streamcore

STREAMSERVICE API 6.0

Table des matières

1 Loading the StreamService Javascript API

The StreamService Javascript API is loaded by placing the following code in the head part of your HTML pages:

```
<script type="text/javascript"
      src="http://<sgm>/streamservice/<base>/streamcore.js.php?lang=en&load=main,ui">
</script>
```

- `<sgm>`: address / name of the SGM
- `<base>`: database name

Optional parameters:

- `lang`: language, `en` or `fr` (`en` by default)
- `load`: list of modules to load, separated by "`,`" (`main,ui` by default)

Available modules are main, ui and history. They are detailed in the following chapters.

A CSS file is also available for quickly giving a good appearance to the elements of the `ui` and `history` modules. You can load it by placing the following code in the head part of your HTML pages:

```
<link rel="stylesheet" type="text/css"
    href="http://<sgm>/streamservice/<base>/stylesheets/ui.css" />
```

The StreamService Javascript API depends on some external open-source Javascript:

- Prototype Javascript Framework (version 1.7)
  http://www.prototypejs.org/
- Script.aculo.us user interface libraries (version 1.9)
  http://script.aculo.us/

A copy of these libraries is available on SGM. They are automatically loaded at API initialization if they were not loaded before the StreamService API.

2 "main" module

2.1 STREAMCORE.SESSION

This is the main object managing a connection to StreamService. It enables login, logout, object search, ...

The connection status is maintained with a cookie, valid for the same database and the same host from where the API was loaded.

2.1.1 new Streamcore.Session(arguments)

- `arguments` [hash]:
  - `onLogin` [function()]: invoked on successful login or at API start if a user is already connected
  - `onLogout` [function()]: invoked on logout or at API start if no user is connected

One of the two function is always called at API start. The API ensures that the HTML document is fully loaded before the call.

Sample:

```
var session = new Streamcore.Session({
  onLogin: function() {
    // application start...
  },
  onLogout: function() {
    // show a login form...
  }
});
```

2.1.2 login(arguments, callback)

- `arguments` [hash]:
  - `user` [string]: user name
  - `pass` [string]: password
- `callback` [function(status)]: invoked when the login completes.

Sample:

```
session.login({user: 'global', pass: 'xxxxxxxx'}, function(status) {
  if (status) {
    // login was successful. Do some stuff before the onLogin session
    // function is called....
  }
  else {
    // authentication failure. Alert user...
  }
});
```

2.1.3 logout(callback)

- `callback` [function(status)]: invoked when the logout completes.

Sample:

```
session.logout(function(status) {
  if (status) {
    // logout was successful. Do some stuff before the onLogout session
    // function is called...
  }
  else {
    // error. Should not happen often!
  }
});
```

2.1.4 getObject(type, lid, id)

Returns the Streamcore.Object with matching type/lid/id.

Sample:

```
var service = session.getObject('services', 0, 0);
```

2.1.5 search(arguments, callback)

Enables the search of Streamcore.Object objects by their name or a part of their name. The search is case insensitive.

- `arguments` [hash]:
  - `name` [string]: name to search
  - `match` [string]: type of search (`exact`, `contains` or `begin`)
  - `type` [string]: types of searched objects separated by ",” (`category`, `site`, `sg`, or `all`)
- `callback` [function(status, result)] : invoked with search status and result. `result` contains an array of the matching `Streamcore.Object` objects.

Sample:

```
session.search({name: 'a', match: 'begin', type: 'category,site'}, function(status, result) {
  if (status) {
    // result contains all objects with name beginning with 'a'...
    result.each(function(object) {
      // object is a Streamcore.Object...
```

```
    });
  }
});
```

2.2 STREAMCORE.OBJECT

This object is returned by Streamcore.Session.getObject or Streamcore.Session.search.

2.2.1 load(parameters, callback)

Parameters loading.

- `parameters` [array of strings]: list of parameters to load, or `null` for loading all parameters
- `callback` [function(Streamcore.Object)] : invoked when the parameters are loaded. The parameters are accessible as object attributes.

Sample:

```
var s = session.getObject('site', 10, 1001);
s.load(['name'], function(o) {
  // now, o.name is accessible...
});
```

2.2.2 modifiable()

Returns `true` if the connected user can modify the object, `false` otherwise.

2.2.3 modify(parameters, callback)

Object parameters modification.

- `parameters` [hash] : hash of parameters / new values.
- `callback` [function(status)] : invoked when the modification completes.

Sample:

```
var s = session.getObject('site', 10, 1001);
if (s.modifiable()) {
  s.modify({name: 'new name'}, function(status) {
    if (status) {
      // modification was successful...
    }
  });
}
```

2.2.4 exec(command, callback)

Execution of a generic StreamShell command on an object.

- `command` [string]: command to execute
- `callback` [function(status, result)] : invoked when the command completes

3 "ui" module

This module contains some useful user interface elements to build a StreamService application.

3.1 STREAMCORE.UI.LOGINFORM

This object builds a form for session login management and language selection.

3.1.1 new Streamcore.UI.LoginForm(session, options)

- `session` [Streamcore.Session]: the StreamService session
- `options` [hash]:

- `onLangChange` [function(lang)]: invoked when user click on a language selection item

3.1.2 toElement()

Returns the DOM element suitable for DOM insertion and customization.

Sample:

```
var loginForm = new Streamcore.UI.LoginForm(session).toElement();
loginForm.id = 'myLoginForm';
$(document.body).insert(loginForm);
```

3.2 STREAMCORE.UI.USERFORM

This object builds a user form displaying the current user name and managing session logout and language selection.

3.2.1 new Streamcore.UI.UserForm(session, options)

- `session` [Streamcore.Session]: the StreamService session
- `options` [hash]:
  - onLangChange [function(lang)]: invoked when user click on a language selection item

3.2.2 toElement()

Returns the DOM element suitable for DOM insertion and customization.

Sample:

```
var userForm = new Streamcore.UI.UserForm(session).toElement();
userForm.id = 'userForm';
$(document.body).insert(userForm);
```

3.3 STREAMCORE.UI.SEARCHFORM

This object builds an auto-completion enabled form managing the search of Streamcore.Object objects by a part of their name.

3.3.1 new Streamcore.UI.SearchForm(session, options)

- `session` [Streamcore.Session]: the StreamService session
- `options` [hash]:
  - `type` [string]: types of searched objects separated by “,” (`category`, `site`, `sg`, or `all`)
  - `onSelect` [function(Streamcore.Object)]: invoked when the user selects an object.

3.3.2 toElement()

Returns the DOM element suitable for DOM insertion and customization.

Sample:

```
var searchForm = new Streamcore.UI.SearchForm(session, {
  type: 'category,site',
  onSelect: function(object) {
    // do something with the selected Streamcore.Object...
  }
});
searchForm.id = 'searchForm';
$(document.body).insert(searchForm);
```

4 “history” module

This module contains objects for getting StreamHistory data.

4.1 STREAMCORE.HISTORY.DATA

4.1.1 new Streamcore.History.Data(object, period, name, graphParams, historyParams)

- `object` [Streamcore.Object]: the object
- `period` [string]: a string representation of the period in the following format: `yyyy`, `yyyymm`, `yyyymmdd`, `yyyymmddhh`, "`yyyymmddhh - yyyymmddhh`", `today`, `last4h`, or `last12h`
- `name` [string]: the graph name (depends on object type)
- `graphParams` [hash]: the graph parameters (depends on `name`)
- `historyParams` [hash]:
  - `width` [integer]: requested PNG width in pixels (600 by default)
  - `titles` [1/0]: enable/disable graph title on PNG (enabled by default)

4.1.2 png()

Returns a PNG image as a Javascript `Image` object.

Sample:

```
var o = session.getObject('site', 10, 1001);
var h = new Streamcore.History(o, 'today', 'avgRate', {}, {width: 400});
$(document.body).insert(h.png());
```

4.1.3 getUrl(format)

- `format` [string]: `png`, `csv` or `xml`

4.2 STREAMCORE.HISTORY.GRAPH

This object manages a set of graphs for a Streamcore.Object. It contains a period selection form, a graph selection form, and adapt graph width to the DOM element containing the manager.

4.2.1 new Streamcore.History.Graph(container, object, options)

- `container` [id or Element]: the DOM id or Element of the container.
  The container must be present in the DOM before the manager is created.
- `object` [Streamcore.Object]: the object
- `options` [hash]:
  - `zoomWidth` [integer]: the width of the zoom image that will appear when user clicks on a graph (600 by default, 0 for disabling zoom)

Sample:

```
var o = session.getObject('site', 10, 1001);
var container = new Element('div').setStyle({width: '400px'});
$(document.body).update(container);
new Streamcore.History.Graph(container, o);
```

4.3 STREAMCORE.HISTORY.LASTVALUES

This object is similar to `Streamcore.History.Graph`, except that last values are displayed instead of graphs.

4.3.1 new Streamcore.History.LastValues(container, object)

- `container` [id or Element]: the DOM id or Element of the container.
  The container must be present in the DOM before the manager is created.
- `object` [Streamcore.Object]: the object

Sample:

```
var o = session.getObject('site', 10, 1001);
var container = new Element('div').setStyle({width: '400px'});
$(document.body).update(container);
new Streamcore.History.LastValues(container, o);
```

4.4 STREAMCORE.HISTORY.DASHBOARD

This object manages a dashboard for a Streamcore.Object. The dashboard contains `Streamcore.History.Graph` or `Streamcore.History.LastValues` objects, and toggles to add or remove them.

4.4.1 new Streamcore.History.Dashboard(container, object, options)

- `container` [id or Element]: the DOM id or Element of the container.
  The container must be present in the DOM before the manager is created.
- `object` [Streamcore.Object]: the object
- `options` [hash]:
  - `type` [string]: dashboard type (`graph` or `lastValues`)

Sample:

```
var o = session.getObject('site', 10, 1001);
var container = new Element('div').setStyle({width: '400px'});
$(document.body).update(container);
new Streamcore.History.Dashboard(container, o, {type: 'graph'});
```

5 Complete Javascript sample

The following sample uses various `ui` and `history` modules components. It displays history graphs for an object selected with a search form.

5.1 INDEX.HTML

```
<!DOCTYPE html>
<head>
<meta charset="utf-8" />

<title>StreamService Javascript API Sample</title>

<link rel="stylesheet" href="http://sgm/streamservice/testlh/stylesheets/ui.css"
      type="text/css"/>

<style type="text/css">
body, td, th, input, select, option, button, textarea {
  font: 11px tahoma,verdana,helvetica,sans-serif;
}
body {
  line-height: 16px; background: #ddd; color: #222;
}
h1 {
  font-size: 20px; font-weight: bold; text-align: center;
}
a {
  color: #000;
}
#sample {
  background: #fff; color: #222; width: 400px; min-height: 300px;
  border: 1px solid #777; margin: 10px auto;
}
#sample .top {
  background: #eee; padding: 2px;
}
#sample .sc-userform {
  padding: 2px;
}
#sample .sc-loginform {
  position: static; margin: 80px 0 0 0; width: 100%;
```

```
}
</style>

<script type="text/javascript"
        src="http://sgm/streamservice/testlh/streamcore.js.php?lang=en&load=main,ui,history">
</script>

<script type="text/javascript" src="./sample01.js"></script>
<script type="text/javascript">
$(document).observe('dom:loaded', function() {
  Sample.init('sample');
});
</script>

</head>
<body>
<h1>StreamService Javascript API Sample</h1>
<div id="sample"></div>
</body>
</html>
```

5.2 SAMPLE01.JS

```
var Sample = {
    container: null,
    main: null,
    session: null,
    object: null,

    init: function(container) {
        Sample.container = $(container);
        Sample.main = new Element('div', {'class': 'main'});
        Sample.session = new Streamcore.Session({
            onLogin: Sample.mainScreen,
            onLogout: Sample.loginScreen
        });
    },

    loginScreen: function() {
        Sample.container.update(
            new Streamcore.UI.LoginForm(Sample.session, {
                onLangChange: Sample.loginScreen
            }).toElement()
        );
    },

    mainScreen: function() {
        Sample.container
        .update(
            new Element('div', {'class': 'top'})
            .insert(
                new Streamcore.UI.SearchForm(Sample.session, {
                    onSelect: function(object) {
                        Sample.object = object;
                        Sample.history();
                    }
                }).toElement().setStyle({'float': 'left'})
            )
            .insert(
                new Streamcore.UI.UserForm(Sample.session, {
                    onLangChange: Sample.mainScreen
                }).toElement().setStyle({'float': 'right'})
            )
            .insert(
                new Element('div').setStyle({'clear': 'both'})
            )
        )
        .insert(Sample.main);
        if (Sample.object) {
            Sample.container.down('input').value = Sample.object.name;
            Sample.history();
        }
       else {
            Sample.container.down('input').activate();
       }
    },

    history: function() {
        Sample.main.update();
        new Streamcore.History.Dashboard(Sample.main, Sample.object, {
           type: 'lastValues'
       });
    }
};
```

6 StreamService URLs

The StreamService Javascript API uses internally URLs to fetch data from the SGM. These URLs can be built and used manually to use StreamService in another language than Javascript.

6.1 URL FORMAT AND CONTENT TYPE

The generic format of StreamService URLs is:

`http://<sgm>/streamservice/<base>/?m=<method>&ssid=<ssid>&<arguments>`

- `sgm`: address / name of the SGM

- `base`: database name
- `method`: method name
- `ssid`: session identifier, returned by the `login` method. It is necessary for every methods (except the `login` method itself...)
- `arguments`: other URI encoded arguments required for method

Every methods returns UTF-8 JSON encoded data (except `history` which returns PNG, XML or CSV).

6.2 METHODS REFERENCE

6.2.1 login

- Arguments:
  - `user`: the user name
  - `pass`: the password
- Returns: the `ssid` string that must be used in following URLs

6.2.2 logout

- Arguments: (none)
- Returns: `null`

6.2.3 whoami

- Arguments: (none)
- Returns: a hash with the following keys
  - `user`: a hash containing the result of the `whoami` StreamShell command (name, permissions, ...)
  - `token`: (not documented)
  - `prefs`: (not documented)

6.2.4 search

- Arguments:
  - `name`: the searched token
  - `match`: type of search (`exact`, `contains` or `begin`)
  - `type`: types of searched objects separated by "`,`" (`category`, `site`, `sg`, or `all`)
- Returns: a hash with the following key:
  - `cli_search`: an array of hashes containing `type`, `lid`, `id` and `name` of found objects

6.2.5 exec

- Arguments:
  - `cmd`: the StreamShell command to execute.
- Returns: (depends on the StreamShell command)

The generic format of a StreamShell command is "`go <type> <lid> <id> ; <command>`".
Here are some useful command samples :

`show` (show object parameters)

`stat` (show real-time statistics)

`modify <param>=<value> <param>=<value>` (modify object parameters)

6.2.6 history

- Arguments:
  - `type`: object type
  - `lid`: object lid
  - `id`: object id
  - `format`: `json`, `png`, `xml`, or `csv`
  - `period`: the period in the following format:
    `yyyy`, `yyyymm`, `yyyymmdd`, `yyyymmddhh`, "`yyyymmddhh - yyyymmddhh`", `today`, `last4h`, or `last12h`
  - `name`: name of the StreamHistory requested graph (depends on object type)
  - `p[...]`: graph parameters (depends on `name`)
  - `h[width]`: requested PNG width
  - `h[titles]`: enable/disable titles on PNG images
- Returns: the history data, in PNG, CSV or XML

6.3 SAMPLE (PYTHON)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from mechanize import Browser
import re
import simplejson
import urllib

class StreamService:
    def __init__(self, host, db, browser=None):
        self.host = host
        self.db = db
        self.url = 'http://'+host+'/streamservice/'+db
        self.browser = browser
        if self.browser is None:
            self.browser = Browser()

    def login(self, user, password=''):
        self.browser.open('%s/?m=login&user=%s&pass=%s' % \
                          (self.url, user, password))
        self._ssid = simplejson.loads(self.browser.response().read())

    def call(self, method, **params):
        result = None
        error = None
        p = dict()
        for k, v in params.items():
            if type(v) == unicode:
                p[k] = v.encode('utf-8')
            else:
                p[k] = v
        p['m'] = method
        p['ssid'] = self._ssid
        url = self.url + '/?' + urllib.urlencode(p)
        self.browser.open(url)
        response = self.browser.response()
        info = response.info()
        encoding = info.get('Content-encoding')
        content_type = info.get('Content-Type').split(';')[0]
        content = response.read()
        result = None
        if content_type == 'application/json':
            result = simplejson.loads(content)
            if isinstance(result, list) and len(result) and \
                    isinstance(result[0], dict) and result[0].has_key('errno'):
                raise Exception(result)
        elif content_type == 'image/png':
            return content
        else:
            if type(content) is not unicode:
                result = content.decode('utf-8')
            else:
                result = content
        return result

if __name__ == '__main__':
    import pprint
    S = StreamService('10.0.150.7', 'confdebase')
    S.login('global', 'xxxx')

    # Real-time stats
    stats = S.call('exec', cmd='go rule 10 1121; stat')
    avgRateLtoR = stats['rate'][0]['avgLtoR']

    # History data
    data = S.call('history', type='rule', lid=10, id=1121,
                  period='last4h', name='avgRate')
    pprint.pprint(data)
```