#### useBraiins BroSets

20 July 2013

Contents

[1. Abbreviations or Acronyms 3](#_Toc362062099)

[2. BroSets 4](#_Toc362062100)

[3. Bros 5](#_Toc362062101)

[3.1 DataBase Objects and Bros 7](#_Toc362062102)

[3.2 Slave Bros 8](#_Toc362062103)

[3.3 Bro References 10](#_Toc362062104)

[3.4 Terms to Describe Bros 11](#_Toc362062105)

[4. Multiple Taxonomy, Jurisdiction, and EntityType Options 12](#_Toc362062106)

[5. Working with BroSets 13](#_Toc362062107)

[5.1 BroSet Import 13](#_Toc362062108)

[5.2 BroSet Export 13](#_Toc362062109)

[6. BroSet Spreadsheets 14](#_Toc362062110)

[6.1 Column Headings Row 14](#_Toc362062111)

[6.2 Comment and Blank Rows 14](#_Toc362062112)

[6.3 BroSet Statements 14](#_Toc362062113)

[6.4 Defined Bro Statements 17](#_Toc362062114)

[6.5 Include BroSet Statements 17](#_Toc362062115)

[6.6 Bro Rows 18](#_Toc362062116)

[7. The Bro Columns 19](#_Toc362062117)

[7.1 BroId 19](#_Toc362062118)

[7.2 Type 20](#_Toc362062119)

[7.3 Level 20](#_Toc362062120)

[7.4 BroName 20](#_Toc362062121)

[7.5 Name0 and N1-N8 20](#_Toc362062122)

[7.6 ShortName 21](#_Toc362062123)

[7.7 Master 21](#_Toc362062124)

[7.8 Ref 21](#_Toc362062125)

[7.9 Folio 21](#_Toc362062126)

[7.10 DataType 22](#_Toc362062127)

[7.11 DboField 23](#_Toc362062128)

[7.12 Sign 23](#_Toc362062129)

[7.13 AcctTypes 23](#_Toc362062130)

[7.14 PostType 23](#_Toc362062131)

[7.15 RO 23](#_Toc362062132)

[7.16 ExclProps 24](#_Toc362062133)

[7.17 AllowProps 24](#_Toc362062134)

[7.18 Members 24](#_Toc362062135)

[7.19 SumUp 26](#_Toc362062136)

[7.20 Check 27](#_Toc362062137)

[7.21 Related 27](#_Toc362062138)

[7.22 Period 28](#_Toc362062139)

[7.23 StartEnd 28](#_Toc362062140)

[7.24 Zones 28](#_Toc362062141)

[7.25 Order 29](#_Toc362062142)

[7.26 Descr 29](#_Toc362062143)

[7.27 Taxonomies 29](#_Toc362062144)

[7.28 Countries 29](#_Toc362062145)

[7.29 EntityTypes 30](#_Toc362062146)

[7.30 Comment 31](#_Toc362062147)

[7.31 Scratch 31](#_Toc362062148)

[7.32 I UsableProps 31](#_Toc362062149)

[7.33 I Sum UsableProps 31](#_Toc362062150)

[7.34 I M Use Member Info 31](#_Toc362062151)

[7.35 I StartEnd 31](#_Toc362062152)

[7.36 I Slave Ids 31](#_Toc362062153)

[7.37 I Slave Filtering 31](#_Toc362062154)

[8. Version Notes 32](#_Toc362062155)

[8.1 24 June 2013 Initial Release 32](#_Toc362062156)

[8.2 1 July 2013 Release 32](#_Toc362062157)

[8.3 5 July 2013 Release 32](#_Toc362062158)

[8.4 20 July 2013 Release 32](#_Toc362062159)

# Abbreviations or Acronyms

Abbreviations or Acronyms used in this document are:

| **Term** | **Description** |
| --- | --- |
| BRO or Bro | Braiins Report Object |
| BroRef | Bro Reference |
| DBO | DataBase Object as described in the Braiins Admin SIM Members page |
| DE | Double Entry |
| Member | Property Member as listed at the Braiins Admin SIM Members page |
| MemberRef | Property Member name used as part of a Bro Reference |
| Prop | Property |
| RG | The Braiins Report Generator |
| RO | ReportOnly |
| SIM | Standardised Information Model, the Braiins way of organising and managing data of which BroSets are a part. |
| SFR | Semantic Financial Reporting, the reporting part of Braiins which makes use of SIM data |
| Sch | Schedule |
| SE | StartEnd |
| SS | SpreadSheet |

# BroSets

Braiins BroSets are sets or collections of Bros (Braiins Report Objects) which are the means by which Braiins data is organised, stored, accessed, summed, and output, including in XBRL form. Bros and BroSets form the Braiins intelligent equivalent of a traditional chart of accounts.

BroSets have a Name and Type as well as other attributes described in section *6.3 BroSet Statements*.

Full (or Main) BroSets for use by an entity are intended to be built up from smaller BroSets for particular data/reporting areas e.g. Fixed Assets. BroSets may be dynamically included/excluded at the time of use for a particular taxonomy, jurisdiction (country), and entity type.

There are two basic types of BroSet, In and Out:

In-BroSets provide data storage (after **In**put), summing, and non-XBRL output. They are “universal” in that they are intended to be general, not specific to any particular taxonomy, jurisdiction, or entity type, though they can encompass taxonomy, jurisdiction, or entity type variations.

In-BroSets work with SIM Folios, Properties, and Members.

Out-BroSets are taxonomy specific and use data from In-BroSets to provide XBRL **Out**put.

Out-BroSets work with the taxonomy’s Hypercubes, Dimensions, Dimension Members, and Tuples.

Both BroSet types include two sub-types:

Incl-BroSets are building block BroSets intended to be **Incl**uded by a Main-BroSet to build up a full BroSet. An Incl-BroSet cannot be used by itself for posting or report generation. An Incl-BroSet can include other Incl-BroSets, any number of them, to any depth.

Main-BroSets are full BroSets intended for In and Out BroSet use as described above. A Main-BroSet can be complete in itself, but preferably for ease of development, maintenance, re-usability, and to take advantage of dynamic inclusion/exclusion of BroSets by taxonomy, jurisdiction, and EntityType, Main-BroSets should be built up from included Incl-BroSets.

An Out-Main-BroSet must include one and only one In-Main-BroSet as its data source. (It can include multiple Out-Incl-BroSets and the In-Main-BroSet can include multiple In-Incl-BroSets.)

Main-In-BroSets are the only ones which a user will ever “see”.

BroSets are maintained in Spreadsheets which are imported into Braiins.

The remainder of this version of the document describes In-BroSets only.

# Bros

Bros (Braiins Report Objects) are the “accounts” of Braiins. They are organised or grouped into BroSets as described in section *2. BroSets*.

Bros are processed by the Braiins Bro Engine, named BroClass internally, using OOP (Object Oriented Programming) techniques.

Bros are intended to be used for main accounting concepts, not details. Bros should be relatively few in number versus the number of accounts in a traditional 5 or 7 digit code general ledger.

The detail is covered by means of any number of SIM Property Members which may be attached to a piece of data to describe it fully i.e. to add the detail. Property Members are themselves grouped into Properties which are in turn grouped into Folios (groups of Properties). See the Braiins Admin SIM Folios, Properties, and Members pages. A Bro Reference or BroRef is used to refer to a particular Bro or piece of data held in a Bro as described in section *3.3 Bro* References.

Conceptually a Bro encompasses both knowledge about its place in the world; what it can hold, how it relates to other Bros, how to process its data; as well as the data itself, which may comprise many individual pieces of data.

Bros have a Name and optionally a ShortName, both of which must be unique across all BroSets that are pulled together to form a larger BroSet.

Bros also have a numeric reference in the range 1 to 99999 called a BroId which is used internally by Braiins for speed and efficiency reasons instead of the Bro name. BroIds are allocated automatically by BroSet Import, but groups of Bros can be given BroIds within a range by giving them a starting BroId value. BroIds must also be unique across all BroSets that are pulled together to form a larger BroSet.

The requirement for BroId uniqueness means that BroId ranges should be planned before creating BroSets and Bros. The 1 to 99999 available range provides plenty of room for this.

Overlapping BroId ranges can be used for different BroSets that will never be used together e.g. for two different versions of a TFA BroSet.

Bros are organised in a tree structure using Sets (or nodes) and Elements. An element is always a member of a set i.e. it always has a parent. A member of a set can be another set and this can continue to any depth or level, though currently BroSet export and import allow for only up to 9 levels.

A Set Bro cannot cross BroSets i.e. a Set Bro must be complete within one BroSet.

A Bro holds one value per period for the so called ‘Base’ Bro (the Bro without any property references), and additional values for each member reference value that has been posted.

Slave Bros are a special type of Bro which may be used to copy Bro information, optionally filtered, to another branch of the Bro Tree to enable natural and complete tree construction. Slave Bros are described in section *3.2 Slave* Bros.

Bros have a DataType to define what kind of data they can store and process. There are 11 different DataTypes (Money, String, Date, Boolean, Integer, Decimal, Enum, Share, PerShare, Percent, MoneyString) but the most common ones are Money whose use is obvious from its name, and String for text. Money values are stored as debit (+ve) or credit (-ve) integer (whole number, no cents/pence) balances.

Element Bros can hold data. Set Bros may also hold data if defined accordingly. A Set Bro that has a summing numeric DataType (Money, Integer, Decimal, Share) often but not necessarily holds the sum of its children.

**Posting Bro Uniqueness**Any possible posting must be unambiguous, which is achieved simply by virtue of Bro Names (or ShortNames) being unique.

**Set and Set Member Posting Mutual Exclusivity**Sets and Set Members are mutually exclusive for posting i.e. a Bro (whether set or element) cannot be posted to if any ancestor has been posted to, and a Set Bro cannot be posted to if any ancestor or descendant has been posted to.

**Property Posting Mutual Exclusivity**Posting to a Bro with and without a member reference are mutually exclusive.

**Member Posting Mutual Exclusivity**Some Members are also mutually exclusive as shown in the Braiins Admin SIM Members listing.

**Automatic Summing**Automatic summing is performed for Summing Type Bros (those of type Money, Integer, Decimal, and Share) in three ways:

* within a given Bro where the Base value for the Bro is the sum of the member values
* for members that are summed as shown in the Braiins SIM Members listing
* up the Bro tree via Sets (for both base and member values, subject to allowable property restrictions) unless stopped via a ‘No SumUp’ property.

## DataBase Objects and Bros

DataBase Objects (DBOs) are used by Braiins to hold Entity, People, Address, and Contact data that can apply to multiple entities of an Agent. (DBOs do not cross Agents.)

DBOs provide more natural groupings of information than the XBRL approach, whilst allowing more flexibility, especially as to naming, cross entity usage for Groups, and avoiding any limits on numbers.

DBO data is accessible to Bros and the Report Generator either directly or via Bros.

Some SIM Properties and Members also make use of DBOs. See the Braiins Admin SIM Members page and its notes for more detail.

**DBO Read Bros**

A Bro can have read access to DBO data by means of the Bro column *7.11 DboField*. DBO Read Bros are automatically set to being RO.

Typically DBO Read Bros would only be used for a Master Bro that is to be copied to an Out-BroSet Slave Bro. [Note to Charles: This may change depending on how mapping to XBRL ultimately gets done.]

**DBO Posting Bros**

If DBO data is potentially to be changed or added to via posting, then an In-BroSet should include one Bro, called a DBO Posting Bro, for each such DBO. There is no need to add such DBO Posting Bros for DBOs that will not be changed via posting e.g. for a person.

DBO Posting Bros must have a DataType of Boolean and just one UsableProp, the DBO Property, or a Members column value which allows a member for the DBO Property. The members for such Bros can then be set or unset via posting.

Only SIM Members can be set via posting, not base DBO data such as an entity name, or a person’s name, as that kind of data is managed via the Braiins Desktop, which is why DBO posting Bros must have a DataType of Boolean.

The DboField column cannot be used with DBO Posting Bros.

**DBO Property Use**

DBO Properties can also be used to describe other data e.g. accounting data such as which Officer a remuneration posting applies to. For Bros using a DBO Property in this way the DBO Property will always be one of a number of AllowProps for the Bro, which tells Braiins that the Bro is not a DBO Posting Bro. With such Bros only the Ref member can be used e.g. Officer.BondJ. All other members of the DBP Property, if any, are automatically excluded.

## Slave Bros

A Slave Bro is a Bro which copies or mirrors (except for a Summing Set Slave – see below) the values (or a filtered subset of the values) of another Bro, known as its Master Bro, in a different location in the Bro tree, to facilitate natural tree structures, and summing. The values which are copied from Master to Slave may be filtered by folio, property, or member and Slave Bros may be included/excluded by taxonomy, jurisdiction, or entity type as for other Bros.

The copying may be for the values of a prior year – 1, 2, or 3 years backwards in time from the current year. This allows cross year figures to be introduced into Bros for start/end and movement purposes. Such Bros are called Prior Year Slaves. A Money Prior Year Slave must have a PostType of Sch as such cross year figures cannot be DE. BroSet Import sets Money Prior Year Slaves to Sch if not input as Sch.

If a Slave Bro has a Folio (it doesn’t have to), it must be either the same as the Folio of its Master, or a subset of the Master’s Folio, if the Master has one. If no Folio is input for a Slave, it inherits its Master’s Folio, if any.

A Slave Bro is always RO as slaves cannot be posted to directly.

A Slave cannot also be a Master, though a Master can have multiple Slaves.

If the Master is a Start/End Bro both the Start and End values are copied to the Slave. If just the Start values are needed these may be obtained in another Bro defined as a Prior Year Slave.

Master values are not copied to a Summing Set Slave. Instead the set is summed as for any other set and the result of the summing is checked for equivalence with the Master via an ‘Equal To Either{ Year#} Master Bro Name’ Check which is automatically added to the Bro by BroSet Import if no slave filtering is involved.

Slave attributes which can be entered for a Slave and which can differ from those of the Master unless noted otherwise are:

* Level (via the number of segments in the Name)
* Type (Set or Ele)
* Name (must be different)
* ShortName (must be different)
* Ref
* Folio (can be a subset of the Master’s)
* Sign (for a Money Bro, with a warning)
* AcctType
* PostType (cannot be DE if the Master is Sch but can be Sch for a DE Master. Must be Sch for a Money Prior Year Slave.)
* RO (always set for the Slave)
* SumUp
* Check
* Period (must be the same as the Master’s but can be entered for the Slave as a check)
* StartEnd (must be the same as the Master’s but can be entered for the Slave as a check)
* Zones
* Order
* Descr
* Taxonomies
* Countries
* EntityTypes
* Comment

Scratch

Filtering properties for Slaves are:

* Folio (since this can be a subset of the Master’s)
* ExclProps
* AllowProps

Members

Property always inherited from the Master

* DataType

Properties inherited from the Master, if not set for the Slave:

* Folio
* Sign
* Period
* StartEnd

## Bro References

In formats and when importing/posting, Bros are referenced by Name or ShortName, optionally with property member references and/or a start|end value as follows:

<BroName | BroShortName>{,MemberRef...}{,<start | end>}

optionally with spaces between segments, where

BroName is a full Bro name, including dots between level sections of the name

BroShortname is a Bro’s ShortName if defined

The optional member references and/or a start|end value {,MemberRef...}{,<start | end>} are comma separated and may come in any order i.e. even with {,<start | end>} in the middle of a series of member references.

MemberRef is a member reference of PropertyName.MemberName as listed in the Braiins Admin SIM Members page.  
  
For posting, a “#” MemberName must be replaced by an actual DBO reference.  
  
For RG use the ".MemberName" part is optional if used with a numeric Bro, meaning the sum of all the Member values. This applies to .# and well as explicit Members.   
  
The “PropertyName.” part of a MemberRef can be omitted if there is a ".MemberName" section and if there is a MemberRef to the left of this MemberRef which includes the desired “PropertyName.” e.g.  
EntityInfo.Officers, Officer.WoodgateC, Director, Exec, Chairman, SignAccounts  
rather than  
EntityInfo.Officers, Officer.WoodgateC, Officer.Director, Officer.Exec, Officer.Chairman, Officer.SignAccounts  
  
There can be any number of comma separated MemberRefs in any order, but not duplicates.   
  
MemberRefs are converted to PMemIds for internal use by Braiins.  
  
Validity checks are applied to MemberRefs e.g. versus the Bro's usable properties, potentially as modified for global options, etc.  
  
Mux checks also apply.

,<start | end>  
‘,start’ or ‘,end’ can be used to specify start or end (opening or closing) period for SumEnd, PostEnd, or Stock type StartEnd Instant Bros, and only for such Bros.  
  
The default is end, so use of end is optional.  
  
Start for posting is only applicable for the first year of data in Braiins i.e. for a year without a prior year for Braiins to obtain the start value from. If used when posting to a year which does have prior year values, the value is ignored. Start for reading via the RG can be used in any year.   
  
end is RO for SumEnd Bros

## Terms to Describe Bros

Bros are described using various terms from the following list, sometimes prefixed with ‘non-‘. More details on some of these terms and others are given in section *7. The Bro Columns*.

| **Term** | **Description** |
| --- | --- |
| Set {Bro} | Bro which is a Bro tree node with child Bros which can be Sets or Elements. Slave Sets cannot be empty. |
| Element or Ele {Bro} | Stand-alone Bro which is a member of a Set. |
| DataType | Bros can have a DataType and sometimes the DataType is used to describe a Bro e.g. a Money Bro. |
| Summing | Numeric Bro with a DataType of Money, Integer, Decimal, or Share that can be summed automatically by Braiins if set to SumUp. |
| SumUp | Bro that is set to sum up to its Set parent i.e. one for which the SumUp column = ‘+’ |
| Numeric | Bro with a numeric DataType, which is all of them except for String. See Bro Column *7.10 DataType.* |
| DBO | DBO {<Read | Posting>} Bro as described in section *3.1 DataBase Objects and Bros* |
| DboField | Bro with DboField defined = a Bro for read access of DBO data |
| DE or Sch | Bro with a PostType of DE (Double Entry) or Sch (Schedule). |
| RO or ReportOnly | An RO or ReportOnly Bro can only be used for reporting purposes and cannot be posted to i.e. it is a Non-Posting Bro. Element Bros which are not Slaves cannot be RO as otherwise there would be no way to get data into them. |
| Non-Posting | Another way of saying ‘ReportOnly’ |
| Posting | A Bro which can be used for Posting, and is not ReportOnly |
| Instant | A Bro with Instant period which is one whose data applies to a specific date. |
| Duration | A Bro with Duration period which is one whose data applies to a period. |
| StartEnd | An Instant Bro which has Start and End or opening and closing values. |
| Standard | A standard or normal Bro which is not a Slave. |
| Slave | A Slave Bro is a Bro which copies the values (or a filtered subset of the values) of another Bro, known as its Master Bro, in a different location in the Bro tree. |
| Master | A Master Bro is a Bro which has one or more Slave Bros replicating its values. It can be a Set or an Element. |

# Multiple Taxonomy, Jurisdiction, and EntityType Options

BroSets should be constructed to be general or independent of Taxonomy, Jurisdiction, and EntityType as much as possible. Differences such as with/without Tax entries (different sets of books), would be covered via dynamic Journals/DataSets, not BroSets.

However, for those cases where tweaking for relatively small Taxonomy, Country (Jurisdiction), and EntityType variations would avoid the need to create wholly separate BroSets just to cater for these differences, BroSets and Bros can be included/excluded in a full Main BroSet according to Taxonomy, Country (Jurisdiction), and EntityType in 4 ways:

A Main-BroSet can be defined to be Excluded/Included by Taxonomy, Country, or EntityType as described in section *6.3 BroSet Statements.* If a Main-BroSet is excluded for the particular input or reporting operation chosen, then the BroSet cannot be used at all.

An Incl-BroSet can be defined to be Excluded/Included by Taxonomy, Country, or EntityType as above. If the Main-BroSet is available for the chosen operation but the Incl-BroSet is not, then the Main-BroSet is adjusted dynamically at the time of use to remove the excluded Incl-Broset(s).

An individual Set Bro (standard or slave) can be defined to be Excluded/Included by Taxonomy, Country (Jurisdiction), or EntityType as described in the Bro Columns sections *7.27 Taxonomies,7.28* Countries*,* and *7.29* EntityTypes. If the Set Bro’s BroSet is available for the chosen operation but the Set Bro is not, then the Main-BroSet is adjusted dynamically at the time of use to remove the excluded Set Bro, which also excludes all of that Set Bro’s children.

An individual Element Bro (standard or slave) can similarly be defined to be Excluded/Included by Taxonomy, Country, or EntityType. If the Element Bro’s BroSet and Set Bro is available for the chosen operation but the Element Bro is not, then the Main-BroSet is adjusted dynamically at time of use to remove the excluded Element Bro.

Inclusion requires a pass for all three tests: Taxonomy, Country, and EntityType.

Exclusion occurs with a fail on any one of the tests.

Note that the three factors being tested (Taxonomy, Country, and EntityType) can be related to a degree in that an EntityType can apply to certain taxonomies and countries, and as an Entity must have an EntityType, an EntityType will always be in play. (The EntityTypes DB table design allows for multiple countries and multiple taxonomies, including none, per EntityType though as of June 2013 no multiple ones have been defined.)

These options should permit the building of flexible BroSets to cope with Taxonomy, Jurisdiction, and EntityType variations. From all perspectives (development, maintenance, and operational efficiency), the higher up the above list that variation control is applied the better.

If the Taxonomy, Country, or Entity Type differences are major, it would probably be better to construct wholly separate BroSets than to make complicated use of this feature. It will be a judgement call.

**Examples** (silly ones just to illustrate the use of this feature):

If IR-IFRS existed and included elements for a note on the environmental impact of windmills on bogs, then a BroSet for the Bros needed for this note could be made specific to use of the IR-IFRS Taxonomy i.e. when the In-BroSet is being used for data entry with the ultimate target Taxonomy being IR-IFRS.

* If the Irish standards required ‘Guinness expense’ to be shown separately, then the Bro in the Expenses section for ‘Guinness expense’ could be made specific to an Irish registered entity.

# Working with BroSets

BroSets are maintained in SpreadSheets as described in section *6. BroSet Spreadsheets*.

BroSets are worked on in Braiins Admin via:

* BroSet Import
* BroSet Export
* BroSet List
* BroSet Delete

BroSet Build Structs

Individual Bros within a BroSet can be viewed online via Bros Lookup.

Some details on the first two options follow.

## BroSet Import

The Braiins Admin BroSet Import module takes data from a BroSet spreadsheet (while ignoring ‘I’ or Info columns), to build the BroInfo database for the given BroSet, replacing all previous entries for that BroSet.

Only one BroSet, the one named in the “BroSet Name: nnnnn” statement of the spreadsheet is affected i.e. BroSet Import is for one BroSet at a time, even if other BroSets are included by the BroSet being imported.

Bros are imported in the order in which they appear in the SS.

Import first checks the SS data for validity, and does not rebuild the BroSet database unless no errors are found.

The Braiins Admin module “Build Bro Structs” runs after a successful Import to create the Bro “Structs” used internally by BroClass. It can also be run from the Menu as a standalone task, though this normally isn’t necessary.

*Hint:* When importing it is only necessary to upload SS columns as far as Scratch since the following ‘I’ or Info columns are ignored by Import. Doing this saves some bandwidth and upload time.

## BroSet Export

The Braiins Admin BroSet Export module creates a tab delimited text file /Admin/BrosAndTx/BroSet-BroSetName-<Names|Ids>-yyyy-mm-dd\_HH\_MM.txt for the selected BroSet, with a link to the file, of all Bros in the BroSet, for all the columns listed in the Spreadsheet Columns table below.

Export files may also be ftp copied from the server.

“Names” from <Names|Ids> in the file name means that the option to export Folios, Properties, and Members by name was chosen, whereas “Ids” means that the Id option was chosen. Either form can be re-imported, but while Folios, properties, and Members are in a state of flux it is better to use the Names version, even though it is more verbose. This is the default.

Export cleans up the SS by leaving columns blank where Import can deduce a value from previous row values.

The export file may be imported into Excel using just Ctrl A, Copy, Paste.

# BroSet Spreadsheets

A BroSet Spreadsheet consists of 6 sections, only one of which is mandatory:

* Bro Column Headings Row
* Comment Rows
* BroSet Statements to name and describe the BroSet 🡸 Mandatory
* Defined Bro Statements
* Include BroSet Statements
* Bro Columns

Details of these 6 sections are:

## Column Headings Row

The first row of a BroSet spreadsheet is an optional row of Bro Column headings that match the Bro Columns described below. BroSet Export includes the Headings row if the BroSet includes any Bro rows.

When importing a BroSet the Headings row can be included or not. If it is included, it is checked versus the BroSet Type. When working with a BroSet spreadsheet it is often convenient to freeze the Headings row. Then, when selecting the spreadsheet for uploading to BroSet Import, the initial or standard Excel selection excludes the frozen Headings row. The Headings row can be added to the selection, but it is fine and simpler to just upload the selection without the Headings row.

## Comment and Blank Rows

Any number of comment and blank rows may be used anywhere in a BroSet spreadsheet other than before a Headings row if used.

BroSet Import treats any row that doesn’t match one of the other row types as a comment row.

Alternatively a row may be specifically specified as being a comment row by starting it with a “#”. This is useful for commenting out a real row during development if it is intended to restore it later.

## BroSet Statements

BroSet Statements define the Type, Name, Descr(iption), and up to 6 other optional attributes of a BroSet.

A BroSet must include at least the first three BroSet statement (Type, Name, and Descr) described in the table below, and as many of the others as is desired. These BroSet Statement rows need to appear before any Include BroSet or Bro rows.

BroSet statements take the form:

BroSet Type{:} Value(s) {# comment}  
where Type is one of <Type | Name | Descr | SortKey | Taxonomies | Countries | EntityTypes | DateFrom | DateTo>

Examples:

BroSet Type: In-Incl # Part of an In-Main set of BroSets

BroSet Name: TFAs11 # To be included by the BS BroSet

BroSet Descr: TFAs excluding Impairments, UK-IFRS, not UK-GAAP-DPL # Experimental  
BroSet SortKey: Notes21  
BroSet Taxonomies: UK-IFRS,-UK-GAAP-DPL  
BroSet Countries: Wales,NI,-Scotland # Just an example!  
BroSet EntityTypes: -UK Charity  
BroSet DateFrom: 15-01-01 # Updated taxonomy applies  
BroSet DateTo: 13-12-31 # Being replaced by TheNewCharlesSpecial BroSet

Normally the BroSet statements would be text in just one cell of the spreadsheet but multiple cells can be used if preferred, especially for the optional comment.

Details of each BroSet Statement Type:

| **Type** | **Value(s)** |
| --- | --- |
| Type | <In-Main | In-Incl | Out-Main | Out-Incl> to define the type of BroSet this is, as described in section *2. BroSets* |
| Name | The Name of the BroSet which needs to be unique. Names are case sensitive, and can include spaces.  Warning: If a new spreadsheet (tab) is copied from a previous one and the name is not changed, importing the new BroSet will overwrite the previous one in the database of BroSets.  Editing the name of a previously imported BroSet and importing it again will cause another BroSet to be added to the database of BroSets. The previous BroSet can be deleted if it is no longer wanted. |
| Descr | A description of the BroSet. This text is what a user would see for a Main-In-BroSet. (Users will never “see” other BroSet types.) |
| SortKey | Optional text to control the order in which BroSets list, including Main-In-BroSets for users. If no Sortkey is provided, the BroSet Name is used. |
| Taxonomies | Optional comma separated list of any number of Taxonomies entered as Taxonomy Id or Taxonomy Name or both e.g. 2 or UK-GAAP-DPL or 2 UK-GAAP-DPL. See section *0.*  *Multiple Taxonomy, Jurisdiction, and EntityType* Options for an overview of the use of the Taxonomies option.  A Taxonomy in the list may be prefixed by a “-“ to mean ‘Not” i.e. Not that Taxonomy. A given Taxonomy can be included only once i.e. with or without a ‘-‘ prefix.  Two lists of Taxonomies for the BroSet are generated from this statement: - the Allow (or include) list of Taxonomies - the Not (or exclude) list of Taxonomies  The two lists are used as follows to control when the BroSet is used:  If the user activity (posting or reporting) does not involve a particular target Taxonomy, then nothing is done with these lists, and the BroSet is used (included), provided it also passes the other BroSet Statement tests.  If the user activity does involve a particular target Taxonomy:  the BroSet is used (included) if:   * the Taxonomy is in the Allow list * the Allow list is empty, and the Taxonomy is not in the Not list, which is also the case if the BroSet does not have a BroSet Taxonomies statement at all * the other BroSet Statement tests also pass   the BroSet is not used (i.e. it is excluded) if:   * the Taxonomy is in the Not list * the Allow list is not empty but the Taxonomy is not included in it |
| Countries | Optional comma separated list of any number of Countries entered as Country Id, Country Ref (ShortName) or both e.g. 1 or UK or 1 UK. See section *0.*  *Multiple Taxonomy, Jurisdiction, and EntityType* Options for an overview of the use of the Countries option.  A Country in the list may be prefixed by a “-“ to mean ‘Not” i.e. Not that Country. A given Country can be included only once i.e. either with or without a ‘-‘ prefix.  Two lists of Countries for the BroSet are generated from this statement: - the Allow (or include) list of Countries - the Not (or exclude) list of Countries  The two lists are used as follows to control when the BroSet is used:  If the user activity (posting or reporting) does not involve a particular target Country, then nothing is done with these lists, and the BroSet is used (included), provided it also passes the other BroSet Statement tests.  If the user activity does involve a particular target Country:  the BroSet is used (included) if:   * the Country is in the Allow list * the Allow list is empty, and the Country is not in the Not list, which is also the case if the BroSet does not have a BroSet Countries statement at all * the other BroSet Statement tests also pass   the BroSet is not used (i.e. it is excluded) if:   * the Country is in the Not list * the Allow list is not empty but the Country is not included in it |
| EntityTypes | Optional comma separated list of any number of EntityTypes entered as EntityType Id or Country Ref followed by EntityType ShortName or both e.g. 5 or UK Private Company or 5 UK Private Company. See section *0.*  *Multiple Taxonomy, Jurisdiction, and EntityType* Options for an overview of the use of the EntityTypes option.  An EntityType in the last may be prefixed by a “-“ to mean ‘Not” i.e. Not that EntityType. A given EntityType can be included only once i.e. either with or without a ‘-‘ prefix.  Two lists of EntityTypes for the BroSet are generated from this statement: - the Allow (or include) list of EntityTypes - the Not (or exclude) list of EntityTypes  The two lists are used as follows to control when the BroSet is used:  User activity (posting or reporting) always involves an EntityType as any activity involves an Entity and an Entity always has an EntityType. Usage is:  the BroSet is used (included) if:   * the EntityType is in the Allow list * the Allow list is empty, and the EntityType is not in the Not list, which is also the case if the BroSet does not have a BroSet Entities statement at all * the other BroSet Statement tests also pass   the BroSet is not used (i.e. it is excluded) if:   * the EntityType is in the Not list * the Allow list is not empty but the EntityType is not included in it |
| DateFrom | Optional date in {20}YY-MM-DD form for a date from which the BroSet is valid. The test is applied to the period end date. |
| DateTo | Optional date in {20}YY-MM-DD form for a date up to which the BroSet is valid. The test is applied to the period end date. |

## Defined Bro Statements

Defined Bro Statements apply to Main-In-BroSets only.

In processing BroSets and Bro data, BroClass and the RG need to know about certain specific Bros. If fixed or defined Names were to be adopted for these Bros, BroSet Defined Bro Statements would not be needed, but in the absence of such fixed names, and to allow Bro developers name flexibility, these statements provide links between names that BroClass and the RG know about, and the actual Bro Names used by BroSet developers.

The statements for a Main-In-BroSet are:

OfficersBro = Full Name of the Officers Bro # Optional comment

TPAsBro = Full Name of the TPAs (Third Party Agents) Bro # Optional comment

All of these statements are required for a Main-In-BroSet.

Others will be added as work proceeds.

The absence of these statements during BroSet Import is not treated as an Error because the required Bros might not have been added to the BroSet yet, but rather as a special kind of warning, which does not prevent the BroSet from being included by other BroSets or being exported, but it does prevent the BroSet from being used for live processing.

## Include BroSet Statements

A BroSet can include other BroSets as described in section *2. BroSets*.

The syntax of the statement to include another BroSet is  
Include BroSet BroSetName {# Optional comment}

An example is

Include BroSet TFAs11 # TFAs excluding Impairments, UK-IFRS, not UK-GAAP-DPL

Include statements can be used anywhere in a BroSet SS after the BroSet Statements except in between the Bro Rows of a Set Bro i.e. the first Bro Row after an include statement, if there is one, is expected to be for a Level 0 Set Bro.

Normal practice, however, is to group all the Include statements near the start of the BroSet e.g. after the BroSet Statements and Defined bro Statements (if any) and before the Row Statements (if any).

Include statement do not need to be in any particular order i.e. it is not necessary to include BroSets according to the order of the BroIds that they contain.

Include BroSet Statements are needed with Main-BroSets to build up the full set of Bros.

Include BroSet Statements may also be needed with Incl-BroSets in order for the Incl-BroSet to Import without error if it references Bros in other BroSets for

* Master Bros
* Related Bros
* Check Bros

StartEnd MvtList Bros

Include statements can occur multiple times in a series of includes such as  
BroSet A includes BroSet B  
BroSet A includes BroSet C  
BroSet B includes BroSet C

Then Importing BroSet A results in BroSet C being included twice, once by A and once by B. If that happens the second or subsequent includes are silently ignored. Allowing multiple includes like this is needed to cover cases where an Incl-BroSet references Bros e.g. Masters in other BroSets and is being imported by itself during its development.

## Bro Rows

Finally there are the spreadsheet rows which define the Bros themselves. These rows will usually, though not always, form the bulk of a BroSet spreadsheet. (There could be real, working BroSets with no Bro rows at all e.g. a Main-BroSet that includes other BroSets to define all the required Bros.)

Bro rows use many columns of a BroSet spreadsheet which are described in the following section *7. The Bro Columns*.

# The Bro Columns

The Columns in a BroSet SS used for defining a Bro in a Bro row are described in the table below, with comments for inserting/editing guidance.

All columns in a Bro row may be edited subject to the restrictions described in the table, though editing ‘I’ or Info columns without a background colour achieves nothing as Import ignores these columns.

Bros may be deleted just by removing the row, though if the Bro is a Set, all Set members (children) will also need to be deleted.

Bros may be moved if due attention is paid to Sets, both children and parents.

A new Bro may be added by inserting a row into the SS and completing just the relevant columns without bothering about Info columns.

Rows may be left part complete if sufficient information is provided.

Anything after the Scratch column is ignored by Import, but will be re-generated by an Export.

The Type Characters and background colours mean:

E **E**ditable as described in Comments, used by Import

S Applicable only to a Slave Bro  
 I **I**nformation purposes only, not used by Import. These columns have no background colour.

| **Name** | **Type** | **Comments** |
| --- | --- | --- |
| BroId | I E | The BroId in this column is generated by BroSet Import and included by BroSet Export. It must be within the range 1 to 99,999.  The BroId column is not normally used by BroSet Import, so mostly this is an ‘Info’ column.  However, the column can optionally be used as an ‘E’ column to Set or start a new BroId sequence or range. This is achieved by entering the desired BroId into the column prefixed by ‘=’ e.g.  =3000  This is interpreted as a command to Set the BroId for the Bro on this row to the number given, as the start of a new grouping within the Bro Tree. Subsequent BroIds will increment from this number.  **To use this feature** t**he ‘=’ must be visible in the SS** i.e. the cell value needs to be text. BroSet Export achieves this by outputting a space before the ‘=’. The number for the BroId must be in the range 1 to 99,950.  Any Incl-BroSet should start with an ‘=nnnnn’ BroId column to set the start of the range of BroIds for the BroSet.  BroIds do not need to be sequential i.e. there can be gaps in the BroId sequence.  Data is stored within the Braiins DB using BroIds. Thus once live data is being stored, BroIds can only be changed with a DB update.  Other than if the ‘=’ option is being used, there is no need to edit Ids, or keep them consistent. If a Bro is deleted there is no need to renumber the Ids of following Bros.  The BroIds of new Bros created in the SS can be left blank, or any ‘code’ can be used. |
| Type | E | The possible Bro Types are:   * Set * Ele (meaning Element) * Set Slave * Ele Slave * Set Master   Ele Master  Export fills in complete values here, but when creating/editing Bros the only ones which really matter are ‘Set’ and ‘Ele; as Import works out the others, or there is another way of specifying them. Even ‘Set’ is not critical, as if the names of following rows indicate that an ‘Ele’ should really be a Set, Import converts it from Ele to Set.  A Set may hold Sets and Elements. An Element must always be a member of a Set.  All Level 0 Bros must be Sets.  The type of a Bro may be changed if the Name is adjusted appropriately. |
| Level | I | Level starts at 0 for a top level Bro and increases by 1 for each Set level.  Level corresponds to the number of “.”s in the Bro’s full name or the number of segments in the name less 1.  Level is an Info item, as Import uses the BroName as the primary source of “level” information. |
| BroName | I | The BroName column shows the full name of a Bro for Info purposes. Bro names consist of dot separated segments or sections corresponding to set or level names. Bro names also appear in the following columns, split into segments. Any edits should be performed there.  A Bro’s name defines a Bro’s place in the Bro tree. It is used in formats and imports.  Export prefixes Bro names with 2 spaces per level for visual purposes. These spaces are not part of the actual names.  Bro names must be unique.  Bro names must start with a letter. After that any alphanumeric character plus \_ (underscore) may be used. Segment names after the first may start with a digit or be purely numeric. The size limit is 300 characters including dots. |
| Name0 and N1-N8 | E | Bro Name segments by level from level 0 to level 8.  Bro names may be edited via these columns, provided care is taken to change all set member names appropriately if a set name is changed.  Optionally a blank value may be used to repeat the value of the column in an ancestor row if defined. Using this feature makes trees easier to read and visualise in the SS. Export outputs name segments in this form, even if they were specified in full in the import.  A Bro may be moved in the Bro tree by moving it in the SS and editing its name accordingly. If such a move is performed, the BroId need not be changed. |
| ShortName | E | The ShortName for a Bro is an optional shorter form of the BroName to be used everywhere a BroName can be referenced i.e. in Bros to reference a Master Bro, a Check Bro, a related Bro, and a StartEnd MvtList Bro; in posting selection screens; in imports; in trial balances and data trail reports, and in formats.  A ShortName is not a level name. It is a replacement for the full Bro name.  No parent.child (set.member) rules are enforced for ShortNames so a ShortName for a Bro need not reflect the Bro tree structure. Clearly care should be taken with this to avoid causing confusion, especially when Bros are presented in a tree structure, where it could appear as if the full name is the short names joined together, giving a name longer than the full name.  The maximum length of a ShortName is 48 characters.  If no ShortName is defined for a Bro, the full name, including dots between level section names, applies.  Bro ShortNames must start with a letter. After that any alphanumeric character plus ‘\_’ (underscore) but not ‘.’ may be used.  Bro ShortNames must be unique versus all other short and full Bro names. |
| Master | S | The Master column is only relevant to Slave Bros to specify the Master of the Slave. The format of the column is: {Year#}{ MasterBroId} Master <BroName | BroShortName>  The optional Year# argument defines the Bro as a Prior Year Slave. # must be 1, 2, or 3 meaning 1, 2, or 3 years before the current year.  The MasterBroId argument is inserted into the column by BroSet Export for info purposes. It is ignored during Import.  The Master <BroName | BroShortName> argument is inserted by BroSet Export using the ShortName if that is defined for the Master. |
| Ref | E | Optional free form text field that is intended to serve as a hint or guide (or reference) as to which NL code or code range, or other item e.g. Memo code or ReportPad name in the case of SAPA, would or could apply to the Bro.  This information will be used in import mapping, and in directing a user to the appropriate part of the Bro Tree during on screen posting.  Currently the field consists of just one section that can contain anything, like the Comment field.  However, the field could be extended to containing multiple sections if it becomes clear that this would be useful.  The maximum length of a Ref field is 48 characters. This is arbitrary and can be increased if needed. |
| Folio | E | Optional Folio Name or Folio Id.  The Folio of a Bro defines the list of Properties that can potentially be used with the Bro to describe pieces of data, as per the Braiins Admin SIM Folios page, though that list can be amended for a particular Bro via the ExclProps, AllowProps, and Members columns.  The UsableProps for a Bro derived from its Folio, ExclProps, and AllowProps columns are shown in the I UsableProps column.  A Bro without a Folio can still have UsableProps courtesy of the AllowProps column.  If a Bro does not have a DataType (next column) it cannot hold data and so Folio is meaningless and must be blank.  The Folio of a Slave can be a subset Folio of the Master’s Folio. This provides one form of filtering of the values copied from Master to Slave. |
| DataType | E | All Bros have a DataType, even if nothing or None, as below in alphabetical order:   |  |  |  |  | | --- | --- | --- | --- | | **DataType** | **Numeric** | **Summing** | **Comments** | | (nothing i.e. empty column) |  |  | Only applicable to a Set Bro that is just a holder or tree node, which will not ever be output itself. Such Bros cannot hold data, cannot be posted to, and cannot have Folio, Props, or Members columns defined. | | Boolean | ● |  | Shown as true/false, stored as 1/0, able to be used in RG expressions as 1/0. | | Date | ● |  | Shown as YYYY-MM-DD or other date format, stored as a days number, able to be used in RG expressions. | | Decimal | ● |  | Floating point number i.e. with decimal fraction, rarely used. | | Enum | ● |  | Integer number representing one of a set of options, shown as the text value of the option, able to be used in RG expressions | | Integer | ● | ● | Integer number e.g. for number of staff | | Money | ● | ● | Money value held as integer i.e. no pence or cents. Is the most commonly used DataType. | | MoneyString | ● | ● | A Money and a String value, stored as a pair. Used when a Money value must have a narrative attached. Can only be used with an Ele Bro, not a Set as strings can’t be summed. However, a MoneyString Bro can SumUp to a Money Set with just the Money part being summed. | | None |  |  | Same as nothing or an empty column – see top of table | | Percent | ● |  |  | | PerShare | ● |  |  | | Ratio | ● |  | Ratio expressed as numerator:denominator e.g. 77:81 as used in a share consolidation. [Not implemented yet] | | Share | ● | ● | Integer number of shares | | String |  |  | Text |   The DataType of a Bro specifies how values are stored in the DB, whether the Bro is a Summing Bro, how the Bro may be used in RG expressions, and how values are formatted for inclusion in reports. |
| DboField | E | A DBO (DataBase Object) Field name e.g. Entity.Ref, Entity.Name, Entity.Identifier, Entity.Country acts as a read reference to the DBO field data i.e. at posting time the DBO field value is ‘posted’ to the Bro.  DboField can only be used with standard element Bros i.e. not Slaves and not Sets.  A Bro with a DboField value (a DboField Bro) is automatically set to RO.  A DboField Bro must have no UsableProps i.e. no Folio, no AllowProps, and no Members.  A DboField Bro cannot be a DBO Posting Bro.  See section *3.1 DataBase Objects and Bros* for more.  [Full list of field names to come.]  [Full usage still to be implemented.] |
| Sign | E | Sign applies only to Money Bros. Allowable values are ‘Debit’, ‘Credit’, and ‘Either’. ‘Either’ applies when a Bro is just as likely to have a Credit balance as a Debit one.  This sign property is used by the report generator in conjunction with the sign of the balance, any Zone sign setting, report generator row/cell sign settings, and brackets in title text, to decide on whether balances are to be bracketed. See ‘Table Balance Sign Handling’ in RGlanguage.txt for details. |
| AcctTypes | E | All Standard Bros must have at least one AcctType (Account Type), and can have up to 4 AcctTypes in a comma separated list, chosen from:   * BS = Balance Sheet * IS = Income Statement * CF = Cash Flow * DPL = Detailed P&L * HPL = Historical P&L * SCI = Statement of Comprehensive Income * Notes * Info * Other   The AcctTypes define the accounting uses of the Bro, though as of June 2013 no use is made of this column. It could potentially be used in conjunction with DataType, ReportOnly, PostType, UsableProps, and Zones to control posting and Bro use. Though does it add anything not covered by Zones and PostType?? |
| PostType | E | PostType applies only to Money Bros. Allowable values are:   * Nothing which defaults to DE on import * DE meaning Double Entry or ”Chart of Accounts” type posting is required * Sch meaning that Single Entry Schedule type posting is required |
| RO | E | RO or ReportOnly applies to standard and Slave Bros of any DataType.  Allowable values are:   * Nothing meaning that Posting is allowed   RO meaning the Bro is ReportOnly or Non-Posting  Slave Bros are automatically set to be RO. (Slaves mirror their Master’s values so cannot be posted to directly.)  Standard Element Bros which are not DboField Bro or not stock movement calculation value Bros should not be set to RO as otherwise there would be no way of getting data into the Bro. |
| ExclProps | E | For Bros with a Folio, ExclProps is an optional comma separated list of up to 20 Property Ids, or Property Names, as per the Braiins Admin SIM Properties page to be excluded from use with the Bro i.e. that are not usable when posting to the Bro, or, for a Set Bro, when balances from other Bros are being summed to the Bro.  Individual members of an excluded property can be brought back via the Members Allow list option.  ExclProps is mutually exclusive with AllowProps.  For Non-Slave Non-Summing Bros with UsableProps but no AllowProps list the Restated Property is automatically added to ExclProps if not already there.  For a Slave Bro, ExclProps acts as a filter i.e. values for properties in the ExclProps list are not copied from the Master to the Slave. |
| AllowProps | E | AllowProps is an optional comma separated list of up to 20 Property Ids, or Property Names, that are allowable for use with a Bro of any type, whether with or without a Folio.  AllowProps is mutually exclusive with ExclProps.  If the Bro has a Folio the properties in the AllowProps list must be members of the Folio.  If AllowProps is used, the AllowProps list becomes the Bro’s UsableProps list.  For a Slave Bro, AllowProps acts as a filter i.e. only values for properties in the AllowProps list (and any Allow Member matches) are copied from the Master to the Slave. |
| Members | E | The Members column allows for four different types of special Property Member handling for non-Slave Bros, and two types of filtering for Slaves – see Exclude, Allow and Slave Filtering at the end of this section for more re Slaves.  The first three types apply only to Bros which have UsableProps, either via a Folio or AllowProps:  **Mandatory** (m): Members which are Mandatory for a non-Slave Bro, one per Prop. The Prop containing a member must be in the Bro’s UsableProps list, or the Member must be one of the Allow Members list described below. This list is Mux with the Default and Exclude lists. When posting, any of the Mandatory Member(s) not included in a BroRef are added automatically.  **Default** (d): Members which are Defaults for a non-Slave Bro, one per Prop. The Prop containing a member must be in the Bro’s UsableProps list, or the Member must be one of the Allow Members list below. This list is Mux with the Mandatory and Exclude lists. When posting, any of the Default Member(s) not included in the BroRef and for which there is no other Member from the same Prop as the Default Member, are added automatically.  **Exclude** (x): Members which are Excluded from use with the Bro, Slave or non-Slave. The Prop containing a member must be in the Bro’s UsableProps list. This list is Mux with the Mandatory, Default, and Allow lists.  For a Slave Bro, Exclude Members act as a filter when copying from Master to Slave. Master values with a member matching an Exclude Member filter are not copied from the Master to the Slave.  The fourth type, **Allow** (a), has several functions depending on the Bro type/column values, as follows:  **Bro with ExclProps**: Allow Members can be used with a non-slave Bro with properties excluded from use with the Bro via ExclProps to re-enable just one or a few of the Members in the excluded properties. The same result could be achieved by leaving the properties as usable and excluding all the unwanted Members, but that could result in a long Exclude Members list.  **Bro with no UsableProps**: Allow Members can also be used with non-Slave Bros that do not have UsableProps, in a similar way to how AllowProps can be used with such Bros. This provides the minimum and most specific definition of a Bro – just allowable members.  **Slave Bro**: Allow Members can be used with a Slave Bro as a filter. If Allow Members are specified for a Slave, only Master values matching an Allow Member (or which match an AllowProps Property) are copied from the Master to the Slave.  **Source Format**  Members column values consist of a CS list of PMemIds or Member Names with each PMemId/Name prefixed by the code letter shown above after the name of list type i.e. <m | d | x | a> and a colon. The <m | d | x | a> Member types can be intermingled in the Members column, and the entries can be in any order apart from in one rare case mentioned below in Summing Use of Mandatory/Default Members.  Any Members which cannot be used when posting i.e. ‘RO’ or ‘Z’ types are ignored.  Examples of Members use are: [djh?? Correct examples once the Members are stable]  m:145 or m:NegativeGoodwill to make member IFAclasses.NegativeGoodwill mandatory with negative goodwill related Bros.  m:148 or m:DevelopmentCosts to make member IFAclasses.DevelopmentCosts mandatory for use with Asset development cost related Bros.  d:1319 or d:CoS to make member ExpenseType.CoS the default for purchases related Bros  x:423 or x:CoSec to exclude the the Officers.CoSec member from Director and PartnerLLP Bros.  x:1376,x:1378,x:1379,x:1380 to exclude >1 year details from Ageing for a debtor Bro. If the Bro was specifically for <1 year ageing then m:1375 could be used instead.  a:1184,a:1284 to Allow English and Welsh for a Bro from which Languages had been excluded via ExclProps 42.   * a:1375,a:1376 for a Bro to allow <1 or >1 Ageing The same end could be achieved via use of AllowProps 48 and then excluding the unwanted Members i.e. x:1378,x:1379,x:1380. (There is no need to exclude 1377 as that is an ‘R’ or non-posting Member.) The a:1375,a:1376 method is shorter and clearer.   **Summing Use of Mandatory/Default Members**  When Bros are either SumEnd or Set summed, if the sum is for a mixture of Bros with and without Members in use, then a difference arises between the sum of the Primary Member values, and the Base Sum which includes both the with and without Member values. The difference is allocated to a member so that after summing is finished the Base Sum for a Bro will equal the sum of its Primary Member values.  If the target Bro of the sum has a Mandatory Member list then the first Member in the Mandatory list is used for allocating the difference. This is the only time that the order of Mandatory Members in the Members column has meaning, though the most usual case is for the list to contain only one Member, in which case the order of Members in the Members column still does not matter.  If the target Bro of the sum has no Mandatory Member list but does have a Default Members list, then the first Member in the Default list is used for allocating the difference. This is the only time that the order of Default Members in the Members column has meaning, though the most usual case is for the list to contain only one Member, in which case the order of Members in the Members column still does not matter.  If the target Bro of the sum has neither a Mandatory Members list nor a Default Members list, then the Unallocated Member (Id 999) is used for allocating the difference.  **Slave Filtering**  Exclude (x:) and Allow (a:) Members may be specified for Slave Bros. These act as filters to fine tune the copying of data from Masters to Slaves. They are mutually exclusive. These filters work in conjunction with the Slave’s Folio which can be a subset of the Master’s Folio, plus ExclProps, and AllowProps slave filtering.  Master values with a member matching an Exclude Member filter are not copied from the Master to the Slave.  If Allow Member filters are in use, only Master values matching an Allow Member (or which match an AllowProps Property) are copied from the Master to the Slave. |
| SumUp | E | SumUp applies to Summing Bros and defines what auto summing does with the Bro balances in regard to automatic summing up the Bro tree. All Summing Bros not at level 0 must have a SumUp value. Allowable values are:   |  |  | | --- | --- | | Nothing | Level 0 Bro for which no summing ‘up’ can be performed. | | + | Specifies that the balances of the Bro sum to the parent Set balances.  This is the default on creation of a new Summing Bro for which no SumUp value is supplied. | | No | Specifies that the balances of the Bro are not to be summed to the parent Set i.e. that summing up the tree stops at this point.  No is typically used with a Set at the head of a duplicate or analysis branch of the Bro tree, whose sum should not be summed up the tree any further e.g. with PL.Revenue.Analysis so that the restated Revenue figures for analysis purposes are not summed to the PL.Revenue or PL sets which would result in the revenue sums being doubled. | | NA | Setting when auto summing does not apply because the Bro’s DataType is different from the parent Set’s DataType. | |
| Check | E | The Check column provides a means of specifying auto summing checks for Summing Bros, including Slave ones, by defining whether the Bro’s value should be equal to, or equal and opposite to, the value of a ‘Check’ Target Bro. The two Bros involved must have the same DataType.  The check is applied to a single balance only, the Base value.  The Check Column takes the following format:  {TargetBroId }<Equal To | Equal & Opp To>{ <Either | Both | Check | Target>}{ Year#} <TargetBroName | TargetBroShortName>  BroSet Export prefixes the Check field with the Check’s TargetBroId, but this is ignored during Import. When creating a new Check property it is not necessary to include the prefix of the TargetBroId.  The optional { <Either | Both | Check | Target>} field defines when the check is to be Performed, with the options meaning:   * Either if Either Bro has a value * Both if Both Bros have a value * Check if the Check (this) Bro has a value   Target if the Target Bro has a value  The default is Either if no { <Either | Both | Check | Target>} field is defined.  The optional { Year#} allows a prior year to be specified for the Check similarly to the Year# option for Slaves.  A Check of ‘Equal To Either{ Year#} Master BroName’ is automatically added to a Summing Set Slave as explained in the Slave Bros section. |
| Related | E | The Related column allows Element Bros that are ‘related’ or ‘associated’ to be tied together for checking during posting or importing. The syntax is:  <M | O | U> Related <BroName | BroShortName>  Where:  M = Mandatory once  O = Optional once  U = optional an Unlimited number of times  The Related Bros must form a loop e.g.  Bro A has a Related column value of ‘M Related B’  Bro B has a Related column value of ‘O Related C’  Bro C has a Related column value of ‘U Related A’  where Related B, Related C, Related A form the A -> B -> C -> A loop, and define A, B, and C as related Bros i.e. a list A,B,C of related Bros. The statement ‘M Related B’ for Bro A does not mean that there is a specific relationships between A and B, just that A and B form part of a list of related Bros.  Another way of viewing it is:  Bro A use is ‘M’ if any Bro in the A,B,C list is posted to  Bro B use is ‘O’ if any Bro in the A,B,C list is posted to  Bro C use is ‘U’ if any Bro in the A,B,C list is posted to  One ‘related’ Bro per Bro row rather than specifying the full list of related Bro (like StartEnd MvtLists) for each Bro was used to avoid repetition and to make editing easier.  For these A, B, C example Bros, posting can check that if a posting is made to C there must also be a posting to A as it is an ‘M’ Related Bro, and that a single posting to B is ok, but not more as it is ‘O’ for optional once, while any number of postings could be made to C which is a ‘U’ type.  Note that in simple cases of just two items, a Money value, and a narrative, a single Bro with DataType MoneyString can be used instead of two M Related Bros, one for the Money data and one for the String data. |
| Period | E | Every Bro which can hold data i.e. every one with a DataType, has a Period of either Duration or Instant.  ‘Duration’, ‘Instant’, or nothing (a blank column) are valid column values.  Nothing defaults to ‘Duration’. |
| StartEnd | E | StartEnd is used with Instant Summing Bros which have Start/End (Opening /Closing) characteristics to define how BroClass processes the data values.  The StartEnd column syntax is: <SumEnd | PostEnd | Stock> MvtList where MvtList is a comma separated list of the Movement Bros by BroName or BroShortName and:  **SumEnd**   * Start = previous period End and is automatically RO * End = Start + the Sum of the MvtList Movement Bros * End is automatically RO * TB includes: Start balance AND the Movement Bros   **PostEnd**   * Start = previous period End and is automatically RO * End is to be posted. Zero if no posting. * The MvtList provides a check, or means of calculating missing values. * TB includes: End balance but NOT the Movement Bros   **Stock** Similar to PostEnd with the additional property of being a Stock Bro. The differences from PostEnd are:   * TB includes: P&L: Start and End with sign reversed  BS: End * Movement is to be taken into account when checking that a posting journal balances.   **StartEnd Bro References**  For Instant Bros with StartEnd values, short form BroRefs for the Start values have an ‘s’ suffix, while full text BroRefs include “,Start’.  A Bro reference for a StartEnd Bro can include an optional {,<start|end>} value. If neither is used the defaults is end. |
| Zones | E | Zones is a comma separated list by Ref of up to 10 Zones which apply to the Bro.  See Braiins Admin Zones for a list of Zones.  If a Bro that is being output in a report has zones defined, the first zone in the list which has a sign defines the zone sign, which is one of the factors taken into account when processing the sign of a balance. See also the Sign column above.  During compilation of a format, if allowable zones have been set via a [zones ....] statement, and a Bro is used which has zones defined, a check is performed to see if one of the Bro’s zones appears in the [zones ....] statement list. If not, a compilation error occurs as use of the Bro at that location in the format is deemed to be invalid. |
| Order | E | Order is an optional integer number intended to define the sort order for elements within a Set if an order different from the Bros Id order is required for report List type output purposes. As of June 2013 no RG statement makes use of this option. |
| Descr | E | Descr is the Description or title text of up to 400 characters in length that is output by the report generator when the Bro is used in a title column of a table, or when the Bro is used with a ‘descr’ prefix in an expression.  If the Descr property of a Bro that has not been given a Descr property is referenced in a format, the name of the Bro is output as a reminder to “give me a Descr or hard code the required description in the format.” |
| Taxonomies | E | Optional comma separated list of any number of Taxonomies entered as Taxonomy Id or Taxonomy Name or both e.g. 2 or UK-GAAP-DPL or 2 UK-GAAP-DPL. See section *0.*  *Multiple Taxonomy, Jurisdiction, and EntityType* Options for an overview of the use of the Taxonomies column.  A Taxonomy in the list may be prefixed by a “-“ to mean ‘Not” i.e. Not that Taxonomy. A given Taxonomy can be included only once i.e. with or without a ‘-‘ prefix.  Two lists of Taxonomies for the Bro are generated from this statement: - the Allow (or include) list of Taxonomies - the Not (or exclude) list of Taxonomies  The two lists are used as follows to control when the Bro is used:  If the user activity (posting or reporting) does not involve a particular target Taxonomy, then nothing is done with these lists, and the Bro is used (included), provided it also passes the other Bro tests.  If the user activity does involve a particular target Taxonomy:  the Bro is used (included) if:   * the Taxonomy is in the Allow list * the Allow list is empty, and the Taxonomy is not in the Not list, which is also the case if the Bro does not have a Taxonomies column value at all * the other Bro tests also pass   the Bro is not used (i.e. it is excluded) if:   * the Taxonomy is in the Not list * the Allow list is not empty but the Taxonomy is not included in it   If a Bro that is excluded is a Set Bro, then all its child Bros are also excluded. |
| Countries |  | Optional comma separated list of any number of Countries entered as Country Id, Country Ref (ShortName) or both e.g. 1 or UK or 1 UK. See section *0.*  *Multiple Taxonomy, Jurisdiction, and EntityType* Options for an overview of the use of the Countries column.  A Country in the list may be prefixed by a “-“ to mean ‘Not” i.e. Not that Country. A given Country can be included only once i.e. either with or without a ‘-‘ prefix.  Two lists of Countries for the Bro are generated from this statement: - the Allow (or include) list of Countries - the Not (or exclude) list of Countries  The two lists are used as follows to control when the Bro is used:  If the user activity (posting or reporting) does not involve a particular target Country, then nothing is done with these lists, and the Bro is used (included), provided it also passes the other Bro tests.  If the user activity does involve a particular target Country:  the Bro is used (included) if:   * the Country is in the Allow list * the Allow list is empty, and the Country is not in the Not list, which is also the case if the Bro does not have a Countries column value at all * the other Bro tests also pass   the Bro is not used (i.e. it is excluded) if:   * the Country is in the Not list * the Allow list is not empty but the Country is not included in it   If a Bro that is excluded is a Set Bro, then all its child Bros are also excluded. |
| EntityTypes |  | Optional comma separated list of any number of EntityTypes entered as EntityType Id or Country Ref followed by EntityType ShortName or both e.g. 5 or UK Private Company or 5 UK Private Company. See section *0.*  *Multiple Taxonomy, Jurisdiction, and EntityType* Options for an overview of the use of the EntityTypes column.  An EntityType in the last may be prefixed by a “-“ to mean ‘Not” i.e. Not that EntityType. A given EntityType can be included only once i.e. either with or without a ‘-‘ prefix.  Two lists of EntityTypes for the Bro are generated from this statement: - the Allow (or include) list of EntityTypes - the Not (or exclude) list of EntityTypes  The two lists are used as follows to control when the Bro is used:  User activity (posting or reporting) always involves an EntityType as any activity involves an Entity and an Entity always has an EntityType. Usage is:  the Bro is used (included) if:   * the EntityType is in the Allow list * the Allow list is empty, and the EntityType is not in the Not list, which is also the case if the Bro does not have an EntityTypes column value at all * the other Bro tests also pass   the Bro is not used (i.e. it is excluded) if:   * the EntityType is in the Not list * the Allow list is not empty but the EntityType is not included in it   If a Bro that is excluded is a Set Bro, then all its child Bros are also excluded. |
| Comment | E | Optional free form comment text of up to 500 characters in length. Comments are included in Bro Exports, are shown by Bros Lookup, and are available as an optional column in Bros List.  Also, whole rows may be commented out by starting them with ‘#’, ‘;’ or ‘//’. Import will skip such rows but preserve them (with column detail kept) for a subsequent Export. |
| Scratch | E | Optional free form comment text of up to 500 characters in length. The difference from Comment above is that Scratch is not shown in Bros Lookup or Bros Listing. It is included in an Export. |
| I UsableProps | I | Entries in this column are Info entries generated by BroSet Export.  UsableProps is a comma separated list of the usable properties for a Bro as derived from its Folio, ExclProps, and AllowProps options. Property Member exclusions via the Members column are not shown here. |
| I Sum UsableProps | I | Entries in this column are Info entries generated by BroSet Export.  Sum UsableProps is the list of UsableProps which can be used when posting to a Summing Bro, based on its UsableProps and the UsableProps of its SumUp ancestors.  If a Prop has been specifically excluded from use with a Bro via the ExclProps column, it will not be added back into Sum UsableProps courtesy of use with an ancestor.  If Sum UsableProps is the same as UsableProps no entry appears in this column.  Bros without a Folio or Bros which are not Summing Bros set to SumUp do not have Sum UsableProps. |
| I M Use Member Info | I | Entries in this column are Info entries generated by BroSet Export.  The column shows ‘M Use Member’ if the Bro includes a Prop in its UsableProps list which includes M Use Members.  See the Braiins Admin SIM Members Notes for more information about M Use Members. |
| I StartEnd | I | Entries in this column are generated by BroSet Export to show:  the StartEnd MvtList of StartEnd Bros with BroIds rather than BroNames. Bros Lookup also gives both forms of the StartEnd MvtList.   * “In MvtList of <SumEnd | PostEnd | Stock> Bro BroId” for Bros which are in the StartEnd MvtList of another Bro |
| I Slave Ids | I | The BroId(s) of the Slaves of a Master Bro. |
| I Slave Filtering | I | Lists the Slave Filtering options in use, if any, from <Folio subset | ExclProps | AllowProps | Members> |

# Version Notes

## 24 June 2013 Initial Release

Initial release covering In-BroSets.

## 1 July 2013 Release

* Section 4 Multiple Taxonomy, Jurisdiction, and EntityType Options expanded to provide more explanation and to include a couple of examples.
* Two BroSet Statements removed from section 6.3 BroSet Statements as being overkill:
* BroSet MemberLevel: n
* BroSet EntityLevel: n

Column 7.13 AcctTypes column values changed to use IFRS terms as suggested by Charles

Column 7.21 Related column description made clearer

## 5 July 2013 Release

BRL changed to SIM

Section 3.1 DataBase Objects and Bros added

Terms to describe Bros moved to the end of Section 4

Column 7.11 DboField description expanded

## 20 July 2013 Release

Edits for change of Property Item to Property Member