

This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. The article is believed to be still valid for the original topic scope.



Introduction to fetching content with services

A widget can access online content through the widget service interface. All communication objects are Value structures (see the [API documentation](#)). In this topic, the structures are specified in a Lisp-style language that the WidSets mobile server uses for validation of incoming and outgoing messages.

The basic description for making widget calls:

```
namespace call is Service
{
  input = (choice (list (bind (const id) (int widgetid))
                        (bind (const sid) (string serviceid))
                        (bind (const act) (string actionid))
                        (bind (const arg) (any argument))))

                        (list (bind (const id) (int widgetid))
                              (bind (const typ) (string serviceType))
                              (bind (const ver) (int serviceVersion))
                              (bind (const act) (string actionid))
                              (bind (const arg) (any argument))))))
}
```

A widget service has the following basic properties:

- **type**

There are several different types of services that can perform different tasks for a widget. A widget can use multiple typed services to perform server side functions. For example, a widget can have several *syndication* services that fetch different RSS feeds simultaneously.

- **version**

A certain version of a widget service has its unique input and output. Input refers to a Value coming from the widget to the server and output refers to the response Value the server sends back to the widget.

- **actions**

The actions of a widget service are separate functions with their own input and output. In the call function the id of the action is given, or alternatively the combination of the type and version with generic services (See below). The input of the action is given separately in the argument.

Generic calls

The input of the widget service call is either a direct reference to a service defined in the widget's XML (*widget.xml*), or a more generic call to a certain action of the service. Generic calls for actions cannot naturally

Getting_content_with_Services

have any configuration (specified in the XML). Some actions of widget services can be called in a generic manner and some cannot be. For example, getting an image from an URL with the action [geturl](#) of the [image](#) service is generic, but getting the latest items of an RSS feed with the [syndication](#) widget service action [getItems](#) is not because the syndication service needs configuration in the [widget.xml](#).

The message sent to any widget service has the following structure:

```
Message = (list (string action)
             (any actionArguments))
```

Example:

```
("call" 805 feed1 (get 1173954300))
```

This example calls the service that has the id *feed1* for the widget *805*. The service action is *get*, and the argument for the action is *1173954300*. The service is a *syndication service* that will return feed content with the *get* action. The argument *1173954300* is a *timestamp* indicating to the service that the widget needs only content that is newer than the *timestamp*.

The developer programming widget service calls into the widget script can get a list of services with their types and ids. Usually the developer has already defined the services in the XML configuration for the widget, and this way knows the id of the service to be called.

Calls are made with the asynchronous API function `void call(Object state, String service, Value argument)`. When a successful call returns, the function `onSuccess(Object state, Value returnValue)` is called. When an error occurs, the function `onFailure(Object state, String returnValue)` is called. You can determine which call the answer is for by comparing the returned state object.

There are two classes of services: services that are fetched server-side on the background, and services that are invoked on demand from the widget script. Fetched services are suitable for the mobile world because they do not use unnecessary mobile bandwidth. New content notifications are pushed to the client scripts, and the script code can have built-in logic to handle the notification event. For example, it can set a flag indicating that it will fetch the new content the next time the user maximizes the widget, and meanwhile update the minimized icon so the user notices that something interesting has happened.

The intention in the future is to have more fetched services that can perform customizable and complex tasks server-side, and notify the client when the processing is ready. One of the targets is to minimize the actual data size going to the client.

See also

- **Getting content with Services**
- [Available content fetching services](#)
 - ◆ [Syndication service](#)
 - ◆ [Webfeed service](#)
 - ◆ [HTTP service](#)
- [Fetcher details](#)
 - ◆ [Feed formats](#)
 - ◆ [HTTP authentication](#)

- Advanced filters
 - ◆ Filter expressions reference