NativeSPStorageService

The `<StorageService>` element configures back-end plugins used for persistent storage of information across requests, and in some cases, restarts. Generally used only within the `shibd` service.

Other plugin components are layered on top of this service, and you can define multiple versions and reference them as needed, although typically only one is used.

On Version 2.4 and above, omitting this element will result in an in-memory plugin identified as `id="mem"`, per the example shown below for the Memory StorageService.

Common Attributes

- `type` (string)
  - Name of plugin type.
- `id` (XML ID)
  - A unique identifier within the XML file that labels the plugin instance.

Memory StorageService

Identified by `type="Memory"`, stores data in-memory and does not persist it across restarts of the `shibd` service.

```
<StorageService type="Memory" id="mem" cleanupInterval="900"/>
```

On Version 2.4 and above, the example above is the default plugin created if no StorageService is defined in the configuration.

Attributes

- `cleanupInterval` (time in seconds) (default is 900 on 2.4+)
  - Interval in seconds between background cleanup of service, when expired entries are swept from memory.

ODBC StorageService

Identified by `type="ODBC"`, stores data using an ODBC-compliant database. Persists across restarts and supports shared access across nodes of a cluster, provided the ODBC driver and the database support transactions. No locking is performed inside the plugin itself.

For more detailed information on using this plugin, and tips for specific databases, check the how-to topic. Note that in addition to the code below, this plugin (or its library `odbc-store.so`) has to be loaded by the Service Provider as is described on NativeSPOutOfProcess.

```
<StorageService type="ODBC" id="db" cleanupInterval="900">
    <ConnectionString>
    DRIVER=drivername;SERVER=dbserver;UID=shibboleth;PWD=password;DATABASE=shibboleth;APP=Shibboleth
    </ConnectionString>
</StorageService>
```

Attributes

- `cleanupInterval` (time in seconds) (default is 900)
  - Interval in seconds between background cleanup of service, when expired entries are deleted from the database.
- `isolationLevel` ("SERIALIZABLE", "REPEATABLE_READ", "READ_COMMITTED, or "READ_UNCOMMITTED")
  - Transaction isolation level for database access. Defaults to SERIALIZABLE to prevent race conditions while updating records. Reduce at your own risk.

Child Elements

- `<ConnectionString>`
  - Element contains an ODBC connection string. Exact contents differ across drivers. Note that you may wish to use a so-called "named" connection defined in odbc.ini (or for Windows inside the ODBC control panel applet or registry) to keep the password out of this file.
- `<RetryOnError>` (integer content)
  - Instructs the plugin to retry certain operations up to 3 times if the element's content is returned as a native ODBC status code from the operation. Can be used with some drivers to detect rolled back transactions and retry them by including the driver-specific status code that indicates the condition.

Memcache StorageService

Identified by `type="MEMCACHE"`, stores data in one or a cluster of http://www.danga.com/memcached servers. Persists across restarts, it's pretty fast and supports shared access across nodes of a cluster. Locking is performed inside the plugin itself. Note that the memcache store itself is volatile so if the memcache servers restart, all the state is lost.

This plugin is generally only available in custom builds. Also note that in addition to the code below, this plugin (or its library `memcache-store.so`) has to be loaded by the Service Provider as is described on NativeSPOutOfProcess.

```
<StorageService type="MEMCACHE" id="mc" prefix="SERVICE_PREFIX:">
  <Hosts>
    10.135.64.71:11211, 10.135.64.72:11211
  </Hosts>
</StorageService>

<StorageService type="MEMCACHE" id="mc-ctx" prefix="SERVICE_PREFIX:" buildMap="1">
  <Hosts>
    10.135.64.71:11211, 10.135.64.72:11211
  </Hosts>
</StorageService>

<SessionCache type="StorageService" cacheTimeout="28000" StorageService="mc-ctx" StorageServiceLite="mc" />
<ReplayCache StorageService="mc"/>
<ArtifactMap StorageService="mc" artifactTTL="180"/>
```

Attributes

- `prefix` (string)
  - String prefix that gets appended to the keys so that there are no key name conflicts when different applications use the same memcache servers.
- `buildMap` (1 or 0)
  - The memcache store is able to function in two modes: with or without a context map. The SessionCache requires that we keep an additional data structure to deal with contexts while the ReplayCache and ArtifactMap do not. **Default**: 0.
- `sendTimeout` (int microseconds)
  - Sets the timeout used for sending data on the socket. **Default**: 999999
- `recvTimeout` (int microseconds)
  - Sets the timeout used for receiving data on the socket. **Default**: 999999
- `pollTimeout` (int milliseconds)
  - Sets the timeout used for polling the socket. **Default**: 1000
- `failLimit` (int)
  - Sets the maximum number of errors that can happen when talking to a server before marking it as DEAD. **Default**: 5
- `retryTimeout` (int seconds)
  - Sets the timeout used for trying to use a DEAD server. **Default**: 30
- `nonBlocking` (1 or 0)
  - Sets the behavior of the memcached lib. Non-blocking mode gives better performance and recovers better from errors. **Default**: 1.

Child Elements

- `<Hosts>`
  - Element contains a list of memcache hosts.