

**Tripal Web Service BrAPI**

A Drupal module that implements the Breeding API (BrAPI) standardized specifications for Tripal websites that host genotypic/phenotypic data stored in a CHADO Database. BrAPI is an interface for data interchange between crop breeding applications.

|  |  |
| --- | --- |
| Breeding API (BrAPI) | [http://docs.brapi.apiary.io](http://docs.brapi.apiary.io/) |
| Tripal | [http://tripal.info](http://tripal.info/) |
| CHADO | <http://gmod.org/wiki/Chado> |
| Drupal | <https://www.drupal.org> |

**REQUIREMENTS**

|  |  |
| --- | --- |
| Tripal 3 (with Tripal Web Service extension) | [http://tripal.info](http://tripal.info/) |
| CHADO | <http://gmod.org/wiki/Chado> |
| Drupal 7.x | <https://www.drupal.org> |
| Intermediate knowledge of SQL Query |  |

**CONFIGURATION**

TRIPAL\_WS\_BRAPI module generates a number of system variables that will give end users technical control over tis general operation and call request response mechanism. Upon installation of this module, the following table summarizes all system variables created and the default value each one is set to.

|  |
| --- |
| FILE: tripal\_ws\_brapi/includes/config.inc |

Summary table showing system variables.

|  |  |  |
| --- | --- | --- |
| Configuration Variable | Use | Default |
| $config\_ws\_brapi | Holds the name of the module. To prevent variable name conflict with other modules, all system variables are prefixed with the module name. | Do not change |
| $config\_version | The default BrAPI version of this implementation. | 1.3 |
| $config\_resultset\_limit | Limit the number of items/data per page returned by a BrAPI call request. | 100 items per page |
| $config\_supported\_method | REST request methods supported. | GET and POST |
| $cofig\_menu\_level | An arbitrary levels may be added to request URL as outlined in BrAPI URL structure specifications. | web-services |
| $config\_hook | External modules can implement a new call or override existing call through the use of Drupal hooks. The string value of this configuration is used to signal this module that a module is implementing calls. | tripal\_ws\_brapi\_call |
| $config\_allow\_override\_hook | In cases where an external module implements a call that is identical in name and version to an existing call, this option will decide which of the two call instances to apply. | no – use the local version of the call |

All configuration variables can be modified as desired, except for the module name. Be sure to save your changes before installing the module. Alternatively, most of these system variables can be altered using the configuration page after the installation routine.

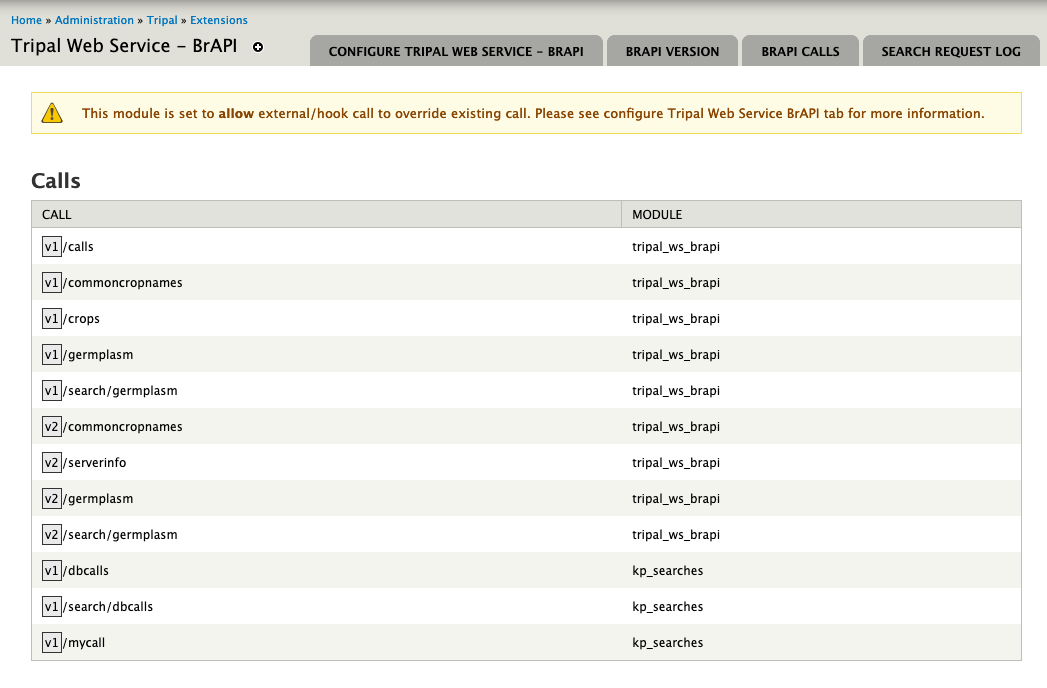


Figure 1 – Module configuration page showing all active BrAPI calls and the module each call is hosted. Series of page tabs show sections of this module that can be configured.

Configuration page allows system administrator to set different values to system variables outlined above. In addition, all active BrAPI calls or simply calls, as well as, search request call logs are summarized in this interface. This page can be accessed using

A . **Drupal administration context menu/Tripal/Extensions/Tripal Web Service – BrAPI**

Alternatively, copy and paste this URL into the browser’s location bar.

B . **host/ admin/tripal/extension/tripalwsbrapi/configure**

Sections are laid out in page tabs and are labelled to indicate which are of this module it covers. Each tab as seen in Figure 1, moving from left to right is summarized in the table below.

|  |  |
| --- | --- |
| CONFIGURE TRIPAL WEB SERVICE BRAPI | Configure overall module operation such as menu levels, HTTP request methods to support and call override strategy. |
| BRAPI VERSION | Add BrAPI version(s) to support. |
| BRAPI CALLS | Configure call restrictions or filter conditions used by each call when performing queries. |
| SEARCH REQUEST LOG | A summary of all search request made. See [https://brapi.docs.apiary.io/#introduction/search-services](See%20https:/brapi.docs.apiary.io#introduction/search-services) |

Configure Tripal Web Service BrAPI module

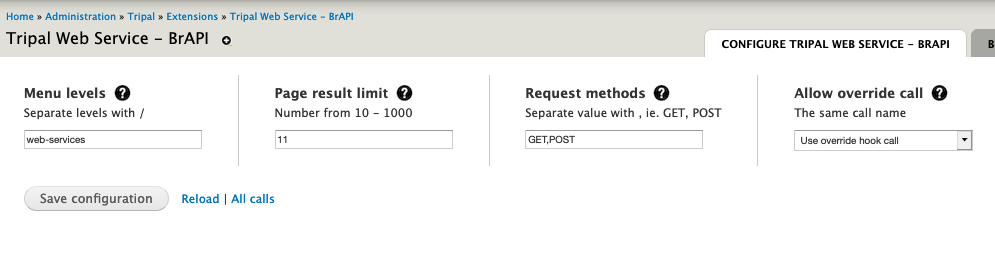


Figure 2 – Configuration page showing module settings.

Most system variables described can be accessed and modified using this form. Each field item (in all sections) can be described with a help or information text by hovering the mouse pointer on to help (question mark) icons. Click Save configuration button to save changes each time an option is modified.

NOTE: Clear cache each time when setting menu levels options.

BrAPI Version



Figure 3 – Configuration page showing BrAPI version settings.

To support multiple versions of BrAPI in a single implementation, this page enables system administrators to plan and set additional version numbers. BrAPI only requires the major version number (leftmost digit also seen in request url – brapi/v1/..) when requesting a call, while calls can be versions 1.2 and/or 1.3 etc. This version construct can be arranged using this page.

1. Select major version number from the list in Major version select field.
2. Select minor version number from the list in Minor version select field.
3. Click Add version button to save.
4. Each version added will be sorted and grouped according to major version number shown in the summary table below the form.
5. To add more version to a major version, re-select the major version number and select a minor version number then click Add version button. Minor version select field keeps track of what has been added, thus making sure no the same minor version number can be added more than once.
6. Additional version will be sorted accordingly as they are added to the group.
7. With multiple version, select a default, among the list of minor versions, to set as the default version of a major version number.
8. Use Reset button to drop all other versions except the default already set.
9. To remove version (including minor versions and default), click Remove button.

NOTE: A version unsupported error message will be returned when attempting to request a call with undefined or not configured version number.

NOTE: This module can only default to a single version at a given time per major version ie BrAPI 1.3. Other version such as BrAPI 1.2 can be implemented along side BrAPI 1.3, but requires switching from either versions as desired.

BrAPI Calls

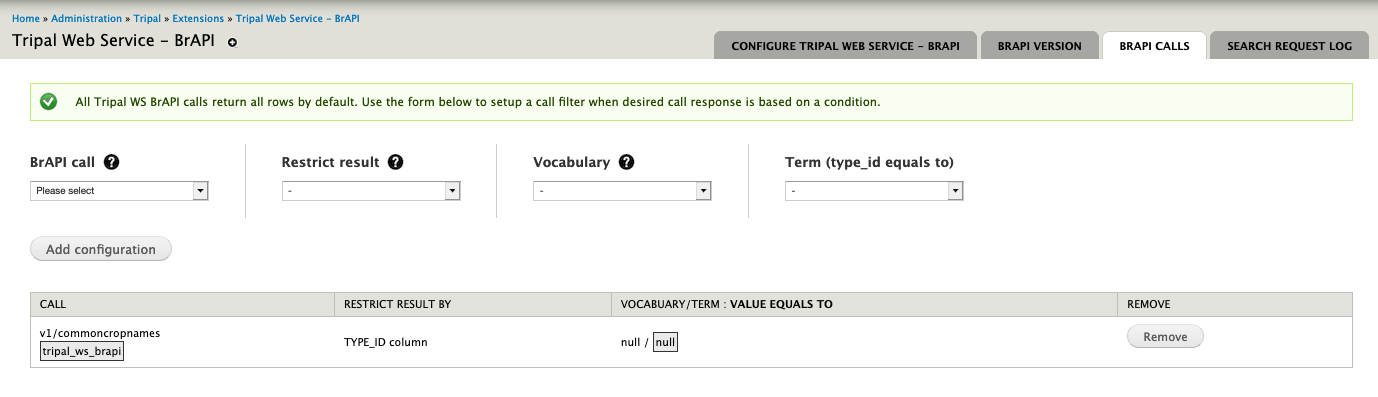


Figure 4 – Configuration page showing call settings.

Every call setup (see setting up calls) that performs a query to a CHADO database table can be configured using this form. This page enables system administrators to create additional restriction or filter criteria based on values stored in columns CHADO.table – type\_id column and CHADO.property table.

Each select field contains summarized values from either of this table columns for quick and easy selections. A summary table below this form outlines all restrictions to a call. A row can be interpreted as (from left to right column)

**“Call, version *X*, hosted by *Y* module, titled *ABC*, restricts its results by type\_id/value column, where its type\_id value is of type *W* cv, equals to *H* cvterm.”**

To setup a database query filter/restriction to a call.

1. Select a BrAPI call from the select box.

NOTE: when a call does not involve querying of data from a database table, a warning message will pop up instructing user that call cannot implement a query condition.

1. Each call can either use the column type\_id or a property table. Restrict select field will analyze data stored and decide if it could support either option. Select an option.
2. Once a restrict option has been selected, subsequent fields will auto-populate with relevant values, once again based on values or records stored.
3. Select option when prompted.

NOTE: another field labelled Value will present when restrict is set to prop.

1. Click Add configuration button to save.
2. When additional term is required, re-select the same call title. All select field elements will auto-fill with values that has been previously selected for easy and quick selection. Select additional values.
3. All configurations will be summarized in the summary table below.

Search Request Log

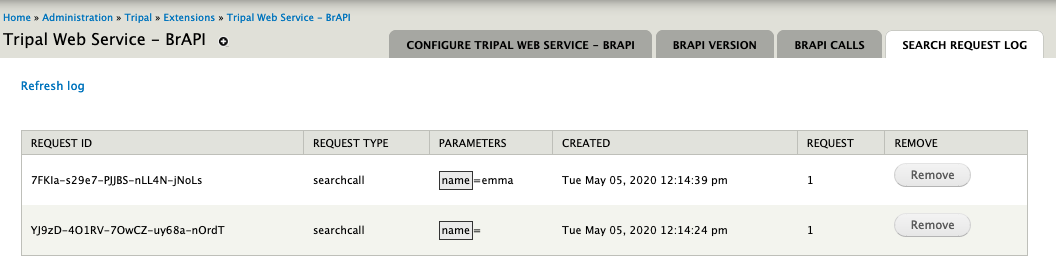


Figure 5 – Configuration page showing search request log.

As described by BrAPI search services (<https://brapi.docs.apiary.io/#introduction/search-services>), all search call request using POST request method will be saved. This page summarizes all search call history, along with every call details, parameters, date created and number of times it has been requested. A Remove button is available to erase a log item permanently.

**TECHNICAL DETAILS**

Tripal Web Service BrAPI Call Types: This module structures call into 3 types namely, Custom call, Database call and Search call.

Search

Database

Custom

CALL

|  |  |
| --- | --- |
| Custom | Data source can be supplied by user either by typing a list or by defining an array of values. No database table involve. |
| Database | Data source comes from a chado table and can be configured to restrict result based on the type\_id column or value column of the corresponding prop (property) table. |
| Search | Similar to Database call, search call operates like a database call except each request undergoes a two-stage process. First it will post a request that will result in a unique id number then the second stage, using the id number returned will produce the response. A log keeps a record of requests made (see Manage Search Request Log) |

This modules has built-in calls namely v1/calls, v1/germplasm, v1/crops, v1/commoncropnames and search call v1/search/germplasm, alternatively, all of this call can be viewed as BrAPI call response by requesting v1/calls call.

**Host/web-services/brapi/v1/call or host/web-services/brapi/v2/serverinfo**

Apart from these predefined calls mentioned, an external module (hosted by the same Drupal website) can implement a call or override a call without necessarily storing call assets in the same directory as this module, which will enable developers to extend functionality.

**SETTING UP BRAPI CALLS**

**A. General information about calls: File structure.**

All calls (directory and files) must be saved in a calls/ directory of this module. This file structure also applies to external modules implementing a set of call. Tripal\_ws\_brapi can support 2 file structures observed in BrAPI 1.3 and BrAPI 2.0, illustrated in the folder diagram below, labelled as A and B, respectively.

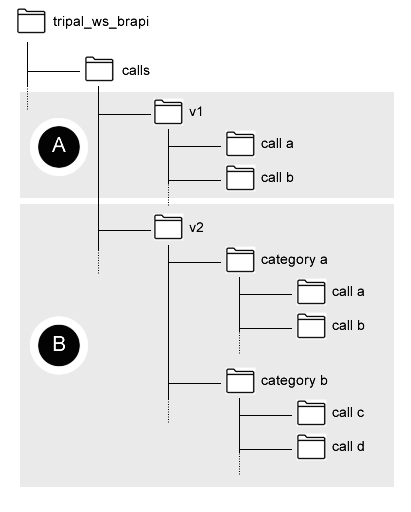


Figure 6 – Call directory structure supported. A: basic structure where calls fall directly under version folder and B: where calls are grouped using a category sub-folder.

Every call directory contains 2 PHP .inc files, 1 is a base call class file and 2 search implementation class file. #2 file is optional when a call does not require search functionality.

See figure below showing an example of implement the v1/germplasm call using file structure A or file structure B:

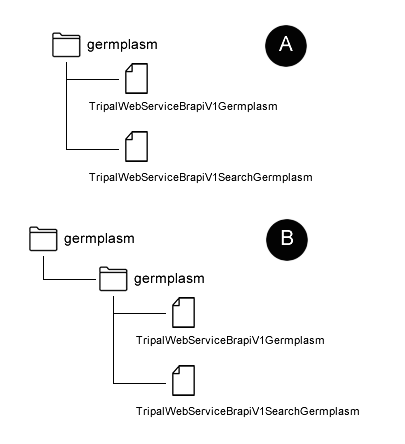


Figure 7 – Call file structure show both directory structure options

**B. General information about calls: Naming file and class.**

Call name or title should be the identical call title defined by BrAPI specification. Directory name should match this title with all letters in lowercase form. Filename must be in the following format:

TripalWebService + V + Major Version Number + Call name or title with the first letter in capitalized form. For example:

TripalWebServiceV1Germplasm.inc (to include the file extension which is .inc).

A search implementation uses the same naming arrangement but has the keyword Search inserted between the version number and the call title. For example:

TripalWebServiceV1SearchGermplasm.inc (to include the file extension which is .inc)

Both of these naming methods apply to when implementing a call that will be hosted outside the module with the exception that each name must be prefixed with the keyword External. For example:

ExternalTripalWebServiceV1Germplasm.inc or ExternalTripalWebServiceV1SearchGermplasm.inc

Class name will match the filename created using this method.

**C. General information about calls: External module implementing a hook must implement hook using hook string (see configuration).**

To register an external module implementing a BrAPI call, use the hook string configuration and implement the following hook in the .module file:

/\*\*

\* Implement BrAPI calls.

\*/

function HOOK\_tripal\_ws\_brapi\_call() {

// Indicate this module implements BrAPI calls.

return TRUE;

}

Where the HOOK is the module name and tripal\_ws\_brapi\_call is the hook string defined in the configuration.

**CREATE A CUSTOM CALL**

Creating a custom call, or other call types, is as easy as setting up parameters and defining a query or callback function that is responsible for generating relevant data as the response or result from executing a call.

Class definition

class TripalWebServiceBrapiV1Yourcallname extends TripalWebServiceCustomCall {

public $call\_parameter = [

// Key : Expected value for this key.

‘parameter 1’ => ‘data type’,

‘parameter 2’ => ‘data type’,

‘parameter …’ => ‘data type’,

];

// Keyword used to identify result items.

protected $call\_payload\_key = 'data';

// Unit of response for this call.

// With corresponding minor version a response is for.

public $response\_field = [

'1.3' => ['response field 1', 'response field 2', ' response field …'],

];

// Call parameters as provided in the request url.

public $call\_asset;

// Class name.

public $class\_name;

// PREPARE QUERY/RESULT:

// Callback to fetch/create data.

public function getResult() {

// Define values matching the number of elements defined in $response field.

return [

];

}

}

1. Rename the class name using the format defined relating to call class filename (titled Yourcallname in the code snippet above).
2. List the parameters that user can apply to request specific items from the result. Each parameter can be set to a data type which will ensure that only appropriate entries are permitted.

|  |  |
| --- | --- |
| int | (single value) numbers, including 0. |
| text | (single value) text, alphanumeric value. |
| array-int | (array, multiple values) elements are numbers. |
| array-text | (array, multiple values) elements are text value. |
| hash-code | (single value) xxxxx-xxxxx-xxxxx-xxxxx-xxxxx alphanumeric format. |

1. Define the unit of data and its elements in $response\_field. Set the key to the target BrAPI version number. ie 1.3 or 1.2.
2. Set the $call\_payload\_key to a string value. This variable will render as the key in the response. ie data (BrAPI 1.3) and call (BrAPI 2.0) used by /calls and /serverinfo calls, repectively.
3. Define the result in the only method of this class - getResult().

NOTE: ensure that the number of items in the data array should match the items in the $response\_filed.

Include a mechanism to handle each parameters defined in #2. Parameters requested in the url are available in

$this->call\_asset[‘parameters’] property and

$this->call\_asset[‘parameters’][‘yourparameter’] to access the value.

1. Save the file.
2. Test your call using host/web-services/brapi/v + version/yourcallname.

**CREATE A DATABASE CALL**

Similar to creating a custom call, a database call uses a CHADO table (as source table) and requires 3 callback function or method to handle querying the source table. Two of the methods represent query type that matches the configuration settings applied to calls – by type\_id or by value of prop table.

Class definition

class TripalWebServiceBrapiV1Yourcallname extends TripalWebServiceDatabaseCall {

public $call\_parameter = [

// Key : Expected value for this key.

‘parameter 1’ => ‘data type’,

‘parameter 2’ => ‘data type’,

‘parameter …’ => ‘data type’,

];

// Keyword used to identify result items.

protected $call\_payload\_key = 'data';

// Unit of response for this call.

// With corresponding minor version a response is for.

public $response\_field = [

'1.3' => ['response field 1', 'response field 2', ' response field …'],

];

// Chado table, source data.

public static $chado\_table = 'chado.table ie stock';

// Call parameters as provided in the request url.

public $call\_asset;

// Class name.

public $class\_name;

// Unfiltered result.

public function getResult() {

// Define values matching the number of elements defined in $response field.

return [

];

}

// Result matching type\_id

public function getResultByTypeid() {

$result = '';

return $result;

}

// Result matching value in property table.

public function getResultByPropertyTable() {

$result = '';

return $result;

}

}

NOTE: This class extends to a different class than the one used in defining custom calls and it is important to specify the source table ($chado\_table property).

To apply parameters (defined in the class) to a call, include each item with the request url. For example:

Host/web-services/brapi/v1/germplasm?commonCropName=Lentil

**CREATE A SEARCH CALL**

Class definition

class TripalWebServiceBrapiV1SearchYourcallname extends TripalWebServiceSearchCall {

public $call\_parameter = [

// Key : Expected value for this key.

‘parameter 1’ => ‘data type’,

‘parameter 2’ => ‘data type’,

‘parameter …’ => ‘data type’,

];

// Keyword used to identify result items.

protected $call\_payload\_key = 'data';

// Unit of response for this call.

// With corresponding minor version a response is for.

public $response\_field = [

'1.3' => ['response field 1', 'response field 2', ' response field …'],

];

// Call parameters as provided in the request url.

public $call\_asset;

// Chado table, source data.

public static $chado\_table = 'organism';

// Class name.

public $class\_name;

// PREPARE QUERY/RESULT:

// Callback to fetch/create data.

// Search query using parameters defined.

public function getResult() {

// Define values matching the number of elements defined in $response field.

return [

];

}

}

NOTE: This class extends to a different class than the one used in defining database calls and custom calls and it is important to specify the source table ($chado\_table property). Class name now contains a Search keyword as described in naming class section. The class this class extends to handles both POST (log search request) and GET requests.

Search call operates differently compared to other calls – custom call and database call. Search call needs to POST the call (with parameters) and at this stage a hash code is returned. A call can then be requested using the has code to view the result or response.

|  |  |
| --- | --- |
| POST Search Request | GET Search Request |
| $ch = curl\_init();  curl\_setopt($ch, CURLOPT\_URL, "host/tripaltest/web-services/brapi/v1/search/searchcall");  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE);  curl\_setopt($ch, CURLOPT\_HEADER, FALSE);  curl\_setopt($ch, CURLOPT\_POST, TRUE);  // Parameter  curl\_setopt($ch, CURLOPT\_POSTFIELDS, "{\"parameter\" : [\"value\"]}");  curl\_setopt($ch, CURLOPT\_HTTPHEADER, ["Content-Type: application/json"]);  $response = curl\_exec($ch);  curl\_close($ch);  var\_dump($response); public $call | host/tripaltest/web-services/brapi/v1/search/searchcall?searchResultDbId=7FKIa-s29e7-PJJBS-nLL4N-jNoLs |
| Add parameters in // Parameter line. Parameter in JSON format. | Using the hash code returned, get the call response. |
| RESPONSE: hash code  7FKIa-s29e7-PJJBS-nLL4N-jNoLs | Call response JSON. |

A copy of the POST request and the hash code can be accessed in the configuration page. To perform the same search request, use the same hash code to GET request call to retrieve the same response. This call request and its parameters can be accessed multiple times so long as the log entry is not deleted.

**CREATE A CALL ALIAS OR CALL OVERRIDE**

In some cases, calls from different versions implement the same process carried out by an existing call. To eliminate the need to copy and paste codes, call can point to an existing call and function identical to the source call.

Class definition

<?php

/\*\*

\* Call: Calls, class definition.

\* The '/calls' call is used to find the available BrAPI calls on a particular server.

\*/

class TripalWebServiceBrapiV2Serverinfo extends TripalWebServiceCustomCall {

// Call parameters as provided in the request url.

public $call\_asset;

// Override the base class payload key.

protected $call\_payload\_key = 'call';

// Call existing and identical call already setup.

// Declare existing call.

//

// module name / char v + major version number / call name.

public static $is\_alias\_of = 'tripal\_ws\_brapi/v1/calls';

}

1. To override the $call\_payload\_key of the source call, set the value of the property $call\_payload\_key.
2. Extend the class to whichever the type of call.
3. Finally, set the source call by specifying the module it is hosted followed by the version number then the call title. For example, the call – mygermplasm that mimics germplasm call, located in tripal\_ws\_brapi module:

**tripal\_ws\_brapi/v1/germplasm**

In cases when call wants to point to a search call, add search/ level between version number and the call title.

**tripal\_ws\_brapi/v1/search/germplasm**

NOTE: ensure correct override configuration settings when implementing an override, by using the exact same call title but is hosted by module external to tripal\_ws\_module.

When implementing from an external module, prefix the class and class filename with External keyword (see naming class above). For example:

class ExternalTripalWebServiceBrapiV2Serverinfo extends TripalWebServiceCustomCall

Summary table of Calls when implementing a call Alias and/or Override:

|  |  |
| --- | --- |
| tripal\_ws\_brapi Calls | Call alias property - $is\_alias\_of: |
| v1/calls | tripal\_ws\_brapi/v1/calls |
| v1/commoncropnames | tripal\_ws\_brapi/v1/commoncropnames |
| v1/crops | tripal\_ws\_brapi/v1/crops |
| v1/germplasm | tripal\_ws\_brapi/v1/germplasm |
| v1/search/germplasm | tripal\_ws\_brapi/v1/search/germplasm |
| Other calls | TO DO. |

**CREATING A TEST ENVIRONMENT**

A deplorable code is available in test/ directory that can be executed in a PHP execute window such as devel/php. The code populates stocks and organism tables (used by commoncropnames and germplasm calls) that can be used to test and try out calls, configuration and functionality of this module.