Implementing Single Sign-On using ADFS/SAML2

v1.0, September 2014

Contents

Summary

This guide outlines a process which will allow you to achieve the following:

- Understand how Single Sign-On (SSO) using SAML2 works in the context of WebWhiteboard.
- Understand the infrastructure required on your establishment's side in order to set up SSO.
- Configure your SSO infrastructure to talk to WebWhiteboard.
- Send us the information we need in order to complete the process and enable SSO integration for your organisation's WebWhiteboard account

We are assuming the following:

- You are familiar with WebWhiteboard (WWb).
- You are familiar with Windows, Active Directory and configuring Windows Server operating systems.
- You are unafraid to research concepts that are new to you and 'read around the subject' if necessary.

This guide has been written from the perspective of an establishment with a Windows Active Directory network, using Active Directory Federation Services (ADFS) as the SSO server, as this is the position that most of our clients will be in.

You can use any SAML2 server with WebWhiteboard, including Shibboleth. The server should be configured in the same way as ADFS, the only difference will be *how* you configure it.

If you're already familiar with SAML2/ADFS, feel free to skip to the 'Configuring ADFS to talk to WebWhiteboard' section.

Introduction to SAML2 authentication

SAML2 stands for **Security Assertion Markup Language v2**, and is a standard method of exchanging authentication and authorization[1] data between parties (the end user, the service provider – that's us, and the identity provider – that's you, or more specifically your internal domain).

For those new to the idea of SSO or authentication frameworks SAML can be difficult to understand, with lots of opaque vocabulary and complicated processes to wrap your head around. Hopefully this guide will help cut through some of the fog.

Single Sign-On – what exactly are you on about?

Single Sign-On is one of those 'buzzwords' that many people claim to be able to support in order to tick a box and get enterprise IT buyers interested, but is often poorly understood.

At its simplest, SSO means that instead of a user logging in to a service directly with their username and password and that service deciding whether the user should be let in, the service redirects the user to an 'identity provider' (IdP) to get a ticket. The service then checks that ticket and decides what to do with the user.

In this specific case, the IdP is your Active Directory domain – the user will enter their standard domain username and password into a form hosted by something living on your network, which then passes something like a ticket to WebWhiteboard. If WWb views the ticket as valid, the user is let in.

The advantages here are that the user doesn't have to have a separate username and password and that you don't have to worry about creating a WWb account for all of your users. It'll happen automatically when a new user first logs in.

So what's SAML got to do with all this?

SAML2 is the language that the ticket is written in. It's based on XML, and usually sent in an encrypted form to prevent snooping.

The SAML2 ticket will contain a number of 'claims'[2] – things it is telling the service provider about the user. These usually include some form of unique ID (such as a GUID or Windows SID, or even an email address), a list of groups to which the user belongs and perhaps the organisation or domain to which the user belongs.

They may also contain supplementary information which isn't needed to identify a user but may improve the user's experience with the receiving service, such as the user's name.

WebWhiteboard needs two things from your organisation's domain – a unique ID for each user, and your domain name (so that we can put the user in the right group our end). You can also supply a few other things should you wish – more on that in a little bit.

By now you might be asking "but how does the service know whether I want to let the user in or not?"

The simple answer is that you simply tell the server to what type of user it should issue tickets! No ticket = no access. For instance, you can tell your server to only issue tickets to users who are in the 'Staff' group.

Getting started with Active Directory Federation Services

This is the exciting bit you've been waiting for – getting something set up and ready to talk to WWb! If you've already got ADFS set up (e.g. for Office 365) you're probably bored stiff by this point and can skip ahead to the 'Configuring…' section.

---

[1] Authentication covers who you are; authorization covers what you are permitted to do

[2] Identity Providers are sometimes called 'Claims Providers' for this reason. For instance, ADFS uses this terminology.

Installing ADFS

You'll need a server running at least Windows Server 2008 in order to install ADFS (your domain's functional level doesn't matter).

You'll also need an SSL certificate for the server you're putting ADFS on – for testing this can be a self-signed cert. For production use, a full certificate issued by a Certification Authority is highly recommended.

Explaining the process of getting ADFS installed here would be pointless as there are countless guides on the Internet and the process will differ slightly depending on which iteration of Windows Server you're using.

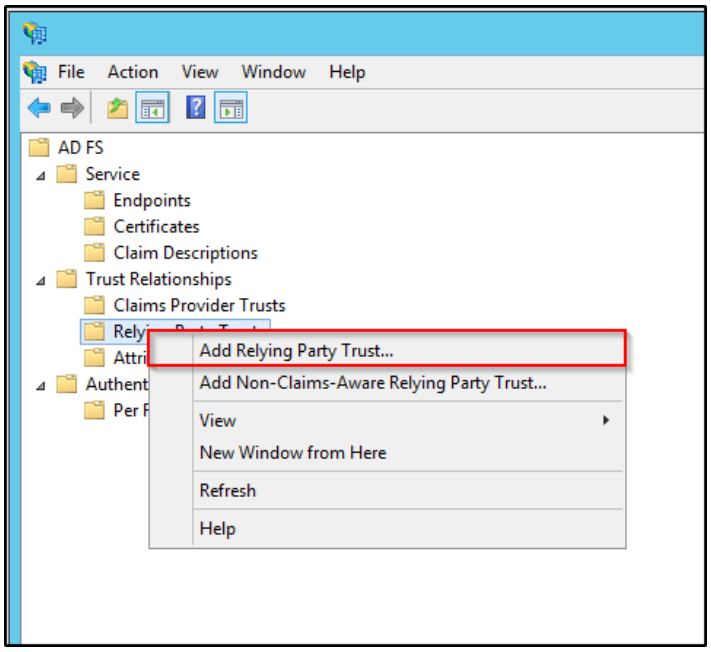
If you're using Server 2012 R2, we'd recommend following the Microsoft Office 365 team's excellent guide on their TechNet blog (part 1, part 2).

Double-check that the service can be accessed from *outside your organisation* – if it isn't then SSO users will not be able to sign in from home!

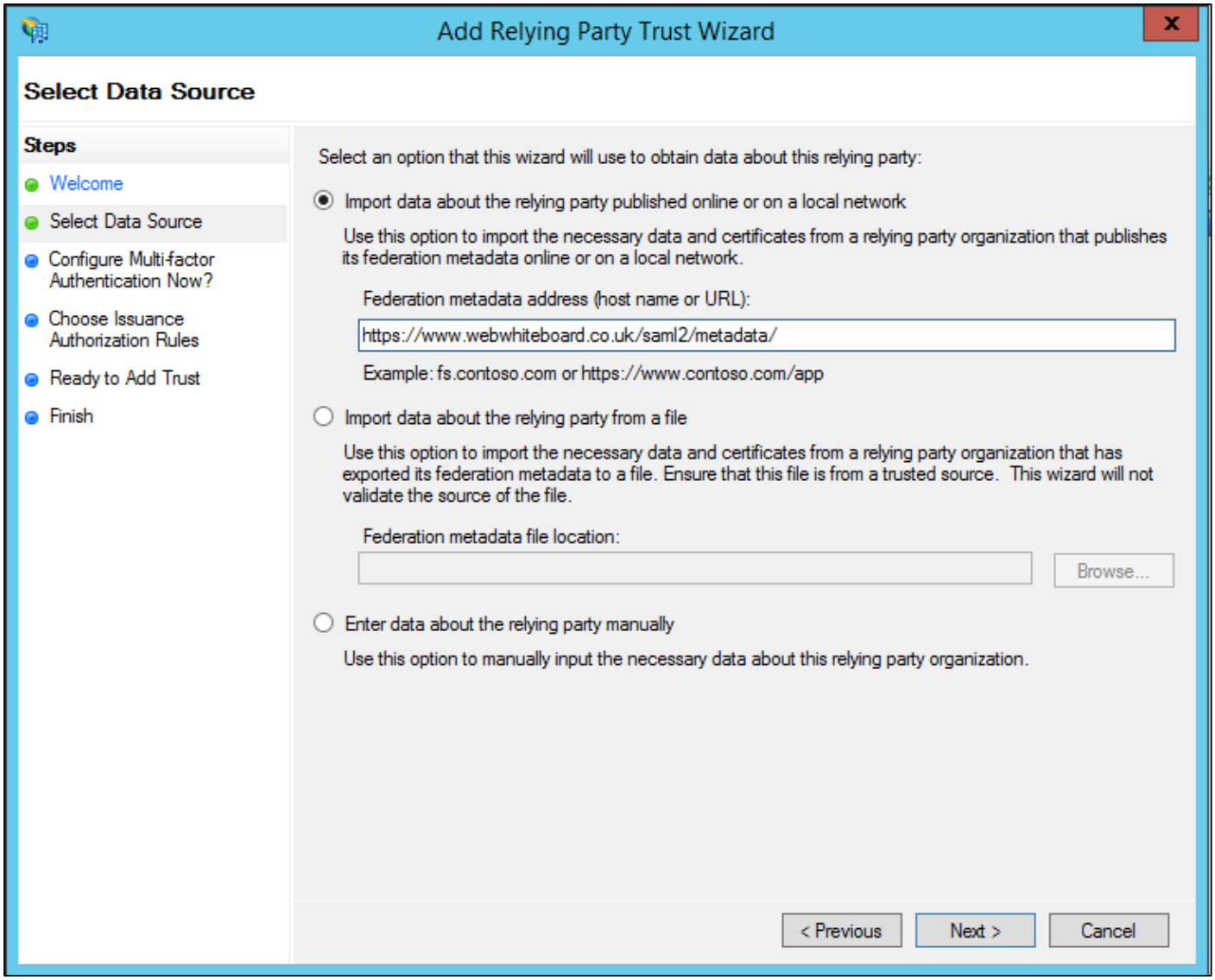
Configuring ADFS to talk to WebWhiteboard

The first thing we'll need to do is tell ADFS about WebWhiteboard, and where it should send those SAML2 tickets we talked about earlier.

1. Load up the ADFS console (***Start > Active Directory Federation Services*** on 2012, ***Start > Administrative Tools > ADFS*** on 2008).
2. Expand ***Trust Relationships***, right-click ***Relying Party Trusts*** and click ***Add Relying Party Trust…***
   A 'Relying Party' is what ADFS calls anything that uses the tickets it gives out for authentication. They *rely* on it to authenticate users.

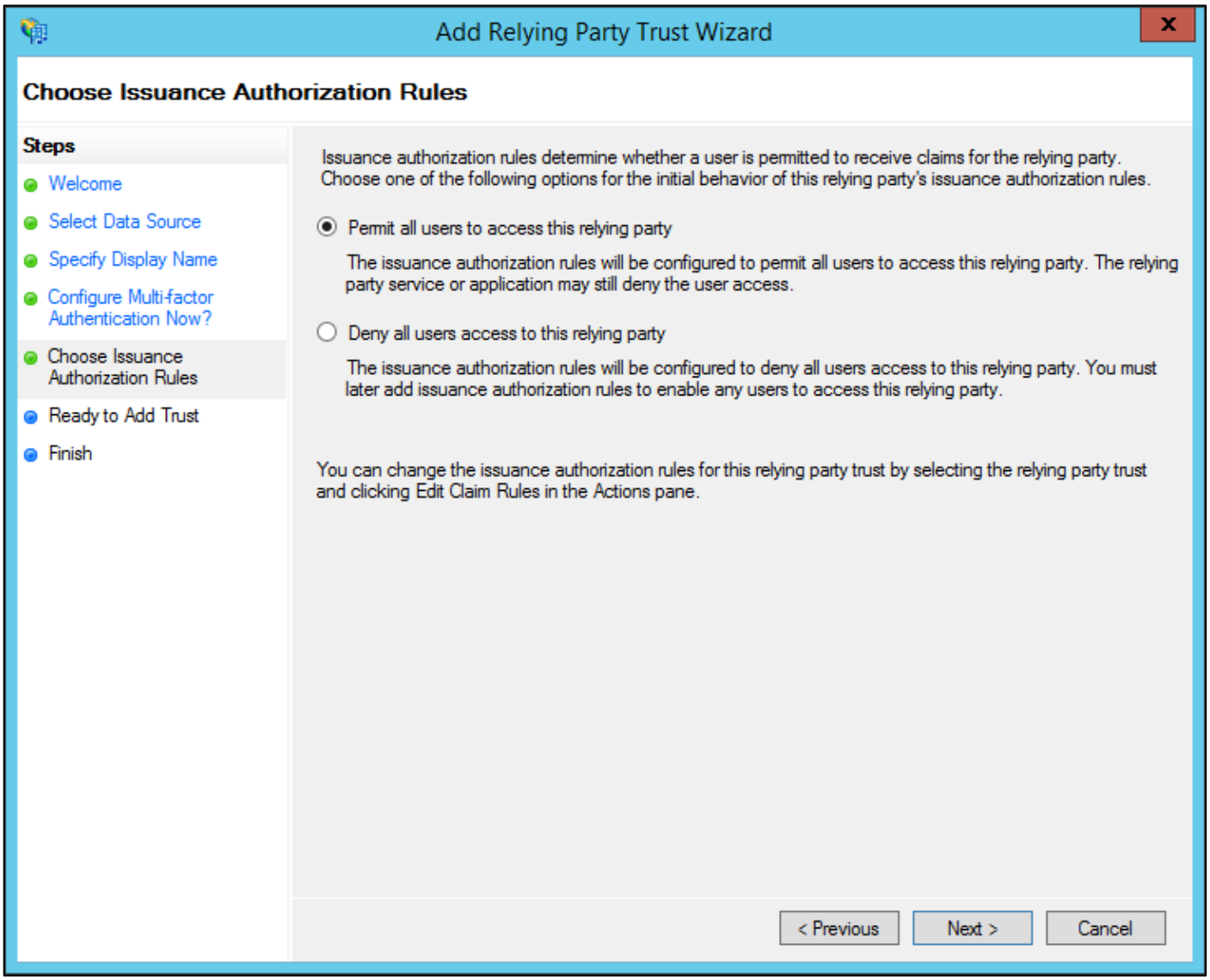


3. Click ***Start*** and then enter `https://www.webwhiteboard.co.uk/saml2/metadata/` in the top text box. This points ADFS to WebWhiteboard's metadata: an XML document that tells it about WWb's systems, where to send those tickets to, where to redirect users to after they've logged in and some other things.



4. On the next screen, enter a name and a description for the WWb relying party if you want and click ***Next***.
5. Leave the Multi-factor authentication settings as they are and click ***Next*** again.

6. On this screen you can set your 'Issuance Authorization Rules' – telling ADFS who it should allow to access WWb. To make life easier initially, select ***Permit all users…*** and then click ***Next***. We'll come back to Issuance Authorization later on.

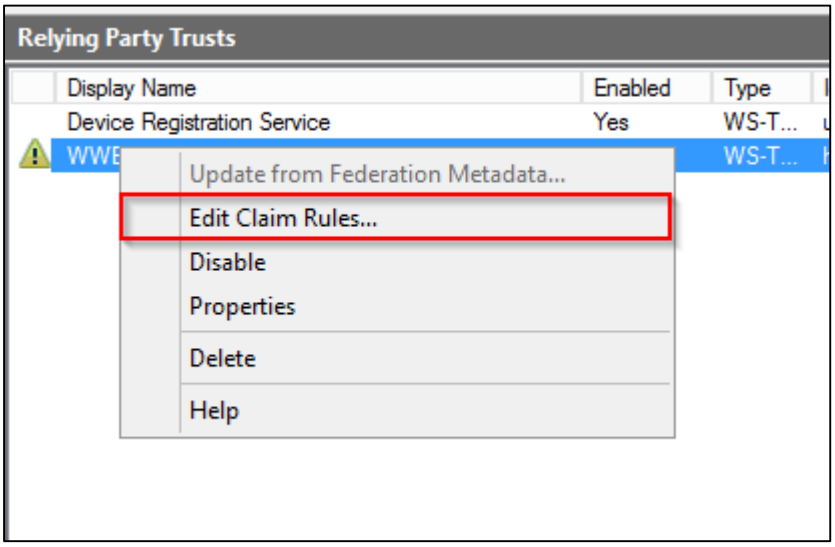


7. The next screen shows you the details of the trust relationship you're about to create. Have a look through the various tabs and when you're happy, click ***Next*** and then ***Finish***.

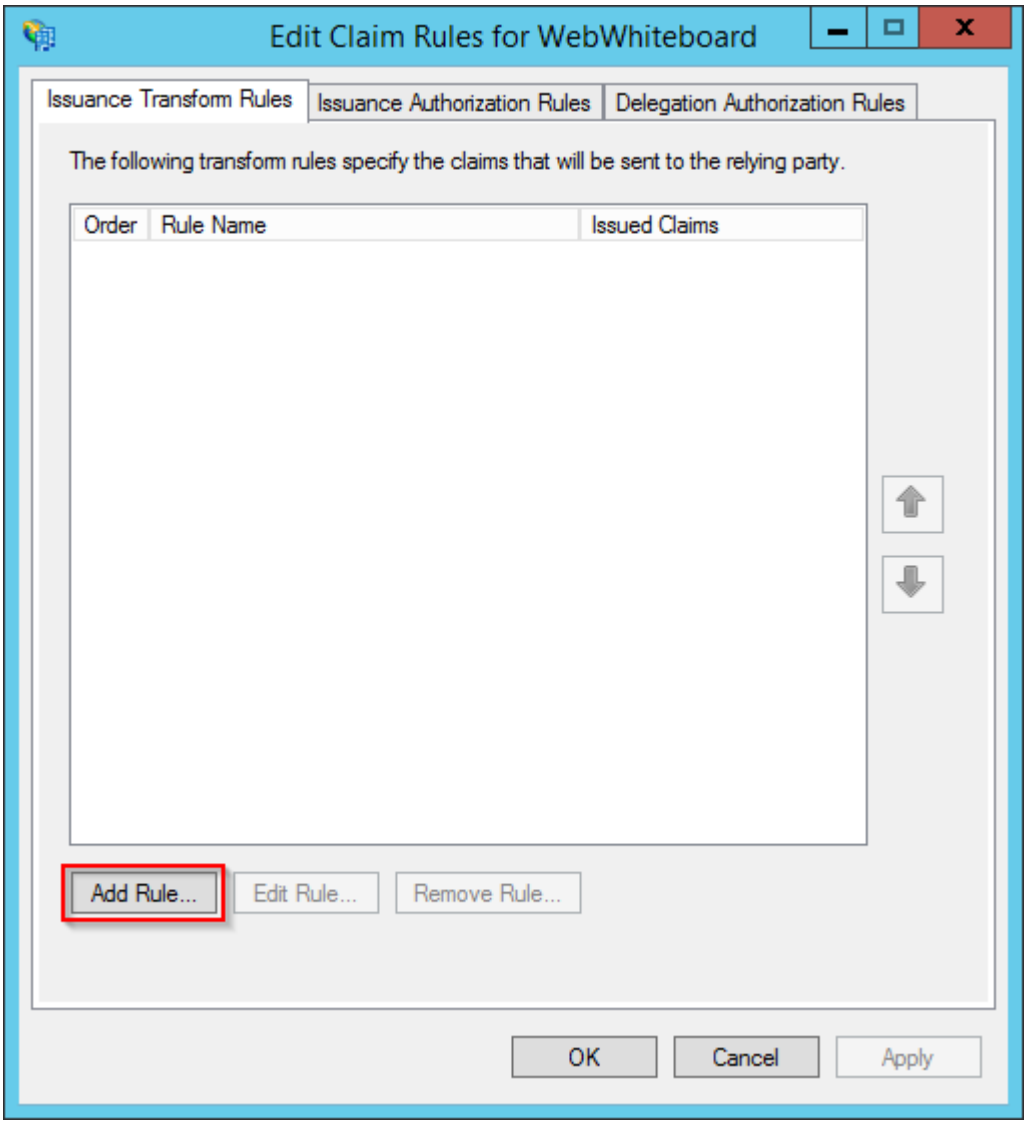
Configuring ADFS to send claims to WebWhiteboard

Okay, so ADFS knows about WWb, but what will it actually send? Remember we talked about the various 'claims' that were passed from the identity provider (ADFS) to the service provider (WWb)? Well we're about to sort them out.

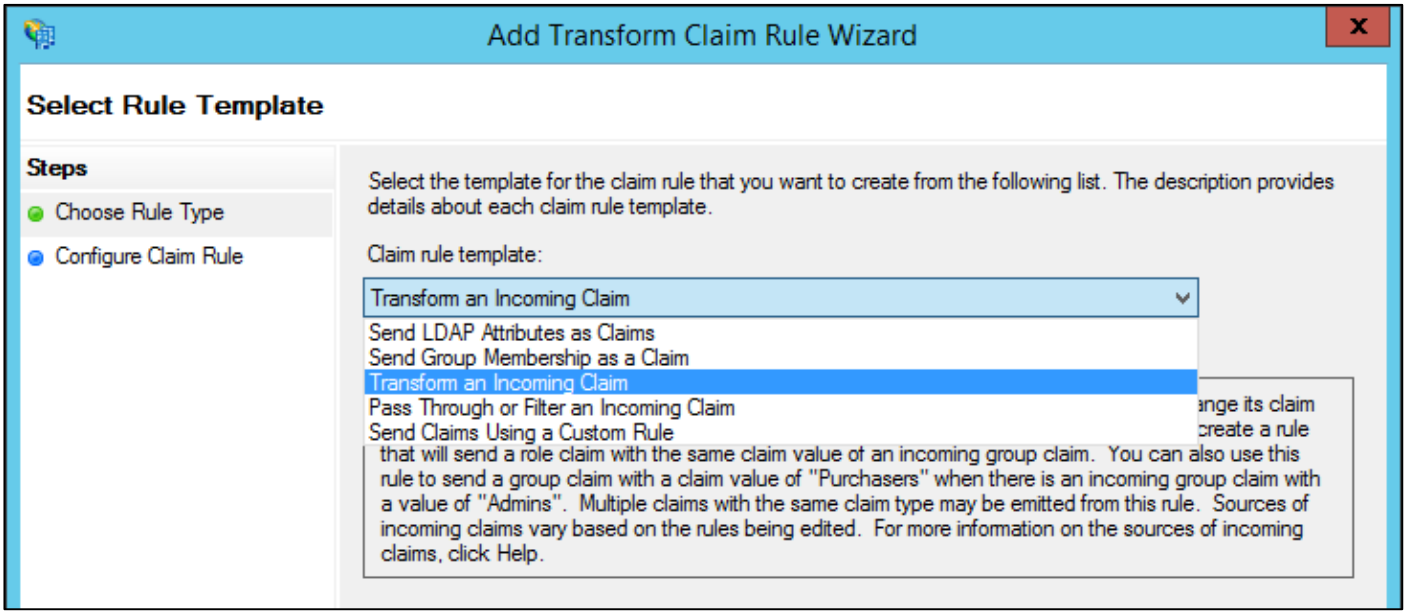
1. Right click the trust you've just created, and click ***Edit Claim Rules***.



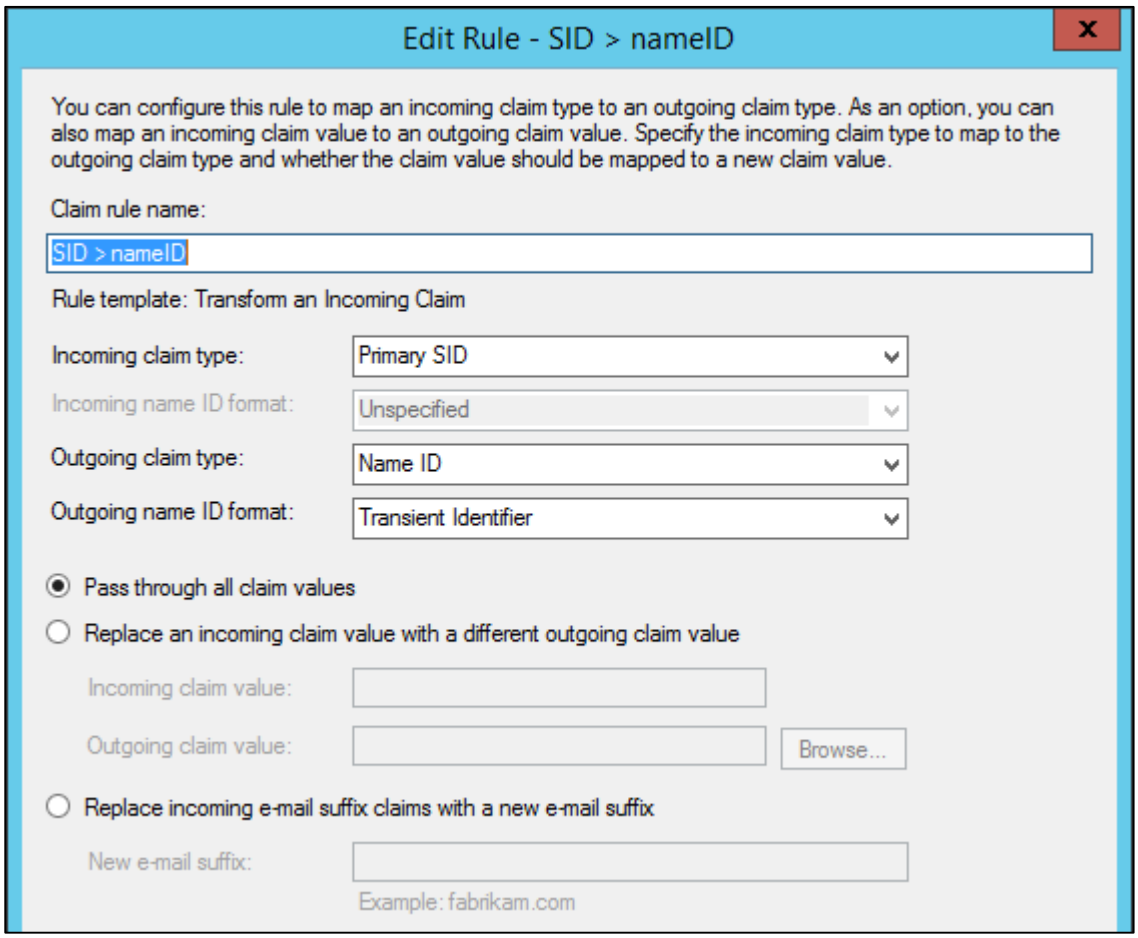
2. You can do all sorts of interesting things with Claim Rules, but for now we're just interested in getting things working. Click ***Add Rule…*** to get started.



3. Firstly, we need to supply WWb with a unique ID for each user – we're going to use the Object SID for this purpose, as the SID is invariant for the life of a user account. The account name, username or other properties can be changed and the SID will remain the same.
   Select ***Transform an Incoming Claim*** as the template and click ***Next***.



4. On this screen, enter a name for the rule and enter the settings as shown below, then click ***Finish***.
   a. Incoming claim type: ***Primary SID***
   b. Outgoing claim type: ***Name ID***
   c. Outgoing Name ID format: ***Transient Identifier***
   d. Pass through all claim values

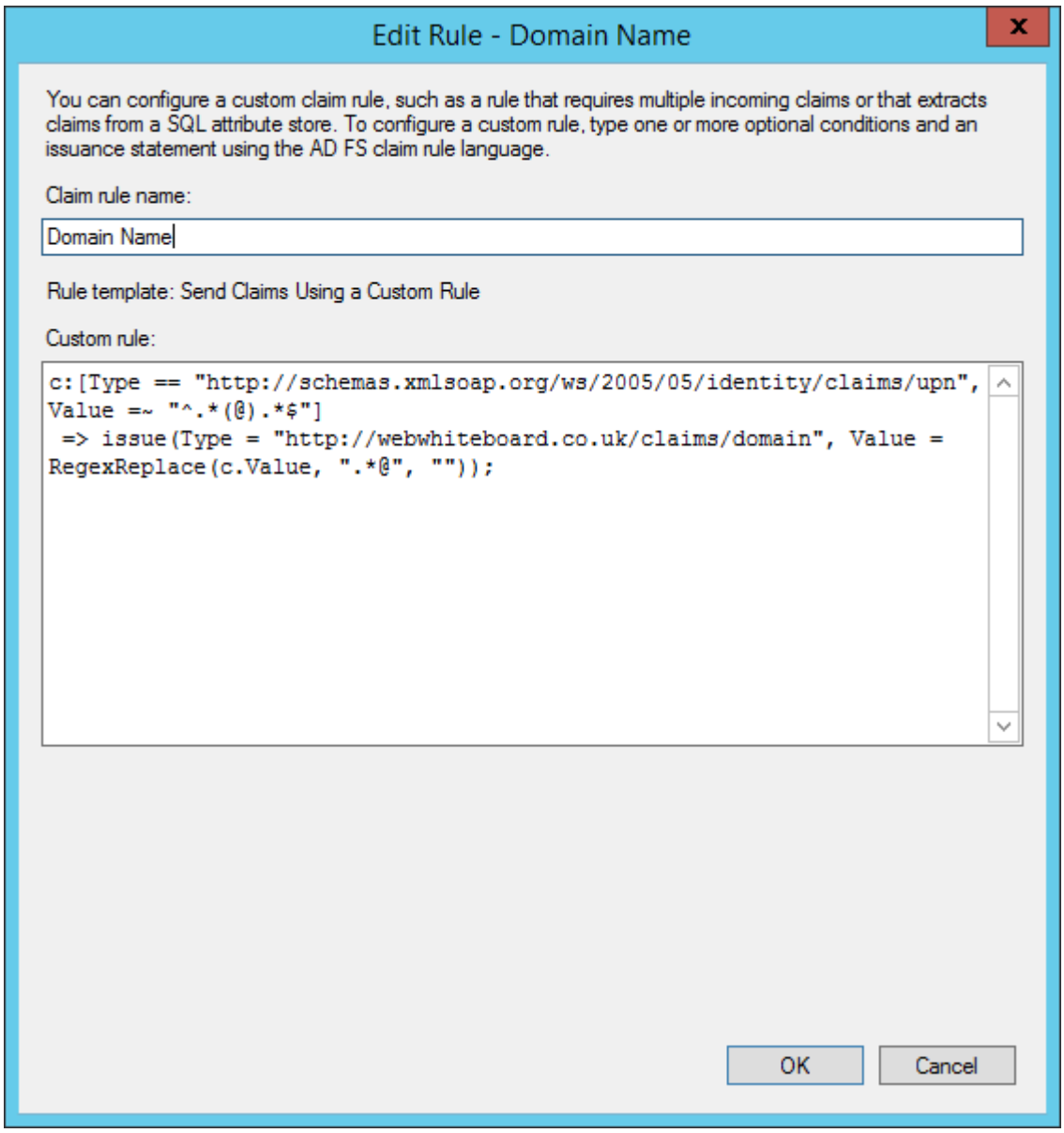


5. Next up is the domain name – this will naturally be the same for all users, and is needed so that WWb can tell to which institution a user belongs. Click ***Add Rule…*** again, but this time select ***Send Claims Using a Custom Rule*** as the Claim Rule Template.

   Copy and paste the following into the ***Custom rule*** text area:

   ```
   c:[Type == "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn", Value =~ "^.*(@).*$"] => issue(Type = "http://webwhiteboard.co.uk/claims/domain", Value = RegexReplace(c.Value, ".*@", ""));
   ```

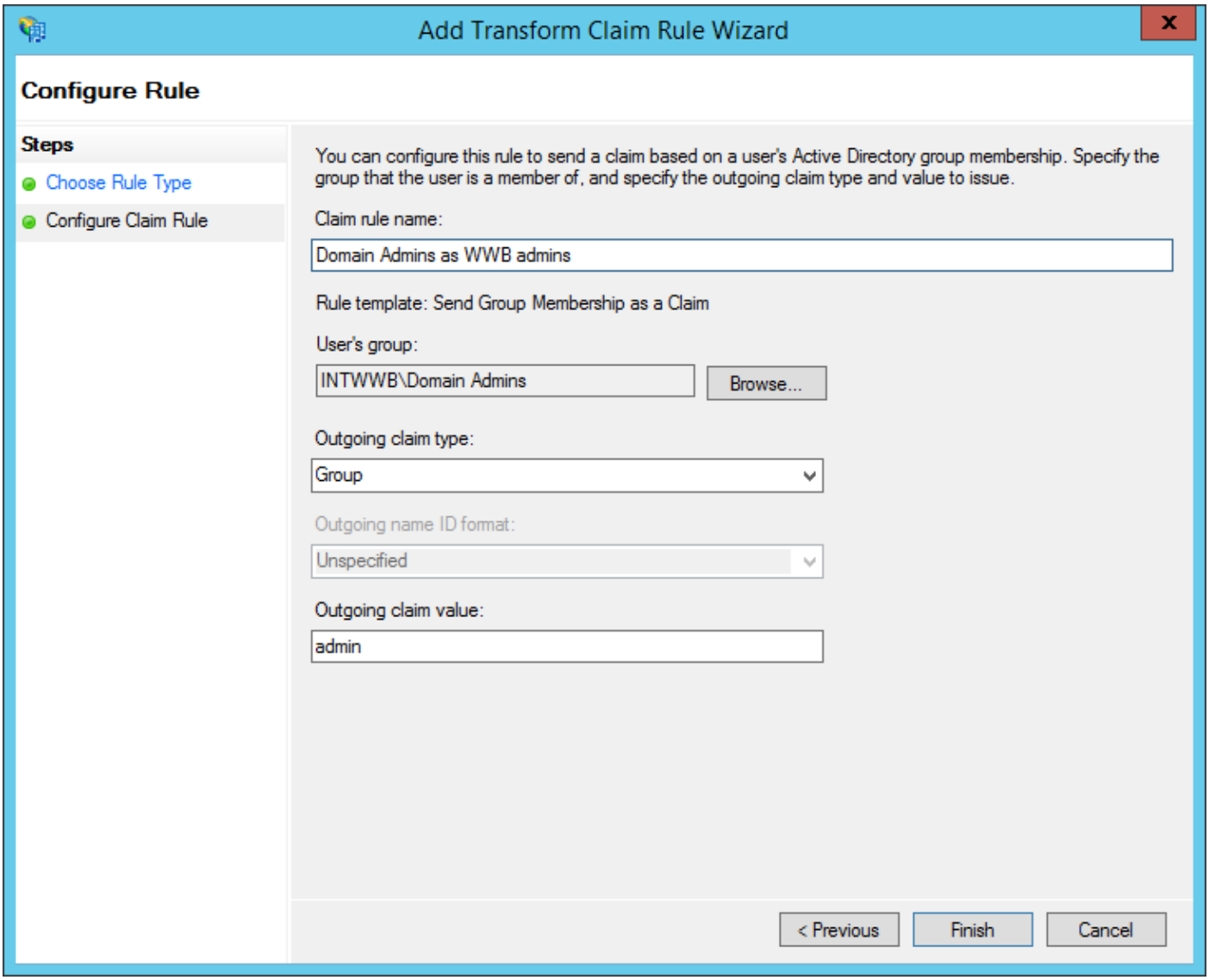
   This rule takes a user's Principal Name (in the form `username@domain.tld`, extracts the domain and sends it on.



6. Congratulations, you've just configured your ADFS server for SSO integration with WebWhiteboard!

   If you want, you can send an extra bit of info which will grant administrator access to your WWb account for a particular group of AD users (e.g. Domain Admins). This is completely optional, but if you want to do it, read on.
   It's worth knowing that if a user is later removed from the group that grants access WWb will pick this up and amend the user's permissions appropriately.

7. Click ***Add Rule…*** again, and select ***Send Group Membership as a Claim*** as the Claim Rule Template.
8. Select the AD group you want to give WWb administration rights to by clicking ***Browse…***, and set the other options as follows:
    a. Outgoing claim type: ***Group***
    b. Outgoing claim value: ***admin***



9. If you want multiple groups to be assigned admin rights, create another rule like this one but select a different AD group for the ***User's group*** setting.

10. You can also send claims for the user's name and email address, if they are stored in AD and you want them to appear in WebWhiteboard.
    a. Create a rule, selecting ***Pass Through or Filter an Incoming Claim*** as the template.
    b. For the ***Incoming Claim Type***, select either ***Given Name***, ***Last Name*** or ***E-Mail Address*** depending on which you want to send (you'll need to create one rule for each).
    c. Leave all other settings as they are and click ***Finish***.

Enabling SSO integration for your account

Now you've set up and configured ADFS, you need to tell us about it! That way we can configure on our end and get things lined up. It doesn't take long at all, so you won't have to wait around.

We hope to make this process completely automated in the near future, but for now it requires some old fashioned manual bit flipping.

1. Navigate to:
   `https://<YOUR_ADFS_SERVICE_NAME>/FederationMetadata/2007-06/FederationMetadata.xml`

   You should get an XML file either offered for download or displayed directly in your browser. You don't need to do anything with it, we just wanted to check that things are configured correctly.

2. Send an email to support@serenitysoftware.co.uk containing the following:
   a. Your organisation's name
   b. The URL of your FederationMetadata.xml that you checked in the previous step
   c. The domain in which your AD users reside
      (e.g. for a user with the UPN `jbloggs@ad.reynholmindustries.com` the domain we're after is `ad.reynholmindustries.com`)

3. Await with bated breath the reply telling you that everything's set up and ready to go! Once it is you can get testing – new users are created on the WWb end as needed, there's no 'pre-staging' involved.

4. If things don't work at first, double-check that everything is configured correctly, and that the ADFS service can be accessed externally. If you don't have any joy, contact us at support@serenitysoftware.co.uk giving as much information as possible (we love screenshots/transcripts of browser errors and event log excerpts☺).