ASP.NET State Service Protocol Specification

1. Introduction

An ASP.NET **web server** such as Microsoft Internet Information Server or the ASP.NET Development server can store and retrieve session data for web applications to and from a **state server**.

This document describes the communication protocol between the web server and the state server. The behavior of the state server described in this document is in accordance with Microsoft's implementation of the state server version 2.0.50727.3053 found in the ASP.NET 3.5 installation.

2. Overview

A state server facilitates rapid out-of-process session state storage for clients. Clients connect to the state server to store, retrieve and delete session data. Each piece of session data has a unique identifier which clients present to access the data, and an expiry date after which the state server deletes the session data to help conserve memory. In this model, a web server acts as a client and connects to the state server to store and access session data. This makes it possible for multiple web servers to access the same session data for a user in a web farm set-up.

The state service communication protocol is based on **HTTP**. To communicate, the web server connects as a HTTP client to the state server, which responds like a HTTP server. The web server sends HTTP requests to the state server, which responds to each request with an appropriate HTTP response.

Each request from the web server requests a certain action to be performed on a particular piece of session data by the state server. For instance, the web server may request that the state server store session data, delete session data, or retrieve session data.

A **unique session identifier** is used by the web server and the state server to identify a particular session and is unique for each web application user. It comprises the application domain identifier, the web application identifier and the user session identifier.

When the state server receives a request from the web service, it may perform the requested action and respond with an **OK Response**. If the state server is unable to perform the action, it may respond with a **Not Found Response** if it was unable to find the session identifier, a **Locked Response** if the session identifier was locked by a previous request or a **Bad Request Response** if the request was invalid or an error occurred while processing the request.

3. Concurrency Model

In accordance with basic HTTP, the web server and state server communicate on a sequential Request – Response basis. The web server sends a request and waits for the state server to send a response. After a response is received, the web server can initiate another request and so on.
The web server can also send multiple requests together without waiting for the corresponding responses. This technique, known as HTTP pipelining, must be handled gracefully by the state server. A client (web server) implementation that pipelines requests must be able to handle pipelined responses.

A request can specify for a session to be locked. In this case, the state server issues a **Lock-Cookie** (a 32-bit positive integer) after locking the session and passes the Lock-Cookie to the web server in the response.
If another request is received by the state server for an action to be performed on the locked session, the request must present the correct Lock-Cookie value to unlock the session for the action to be processed. If a Lock-Cookie is not provided or an incorrect value is provided, the state server responds with a **Locked Response**. The web server will keep retrying the request until the lock is released and the request processed. If the lock is not released within a set time period, the web server will forcefully unlock the session. This mechanism enables an instance of a user's web application to safely work with the session data exclusively without any interference from another instance of the same user's web application.

If multiple requests for a particular unique session identifier are received simultaneously by the state server (from different web servers, from different connections from one web server or pipelined requests on the same connection from one web server), the state server must queue them and process them one after the other. Different requests must not be allowed to operate on the same session data at the same time. The concurrency technique used by the state server to achieve this is implementation specific.

4. Messages

Messages are exchanged between the web server (client) and the state server. Messages are classified as either requests or responses. The web server transmits requests to the state server which processes them and transmits responses back to the web server.

4.1 Common Headers:

The following headers are commonly found in messages:

Host: This is a request-only header. It contains the network name or IP address of the computer sending the request. The Microsoft implementation of the state server ignores this header; however

other implementations of the state server or web server should require this header because it's a compulsory HTTP/1.1 request header.

Content-Length: This header applies to both requests and responses. This header contains the length of the body of the HTTP message in bytes. Session data is kept in the body of a request or response. Messages that do not have session data can omit this header or set its value to 0.

LockCookie: This header applies to both requests and responses. The value of this header is a positive 32-bit integer used to lock and unlock a session. The lock-cookie is assigned by the state server to a session when it locks the session. A web server (client) that wishes to unlock the session must pass the correct lock-cookie value to the state server.

When this header is in a response, it indicates the value of the lock-cookie assigned to the session. When this header is in a request, it indicates the value to be compared to the assigned lock-cookie, to unlock the session, if the session is locked.

The state server must not allow the web server (client) assign or influence a particular value to be assigned as the lock-cookie. The state server must select its own lock-cookie value. This can be done sequentially or randomly or in any arbitrary fashion. The Microsoft state server implementation assigns lock-cookies sequentially starting with a value of 2. The Microsoft client implementation (web server) accepts any positive 32-bit integer ranging from 0 to 2,147,483,646.

X-AspNet-Version: This is a response-only header. It contains the version of the state server, which can be used by web servers to implement versioning behavior.

4.2 Request Messages

A generic request message is in the format below. Each header line is terminated by a carriage return character and a line feed character (CRLF). A message can contain extra headers depending on the action being requested. A message terminates with a double CRLF and session data (if any)

```
VERB Unique-Session-Identifier HTTP/1.1
Host: Hostname
Content-Length:Length-of-Session-Data

[Session-Data]
```

Web server (client) implementations should note that the Microsoft implementation of the state server is inflexible with request headers. It expects the verb, header name and header value to be in an exact case as shown in the examples below. It also expects a single space character between the header name and header value for headers with a textual value and no space between the header name and header value for headers with a numeric value. Any deviation from this format will result in an error.

Verb: A HTTP Request verb that indicates what kind of action is being requested, examples of verbs are GET, PUT and DELETE.

Unique-Session-Identifier: This identifies a session. The state service should treat this value opaquely and use it as a lookup key to retrieve, delete, insert or update session data as requested.

The unique session identifier is usually comprised of three parts: a web application identifier, an application domain identifier, and a user session identifier in the following format:

web-application-identifier(application-domain-identifier)/user-session-identifier

Each part is URL encoded and then merged to form the unique session identifier. For example: A web application with the following parameters:

Web Application ID: `/3e50a960`
Application Domain ID: `iE+KOE6bwMI7BuHXun98zlcnkb8=`
User Session ID: `miztsjiek5gvzu55km3xun55`

will produce a unique session ID of `%2f3e50a960(iE%2bKOE6bwMI7BuHXun98zlcnkb8%3d)%2fmiztsjiek5gvzu55km3xun55`

The state service can use this format information internally to optimize performance. For example, in order to reduce hash table collisions, the state service can store sessions from different application domains in separate hash tables.
However, if such an implementation is unable to parse the unique session identifier into the expected format, it must have a fallback mechanism to work with the unique session id.
Any string in any format is a valid unique session identifier and must be accepted by the state server.

Session-Data: An 8-bit sequence of session data. This data is completely opaque to the state service and must be returned verbatim as it was sent by the web server.

4.2.1 Get Exclusive Request:

The Get Exclusive request is sent by the web server when it wishes to receive a session exclusively. When the state server receives this request, it looks up the session in its internal table. If the session is not found, it replies with a Not Found response. If the session is found but is already locked, it replies with a Locked response. If the session is found unlocked, it assigns a lock-cookie to the session, locks the session and replies with an OK[A] response. If the request is not in the required format or there is an error processing the request, the state server replies with a Bad Request response.

Format:

```
GET Unique-Session-Identifier HTTP/1.1
Host: Hostname
Exclusive: acquire
[CRLF,CRLF]
```

Responses:

OK[A], Not Found, Locked, Bad Request

Example:

```
GET
%2f3e50a960(iE%2bKOE6bwMI7BuHXun98zlcnkb8%3d)%2fmiztsjiek5gvzu55km3xun55
HTTP/1.1
Host: localhost
Exclusive: acquire
[CRLF,CRLF]
```

4.2.2 Get Request:

The Get request is sent by the web server when it wishes to receive a session non-exclusively. When the state server receives this request, it looks up the session in its internal table. If the session is not found, it replies with a Not Found response. If the session is found but is already locked, it replies with a Locked response. If the session is found unlocked, it replies with an OK[B] response. If the request is not in the required format or there is an error processing the request, the state server replies with a Bad Request response.

Format:

```
GET Unique-Session-Identifier HTTP/1.1
Host: Hostname
[CRLF,CRLF]
```

Responses:

OK[B], Not Found, Locked, Bad Request

Example:

```
GET
%2f3e50a960(iE%2bKOE6bwMI7BuHXun98zlcnkb8%3d)%2fmiztsjiek5gvzu55km3xun55
HTTP/1.1
Host: localhost
[CRLF,CRLF]
```

4.2.3 Release Exclusive Request:

The Release Exclusive request is sent by the web server when it only wishes to unlock a locked session. When the state server receives this request, it looks up the session in its internal table. If the session is not found, it replies with a Not Found response.
If the session is found already locked, the state server compares the lock-cookie in the request to the lock-cookie assigned to the session. If they match, the state server unlocks the session and replies with an OK[C] response. If they do not match the state server replies with a Locked response.
If the session is found unlocked, it replies with an OK[C] response.
If the request is not in the required format or there is an error processing the request, the state server replies with a Bad Request response.

Format:

```
GET Unique-Session-Identifier HTTP/1.1
Host: Hostname
Exclusive: release
LockCookie:Lock-Cookie
[CRLF,CRLF]
```

Responses:

OK[C], Not Found, Locked, Bad Request

Example:

```
GET
%2f3e50a960(iE%2bKOE6bwMI7BuHXun98zlcnkb8%3d)%2fmiztsjiek5gvzu55km3xun55
HTTP/1.1
Host: localhost
Exclusive: release
LockCookie:3
[CRLF,CRLF]
```

4.2.4 Remove Request

The Remove request is sent by the web server when it wishes to delete a session. When the state server receives this request, it looks up the session in its internal table. If the session is not found, it replies with a Not Found response.
If the session is found already locked, the state server compares the lock-cookie in the request to the lock-cookie assigned to the session. If they match, the state server deletes the session entry and replies with an OK[D] response. If they do not match the state server replies with a Locked response.
If there is no lock-cookie header present in the request and the session is locked, the server replies with a Locked response.
If the session is found unlocked, the lock-cookie header is ignored (if present) and the state server deletes the session entry and replies with an OK[D] response.
If the request is not in the required format or there is an error processing the request, the state server replies with a Bad Request response.

Format:

```
DELETE Unique-Session-Identifier HTTP/1.1
Host: Hostname
LockCookie:Lock-Cookie (Optional header)
[CRLF,CRLF]
```

Responses:

OK[D], Not Found, Locked, Bad Request

Example:

```
DELETE
%2f3e50a960(iE%2bKOE6bwMI7BuHXun98zlcnkb8%3d)%2fmiztsjiek5gvzu55km3xun55
HTTP/1.1
Host: localhost
LockCookie:3
[CRLF,CRLF]
```

4.2.5 Reset Timeout Request:

The Reset Timeout request is sent by the web server when it wishes to extend the expiration date of a session. When the state server receives this request, it looks up the session in its internal table. If the session is not found, it replies with a Not Found response.
If the session is found unlocked or locked, the state server extends the expiry date for that piece of session by the amount specified in the Set Request that created or last updated the session and replies with an OK[D] response.

If the request is not in the required format or there is an error processing the request, the state server replies with a Bad Request response.

Format:

```
HEAD Unique-Session-Identifier HTTP/1.1
Host: Hostname
```

Responses:

OK[D], Not Found, Bad Request

Example:

```
HEAD
%2f3e50a960(iE%2bKOE6bwMI7BuHXun98zlcnkb8%3d)%2fmiztsjiek5gvzu55km3xun55
HTTP/1.1
Host: localhost
[CRLF,CRLF]
```

4.2.6 Set Request

The Set request is sent by the web server when it wishes to create or update a session. An **ExtraFlags** header value of 1 indicates that the state server should store the session data only if the session entry does not exist. The **Timeout** header value specifies the length of time in minutes the session is stored. If the Timeout header is not present, the timeout defaults to 20 minutes.

When the state server receives this request, it takes note of the ExtraFlags header value (if present) and looks up the session in its internal table.

If the ExtraFlags value is set to 1 and the session is not found, the state server creates a new entry for the session in its internal table using the Unique-Session-Identifier as the lookup key, sets that session entry's internal ExtraFlags value to 1, stores the session data in the newly created entry and replies with a OK[D] response

If the ExtraFlags value is set to 1 and the session is found locked or unlocked, the state server does *not* update the session data and replies with a OK[D] response

If the ExtraFlags value is set to 0 or the ExtraFlags header is absent, and the session is not found, the state server creates a new entry for the session in its internal table using the Unique-Session-Identifier as the lookup key, stores the session data in the newly created entry and replies with a OK[D] response

If the ExtraFlags value is set to 0 or the ExtraFlags header is absent, and the session is found locked, the state server compares the lock-cookie in the request to the lock-cookie assigned to the session. If they match, the state server unlocks the session, extends the session expiry date, updates the session data and replies with an OK[D] response. If they do not match the state server replies with a Locked response. If there is no lock-cookie header present in the request and the session is locked, the server replies with a Locked response.

If the ExtraFlags value is set to 0 or the ExtraFlags header is absent, and the session is found unlocked, the state server extends the session expiry date, updates the session data and replies with an OK[D] response.

If the request is not in the required format or there is an error processing the request, the state server replies with a Bad Request response.

```
PUT Unique-Session-Identifier HTTP/1.1
Host: Hostname
Timeout:Timeout-in-Minutes (Optional header)
Content-Length: Length-of-Session-Data
ExtraFlags:0-or-1 (Optional header)
LockCookie:Lock-Cookie (Optional header)
[CRLF,CRLF]
[Session-Data]
```

Responses:

OK[D], Locked, Bad Request

Example:

```
PUT
%2f3e50a960(iE%2bKOE6bwMI7BuHXun98zlcnkb8%3d)%2fmiztsjiek5gvzu55km3xun55
HTTP/1.1
Host: localhost
Timeout:20
Content-Length:14
ExtraFlags:1
LockCookie:0
[CRLF,CRLF]
2o?vHGuSX5%4kx
```

4.3 Response Messages

A generic response message is in the format below. Each header line is terminated by a carriage return character and a line feed character (CRLF). A message can contain extra headers depending on the action being requested. A message terminates with a double CRLF and session data (if any)

```
Status-Code Reason-Phrase
Cache-Control: Private
Content-Length: Length-of-Session-Data

[Session-Data]
```

Status-Code: The HTTP response status code indicating the status of the requested action

Reason-Phrase: The textual HTTP response reason phrase. This token is intended to be a human-readable description of the request's status and is ignored by the web server.

Session-Data: An 8-bit sequence of session data. This data is completely opaque to the state service and must be returned verbatim as it was sent when requested by the web server.

4.3.1 OK Response[A]

The state server sends this response to indicate that the request was successful. This variant of the OK response includes the lock-cookie header, the timeout header, session data and optionally the ActionFlags header.

Format:

```
200 OK
X-AspNet-Version: State-service-version
ActionFlags: 1 (Optional header)
LockCookie: Lock-cookie
Timeout: Timeout-in-minutes
Cache-Control: private
Content-Length: Length-of-Session-Data
[CRLF,CRLF]
[Session-Data]
```

Notes:

The **ActionFlags** header is included in the response and its value set to 1 if the last Set Request marked the session's internal ExtraFlags to 1. After the response is sent, the state server resets the

session's internal ExtraFlags value to 0 which indicates that the ActionFlags header is never to be sent again. Specifically, the ActionFlags header can only be sent once throughout the lifetime of a session and that is only even possible if the Set Request that created the session specified an **ExtraFlags** header of 1.

The **LockCookie** header value contains the lock-cookie of the session.

The **Timeout** header value indicates the length of time in minutes the session will exist in the store before it expires.

Example:

```
200 OK
X-AspNet-Version: 2.0.50727
ActionFlags: 1
LockCookie: 2
Timeout: 20
Cache-Control: private
Content-Length: 14
[CRLF,CRLF]
2o?vHGuSX5%4kx
```

4.3.2 OK Response[B]

The state server sends this response to indicate that the request was successful. This variant of the OK response includes the timeout header, session data and optionally the ActionFlags header.

Format:

```
200 OK
X-AspNet-Version: State-service-version
ActionFlags: 1 (Optional header)
Timeout: Timeout-in-minutes
Cache-Control: private
Content-Length: Length-of-Session-Data
[CRLF,CRLF]
[Session-Data]
```

Notes:

The **ActionFlags** header is included in the response and its value set to 1 if the last Set Request marked the session's internal ExtraFlags to 1. After the response is sent, the state server resets the session's internal ExtraFlags value to 0 which indicates that the ActionFlags header is never to be sent again. Specifically, the ActionFlags header can only be sent once throughout the lifetime of a session and that is only even possible if the Set Request that created the session specified an **ExtraFlags** header of 1.

Example:

```
200 OK
X-AspNet-Version: 2.0.50727
ActionFlags: 1
Timeout: 20
Cache-Control: private
Content-Length: 14
[CRLF,CRLF]
2o?vHGuSX5%4kx
```

4.3.3 OK Response[C]

The state server sends this response to indicate that the request was successful. This variant of the OK response optionally includes the ActionFlags header.

Format:

```
200 OK
X-AspNet-Version: State-service-version
ActionFlags: 1 (Optional header)
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

Notes:

The **ActionFlags** header is included in the response and its value set to 1 if the last Set Request marked the session's internal ExtraFlags to 1. After the response is sent, the state server resets the session's internal ExtraFlags value to 0 which indicates that the ActionFlags header is never to be sent again. Specifically, the ActionFlags header can only be sent once throughout the lifetime of a

session and that is only even possible if the Set Request that created the session specified an **ExtraFlags** header of 1.

Example:

```
200 OK
X-AspNet-Version: 2.0.50727
ActionFlags: 1
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

4.3.4 OK Response[D]

The state server sends this response to indicate that the request was successful. This variant of the OK response does not have any optional headers.

Format:

```
200 OK
X-AspNet-Version: State-service-version
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

Example:

```
200 OK
X-AspNet-Version: 2.0.50727
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

4.3.5 Not **Found Response**

The state server sends this response to indicate that the requested session was not found.

Format:

```
404 Not Found
X-AspNet-Version: State-service-version
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

Example:

```
404 Not Found
X-AspNet-Version: 2.0.50727
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

4.3.6 Locked Response

The state server sends this response to indicate that the requested session is locked.

Format:

```
423 Locked
X-AspNet-Version: State-service-version
LockDate: Lock-date-in-ticks
LockAge: Lock-age-in-seconds
LockCookie: Lock-cookie
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

Notes:

The **LockDate** header value is a 64-bit integer value that indicates the date the lock was placed on the session. The value is the number of 100-nanosecond intervals that have elapsed since 12 midnight, January 1, 0001.

The **LockAge** header value is the number of seconds that have elapsed since the lock was placed on the session.

The **LockCookie** header value is the lock-cookie assigned to the session.

Example:

```
423 Locked
X-AspNet-Version: 2.0.50727
LockDate: 633845920412968750
LockAge: 85
LockCookie: 7
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

4.3.7 Bad Request Response

The state server sends this response to indicate that there was an error processing the request or that the request was not in the required format.

Format:

```
404 Bad Request
X-AspNet-Version: State-service-version
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

Example:

```
404 Bad Request
X-AspNet-Version: 2.0.50727
Cache-Control: private
Content-Length: 0
[CRLF,CRLF]
```

5. Connection Management

The web server (client) is largely responsible for connecting to and disconnecting from the state server. The web server connects to the state server to send requests and disconnects from the state server if no more requests are sent after a set time period.

The web server also disconnects from the state server if the state server is taking too long to respond to a request. The Microsoft client implementation (web server) reads this timeout value from the stateNetworkTimeout attribute in the web application's <sessionState> configuration.

The state server should disconnect a client (web server) if the client sends data that does not comply with the HTTP protocol.
The state server can also disconnect a client connection if the client does not transmit any data after a long period of time. The Microsoft implementation of the state service will disconnect a client if the client does not transmit any data within thirty seconds.

Implementations of the state service can also refuse client connections for security reasons.

Revision: 1.00 (7/31/2009)