.g. C#). In these cases, details of the additional schema restrictions can be found in the Message Structure Table (MST) artefact.

Sections 2.2.2, 2.2.3 and 2.2.4 provide detail on format specific schema validation errors.

#### Channel (C# / Schematron)

A set of data integrity rules which the data being received is validated against by the SBR channel. These rules are non-interactive and are resolved without reference to the data maintained by the ATO processing systems.

Channel validation is used for both interactive and non-interactive services with non-interactive services generally having a greater number of business rule focused validations. These rules are described in ATO Validation Rule spreadsheets.

The ATO provides C# to DSPs as a reference implementation. The following are some ways in which the files can be used:

* Unit testing of software products – allowing DSPs to test the output reports that their product produces to see if they will pass through ATO boundary validation checks.
* Could enable DSPs to create a mock service that may allow DSPs to run tests without hitting EVTE.
* DSPs could use third party tools to convert the C# to another programming language.
* Provides some additional clarity of what a business rule means at run-time.
* **Note that it is not recommended to include the provided C# rules in DSP delivered product.**
	+ The rationale for the ATO advice on not implementing C# rules in production is predominately due to breaking rule changes that may be implemented by the ATO where a DSP product cannot be updated in a reactive manner causing discrepancies in rule behaviours between a DSP product and ATO platform.
	+ Those DSPs that include C# in their production products do so at their own risk.

#### Interactive Error Handling (IEH)

Interactive Error Handling rules are business defined rules, managed by ATO processing systems and processed at the time of lodgment. Interactive validation can only occur once the message passes schema and in-channel validation and may include validations against other services and/or the client’s record.

Unlike channel validation, due to the complexities interactive validation presents, including; threshold, values, benchmarks that have the potential to change rapidly, or reference to sensitive client data stored in the ATO receiving systems, these rules are normally not available externally and it is recommended that DSP’s do not attempt to replicate these into their production software.

Where possible, error messages from these rules are described in ATO Validation Rule spreadsheets.

### XBRL validation

For requests that contain a Payload that is in XBRL format the payload validation checks that the request message is valid XBRL that conforms to the SBR reporting taxonomy and definitional taxonomy. These checks are applied prior to the checks specified in the Validation Rules spreadsheet.

The table below lists the error messages that may be returned from XBRL payload validation and included in the response message. In most cases, more specific information about the error will be included in the detailed description.

| **SBR message code** | **Short description** |
| --- | --- |
| CMN.ATO.GEN.XBRL01 | The message did not pass XBRL validation. Please contact your software provider. |
| CMN.ATO.GEN.XBRL02 | The message was rejected due to a system error. Please contact your software provider. |
| CMN.ATO.GEN.XBRL03 | A field contains invalid data (such as letters in numeric or date field). |
| CMN.ATO.GEN.XBRL04 | A mandatory field has not been completed. |
| CMN.ATO.GEN.XBRL05 | Invalid start or end datetime for duration period. |
| CMN.ATO.GEN.XBRL06 | End date is earlier than start date. |
| CMN.ATO.GEN.XBRL07 | Invalid value for end datetime of duration period or end datetime earlier than start datetime. |
| CMN.ATO.GEN.XBRL08 | Invalid value for start datetime of duration period. |
| CMN.ATO.GEN.XBRL09 | Invalid value for instant period datetime. |
| CMN.ATO.GEN.XBRL10 | The logical record or logical document did not pass XBRL validation. Please contact your software provider. |

### JSON Validation

For requests that contain a Payload that is in JSON format the payload validation checks that the request message is valid JSON that conforms to the SBR reporting taxonomy and definitional taxonomy. These checks are applied prior to the checks specified in the Validation Rules spreadsheet.

The table below lists the error messages that may be returned by JSON validation and included in the response message. In most cases, more specific information about the error will be included in the detailed description.

| **SBR message code** | **Short description** |
| --- | --- |
| CMN.ATO.GEN.JSON01 | The message did not pass JSON validation. Please contact your software provider. |
| CMN.ATO.GEN.JSON02 | The message was rejected due to a system error. Please contact your software provider. |
| CMN.ATO.GEN.JSON03 | A field contains invalid data (such as letters in numeric or date field). |
| CMN.ATO.GEN.JSON04 | A mandatory field has not been completed. |
| CMN.ATO.GEN.JSON05 | Invalid start or end datetime for duration period. |
| CMN.ATO.GEN.JSON06 | End date is earlier than start date. |
| CMN.ATO.GEN.JSON07 | Invalid value for end datetime of duration period or end datetime earlier than start datetime. |
| CMN.ATO.GEN.JSON08 | Invalid value for start datetime of duration period. |
| CMN.ATO.GEN.JSON09 | Invalid value for instant period datetime. |

### XML Validation

For requests that contain a Payload that is in XML format the payload validation checks that the request message is valid XML that conforms to the SBR reporting taxonomy and definitional taxonomy. These checks are applied prior to the checks specified in the Validation Rules spreadsheet.

The table below lists the error messages that may be returned by XML validation and included in the response message. In most cases, more specific information about the error will be included in the detailed description.

|  |  |
| --- | --- |
| **SBR message code** | **Short description** |
| CMN.ATO.GEN.XML01 | The message did not pass XML validation. Please contact your software provider. |
| CMN.ATO.GEN.XML03 | A field contains invalid data (such as letters in numeric or date field). |
| CMN.ATO.GEN.XML04 | A mandatory field has not been completed. |
| CMN.ATO.GEN.XML06 | End date is earlier than start date. |
| CMN.ATO.GEN.XML10 | The logical record or logical document did not pass XML validation. Please contact your software provider. |

### SBR Core Services Validation Phasing

Validation, as defined in the *Validation rules* spreadsheet, is applied in phases for ATO web services in the SBR Core Services platform such that validation will not progress to the next phase until the current phase is completely passed.

Following successful authentication, authorisation, and payload validation, the phases based on rule type are as follows:

1. Message Header checks
2. Payload validation (as described above)
3. XBRL contexts, formats, data types, lengths and enumerations
4. presence of mandatory fields (elements)
5. cross-field rules, calculations, comparisons, common module rules
6. cross-form (cross product) rules
7. warnings (for data that *may* be incorrect or incomplete).

As an example, if a company tax return (CTR) business document contained correct header information and passed the XBRL validator, but has one or more instances of incorrect XBRL context, the submission would result in a fail response message that would contain details of the invalid XBRL contexts, plus any format errors, incorrect data types, etc. No checks for missing mandatory fields, cross-field or cross-form errors will have been performed (other than those performed by the XBRL validator).

If any warnings occur, the business document will still be accepted and processed by the ATO. To correct these warnings, an amendment to the return may be submitted (for those business documents that allow for amendments via SBR).

# **Rule expression**

The validation rules are written in ATO structured English. This is a type of pseudo-code used to ensure clarity in rule expression. Section 4 explains each of the terms used in ATO structured English.

NOTE: Although ATO structured English refers to XBRL terminology, the validation rules they have equivalent application to payloads that are in implemented using the JSON/XML data format.

## Form prefix labels

Form prefix labels may be used in validation rules to identify the business document of a given fact. These are used primarily where an interaction may involve more than one business document, such as those tax returns that include schedules. CTR for example, is a primary form (parent) with multiple associated schedules (child forms), such as IEE, CGT, IDS, etc.

For example:

 **CTR**:RP:pyid.xx.xx:Identifiers.AustralianBusinessNumber.Identifier

refers to the ABN field in the CTR business document.

## Context instance labels

Context instance labels describe the context and link a fact to a context. These labels appear in validation rule as a prefix to each fact.

For example:

 **RP.TOFA**:bafpr1.xx.xx:Income.FinancialArrangementsUnrealisedGains.Amount

refers to the fact being reported in the context ‘RP.TOFA’, where the dimension ReportPartyType is set to “ReportingParty” and the dimension FinancialArrangementType is set to ‘TOFA’.

## Absent form or context labels

Where no form or context prefix (as described above) is provided for a fact within a rule, the rule applies regardless of form or context, for the form(s) in which the rule is specified. This allows a given rule to be specified once and re-used across multiple forms or contexts.

## Use of xx.xx in fact names

In the validation rules the version of an element may not be provided within the namespace prefix and instead be replaced with ‘xx.xx’. In an actual business document an XBRL fact must always include the namespace prefix including the correct version of the element. The correct version can be derived from the discoverable taxonomy set available on the SBR website (refer [http://www.sbr.gov.au/software-developers/enabling-sbr-in-my-application/sbr-taxonomy/view-taxonomy](http://sbr.gov.au/ato/Role/Business%20Intermediary)).

For example, a validation rule may include:

 bafpr1**.xx.xx**:Income.FinancialArrangementsUnrealisedGains.Amount

where ‘xx.xx’ refers to the version of the element consumed in the reporting taxonomy:

 bafpr1**.02.02**:Income.FinancialArrangementsUnrealisedGains.Amount.

## Use of aliases

In order to make the validation rules more readable aliases have been used in some rules as a shorthand identifier for each fact. An alias used in a rule is enclosed in square brackets, for example, [CTR123]. The definition for each alias is defined in the *Message structure* spreadsheet.

## Interpretation of NULL

Where a rule involves a calculation or a comparison with a number, NULL XBRL facts (xsi:nil=true or fact is absent) are treated as zero for the purposes of the calculation or comparison.

## Boolean checks

Where a validation rule includes a check of a form or taxonomy element that has a Boolean data type, it is assumed that the element is not NULL and if the element is NULL then the check does not occur. For example, the expression
 ([X] <> TRUE)
where X is an element with Boolean data type, should be interpreted as equivalent to
 ([X] = NULL OR [X] <> TRUE).

Other interpretations such as “([X] <> NULL AND [X] <> TRUE)” are not correct, unless explicitly specified in the rule.

## Use of domain in rules

Where a rule compares an XBRL fact to a range of values, this range may be defined as a ‘domain’. In this case the values of the domain will be as specified in the product-specific Validation Rules spreadsheet.

##  Case sensitivity

Validation rules that involve comparisons with string values are case insensitive. The case used in these rules often reflects the most common usage, however the case is not used in the comparison.

For example:

 IF <a> = ‘Australia’ would be true if <a> was ‘australia’, ‘AUSTRALIA’, ‘Australia’ or ‘AuStRaLiA’.

However, for enumerations that are in the SBR taxonomy, values must match the case of the enumeration code. This is a requirement of XML. Enumerations that are not defined in the taxonomy (for example: suffix codes) are not case sensitive.

## Tuples and context

In most SBR ATO interactions, all facts reported within a tuple instance (including nested tuples) use the same context. In some rare exceptions, there are facts within the same tuple that use different contexts.

##  Common modules

The common modules worksheets within each product-specific Validation Rules spreadsheet list a set of rules that apply to certain common modules that may be present within the product. With some rare exceptions, these same rules consistently apply regardless of the ATO product.

The following list contains examples of the common modules currently used within ATO business collaborations:

* addressdetails2.xx.xx:AddressDetails
* declaration2.xx.xx:Declaration
* electroniccontactelectronicmail1.xx.xx:ElectronicContactElectronicMail
* electroniccontacttelephone1.xx.xx:ElectronicContactTelephone
* financialinstitutionaccount1.xx.xx:FinancialInstitutionAccount
* organisationname1.xx.xx:OrganisationNameDetails
* organisationname2.xx.xx:OrganisationNameDetails
* perioddetails1.xx.xx.PeriodDetails
* personstructuredname3.xx.xx:PersonNameDetails
* personunstructuredname1.xx.xx:PersonUnstructuredName.

For example, for each incidence of an ‘addressdetails2.xx.xx:addressdetails‘ tuple within a message, the ‘addressdetails2’ set of common module rules will apply. Further ‘product-specific’ rules may also apply to that same tuple.

# **ATO structured english**

The validation rules are expressed in ATO structured English. The following table defines terms and characters used throughout these rules. Where that table refers to XBRL concepts, and a Logical Record is specified as being in JSON/XML format the corresponding equivalent JSON/XML concepts should be read in place of those XBRL concepts.

| **Structured English term** | **Intended interpretation** | **Examples** |
| --- | --- | --- |
| - (as in <a> - <b>) | Minus. | <a> - <b>Means value of ‘a’ minus the value of ‘b’. |
| - (as in SET(0-9) | Specifies a range of numeric or alphabetic values within a ‘SET’ construct. | = SET(0-3)Means is equal to 0, 1, 2 or 3.= SET(a-z)Means is equal to a, b, c, d …(etc.)… or z.DOES NOT CONTAIN SET(a-z)Means the field does not include any incidence of a lower case alphabetic character between a and z.  |
| & | Concatenate. Joins the value of a field to a literal or other field. | [TRT1]&”-06-30”Where [TRT1] is 2010, means 2010-06-30. |
| +/- | In numerical calculations, allows for an allowable deviation. | <a> <> <b> +/- 1Means (<a> > <b> + 1) OR (<a> < <b> - 1)<a> = <b> +/- 1Means (<a> >= <b> - 1 ) AND (<a> <= <b> + 1). |
| <> | Not equal to. | IF <a> <> <b>Means if the value of ‘a’ is not equal to the value of ‘b’. |
| {x} | A named (x) set.The use of a named set is required when representing context definitions that contain segment(s) that can have more than one possible value. | RP.{ForeignCountry} ‘{ForeignCountry}’ represents the set of possible context segment values defined by the ‘ForeignCountry’ dimension. This dimension can be either an explicit or typed dimension.When a reference to a specific instance of a set member is being made, the set name is used without curly braces, as in:FOR EACH ForeignCountry IN SET (RP.{ForeignCountry})Note: Where a list of explicit context definitions is defined, the notation “SET(RP.x,RP.y,RP.z)” is used.  |
| {x=y} | A specific value (y) in a named set (x) | Usage example: RP.{ForeignCountry = ‘us’} Refers to the context definitions where the foreign country context segment value = ‘us’. |
| ALGORITHM (with <Idtype> prefix) | Defines a test against a standard algorithm – as indicated by the <Idtype> prefix.<Idtype> can be ABN, TFN, TAN, ARBN or ACN. | IF TFNALGORITHM(<a>) = FALSEMeans if the TFN field <a> fails the TFN algorithm.IF ABNALGORITHM(<a>) = FALSEMeans the ABN field <a> fails the ABN algorithm. |
| ALL OCCURRENCES OF | All instances of a given field or defined set of multiple fields, where the field/s are from a repeatable tuple or context. (For testing values or sets of values in repeating tuples/contexts) | Usage examples:EXAMPLE 1: SUM of multiple fieldsIF SUM(ALL OCCURRENCES OF(<a> - <b>)) <> <c>.Means if the sum of every instance of <a>, minus the sum of every instance of <b> is not equal to the value of <c> (where <a> and <b> are from the same repeatable tuple/context).EXAMPLE 2: Single field conditionIF ALL OCCURRENCES OF(<a>) > 10Means if all instances of <a> are greater than 10 (where <a> is from a repeatable tuple/context).EXAMPLE 3: Single field with multiple conditionsIF ALL OCCURRENCES OF(<a>) = SET(NULL,0)Means if all instances of <a> are NULL or 0 (where <a> is from a repeatable tuple/context).EXAMPLE 4: Multiple field conditionsIF ALL OCCURRENCES OF(<c>) = (<c> WHERE (<a> = NULL AND <b> <> NULL))Means if all instances of <a> is NULL and <b> is not NULL in the same defined set (where both <a> and <b> are from the same repeatable tuple/context <c>). |
| AND | Logical AND. |  |
| ANY CHARACTER OF | Any character within a field. |  |
| (<context set>):ANY ELEMENT | Any element belonging to a context or set of contexts. | 1) RP:ANY ELEMENTMeans the set of all elements belonging to the RP context2) SET(RP,INT):ANY ELEMENTMeans the set of all elements belonging to either the RP or INT contexts.3) IF COUNT(RP:ANY ELEMENT <> NULL ) = 0 RETURN VALIDATION MESSAGE ENDIFMeans if all elements in the RP context are null then return error. |
| ANY OCCURRENCE OF | Any instances of a given field or defined set of multiple field conditions, where the field/s are from a repeatable tuple or context. (For testing values or sets of values in repeating tuples/context). | Usage examples:EXAMPLE 1: Single field conditionIF ANY OCCURRENCE OF(<a>) > 10.Means if any instance of <a> is greater than 10 (where <a> is from a repeatable tuple/context).EXAMPLE 2: Single field with multiple conditionsIF ANY OCCURRENCE OF(<a>) = SET(NULL,0)Means if any instance of <a> is NULL or 0 (where <a> is from a repeatable tuple/context).EXAMPLE 3: Multiple field conditionsIF ANY OCCURRENCE OF(<c>) = (<c> WHERE (<a> = NULL AND <b> <> NULL))Means if any instance of <a> is NULL and <b> is not NULL in the same defined set (where both <a> and <b> are from the repeatable tuple/context <c>). |
| ANY OTHER OCCURRENCE OF | For testing a value in one occurrence against other occurrences.For elements, used to check if any given value for a particular element is repeated in the same element, where the element is part of a repeatable tuple or repeatable context instance. For contexts, used to ensure context instances are unique where contexts with the same dimension segments are repeatable. | IF <a> = ANY OTHER OCCURRENCE OF <a>.Means if the value of element <a> from one instance of a tuple or repeatable context, is equal to the value of another instance of the tuple/context.IF CONTEXT(RP.{ForeignCountry}.{ActivityCode}) = ANY OTHER OCCURRENCE OF CONTEXT (RP.{ForeignCountry}.{ActivityCode}).Means if context instance RP.{ForeignCountry}.{ActivityCode}, is equal to any other context instance with the same dimension segment ForeignCountry and ActivityCode values. |
| CONTAINS | A string search for text within a field.Usage: <a> CONTAINS <B>. | IF <a> CONTAINS “UNKNOWN”.Means if <a> contains or equals the word ‘unknown’. |
| CONTAINS MORE THAN ONE | A text field contains more than one incidence of a given string with the field. | IF pyde.xx.xx:ElectronicContact.ElectronicMail.Address.Text CONTAINS MORE THAN ONE “@”.Means if there is more than one ‘@’ symbol within the email address. |
| CONTEXT | Used to refer to a context instance.Usage: CONTEXT(<A>)where <A> is a context instance abbreviation e.g. RP.GST.CC | WHERE CONTEXT = “RPI.Closing”.Means in instances where the context instance is ‘RPI.Closing’. |
| CONVERT\_TO\_DATE | Combines separate day, month and year fields into a single date.Usage:CONVERT\_TO\_DATE(<day>, <month>, <year>) | IF CONVERT\_TO\_DATE(<A>, <B>, <C>) <> VALID DATEWhere: <A> = the element capturing the day of the month, <B> = the element capturing the month, and <B> represents the element capturing a year. Means: if the values provided for <A>, <B> and <C> when concatenated, are not equal to a valid date. |
| COUNT  | A count of the number of occurrences of a field or context instance. | IF COUNT(RPI) > 1Means if the number of occurrences of the RPI context is more than 1.IF COUNT([FRM1] > 1Means if the number of occurrences of the element [FRM1] is more than 1.IF COUNT(ForeignCountry) IN SET (RP.{ForeignCountry}.{ActivityCode}, RP.{ForeignCountry}) >3RETURN VALIDATION MESSAGEENDIF.Means the count of distinct Foreign Country segment values across all context instances matching ‘RP.{ForeignCountry}.{ActivityCode}’ or ‘RP.{ForeignCountry}’. |
| COUNT(SCHEDULE) | Return the number of occurrences of a schedule that is attached to a parent return.Usage: COUNT(SCHEDULE = <A>).Where <A> is a schedule abbreviation e.g. DIS, IEE. | IF COUNT(SCHEDULE = “IEE”) > 50.Means if the number of occurrence of an IEE schedule in the business document is greater than 50. COUNT(SCHEDULE = IEE) = 0.Means that there are no IEE schedules attached to the form being validated. |
| CURRENT FINANCIAL YEAR | The year that 30 June next occurs relative to the current system date. | IF <a> > CURRENT FINANCIAL YEAR.If current system date is 01/08/2013, for example, means IF <a> is after 2014. |
| DATE(TODAY) | Compares a date against the current date. | IF <a> > DATE(TODAY).Means if <a> is a date in the future. |
| DIMENSION | Test against a specific set of metadata for a particular context. | IF (RprtPyType.xx.xx:ReportingPartyTypeDimension = “RprtPyType.xx.xx:Intermediary”)Means if the Reporting Party Type context is ‘Intermediary’.  |
| DOES NOT CONTAIN | An element has no instance of the stated value or set of values. | DOES NOT CONTAIN SET(“a-z”, “A-Z”, “0-9”).Means that the field has no instance of an alphabetical character (excepting special characters), nor a numeric character. |
| DOMAIN | A globally defined set of values.Usage: <a> = SET(DOMAIN(<B>)).<a> is one of the values defined in <B>. | SET (DOMAIN(COUNTRY CODES)Means the complete set of country codes. Each set of domain values is defined in the Standard enumerations section within this document. |
| ENDSWITH | A string searches for text at the end of a fieldUsage: <a> ENDSWITH <B>. | IF <a> ENDSWITH “ T/A”.Means the condition is true if field <a> contains a value that ends with the text string ‘ T/A’.  |
| FOR ANY OCCURRENCE OF <object> | An instruction to process each instance of a repeating object, one at a time. | FOR ANY OCCURRENCE OF CONTEXT (RP) <test condition>Means for every occurrence of context RP, apply the <test condition>. |
| FOR EACH <object> IN SET(…) | An instruction to process each object in a set/collection of objects, one at a time.  | FOR EACH ForeignCountry IN SET (RP.{ForeignCountry})<test condition>.Means for each unique ‘ForeignCountry’ segment value, apply the <test condition>. |
| FOUND | A string search for text within a field performing a variation of the SET, CONTAINS, STARTSWITH and ENDSWITH functions.Includes the following tests:1. exact match,
2. match with a space on each side of the variable,
3. match with a space after the variable, and
4. match with a space before the variable.

Where multiple strings have been provided, a check for each string is performed.Usage: <A> = FOUND(<B>,<C>). | IF <a> = FOUND(“The trustee”,”The Exec”).Means if <a>:* equals ‘The trustee’ or ‘The Exec’ (exact match), or
* contains ‘ The trustee ’ or ‘ The Exec ’ (a space on each side of either string), or
* starts with ‘The trustee ’ or ‘The Exec ’ (with a space after either string), or
* ends with ‘ The trustee’ or ‘ The Exec’ (with a space before either string).
 |
| INCREMENTAL SEQUENCE | For testing a repeating element is unique and in sequential order with no gaps in the sequence.The element can be either a numerical value or an alphanumeric string with defined pattern. | IF ANY OCCURRENCE OF <a> <> INCREMENTAL SEQUENCE OF ANY OTHER OCCURRENCE OF <a> Means that each occurence of <a> is unique and in sequential order with no gaps in the sequence. For a repeating numerical value <a>, the above expression would result in the following outcomes:Where <a> = 1,2,3,4,5 - valid Where <a> = 2,1,3,4,5 - invalid (not in sequential order)Where <a> = 1,3,4,5,6 - invalid (missing a value in sequence) |
| IN TUPLE | Restricts a test to the value of a field within a particular tuple (where the field may exist in more than one tuple).(Refer also: ‘WHERE IN TUPLE’). | (1) IF <a> IN TUPLE(<b>).Means if the value of <a> within the tuple <b>. (Where <a> may also exist outside tuple <b>). (2) IF (<B> IN TUPLE(<A>)) = NULLORBLANKMeans if tuple <A> does not exist or if <B> in <A> is null or blank.(3) IF COUNT(<B> IN TUPLE(<A>)) > 1.Means if the occurrence of <B> in <A> is more than one.(4) IN TUPLE(<A>) IF <B> = NULLORBLANKMeans if <B> in the tuple <A> is null or blank. In this example the rule will only trigger if <A> exists. |
| LENGTH | Used to define the constraints on the length of a field.(Refer also: TEXT). | IF LENGTH(<a>) < 6.Means if the value of <a> does not contain at least 6 characters.  |
| MONETARY() | Defines a monetary field pattern where a true response is given when a value passes all conditions.As in: MONETARY(<a>,<b>,<c>)Where:<a> = S or U to indicate if field can be signed or not.<b> = Maximum number of digits (including decimal places).<c> = Maximum number of decimal places.For <a> an S indicates a field can be prefixed with a ‘+’ or ‘-‘ sign, but may be omitted. Where <a> is U, the field must not be prefixed with a sign.The value of <b> does not include a decimal point or sign in the total character limit. | <a> <> MONETARY(U,11,0).Means <a> is not equal to a number in the range 0 to 99999999999, or includes a character other than 0 to 9.<a> <> MONETARY(S,11,0).Means <a> is not equal to a number in the range -99999999999 to 99999999999, or includes a character other than 0 to 9, or ‘+’ or ‘–‘ as the first (left-most) character.<a> <> MONETARY(U,13,2).Means <a> is not equal to a number in the range 0.00 and 99999999999.99, or includes a character other than 0 to 9 or a decimal point. (Decimal point may be omitted). |
| NOT | Reverses the value of an element i.e. turns TRUE to FALSE and vice versa. |  |
| NULL | Fact is not there, or has been specified to be null with xsi:nil indicator or is a null non-textual value. | IF <a> = NULL.Means if a (non-textual) value for <a> is blank or if <a> does not exist.  |
| NULLORBLANK | Fact is not there, is null with xsi:nil = true or is a null string.(Applied to Text, Code, Description, and Identifier facts).  | IF <a> = NULLORBLANK.Means if a (textual) value for <a> is blank or if <a> does not exist. |
| NUMBER | Defines a valid numeric field pattern where a true response is given when a value passes all conditions.Usage: NUMBER(<a>,<b>,<c>)Where:<a> = S or U to indicate if field can be signed or not.<b> = Maximum number of digits (including decimal places).<c> = Maximum number of decimal places.For <a> an S indicates a field can be prefixed with a ‘+’ or ‘-‘ sign, but may be omitted. Where <a> is U, the field must not be prefixed with a sign.The value of <b> does not include a decimal point or sign in the total character limit. | <a> <> NUMBER(U,11,0).Means <a> is not equal to a number in the range 0 to 99999999999, or includes a character other than 0 to 9.<a> <> NUMBER(S,11,0).Means <a> is not equal to a number in the range -99999999999 to 99999999999, or includes a character other than 0 to 9, or ‘+’ or ‘–‘ as the first (left-most) character.<a> <> NUMBER(U,13,2).Means <a> is not equal to a number in the range 0.00 and 99999999999.99, or includes a character other than 0 to 9 or a decimal point. (Decimal point may be omitted). |
| NUMERIC | Contains only digits, 0..9 |  |
| OR | Logical OR |  |
| PARENT RETURN | Used on schedules to refer to the main ‘parent’ return associated with the schedule. | IF <a> <> PARENT RETURN(<a>).Means if the value of <a> on the schedule is not equal to the value of <a> on the main form.WHERE PARENT RETURN EXISTS.Means apply the test if the business document includes a schedule associated with the main form. |
| POS | The POS function is used to return ZERO for negative numbers, and the actual value for positive numbers.When value is NULL, function will return ZERO.Usage: POS(<a>) | **Examples:**POS([IITR321]) where; [IITR321] = -52, result is 0 [IITR321] = 0, result is 0 [IITR321] = <NULL>, result is 0 [IITR321] = 1001, result is 1001 [IITR321] = 99.50, result is 99.50 |
| ROUNDDOWN | Round a number down to the nearest specified increment.Usage: ROUNDDOWN(<A>,<B>)Where <A> is the configurable increment to round down to<B> is the value that is to be rounded down. | ROUNDDOWN(0.05,[CTR120])Means round the value of CTR120 to the .05.In this example, if [CTR120] = 99.93, the amount to be supplied for CTR120 is 99.90. |
| SCHEDULE | Used on a main ‘parent’ form to refer to a schedule that could be attached to the parent return.In terms of SBR, a schedule refers to a business document submitted within the same standard business document body structure as a business document for a main tax return form. Usage: SCHEDULE = <A>. | IF COUNT(SCHEDULE = “S25A”) = 0.Means if there is no instance of a Schedule 25A included in the business document body.IF COUNT(SCHEDULE = “RSPT”) > 50.Means if there are more than 50 occurrences of a Rental schedule in the business document body. |
| SET | Defines an explicit set of values, where if one value meets the criteria for comparison, a true response is given. | IF <a> <> SET(“a”,”b”,”c”).Means if <a> does not equal a or b or c.IF <a> = SET(“a”,”b”,”c”)Means if <a> is equal to a or b or c.IF <a> = SET(0-3).Means if <a> is equal to 0 or 1 or 2 or 3 |
| STARTSWITH | A string searches for text at the start of a field.Usage: <a> STARTSWITH <B> | IF <a> STARTSWITH “T/A”.Means the condition is true if field <a> contains a value that starts with the text string ‘T/A’  |
| SUM | The sum of all instances of an element.Usage: SUM(<A>), where <A> is an element that appears in a repeating tuple or is a repeating element. | SUM(<a>)Means the total value of all instances of <a>, when each <a> is added up.  |
| TEXT() | Used to define the maximum length of a textual field.Definition of a valid text field pattern where a true response is given when a value passes all conditions.Usage: TEXT(<a>)Where <a> = Maximum number of charactersTRUE if the tested field is less than or equal to length <a>(Refer also: LENGTH) | <a> <> TEXT(150).Means the maximum number of characters allowable within field <a> is 150. |
| TUPLE | Used to signify that the element (in brackets) is a tuple – a ‘container’ that may contain a group of two or more related data elements. | TUPLE(addressdetails2.xx.xx:AddressDetails)Means the fields that have been defined as belonging to the ‘addressdetails2.xx.xx:AddressDetails’ module. |
| VALID DATE | A date that conforms to the xbrli:dateItemType datatype. | IF CONVERT\_TO\_DATE <> VALID DATEMeans if the converted date is not a valid date. |
| WHERE (tuple/context/set) | Indicates that the rule execution is dependent on the tuple, context or set existence. | Refer WHERE IN CONTEXT, WHERE IN TUPLE and WHERE…IN SET. |
| WHERE IN CONTEXT | Used to validate data elements which are logically grouped within a context.Rule is executed within the bounds of each occurrence of a particular context.Usage:WHERE IN CONTEXT(<A>) IF <B>…. | WHERE IN CONTEXT(RP) IF <B> = NULLORBLANK(Where <B> is a fact in the RP context).Means if the value of <B> within the RP context is null or blank. In this example the rule will only trigger if the RP context exists and if <B> (in the RP context) is null or blank.  |
| WHERE …IN SET | Used to validate data elements which are logically grouped within a set.Rule is executed within the bounds of each occurrence of a set.Usage:WHERE <A> IN SET(<B>).Refer also: {x}. | WHERE ForeignCountry = ’us’ IN SET (RP{ForeignCountry})Means execute the rule if the ForeignCountry dimension value is ‘us’ in the ForeignCountry dimension within the RP context. |
| WHERE…. IN TUPLE(element definition) | The WHERE and IN TUPLE keywords are used when tuple element explicits are required.The element including the tuple definition is to be considered as a whole. | (RP:pyde.xx.xx:AddressDetails.Line1.Text WHERE(TUPLE ELEMENT Address Usage = “BUS”) IN TUPLE(address2)) = NULLORBLANK.Means Line 1 of the business address of the Reporting Party is not present.Rule will trigger if:* RP context is not present, or
* address2 tuple is not present, or
* address2 tuple with Address Usage = ‘BUS’ is not present, or
* Line1.Text in the address2 tuple with Address Usage = ‘BUS’ is not present.
 |
| WHERE IN TUPLE | Used to validate data elements which are logically grouped within a tuple.Rule is executed within the bounds of each occurrence of a tuple.Usage:WHERE IN TUPLE (<A>) IF <B>….(Refer also IN TUPLE. The ‘WHERE’ keyword is optional). | WHERE IN TUPLE(<A>) IF <B> = NULLORBLANK(Where <B> is a fact in tuple <A>)Means if the value of <B> within the tuple <A> is nullorblank. In this example the rule will only trigger if tuple <A> exists and if <B> (in <A>) is null or blank.  |
| Xbrli (\element definition) | ‘xbrli’ is used to denote the reporting taxonomy root. Indicates a given tuple or element is not nested within another tuple.This is used to differentiate elements that are repeated at different levels of the reporting taxonomy within a given business collaboration.  | IN TUPLE (xbrli\organisationname2.xx.xx:OrganisationNameDetails).Means in the tuple ‘organisationname2.xx.xx:OrganisationNameDetails’ that is **not** within another tuple.In this example, the implication is that the ‘organisationname2.xx.xx:OrganisationNameDetails’ tuple also exists under another tuple within the product.  |

# **Validation rules spreadsheets**

The ATO supplies the validation rules (VRs) within Excel spreadsheets. For each ATO product there is one or more spreadsheet(s) covering the rules for that product. This may include several worksheets:

* one containing rules specific to the product;
* zero, one or more describing the common module rules that apply to the product; and
* zero, one or more containing the domain definitions.

All rules are applied by the ATO, so that if a business message does not comply with a validation rule (other than a warning), the message will be rejected.

Where common module rules are included, these apply for every instance of the common module (tuple) within the product.

Rules implemented via XBRL report design are generally not specified within the *Validation rules* spreadsheets.

If an error message is returned as a result of the validation rule, the element against which the rule is specified will be referenced in the location path of the response message.

Where a given validation rule involves more than one element (such as with cross-field and cross-form rules) the rule is specified only once against the element to which a location path will be returned.

The following table describes each column in the *Validation rules* spreadsheets.

| **Column label** | **Description** |
| --- | --- |
| **Seq Num** | This is the sequence number of the fact, heading or tuple as defined in the MST.  |
| **Alias** | The abbreviated identifier for the fact as defined in the MST.  |
| **Label** | The label of the fact, heading, tuple or context information as defined in the MST. |
| **Context instance** | The context instance for a given fact or tuple.  |
| **Element Name** | The name of the definitional taxonomy element, including namespace prefix. |
| **English Business Rule** | A non-technical explanation of the rule. |
| **Legacy Rule** | Used for transition during the 2016/17 financial year.Specifies the validation rule using Structured English.*Please refer to section 3.4 for details on the new Technical Business Rule format and Transition Plan* |
| **Technical Business Rule** | For new services and those being transitioned during the 2016/17 financial year onwards, specifies the validation rule using the new Technical Business Rule formatFor existing service specifications, this column specifies the validation rule using Structured English.*Please refer to section 3.4 for details on the new Technical Business Rule format and Transition Plan* |
| **Applies to XBRL Payloads** | Identifies if the validation rule applies to XBRL payloads. Valid values are: ‘Y’, ‘N’ or ‘n/a’Y - rule is applied to a XBRL payloadN - rule is not applied to a XBRL payloadn/a - service does not use XBRL payloads |
| **Applies to XML Payloads** | Identifies if the validation rule applies to XML payloads. Valid values are: ‘Y’, ‘N’ or ‘n/a’Y - rule is applied to a XML payloadN - rule is not applied to a XML payloadn/a - service does not use XML payloads |
| **Applies to JSON Payloads** | Identifies if the validation rule applies to JSON payloads. Valid values are: ‘Y’, ‘N’ or ‘n/a’Y - rule is applied to a JSON payloadN - rule is not applied to a JSON payloadn/a - service does not use JSON payloads |
| **Rule Type** | Indicates the type of rule – Context, Format, Enumeration, Mandatory, Calculation, CrossForm, or CrossField. These, in part, determine the validation phase, as described above in section 3.4.2. |
| **Schematron ID** | The Schematron ID for the rule. |
| **Message Code** | The response message code (identifier) returned when a request message does not comply with the rule. The messages returned by the ATO are published in the *ATO response messages* XML file on the SBR website. |
| **Message – Short Description**  | The response message short description corresponding to the Message Code. An additional detailed description may be available for the message code which is available in the ATO Response message repository. |
| **Last Updated**  | The date the rule was added or last changed since the prior version or prior business collaboration (where applicable). |

# **Previous Version Control**

| **Version** | **Date** | **Description of changes** |
| --- | --- | --- |
| 1.3 | 14/02/2019 | Updated Section 4 ATO STRUCTURED ENGLISH. Added new structured English term - ‘Incremental Sequence’. |
| 1.2 | 19.07.2018 | Section **2.2.1.1 XBRL Validation** updated to include SBR Message Code CMN.ATO.GEN.XBRL10 - The logical record or logical document did not pass XBRL validation. Please contact your software provider.Section **2.2.1.3 XML Validation** updated to include SBR Message Code CMN.ATO.GEN.XML10 The logical record or logical document did not pass XBRL validation. Please contact your software provider. |
| 1.1 | 07.06.2018 | Section **2.2.1.3 XML Validation** added. |
| 1.0 | 23.11.2017 | The information contained in this document was previously located in the ATO Common Message Implementation Guide (cMIG) version 3.0 dated 15th September 2016 and has been removed to this stand-alone document. |