The session management server (SMS) is a new component of Tivoli® Access Manager for e-business (TAMeb), version 6.0. The SMS provides a broad range of capabilities that change the way Tivoli Access Manager Web security servers (WebSEAL or Web server plug-ins) handle Web-based browser sessions. This paper is to educate you about a session’s lifecycle within the SMS by using real-life use cases. You will gain an understanding of what communication takes place between the different products in relation to SMS. This knowledge will give you the confidence to troubleshoot an environment that contains SMS, if problem determination is required.

View more content in this series

1.0 Web session management

This section gives a brief overview of what a Web session is and how TAMeb implements session state. It introduces an example SMS architecture as a pre-cursor to the use cases outlined in section 2.0.

1.1 What is a Web session?

A Web session is used to maintain state for a user across HTTP connections. In order for state management to be implemented in software, there are two separate requirements:

- A client side session identifier must be maintained, typically a cookie stored in the browser. This identifier is included by the HTTP client as part of each HTTP request, and provides the server with a consistent identifier for a particular HTTP client across multiple connections.
- A server side session table, typically implemented as a hash table. This table is implemented in the server code that binds an HTTP client identifier to a set of information established by the client on a prior HTTP connection. This table can bind a browser user’s current HTTP connection to a previous connection.
Historically, TAMeb Web security servers provided a server side session implementation that did not replicate session information across Web security server instances. The SMS was thus introduced as a service that could be used by TAMeb Web security servers to share browser-based user session information. These Web Security Servers could be:

- Servers that are replicas of each other (that is, behind a load balancer servicing a single DNS host name), thereby providing failover capabilities to a solution; or
- Servers that are part of the same DNS domain, but service different URLs, thereby providing domain level single sign-on (SSO) for services hosted within the same domain, for example, www.host.domain.com and www.host2.domain.com.

The SMS also allows administrators to manage user sessions (view, list, delete, show) from a single view.

### 1.2 Session cookie details

Within a traditional TAMeb deployment, the session cookie is simply a random value generated by the creator of the server side session. With the introduction of the SMS, some additional requirements were placed on the design of the session cookie. The following diagram shows the make-up of the new SMS session cookie.

**Figure 1.1 SMS session cookie implementation**

| session-cookie | Session ID | HMAC | Session Version |

The addition of the Session Version attribute allows the SMS to notify Web security servers of updates to sessions. The HMAC is embedded within each session to protect the SMS from being the subject of a denial of service target. This might occur if a rogue client attempts to attack a Web security server using an invalid session cookie, forcing the Web security server to query the SMS for the invalid session. Having the HMAC embedded in the cookie stops this attack at the Web security server as the Web security server is able to validate the HMAC before querying the SMS for the session.

### 1.3 TAMeb SMS architecture overview

Figure 1.2 shows an example component architecture of a TAMeb environment that includes an SMS server.
2.0 Use cases and session transitions

Each of the use cases refers to this figure to illustrate a particular process flow when SMS activity is triggered by a session event. For simplicity, the load balancer is not shown, although text is used to describe where it might have an effect in each use case. Note that in each of the use cases, we assume that both authentication requests succeed and authorization results are successful. Web security server 1 and Web security server 2 are replicas.

2.1 Establish initial session

Figure 1.3 illustrates the actions completed when a client requests a protected resource for the first time and is asked to authenticate. In this example the load balancer directs the user’s request to Web security server 1.

Figure 1.3 Client requests protected resource for the first time.
2.2 Client requests additional protected resource

Figure 1.4 illustrates the actions completed when a client, having already authenticated with Web security server 1, requests another protected resource. In this example the load balancer directs the user's request to Web security server 1. Note that this process is exactly the same regardless of whether the deployment includes SMS.

Figure 1.4 Client requests an additional protected resource

The next example shows the situation where the load balancer sends a subsequent request to Web security server 2. The user has a valid session with Web security server 1, but has not accessed Web security server 2 before now. This might be as a result of Web security server 1 being no longer available. Figure 1.5 below illustrates the flow.

Figure 1.5 Client requests routed to different Web security server

2.3 Client requests a session logout

Figure 1.6 illustrates the actions completed when a client requests a session logout. The load balancer sends the request to Web security server 1. The user has a valid session.
2.4 Clients session lifetime is exceeded

Figure 1.7 illustrates the actions completed when a client, who has a valid session with Web security server 1, expires. This example shows that session lifetime timeout is maintained by the Web security servers, not the SMS itself.

2.5 Client's session inactivity time period is exceeded

Figure 1.8 illustrates the actions completed when a client, who has a valid session on Web security server 1, but whose session inactivity period is exceeded. An inactive period is exceeded when a
session is not used with a particular Web security server for a set amount of time. This example shows that the inactivity timeout period is not maintained by the SMS itself.

**Figure 1.8 Client's session inactivity time exceeded**

![Diagram showing session inactivity time exceeded](image)

**2.6 Clients session is terminated by systems administrator**

Figure 1.9 illustrates the actions completed when a client's session is terminated by the system administrator. In this example Web security server 2 has been replaced by the TAMEb authorization server, since we are using the SMS command line interface to remove the user's session.
Figure 1.9 Client session is terminated by the systems administrator

The TAMEb systems administrator removes the session from the authorization server by executing a command from the `pdadmin`. An example of this command is below:

```
<pdadmin> server task <acldServerName> sms session terminate session <sessionId> -realm <realmName>
```

Where:

- `acldServerName` is the name of the authorization server within the SMS environment.
- `sessionId` is the identification of the session to be terminated.
- `realmName` is the name of the realm that the session exists within.

### 3.0 Conclusion

This article has identified the common use cases and actions associated with a number of session operations that SMS will encounter. The article should have given you a greater understanding of the role SMS plays during key session state transitions when used within a TAMEb environment. By understanding the use cases above, you should be able to assist in debugging customer environments when SMS is integrated into the environment.