Federated Identity for SharePoint Applications

* 1. You've seen in previous chapters ways that federated identity can help companies to share resources with their partners. The scenarios have included small numbers of partners as well as large numbers of constantly changing partners, sharing Web applications and Web services, and supporting multiple client platforms. These scenarios share an important feature: they all use claims.
  2. In chapter 11, "Claims-Based Single Sign-On for Microsoft SharePoint 2010," you saw how Adatum could expand its single sign-on domain to include SharePoint Web applications. The SharePoint Web applications at Adatum used claims-based authentication, using claims from an external token issuer (ADFS).
  3. In this chapter, you’ll learn how Adatum lets employees at one of its customers, Litware, use the a-Portal SharePoint application that was introduced in chapter 11, "Claims-Based Single Sign-On for Microsoft SharePoint 2010."
  4. **#!CALLOUT!#**
  5. Adatum wants to allow selected partners access to its SharePoint a-Portal Web application.

# The Premise

* 1. The a-Portal SharePoint application has given Adatum sales personnel access to up to date and accurate product information during the sales process, which has resulted in improved customer satisfaction. However, there have been complaints from customers who make purchases through of Adatum's partners that some of the product information has been out of date. This is because Adatum's partners are responsible for keeping the product information that they use up to date. One of these sales partners is Litware. Rick, the CIO of Litware, has seen the a-Portal SharePoint application: he is keen for his sales staff to use a-Portal instead of their own copy of the product information. Adatum has already claims-enabled the a-Portal SharePoint application (for further information see chapter 11, "Claims-Based Single Sign-On for Microsoft SharePoint 2010") and made it available to Adatum employees who work remotely on the internet. Litware has already deployed ADFS, so most of the required federation infrastructure is already available.

# Goals and Requirements

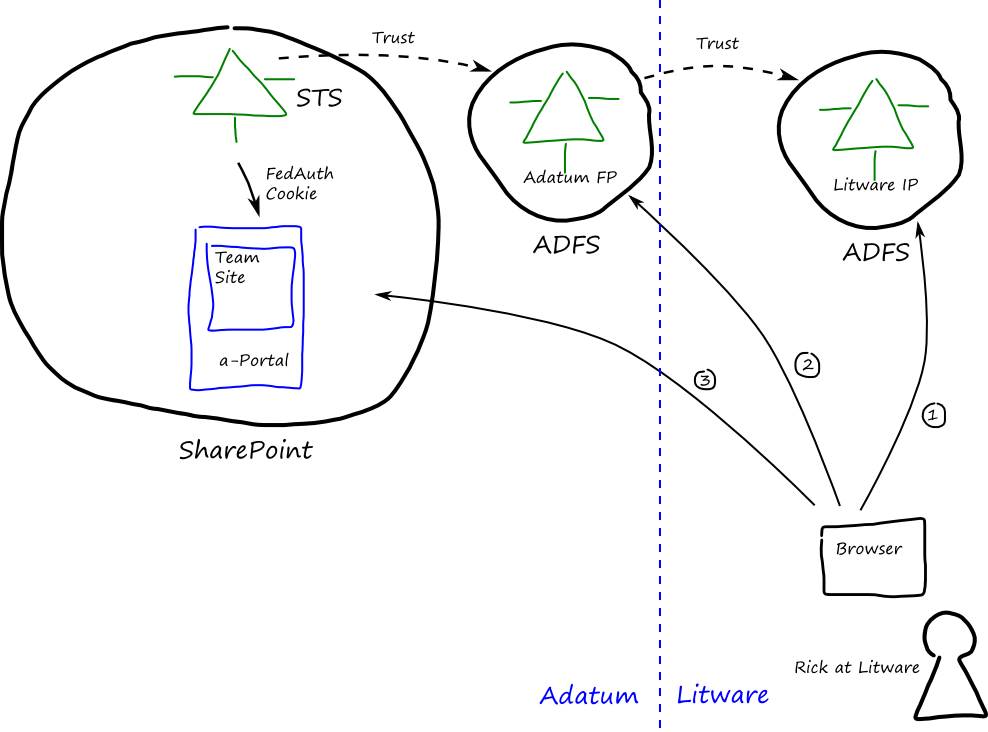
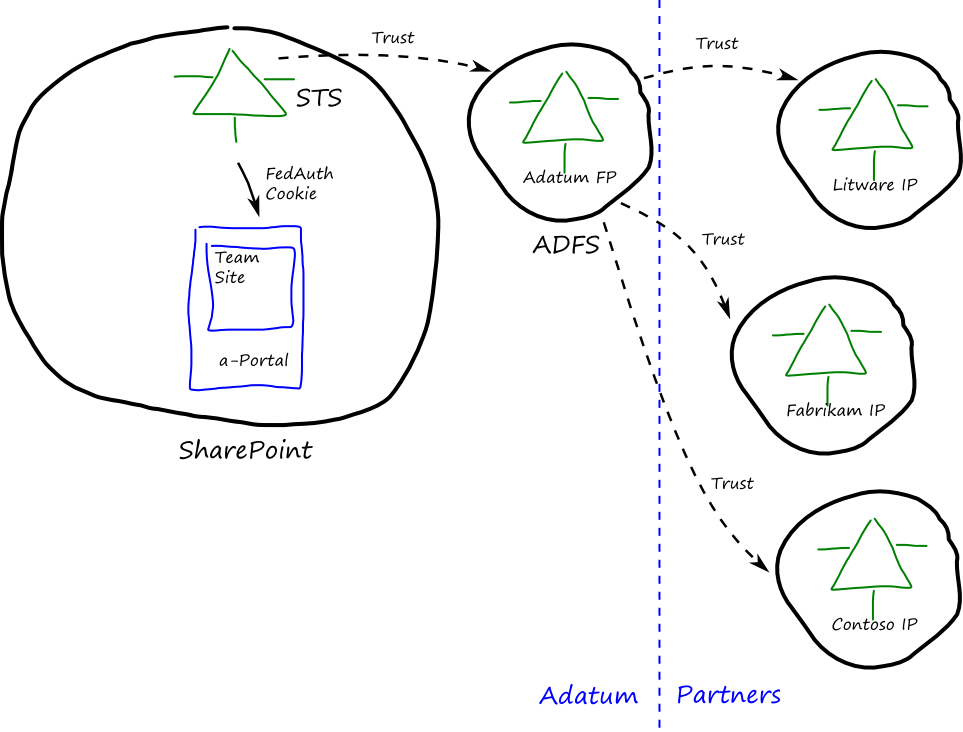
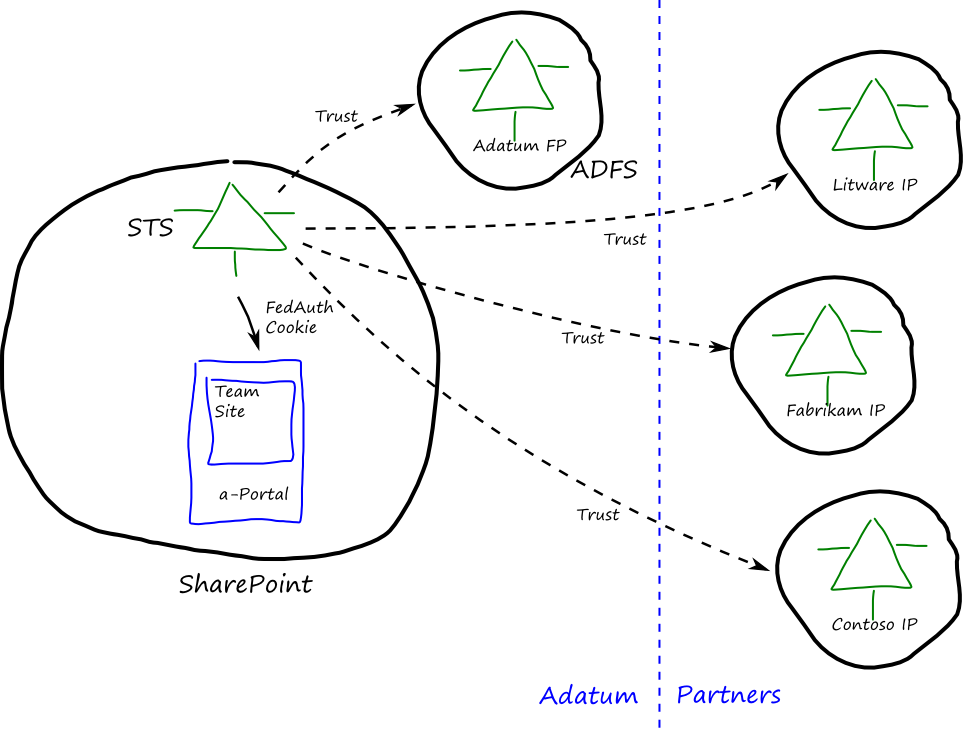
The primary goal of this scenario is to show how to create a SharePoint site that uses federated identities, so that users from Litware can access the Adatum a-Portal SharePoint application without having to sign in again to the Adatum security realm. The types of claims issued by Litware are not the same types as the claims used by a-Portal at Adatum, so it's necessary to include some claims transformation logic to convert the claims issued by Litware. Adatum anticipates that other sales partners will also want to use the a-Portal application, so the solution must be able to accommodate multiple identity providers.

The solution should also ensure that partners are kept isolated. For example, there may be some product information that only Adatum and not Litware sales personnel should see.

For security, Adatum wants to have SharePoint automatically sign users out of the a-Portal application after a period of inactivity. In addition, because users will be accessing the a-Portal application on computers outside the Adatum corporate network, when a user closes the browser and then re-opens it, the user must re-authenticate to gain access to the a-Portal Web application.

# Overview of the Solution

Figure 1 shows an overview of the solution adopted by Adatum and Litware. It shows a new trust relationship between Adatum's issuer, and Litware's issuer. In addition to acting as an identity provider (IdP) for Adatum employees, the Adatum ADFS instance now functions as a federation provider (FP) for partners such as Litware.

* 1. **#!SPOKENBY(Bharath)!#**
  2. Adatum has deployed an ADFS proxy to support authenticating users outside of the Adatum corporate network.
  3. 
  4. Figure 1
  5. Federating identity with Litware
  6. When Rick, a user from Litware, browses to the a-Portal SharePoint Web application, SharePoint detects that Rick is not authenticated, and redirects his browser to the Adatum FP. The Adatum FP then redirects Rick's browser to the Litware issuer.
  7. For details about how to customize the way that SharePoint redirects a user to a token issuer, see the section "The Sign-In Page" in chapter 11, "Claims-Based Single Sign-On for Microsoft SharePoint 2010."
  8. The numbers in the following list correspond to the numbers in Figure 1.
  9. Rick authenticates with the Litware IdP and obtains a SAML token with claims issued by Litware.
  10. Rick's browser redirects back to the Adatum issuer. This FP can apply some custom claims mapping rules to the set of claims from Litware to create a set of claims suitable for the a-Portal Web application. The FP issues this new set of claims as a SAML token.
  11. Rick's browser redirects back to SharePoint. SharePoint validates the token, checks any authorization rules that apply to the page that Rick requested, and if Rick has permission, displays the page.
  12. Adatum considered two alternative models for federating with partners. The first, which is the one that Adatum selected, is shown in Figure 2.
  13. 
  14. Figure 2
  15. The hub model
  16. In the hub model, SharePoint has a single trust relationship with the Adatum FP. The Adatum FP then trusts the partners' issuers. The Adatum FP can apply rules to the claims from the different IdPs to create claims that the SharePoint Web application understands.
  17. Figure 3 shows the alternative model.
  18. 
  19. Figure 3
  20. The direct trust model
  21. In the direct trust model, SharePoint manages a trust relationship with each issuer directly, and uses custom claims providers to manipulate the incoming claims to a common set of claims that the a-Portal Web application understands.
  22. The advantages of the hub model adopted by Adatum are that:
  23. It's easier to manage multiple trust relationships in ADFS rather than SharePoint.
  24. It's simpler to manage a single trust relationship in SharePoint and it avoids the requirement for multiple custom claims providers.
  25. You can reuse the trust relationships in the FP with other relying parties.
  26. You can leverage ADFS features such as integration with auditing tools to track token issuing.
  27. ADFS supports the SAMLP protocol in addition to WS-Federation.
  28. **#!SPOKENBY(Bharath)!#**
  29. An advantage of the SAMLP protocol over WS-Federation is that it supports initializing the authentication process from the IdP instead of the RP, which avoids the requirement for either the relying party (RP) or the FP to perform home realm discovery.
  30. The disadvantage of the hub approach is its performance: it requires more hops to acquire a valid SAML token. With this in mind, Adatum made some changes to the token caching policy in the a-Portal Web application to reduce the frequency at which it's necessary to refresh the SAML token. However, Adatum is using session cookies rather than persistent cookies to store the SAML token references so that if the user closes his browser, then he will be forced to re-authenticate when he next visits the a-Portal Web application.
  31. **#!SPOKENBY(Markus)!#**
  32. Strictly speaking, the session cookie doesn't contain the SAML token, it contains a reference to the SAML token in the SharePoint token cache.
  33. Adatum implemented sliding sessions for users of the a-Portal Web application: after a token issuer authenticates a user, the user can continue using the a-Portal Web application without having to re-authenticate if he remains active. If a user becomes inactive in the Web application for more than a defined period, then he must re-authenticate with the claims issuer and obtain a new SAML token. With the sliding-sessions solution in place:
  + Provided a user remains active in the a-Portal Web application, SharePoint will not interrupt the user and require him to re-authenticate with the SAML token issuer.
  + The a-Portal Web application remains secure because users who become inactive must re-authenticate when they start using the application again.
  1. **#!CALLOUT!#**
  2. It's important that the sliding-session implementation is compatible with the single sign-out solution that Chapter 11, "Claims-Based Single Sign-On for Microsoft SharePoint 2010," describes.

# Inside the Implementation

The following sections describe the key configuration steps that Adatum performed in order to implement the scenario that this chapter describes. The hub model that Adatum selected meant that the changes in SharePoint were minimal: there is still a single trust relationship with the Adatum issuer.

* 1. **#!SPOKENBY(Poe)!#**
  2. The main configuration changes were in ADFS. Adding the trust relationship with Litware and adding the rules to convert Litware claims to Adatum claims.

The following sections describe the changes Adatum made to the a-Portal Web application in SharePoint to support access from partner organizations.

## Token Expiration and Sliding Sessions

* 1. One of the Adatum requirements was that the a-Portal application should automatically sign users out after a defined period of inactivity, but allows them to continue working with the application without re-authenticating so long as they remain active. To achieve this Adatum implemented a sliding session mechanism in SharePoint that can renew the SharePoint session token. For performance reasons, Adatum wanted to be able to extend the lifetime of the session token without having to re-visit ADFS (the FP) or the partner's token issuer.
  2. A cookie (usually named FedAuth) that can exist either as a persistent or in memory cookie represents the SharePoint session token. This cookie contains a reference to the SAML token that SharePoint stores in its token cache. The SAML token contains the claims issued to the user by any external identity and federation providers, and by the internal SharePoint STS.
  3. Before showing the details of how Adatum implemented sliding sessions, it will be useful to understand how token expiration works by default in SharePoint.

### SAML Token Expiration in SharePoint

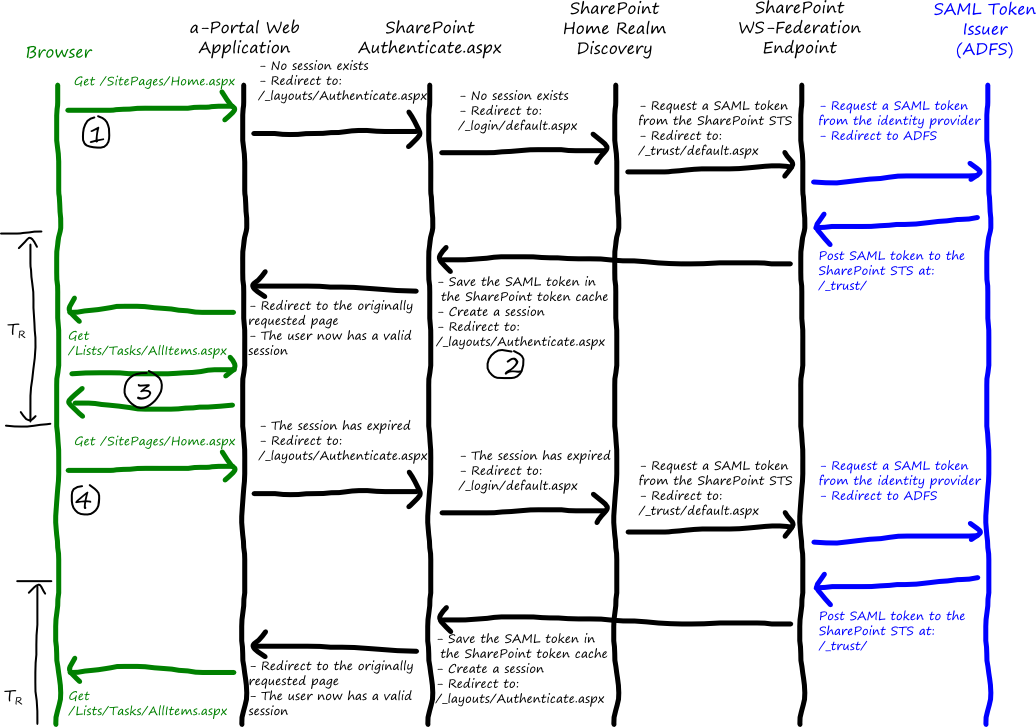
* 1. This section describes the standard behavior in SharePoint as it relates to token expiration.
  2. When Rick from Litware first tries to access the a-Portal Web application, his browser performs all of the following steps in order to obtain a valid SAML token:
  3. Rick requests a page in the a-Portal Web application.
  4. Rick's browser redirects to the SharePoint STS.
  5. Because Rick is not yet authenticated, the SharePoint STS redirects Rick's browser to the Adatum issuer to request a token.
  6. The Adatum issuer redirects Rick's browser to the Litware issuer to authenticate and obtain a Litware token.
  7. Rick's browser returns to the Adatum issuer to transform the Litware token into an Adatum token.
  8. Rick's browser returns to the a-Portal Web application to sign in to SharePoint.
  9. Rick's browser returns to the page that Rick originally requested in the a-Portal Web application to view.

All SAML tokens have a fixed lifetime that the token issuer specifies when it issues the token: in the Adatum scenario, it is the Adatum ADFS that sets this value. Once a token has expired, the user must request a new SAML token from the token issuer. For Rick at Litware, this means repeating the steps listed above. Because this takes time, Adatum does not want users such as Rick to have to re-authenticate too frequently. However, using a token with a long lifetime can be a security risk because someone else could use Rick's computer while he wasn't there and access the a-Portal Web application with Rick's cached token.

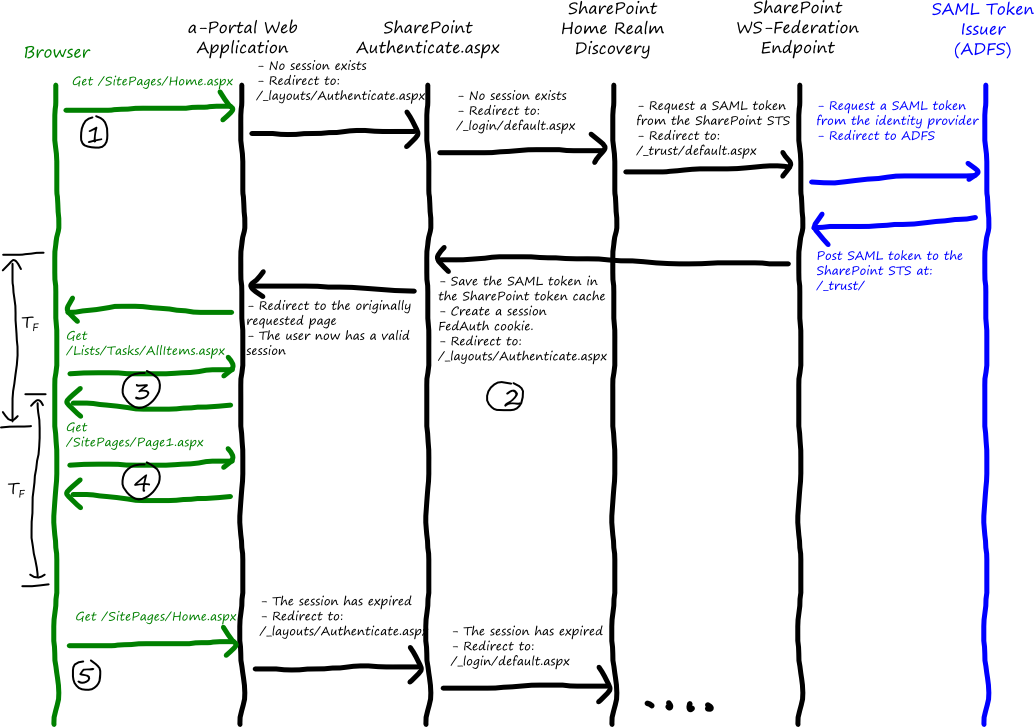
* 1. **#!SPOKENBY(Jana)!#**
  2. When Rick's SAML token expires he may, or may not, need to re-enter his credentials at the token issuer (ADFS): this depends on the configuration of the issuer. In ADFS, you can specify the Web Single Sign-On (SSO) lifetime that determines the lifetime of the ADFS SSO cookie.

The following table describes the two configuration options that directly affect when SharePoint requires a user to get a new SAML token from the issuer.

|  |  |
| --- | --- |
| Configuration value | Notes |
| SAML token lifetime | The token issuer sets this value. In ADFS, you can configure this separately for each RP by using the **Set-ADFSRelyingPartyTrust** PowerShell command.  Once the SAML token expires, the SharePoint session expires, and the user must re-authenticate with the token issuer to obtain a new SAML token.  By default, SharePoint sets the session lifetime to be the same as the SAML token lifetime. |
| LogonTokenCacheExpirationWindow | This SharePoint configuration value controls when SharePoint will consider that the SAML token has expired and ask the user to re-authenticate with the issuer and obtain a new token. SharePoint checks whether the SAML token has expired at the start of every request.  For example, if ADFS sets the SAML token lifetime to ten minutes, and the **LogonTokenCacheExpirationWindow** property in SharePoint is set to two minutes, then the session in SharePoint will be valid for eight minutes:   * + If the user requests a page from SharePoint seven minutes after signing in, then SharePoint checks whether the session is set to expire during the next **LogonTokenCacheExpirationWindow** minutes: in this case the answer is no because seven plus two is less than ten.   + If the user requests a page from SharePoint nine minutes after signing in, then SharePoint checks whether the session is set to expire during the next **LogonTokenCacheExpirationWindow** minutes: in this case the answer is yes because nine plus two is greater than ten. |

* 1. **#!SPOKENBY(Poe)!#**
  2. Make sure that the value of the **LogonTokenCacheExpirationWindow** property is always less than the SAML token lifetime; otherwise, you'll see a loop whenever a user tries to access your SharePoint Web application and keeps being redirected back to the token issuer.
  3. The following script example shows you how to change the lifetime of the SAML token issued by the "SharePoint Adatum Portal" RP in ADFS to 10 minutes.
  4. Add-PSSnapin Microsoft.ADFS.PowerShell
  5. Set-AdfsRelyingPartyTrust –TargetName "SharePoint Adatum Portal" –TokenLifeTime 10
  6. The following script example shows you how to change the **LogonTokenCacheExpirationWindow** in SharePoint to two minutes.
  7. $ap = Get-SPSecurityTokenServiceConfig
  8. $ap.LogonTokenCacheExpirationWindow = (New-TimeSpan -minutes 2)
  9. $ap.Update();
  10. IIsreset
  11. These two configuration settings will cause SharePoint to redirect the user to the issuer to sign in again eight minutes after the user last authenticated with ADFS.
  12. The sequence diagram in Figure 4 shows how SharePoint manages its session lifetime and the SAML token that it receives from the token issuer.
  13. 
  14. Figure 4
  15. Standard token expiration in SharePoint
  16. The sequence diagram shows a simplified view of the sequence of interactions. In reality, SharePoint and the WS-Federation protocol use browser redirects and automatic posts to manage the interactions between the various components so that all of the requests go through the browser.
  17. In the sequence diagram, TR represents the time from when ADFS issues the SAML token after to when SharePoint will try to renew the token. Based in the configuration settings described above, TR is set to eight minutes.
  18. The following notes refer to the numbers on the sequence diagram:
  19. This is the first time that the user visits the a-Portal Web application; there is no valid session so SharePoint redirects the user to begin the sign-in process.
  20. SharePoint creates a session for the user. The lifetime of the session is the same as the lifetime of the SAML token issued by ADFS.
  21. SharePoint uses the session lifetime and the **LogonTokenCacheExpirationWindow** property to determine when the user must sign-in again. At this point, the session is still valid. While the session is valid, the user can continue to visit pages in the SharePoint Web application.
  22. SharePoint uses the session lifetime and the **LogonTokenCacheExpirationWindow** property to determine when the user must sign-in again. At this point, SharePoint determines that the session has expired, so it begins the sig-in process again. If the ADFS SSO cookie has expired, Rick will have to enter his credentials to obtain a new SAML token.
  23. To force users to re-enter their credentials whenever they are redirected back to ADFS, you should set the Web SSO lifetime in ADFS to be less than or equal to **SAMLtokenlifetime** less **LogonTokenCacheExpirationWindow**. In the Adatum scenario, the Web SSO lifetime in ADFS should be set to eight minutes to force users to re-authenticate when SharePoint redirects them to ADFS.

### Sliding Sessions in SharePoint

* 1. Adatum wanted to implement sliding sessions so that SharePoint can extend the lifetime of the session if the user remains active. Adatum wanted to be able to define an inactivity period, after which SharePoint forces the user to re-authenticate with ADFS. In other words, a user will only need to sign-in again if the session is allowed to expire or if the SAML token expires: in this scenario, the session lifetime will be less than the SAML token lifetime.
  2. To implement this behavior, Adatum first configured ADFS to issue SAML tokens with a lifetime of 8 hours. The following PowerShell script shows how you can configure this setting in ADFS for the "SharePoint Adatum Portal" RP.
  3. Add-PSSnapin Microsoft.ADFS.PowerShell
  4. Set-AdfsRelyingPartyTrust –TargetName "SharePoint Adatum Portal" –TokenLifeTime 480
  5. By setting the **LogonTokenCacheExpirationWindow** value to 470 minutes, Adatum can effectively set the session duration to ten minutes.
  6. $ap = Get-SPSecurityTokenServiceConfig
  7. $ap.LogonTokenCacheExpirationWindow = (New-TimeSpan -minutes 470)
  8. $ap.Update();
  9. IIsreset
  10. Adatum then modified the way that SharePoint manages its sessions: SharePoint now recreates a new session before the existing session expires (so long as the user visits the SharePoint Web application before the existing session expires). A user can continue to recreate sessions up to the time that the SAML token finally expires; in this scenario, the user could continue using the a-Portal Web application for eight hours without having to re-authenticate. If the user doesn't visit the Web application before the session expires, then on the next visit they must sign in again. The Visual Studio® solution SlidingSessionModule in the 10‑SharePoint folder from <http://claimsid.codeplex.com> includes a custom HTTP module to deploy to your SharePoint Web application that includes this functionality. The following code sample from the Adatum custom HTTP module shows the implementation.
  11. **#!SPOKENBY(Markus)!#**
  12. Remember: A reference to the SAML token in the SharePoint token cache is stored in the session. The session is represented by the FedAuth cookie.
  13. public void Init(HttpApplication context)
  14. {
  15. // Sliding session
  16. FederatedAuthentication.SessionAuthenticationModule.SessionSecurityTokenReceived +=
  17. SessionAuthenticationModule\_SessionSecurityTokenReceived;
  18. …
  19. }
  20. private void SessionAuthenticationModule\_SessionSecurityTokenReceived(
  21. object sender, SessionSecurityTokenReceivedEventArgs e)
  22. {
  23. double sessionLifetimeInMinutes =
  24. (e.SessionToken.ValidTo - e.SessionToken.ValidFrom).TotalMinutes;
  25. var logonTokenCacheExpirationWindow = TimeSpan.FromSeconds(1);
  26. SPSecurity.RunWithElevatedPrivileges(delegate()
  27. {
  28. logonTokenCacheExpirationWindow =
  29. Microsoft.SharePoint.Administration.Claims.SPSecurityTokenServiceManager
  30. .Local.LogonTokenCacheExpirationWindow;
  31. });
  32. DateTime now = DateTime.UtcNow;
  33. DateTime validTo = e.SessionToken.ValidTo - logonTokenCacheExpirationWindow;
  34. DateTime validFrom = e.SessionToken.ValidFrom;
  35. if ((now < validTo) &&
  36. (now > validFrom.AddMinutes((validTo - validFrom).TotalMinutes / 2)))
  37. {
  38. SessionAuthenticationModule sam =
  39. FederatedAuthentication.SessionAuthenticationModule;
  40. e.SessionToken = sam.CreateSessionSecurityToken(e.SessionToken.ClaimsPrincipal,
  41. e.SessionToken.Context, now, now.AddMinutes(sessionLifetimeInMinutes),
  42. e.SessionToken.IsPersistent);
  43. e.ReissueCookie = true;
  44. }
  45. }
  46. This method first determines the valid from time and valid to time of the existing session, taking into account the value of the **LogonTokenCacheExpirationWindow** property. Then, if the existing session is more than halfway through its lifetime, the method uses the **SPSessionAuthenticationModule** instance to extend the session. It does this by creating a new session that has the same lifetime as the original, but which has a **ValidFrom** property set to the current time.
  47. The sequence diagram in Figure 5 shows how SharePoint handles Adatum's sliding-sessions implementation.
  48. 
  49. Figure 5
  50. Sliding Sessions in the a-Portal Web application
  51. The sequence diagram shows a simplified view of the sequence of interactions. In reality, SharePoint and the WS-Federation protocol use browser redirects and automatic posts to manage the interactions between the various components so all of the requests go through the browser.
  52. In the sequence diagram, TF represents the session lifetime. The session lifetime also defines the inactivity period, after which a user must re-authenticate with ADFS.
  53. The following notes refer to the numbers on the sequence diagram:
  54. This is the first time that the user visits the a-Portal Web application; there is no valid session so SharePoint redirects the user to begin the sign-in process.
  55. SharePoint creates a session for the user. The effective lifetime of the session is the difference between the lifetime of the SAML token issued by ADFS and the value of the **LogonTokenCacheExpirationWindow** property. For Adatum, the lifetime of the session is ten minutes: the lifetime of the SAML token is 480 minutes and the value of the **LogonTokenCacheExpirationWindow** property is 470 minutes.
  56. SharePoint checks the age of the session. At this point, although session is still valid, it is nearing the end of its lifetime so SharePoint creates a new session, copying data from the existing session.
  57. SharePoint checks the age of the session. At this point, it is still near the beginning of its lifetime so SharePoint continues to use this session.
  58. SharePoint checks the age of the session. At this point, the session has expired so SharePoint initiates the process of re-authenticating with the IdP.

### Closing the Browser

* 1. Adatum wanted users to be able to close the browser, but be able to access the a-Portal Web application without re-authenticating if they re-opened the browser before the inactivity period expired. This will happen if the FedAuth cookie that contains the reference to the SAML token is a persistent cookie: this is the default behavior in SharePoint.
  2. The default behavior for SharePoint is to use persistent session cookies: this enables a user to close the browser, re-open the browser, and re-visit a SharePoint Web application without signing in again. Adatum wants users to always re-authenticate if they close the browser and then re-open it and revisit the a-Portal Web application. To enforce this behavior, Adatum configured SharePoint to use an in-memory instead of a persistent session cookie. You can use the following PowerShell script to do this.
  3. $sts = Get-SPSecurityTokenServiceConfig
  4. $sts.UseSessionCookies = $true
  5. $sts.Update()
  6. iisreset

## Authorization Rules

* 1. With multiple partners having access to the a-Portal SharePoint Web application, Adatum wants to have the ability to restrict access to documents in the SharePoint document library based on the organization that the user belongs to. Adatum wants to be able to use the standard SharePoint groups mechanism for assigning and managing permissions, so it needs some way to identify the organization a user belongs to.
  2. Adatum achieves this objective by using claims. Adatum has configured ADFS to add an **organization** claim to the SAML token it issues based on the federated identity provider that originally authenticated the user. You should not rely on the identity provider to issue the organization claim because a malicious administrator at a partner organization could add an **organization** claim with another partner's value and gain access to confidential data.
  3. Chapter 11, "Claims-Based Single Sign-On for Microsoft SharePoint 2010," describes how to add the organization claim to the SharePoint people picker to make it easy for site administrators to set permissions based on the value of the **organization** claim.

## Home Realm Discovery

* 1. If Adatum shares the a-Portal Web application with multiple partners, each of those partners will have its own IdP as shown in Figure 2 earlier in this chapter. With multiple IdPs in place, there must be some mechanism for selecting the correct IdP for a user to use for authentication: the home-realm-discovery process.
  2. Adatum decided to customize the home-realm-discovery page that ADFS provides. The default page in ADFS (/adfs/ls/HomeRealmDiscovery.aspx) displays a drop-down list of the claims provider trusts configured in ADFS (claims provider trusts represent IdPs in ADFS)for the user to select an IdP, ADFS then redirects the user to the sign-in page at the IdP. It's easy to customize this page with partner logos to make it easier for users to select the correct IdP. In addition, this page in ADFS has access to the RP identifier in the **wtrealm** parameter so it can customize the list of IdPs based on the identity of the SharePoint RP Web application. After a user has selected an IdP for the first time, ADFS can remember the choice so that in future, the user bypasses the home-realm-discovery page and redirect the browser directly to the IdPs sign-in page.
  3. **#!CALLOUT!#**
  4. Claims provider trusts represent IdPs in ADFS.
  5. For details about how to customize the ADFS home realm discovery page and configure how long ADFS will remember a user's selection, see this page on the MSDN Web site: <http://msdn.microsoft.com/en-us/library/bb625464(VS.85).aspx>.
  6. Adatum also considered the following options related to the home-realm-discovery page.
  + Automatically determine a user's home realm based on the user's IP address. This would remove the requirement for the user to specify her home realm when she first visits ADFS, however this approach is not very reliable, especially with mobile and home workers and does not provide any additional security because IP addresses can be spoofed.
  + Perform the home realm discovery in SharePoint instead of ADFS. Adatum could customize the standard SharePoint login page (usually located at C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\template\identitymodel\login\default.aspx) to display the list of IdPs, and then append a **whr** parameter identifying the user's home realm to the address of the ADFS sign-in page. However, the SharePoint login page only displays to the user if multiple authentication types are configured in SharePoint: Adatum only has a single authentication type configured for the a-Portal Web application so Adatum would need to override the behavior of the standard login page so that it always displays. By default, all SharePoint Web applications share this login page so SharePoint would display the same list of IdPs regardless of the SharePoint Web application the user is accessing: you can override this behavior and display a separate login page for each SharePoint Web application.
  1. **#!SPOKENBY(Poe)!#**
  2. You should be sure to keep your SharePoint environment up to date with the latest patches from Microsoft.

# More Information

For more information about WIF and Sliding Sessions see this post: <http://blogs.msdn.com/b/vbertocci/archive/2010/06/16/warning-sliding-sessions-are-closer-than-they-appear.aspx>

For more information about automated home realm discovery, see chapter 6, "Federated Identity with Multiple Partners," and chapter 7, "Federated Identity with Multiple Partners and Windows Azure Access Control Service."