CICS Transaction Server for z/OS

Configuring Kerberos for CICS with RACF and Microsoft Active Directory

John D. Taylor

IBM CICS

Table of contents

© Copyright IBM Corporation 2018

© Copyright IBM Corporation 2018

Introduction

This paper describes the steps required to configure Kerberos on IBM z/OS® and Microsoft Active Directory (MAD) to be used with CICS® Transaction Server for z/OS. It is aimed at CICS system programmers who want to configure CICS to use Kerberos with Microsoft Active Directory.

The paper starts with an introduction to Kerberos, and then describes how to configure RACF to use Kerberos with a CICS region. From there, the paper describes how to set up a simple configuration of Microsoft Active Directory and how to create trust between the two Kerberos servers. When each of these configurations are implemented, a simple 'hello world' application is demonstrated. The paper concludes with a troubleshooting section that helps with some common configuration errors and a list of references for further reading.

Throughout the configuration used in this paper, we've specified the use of only AES128 & AES256 as encryption standards to balance reasonable security with interoperability. This is fine for a test system, but care should be taken in choosing which encryption algorithms should be allowed in a production system.

About the author

John D. Taylor is a Software Engineer in CICS development at IBM Hursley Park, UK. He has 20 years of experience as an Application Programmer, Tester and developer of CICS TS and CICS tools within IBM. His areas of expertise include CICS, z/OS and Java™.

© Copyright IBM Corporation 2018

Introducing Kerberos

Kerberos is a network authentication protocol that was developed in the 1980s by Massachusetts Institute of Technology, in cooperation with IBM and Digital Equipment Corporation.

The Kerberos system consists of three components: a client, a server, and a trusted third party, which is also known as a *Key Distribution Center (KDC)*. The KDC interacts with both a client and server to accept the client's request, authenticate its identity, and issue tickets to the service.

The domain served by a single KDC is referred to as a *realm*. A principal is used to identify each client and server in a realm. The principal name is uniquely assigned for all clients and servers by the Kerberos administrator. All principals must be known to the KDC.

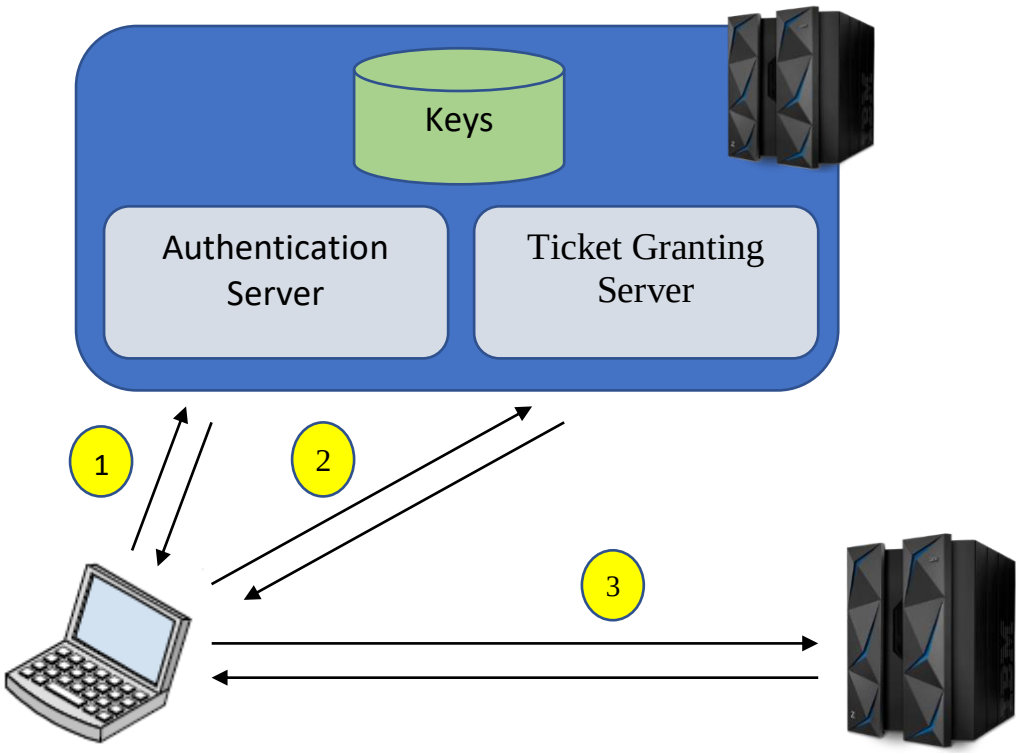


Figure 1 Kerberos message flow

Although the Kerberos protocol consists of several sub-protocols, three exchanges are particularly interesting here. The Kerberos protocol flow, as shown in the figure above, is as follows:

© Copyright IBM Corporation 2018

a. The first phase exchange takes place between a client and the authentication server. In this phase, the authentication server authenticates the user (for example, by validating the user ID and password). After a successful login, the authentication server obtains the user's secret keys and returns a ticket-granting ticket (TGT) to the client.
b. On receiving the TGT, the client sends a request (containing the TGT) for a service ticket to the ticket-granting server (TGS). The TGS authenticates the TGT and then returns a service ticket to the client.
c. Having the service ticket ready, the client is allowed to communicate with the server that is providing a service it wants to use. Optionally, the application server can perform further authentication processing against the client. The application server can verify the service ticket without contacting the KDC.

In the scenario described here, we have two realms, each served by their own KDC: one on a Microsoft Windows server and one on z/OS. These two realms will be configured to trust each other. The name of the realm in which a client is registered is part of the client's name and can be used by the application server to decide whether to honor a request.

By establishing inter-realm keys, the administrators of two realms can allow a client authenticated in one realm to use its credentials in the other realm. The exchange of inter-realm keys registers the ticket-granting service of each realm as a principal in the other realm. A client is then able to obtain a ticket-granting ticket for the remote realm's ticket-granting service from its local ticket-granting service. Tickets issued to a service in the remote realm indicate that the client was authenticated from another realm.

For a complete description of the Kerberos Version 5 protocol, refer to RFC 4120 – The Kerberos Network Authentication Service (V5): https://tools.ietf.org/html/rfc4120.

Before we start

Configuring Kerberos can be straightforward. Troubleshooting configuration problems, however, can be difficult and time-consuming. For this reason, it is very important to plan ahead and implement the configuration carefully.

In this document, we use a simple Java client program to test the Kerberos configuration. Both the source code and a compiled jar file can be found in the CICS GitHub repository here: https://github.com/cicsdev/cics-kerberos-sample.

When making the changes on Windows, it is useful to launch applications as Administrator. To do this, right-click on the application in the **Start** menu and selecting **More -> Run as Administrator**. Although it is not always necessary to run as Administrator, running with the extra authority can avoid some annoying issues that are caused by having insufficient authorities.

© Copyright IBM Corporation 2018

Throughout this configuration, we've specified the use of only AES128 & AES256 as encryption standards and avoided support for weaker standards such as DES, DESD & DES3. The aim was to balance reasonable security with interoperability. This is fine for a test system, but care should be taken in choosing which encryption algorithms should be allowed in a production system.

As there are multiple resource names used in the configuration that must be consistent between systems, we have listed here the values used in our set-up. The values are discussed in more detail in this document as they are used.

Table 1.User IDs and Kerberos principals

Name	Value	Comments
CICS region Kerberos user ID	JT1B	RACF user ID for Kerberos processing in the CICS region
CICS region Kerberos Service principal	jt1b_service	Kerberos identifier for the CICS region Kerberos user ID
RACF client user ID	JT1C	RACF user ID used for client requests (z/OS only)
RACF client principal	jt1c_client	Kerberos identifier for the RACF client user ID
Microsoft Active Directory client user ID	cics4	Windows user ID for client requests (MAD/z/OS cross-realm)
Microsoft Active Directory client principal	cics4@cicsfvs4.ad.hursley.ibm.com	Kerberos identifier for the MAD client user ID
RACF user ID for Microsoft Active Directory client principal	JT1D	RACF user ID mapped to the MAD client principal

It is also useful to describe the relationships between these user IDs and Kerberos principals, so those relationships are shown in Table 2. Relationships between user IDs and Kerberos principals. This table does not introduce any new user IDs. It shows the mapping between the user IDs specified in Table 1.User IDs & Kerberos principals.

Table 2. Relationships between user IDs and Kerberos principals

RACF user ID	MAD user ID	Kerberos Principal	Comments
JT1B	-	jt1b_service	CICS region Kerberos id
JT1C	-	jt1c_client	z/OS-only client principal

© Copyright IBM Corporation 2018

JT1D	cics4	cics4@cicsfvs4.ad.hursley.ibm.com	MAD client principal

Table 3. Host & Realm names

Name	Value
z/OS KDC	winmvs2c.hursley.ibm.com
z/OS Realm	WINMVS2C.HURSLEY.IBM.COM
MAD KDC	cicsmad4.cicsfvs4.ad.hursley.ibm.com
MAD Realm	CICSFVS4.AD.HURSLEY.IBM.COM

© Copyright IBM Corporation 2018

Configuring Kerberos support on z/OS

We will implement our KDC on z/OS using the SAF registry with RACF. The KDC must be running on the same LPAR as the CICS region(s) that will be interacting with it.

Appropriate authorities are required to issue the commands detailed below. Some of the commands may require minor modifications to reflect your installation and conventions.

Required RACF definitions for the KDC

There are several RACF definitions that must be defined for the KDC.

1. Create the SKRBKDC user that will own the KDC STARTED task, using the following command. (Substitute the appropriate DFLTGRP for your systems):

```
ADDUSER SKRBKDC DFLTGRP(SYS1) NOPASSWORD OMVS(UID(0)
PROGRAM('/bin/sh') HOME('/etc/skrb/home/kdc'))
```

Note: if you are using z/OS V2R2 or later, you should use a unique non-zero UID instead, and ensure this ID is given access to the necessary directories and files. To do this, replace UID(0) with AUTOUID and it will give you a new one.

Apply access controls to the ZFS directories and files used by Kerberos as follows:

- Ensure that the SKRBKDC user has write authority to /etc/skrb/home and /var/skrb, and their subdirectories.
- The administrator user should own and have write access to /etc/skrb/krb5.conf. Everyone else should have read access.

If there are any keytab files in the /etc/skrb directory, these should be owned by the server user that uses that keytab file. No other users should have access.

2. Activate the APPL class:

```
SETROPTS CLASSACT(APPL) RACLIST(APPL)
```

3. Define the SKRBKDC application in the APPL class:

```
RDEFINE APPL SKRBKDC UACC(READ)
```

4. Refresh the APPL class:

```
SETROPTS RACLIST(APPL) REFRESH
```

© Copyright IBM Corporation 2018

5. Activate the PTKTDATA class:

```
SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
```

6. Define a key to mask values in RACF for the SKRBKDC application:

```
RDEFINE PTKTDATA SKRBKDC UACC(NONE)
SSIGNON(KEYMASKED(3734343237343131))
```

Choose your own value for KEYMASKED here. Alternatively, you could use KEYENCRYPTED instead of KEYMASKED.

7. Refresh the PTKTDATA class:

```
SETROPTS RACLIST(PTKTDATA) REFRESH
```

8. Define the IRR.RUSERMAP profile in the FACILITY class with READ access for SKRBKDC and the CICS user ID that is used for Kerberos authentication, and refresh the FACILITY class:

```
RDEFINE FACILITY IRR.RUSERMAP UACC(NONE)
PERMIT IRR.RUSERMAP CLASS(FACILITY) ID(SKRBKDC) ACCESS(READ)
PERMIT IRR.RUSERMAP CLASS(FACILITY) ID(JT1B) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

9. Define the STARTED tasks for SKRBKDC and refresh the STARTED class:

```
RDEFINE STARTED SKRBKDC.** STDATA(USER(SKRBKDC))
RDEFINE STARTED SKRBWTR.** STDATA(USER(SKRBKDC))
SETROPTS RACLIST(STARTED) REFRESH
```

10. Define the KERBDFLT RACF REALM for the KDC. The REALM must be KERBDFLT, but you must customize the value for KERBNAME to your system's domain. Set the password and values for ticket life (in seconds).

```
RDEFINE REALM KERBDFLT KERB(KERBNAME(WINMVS2C.HURSLEY.IBM.COM)
PASSWORD(long-non-obvious-password) MINTKTLFE(15) DEFTKTLFE(36000)
MAXTKTLFE(86400))
```

Note that this password can be up to 128 characters long, and due to the importance of the keys generated from this password, it should be a strong (long) password.

11. Refresh the REALM class:

```
SETROPTS RACLIST(REALM) REFRESH
```

Set up the KDC on z/OS

© Copyright IBM Corporation 2018

1. Copy the sample proc from hlq.SEUVFSAM(SKRBKDC) (where hlq is typically EUVF) to the system's PROCLIB.
2. Copy the sample envar file to its default location and change its owner to be skrbkdc:

```
cp /usr/lpp/skrb/examples/skrbkdc.envar /etc/skrb/home/kdc/envar
chown skrbkdc /etc/skrb/home/kdc/envar
chmod 600 /etc/skrb/home/kdc/envar
```

3. Copy the krb5.conf file to its default location and make the file r/w by the admin id, read only by group and other users:

```
cp /usr/lpp/skrb/examples/krb5.conf /etc/skrb/krb5.conf
chmod 644 /etc/skrb/krb5.conf
```

4. Modify the krb5.conf file to specify the Kerberos realm name and the KDC host name. The realm in the krb5.conf file must be the same as the realm that you specified in the RDEFINE REALM command earlier in this procedure. In our example, the realm name and KDC host name are (unusually) both WINMVS2C.HURSLEY.IBM.COM, resulting in this krb5.conf file:

```
;-------------------------------------------------------------------;
;  Simple Kerberos configuration file for WINMVS2C                  ;
;-------------------------------------------------------------------;

[libdefaults]
default_realm = WINMVS2C.HURSLEY.IBM.COM
kdc_default_options = 0x40000010
use_dns_lookup = 1
use_ldap_lookup = 0

; AES128 & AES256 encryption types
default_tkt_enctypes = aes256-cts-hmac-sha1-96,aes128-cts-hmac-sha1-96
default_tgs_enctypes = aes256-cts-hmac-sha1-96,aes128-cts-hmac-sha1-96

[realms]
WINMVS2C.HURSLEY.IBM.COM = {
   kdc = winmvs2c.hursley.ibm.com:88
   kdc = winmvs26.hursley.ibm.com:88
   kpasswd_server = winmvs2c.hursley.ibm.com:464
   kpasswd_server = winmvs26.hursley.ibm.com:464
}
[domain_realm]
.hursley.ibm.com = WINMVS2C.HURSLEY.IBM.COM
winmvs2c.hursley.ibm.com = WINMVS2C.HURSLEY.IBM.COM
```

5. Usually there is no need to modify the contents of the envar file at this time.
6. We can now start the z/OS KDC with a simple START SKRBKDC command. The KDC can be stopped with the command STOP SKRBKDC. Starting the KDC should result in output such as the following in the system log:

```
START SKRBKDC
£HASP100 SKRBKDC ON STCINRDR
IEF695I START SKRBKDC  WITH JOBNAME SKRBKDC  IS ASSIGNED TO USER
```

© Copyright IBM Corporation 2018

```
SKRBKDC , GROUP SYS1
£HASP373 SKRBKDC  STARTED
EUVF04022I Security server start command processed.
```

© Copyright IBM Corporation 2018

Configuring Kerberos support in CICS

Kerberos support in CICS requires a RACF user ID to be associated with a Kerberos service principal. This user ID must have a password associated with it.

In production systems, you should use the CICS SIT parameter called KERBEROSUSER to specify the user id to be used for Kerberos processing. This user ID must not be a protected user ID. Protected user IDs should not be used for Kerberos authentication and Kerberos authentication failures can result in user revocation.

In test systems, use of the KERBEROSUSER SIT parameter is preferred, but if this SIT parameter is not specified, the user ID used for Kerberos processing will be the user ID of the CICS region as seen in message ICH70001I.

Map Kerberos principal names to RACF users

RACF users cannot use Kerberos by default, even though RACF is the user registry for the KDC. The users need to map a Kerberos principal name to a RACF ID so that those IDs can be used for Kerberos.

We need to create a Kerberos Service Principal (user ID) and associate it with the Kerberos user ID for our CICS region (as discussed in Configuring Kerberos support in CICS). To associate the CICS region's Kerberos RACF ID (in this case 'JT1B') with a Kerberos segment, use the following command:

```
ALTUSER JT1B PASSWORD(K1E2R3B!) NOEXPIRED KERB(KERBNAME(jt1b_service)
ENCRYPT(NODES NODES3 NODESD AES128 AES256))
```

Notes:

- The command maps the RACF user id to the Kerberos principal name jt1b_service. As you can see, the Kerberos principal name can include lowercase characters (and is case-sensitive) and can be longer than 8 characters.
- Remember that Kerberos passwords are case-sensitive.
- Unless mixed-case password support is enabled, the ALTUSER command folds the password entered on the command line to uppercase when the user presses the enter command.

In addition, for now we will associate a z/OS user ID with a Kerberos principal defined in our z/OS KDC to be used as a client principal. This allows us to test our z/OS Kerberos configuration without the added complexity of the cross-realm trust to the Microsoft Active Directory Kerberos server. When we know that we can generate Kerberos tickets using just the z/OS KDC, we can then move on to testing with the Microsoft Active Directory KDC with greater confidence.

We use an almost identical command as above to create another Kerberos principal and associate it with another RACF user ID:

© Copyright IBM Corporation 2018

```
ALTUSER JT1C PASSWORD(MTADMIN) NOEXPIRED KERB(KERBNAME(jt1c_client)
ENCRYPT(NODES NODES3 NODESD AES128 AES256))
```

Having issued these commands, we can confirm that the Kerberos principals have been created correctly with the following commands:

```
LISTUSER JT1B KERB NORACF
LISTUSER JT1C KERB NORACF
```

The commands result in the following output:

```
LISTUSER JT1B KERB NORACF
USER=JT1B

KERB INFORMATION
----------------
KERBNAME= jt1b_service
KEY FROM= PASSWORD
KEY VERSION= 001
KEY ENCRYPTION TYPE= NODES NODES3 NODESD AES128 AES256
READY

LISTUSER JT1C KERB NORACF
USER=JT1C

KERB INFORMATION
----------------
KERBNAME= jt1c_client
KEY FROM= PASSWORD
KEY VERSION= 001
KEY ENCRYPTION TYPE= NODES NODES3 NODESD AES128 AES256
READY
```

Validate the Kerberos configuration

At this point, we should have a valid Kerberos configuration running on our z/OS system. We can test this before adding the extra complexity of a second Kerberos realm.

The first very simple test is to check for message DFHXS1400 in the CICS log. The message should look something like:

```
DFHXS1400 JATP2350 Kerberos realm is WINMVS2C.HURSLEY.IBM.COM
```

This message shows the Kerberos realm that we have specified, and shows that we have correctly associated a service principal with the user ID specified on the CICS SIT parameter KERBEROSUSER, or if that is not specified, the user ID of the CICS region as seen in message ICH70001I. If you do not see this message, then check the configuration steps described above before attempting to proceed any further.

© Copyright IBM Corporation 2018

If you already have a suitable application, it should function correctly with this configuration. Alternatively, this section describes a simple test application that can be used to validate this configuration. The test application can also be used to validate the cross-realm scenario later.

One typical usage of Kerberos in CICS is with CICS web services. Here we use a test based on the 'hello world' Web service described in the CICS SupportPac CA1P: Web services samples for CICS TS. This SupportPac provides instructions to set up a simple 'hello world' web service in your CICS region. We will modify the first example from that SupportPac to use Kerberos with a simple Java client application that will request a suitable token from our Kerberos server and use that token to invoke the web service.

1. Download and work through SupportPac CA1P through to the end of Exercise 1. At this point, you have a simple 'hello world' web service running in your CICS region, and Java client with which to invoke it.
2. Copy the provided pipeline soap provider xml file (basicsoap11provider.xml) to create kerberossoap11provider.xml.
3. Add the following lines to the 'service' section of kerberossoap11provider.xml:

```
<service_handler_list>
  <wsse_handler>
    <dfhwsse_configuration version="1">
      <authentication trust="basic" mode="basic-kerberos"/>
    </dfhwsse_configuration>
  </wsse_handler>
</service_handler_list>
```

The updated kerberossoap11provider.xml file should now look like this:

```
<provider_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline">
 <service>
    <service_handler_list>
      <wsse_handler>
        <dfhwsse_configuration version="1">
          <authentication trust="basic" mode="basic-kerberos"/>
        </dfhwsse_configuration>
      </wsse_handler>
    </service_handler_list>
  <terminal_handler>
   <cics_soap_1.1_handler/>
  </terminal_handler>
 </service>
 <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

4. Update the pipeline definition (PROVPIPE) to reference kerberossoap11provider.xml as its configfile in place of basicsoap11provider.xml.
5. If the pipeline PROVPIPE was already installed:
   a. Disable the pipeline (CEMT SET PIPELINE(PROVPIPE) DISABLED)

© Copyright IBM Corporation 2018

b. Discard the pipeline (CEMT DISCARD PIPELINE(PROVPIPE))

6. Install the pipeline (CEDA INSTALL PIPELINE(PROVPIPE) GROUP(CA1PSAMP))
7. Ensure that the pipeline has installed correctly. CEMT INQUIRE PIPELINE(PROVPIPE) should show the pipeline as enabled.
8. Download and run the cicstest-kerberos client Java program:

1. Download the client from the CICS GitHub repository here: https://github.com/cicsdev/cics-kerberos-sample
2. Invoke the client program with a command such as:

```
java -cp cicstest.jar cics.Requester  <host:port> <echoString> <clientPrincipalName> <clientPassword>
<servicePrincipalName>
```

For the configuration we are using here, we use the following command:

```
java -cp cicstest.jar cics.Requester winmvs2c.hursley.ibm.com:12414 HelloWorld
WINMVS2C.HURSLEY.IBM.COM winmvs2c.hursley.ibm.com jt1c_client K1E2R3B! jt1b_service
```

The Java program should run and generate output such as the following:

```
Web service requester for CICS Echo service
-------------------------------------------
Parameters
Host:Port = winmvs2c.hursley.ibm.com:12414
EchoString = HelloWorld
realm =  WINMVS2C.HURSLEY.IBM.COM
kdc = winmvs2c.hursley.ibm.com
clientPrincipalName = jt1c_client
clientPassword = K1E2R3B!
servicePrincipalName = jt1b_service

Target URL is : http://winmvs2c.hursley.ibm.com:12414/ca1p/echoProgProviderBatch


[JGSS_DBG_CRED]  main JAAS config: debug=true
[JGSS_DBG_CRED]  main JAAS config: credsType=initiate only (default)
[JGSS_DBG_CRED]  main config: useDefaultCcache=false
[JGSS_DBG_CRED]  main config: useCcache=null
[JGSS_DBG_CRED]  main config: useDefaultKeytab=false (default)
[JGSS_DBG_CRED]  main config: useKeytab=null
[JGSS_DBG_CRED]  main JAAS config: forwardable=false (default)
[JGSS_DBG_CRED]  main JAAS config: renewable=false (default)
[JGSS_DBG_CRED]  main JAAS config: proxiable=false (default)
[JGSS_DBG_CRED]  main JAAS config: tryFirstPass=false (default)
[JGSS_DBG_CRED]  main JAAS config: useFirstPass=false (default)
[JGSS_DBG_CRED]  main JAAS config: moduleBanner=false (default)
[JGSS_DBG_CRED]  main JAAS config: interactive login? yes
[JGSS_DBG_CRED]  main JAAS config: refreshKrb5Config = true
[JGSS_DBG_CRED]  main Retrieving Kerberos creds from cache for principal=null
[JGSS_DBG_CRED]  main No Kerberos creds in cache for principal jt1_client
[JGSS_DBG_CRED]  main Doing Kerberos login for principal jt1_client@WINMVS2C.HURSLEY.IBM.COM
```

© Copyright IBM Corporation 2018

```
[JGSS_DBG_CRED]  main Doing Kerberos login for principal: jt1_client@WINMVS2C.HURSLEY.IBM.COM
[JGSS_DBG_CRED]  main Kerberos login complete
[JGSS_DBG_CRED]  main Login successful
[JGSS_DBG_CRED]  main kprincipal : jt1c_client@WINMVS2C.HURSLEY.IBM.COM
[JGSS_DBG_CRED]  main jt1c_client@WINMVS2C.HURSLEY.IBM.COM added to Subject
[JGSS_DBG_CRED]  main Kerberos ticket added to Subject
[JGSS_DBG_CRED]  main added key of type aes128-cts-hmac-sha1-96
[JGSS_DBG_CRED]  main added key of type des3-cbc-sha1
[JGSS_DBG_CRED]  main added key of type rc4-hmac
Request message
---------------

<?xml version="1.0"  standalone="no"?> <soapenv:Envelope
xmlns:q0="http://www.ECHOPROG.ECHOCOMM.Request.com"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP-ENV:Header xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"> <wsse:Security SOAP-ENV:mustUnderstand="0"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<wss:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-
profile-1.1#Kerberosv5_AP_REQ" Id="uuide7eafbbf-0143-1ffd-8da2-b17ce09c1b4d"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">YIICDwYJKoZIhvcSAQICAQBuggH+MIIB+qADAgEFoQMCAQ6iBwMFAAAAAACjggESYYIBDjCCAQqgAwIB
BaEaGxhXSU5NVlMyQy5IVVJTTEVZLklCTS5DT02iGTAXoAMCAQGhEDAOGwxqdDFiX3NlcnZpY2WjgcswgcigAwIB
AaEDAgFEooG7BIG4v62ahZT2+hoPhhMo5ZvlLqUNZZaRJ+EDF1D3xLEHTF/rdMWiEjLFOxWeIybD4kDTDAMtviCy
yEUsFXky5OM2KjmH6MRkV4/miFBfy/evM3q3+cmSLD0l1VACvf6EaWJ6B3EGoIW2q3zXOr6zf90kqSn3J+K6yOTj
Q8B2kTLd6ttkmP9ozKStRzO9B7qgVYt2muRoKbPOdZgcfGuHaF8i01dqfBgAuSPGv1hgVbLFmPbgmQ6nZ4Ued6S
BzjCBy6ADAgERooHDBIHA518/386TINNsNXY9SjBOtQ832nJjbXl2x/t8ffYQw2+rSjuDu0NWV8nsiZRxwbBYsa/OE4
HasFTjVohUsoYZTb+Eca0XjRP6qObRT9b0SbnVKAove+i9DhyaT5Y6tLEiTxnk5tSi5Kst9EIwFPPz1Eov3EWkAXQ2aa
pMtK7nujYWmjyHtlrN8JmdEq6SnSezYG0dEl6gqDViqYrtzDpDABfZf2kMhyBrVlrJjdovTPAoaZCFgaLBeTszWlO2Oq
5f</wss:BinarySecurityToken> </wsse:Security> </SOAP-ENV:Header> <soapenv:Body>
<q0:ECHOPROGOperation> <q0:echo_string> HelloWorld</q0:echo_string> </q0:ECHOPROGOperation>
</soapenv:Body> </soapenv:Envelope>

Attempting to read response
---------------------------

<SOAP-ENV:Envelope xmlns:q0="http://www.ECHOPROG.ECHOCOMM.Request.com"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Body><ECHOPROGOperationResponse
xmlns="http://www.ECHOPROG.ECHOCOMM.Response.com"><echo_string>CICS--&gt; HelloWorld
</echo_string></ECHOPROGOperationResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>

End of response
```

If you get similar output then you are ready to move on to the next section where we add a second Kerberos realm defined in Microsoft Active Directory.

If you did not see similar output, first follow the debugging advice covered in the CA1P SupportPac documentation to address any web service issues, then refer to the Troubleshooting section in this paper for advice about debugging Kerberos configurations.

© Copyright IBM Corporation 2018

Configuring Kerberos support in Microsoft Active Directory

The focus of this paper is on ensuring Kerberos interoperability between Microsoft Active Directory, RACF and CICS. If you already have Microsoft Active Directory configured to use Kerberos, then you can skip this section.

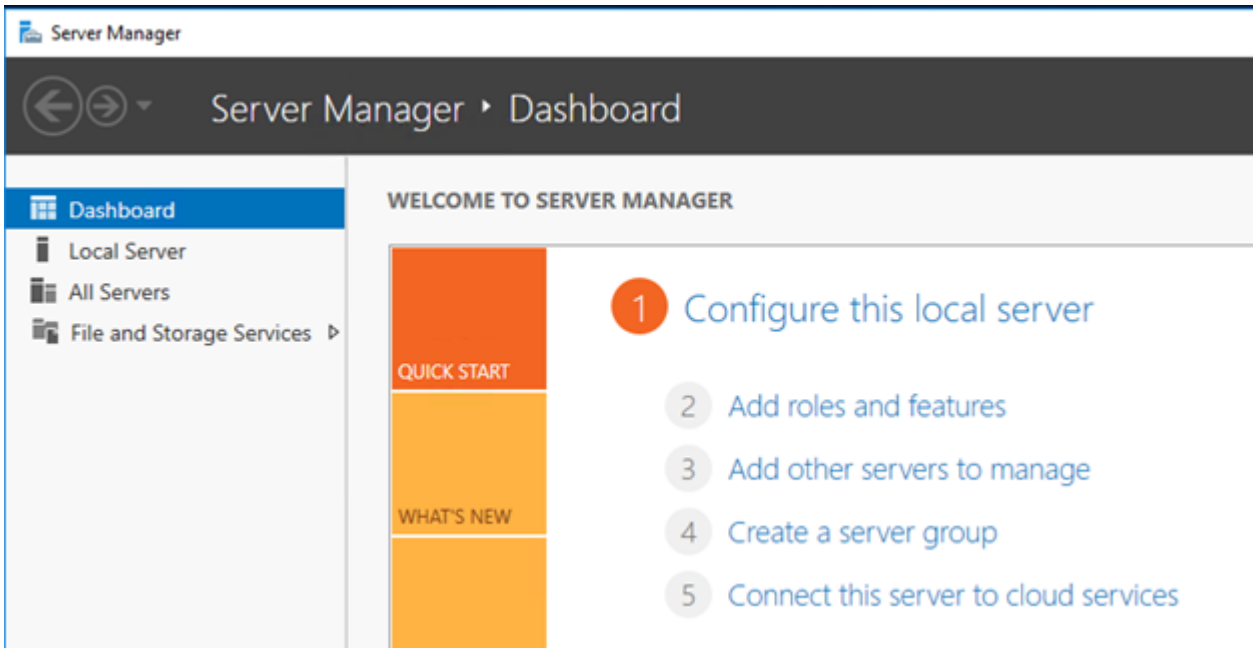
If you do not have Microsoft Active Directory configured to use Kerberos, then this section describes a process to set up Microsoft Active Directory to be used in a ***test*** environment. This is not a comprehensive description of how to configure Active Directory to be used in a production environment. Such a description is beyond the scope of this document.

We are using Microsoft Active Directory as one of our Kerberos servers. We used Windows Server 2016, but the advice in this paper should apply to other versions of Windows Server (with minor changes due to differences between versions).

First, we ensure that Microsoft Active Directory is installed on Microsoft Windows Server, then we configure Kerberos support.

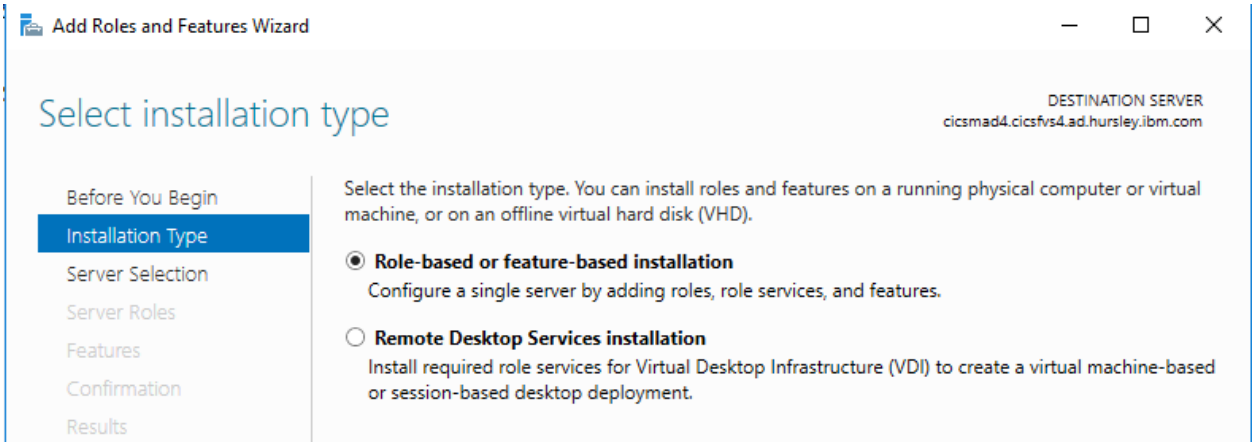
Install Microsoft Active Directory

1. From the Start menu, select Server manager.
2. On the top panel, select **Add roles and features**

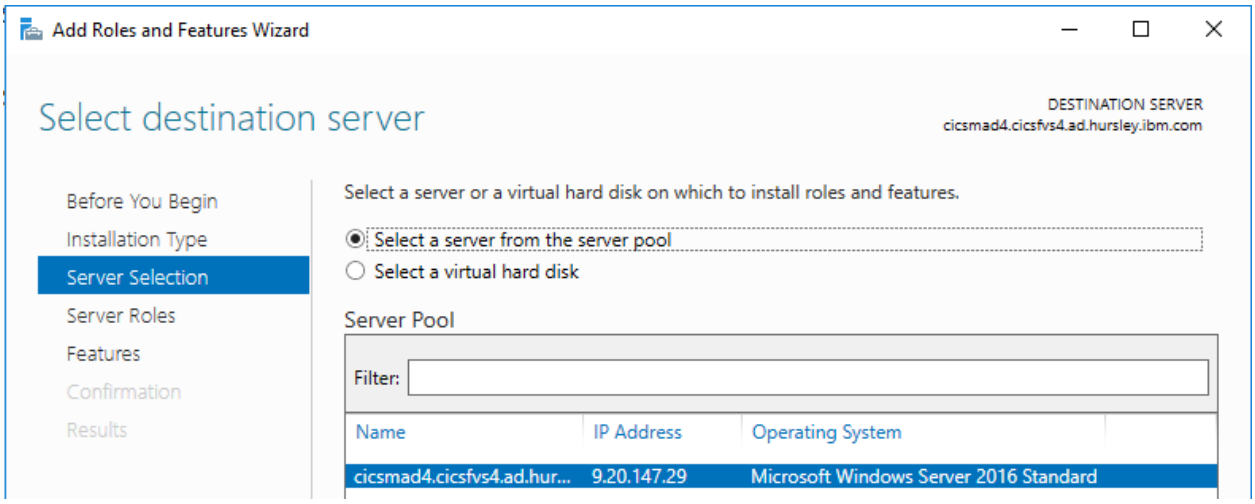


3. The Add Roles and Features Wizard launches. On the **Before you begin** page, click **Next**.
4. On the **Select installation type** page, select **Role-based or feature-based installation**, then click **Next**.

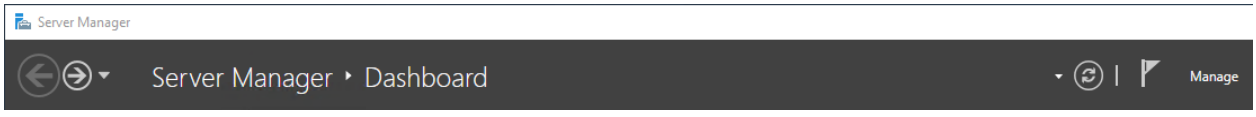
© Copyright IBM Corporation 2018



5. On the **Select destination server** page, select **Select a server from the server pool** and select this server. Click **Next**.



6. On the **Select server roles** page, if **Active Directory Domain Services** is not selected, select **Active Directory Domain Services**. A dialog pops up prompting you to add features. Click **Add features**.
7. Back on the **Select server roles** page, click **Next**.
8. On the **Select Features** page, just click **Next**.
9. Read (Active Directory Domain Services). Click **Next**.
10. Read (confirm installation selections). Click **Install**. The features install. This might take a few minutes.
11. When the install completes, close the dialog. Although you are not prompted to do so, it is worth rebooting the system at this point. Back on the **Server manager** window, click on the flag icon in the top right.



12. Click on Promote this server to a domain controller.

© Copyright IBM Corporation 2018

13. Select **Add a new forest**. We used a root domain name of 'cicsfvs4.ad.hursley.ibm.com'. Click **Next**.

14. For **Forest functional level** and **Domain functional level,** we left both as **Windows Server 2016**. We left the DNS server and global catalog selected. We entered a DSRM password. This password is not needed again in the process described here. Click **Next**.

15. Enter a suitable value for **NetBIOS name**, or accept the default. This name is not needed again in the process described here. Click **Next**.

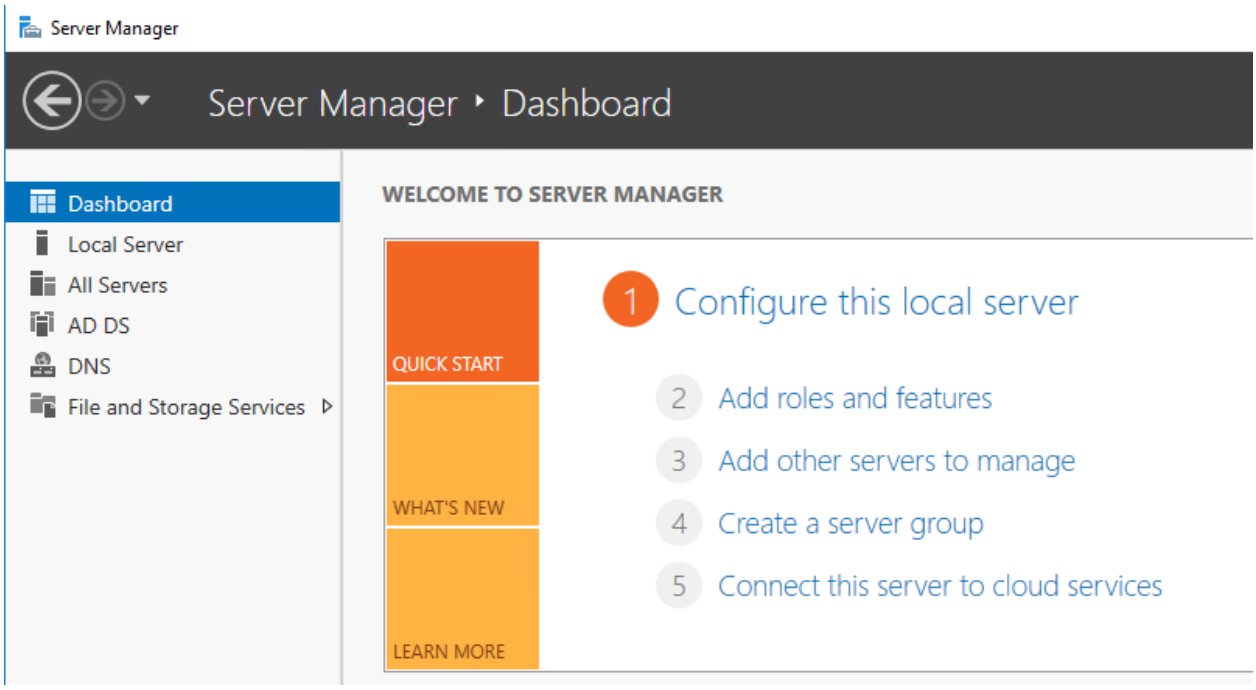
16. For the **database folder**, **log files folder** and **sysvol folder**, accept the default values: C:\Windows\NTDS, C:\Windows\NTDS, and C:\Windows\SYSVOL respectively. Click **Next**.

17. A summary of the selected configuration is displayed. Click **Next**.

18. A list of warnings is displayed, followed by the message "All prerequisite checks passed successfully. Click Install to begin installation." Click **Install**.

19. The install and configuration may take a few minutes, then a reboot dialog is displayed. Click **OK** to allow the reboot.

20. Log back in after the reboot. Server Manager launches. Note that the column on the left now contains entries 'AD DS' and 'DNS'.



You now have Active Directory installed and the Windows Server is configured as a domain name server.

At this point, it is a good idea to do a quick validation of our configuration so far. We can do this with a couple of simple commands in the windows PowerShell:

```
Get-ADDomain | fl Name,DomainMode
Get-ADForest | fl Name,ForestMode
```

© Copyright IBM Corporation 2018

Running these, we get the following output:

```
PS C:\Windows\system32> Get-ADForest | fl Name,ForestMode

Name        : cicsfvs4.ad.hursley.ibm.com
ForestMode  : Windows2016Forest


PS C:\Windows\system32> Get-ADDomain | fl Name,DomainMode

Name        : cicsfvs4
DomainMode  : Windows2016Domain
```

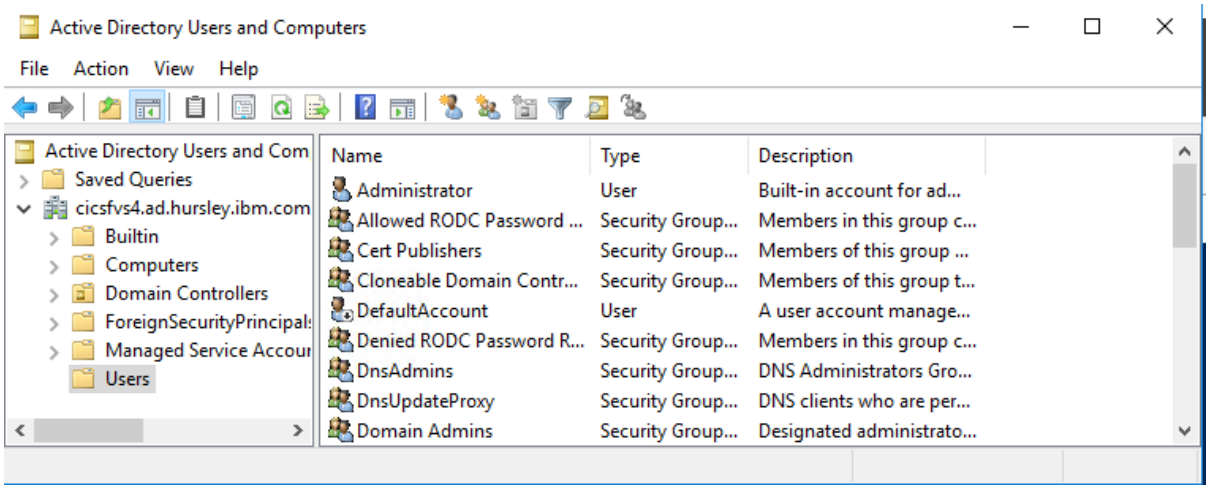
These results are what we expect, so we can move on to the next step.

Add a Kerberos principal name

We need at least one Kerberos principal name defined in our Microsoft KDC. This principal will be used as a client identity when we test the integration with the RACF KDC and CICS.

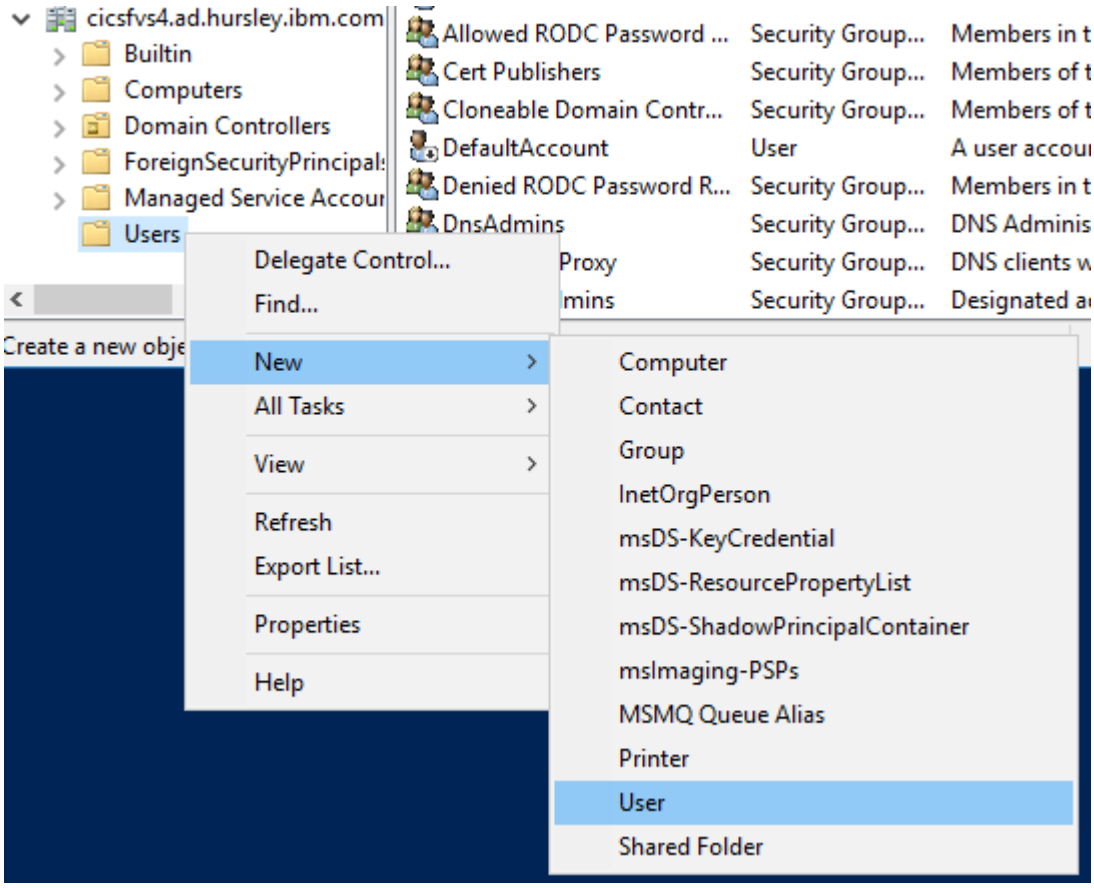
We can create a Kerberos principal by simply defining a new user in Active Directory.

1. Launch **Active Directory Users and Computers**. In the tree on the left, expand the node for this computer and select **Users**:

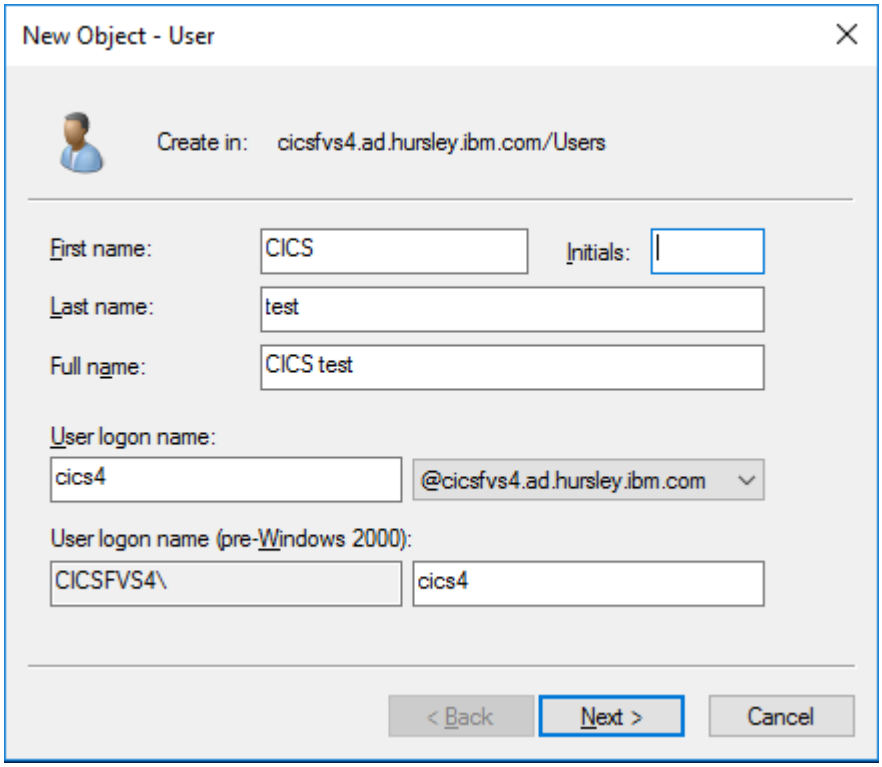


2. Right-click on **Users** and select **New -> User**:

© Copyright IBM Corporation 2018

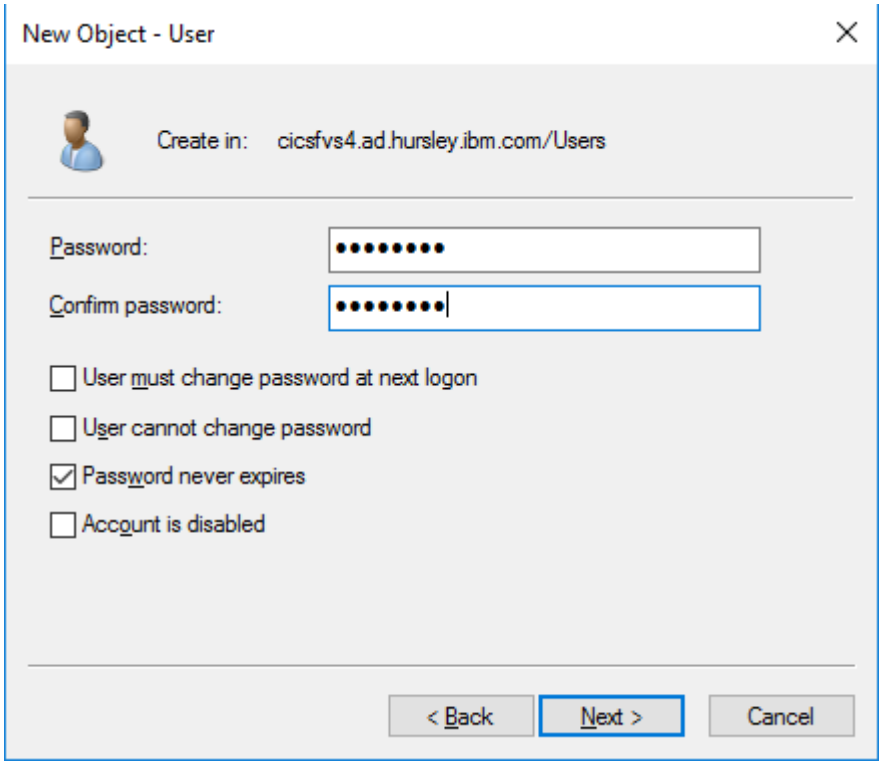


3. Enter some suitable values in the **New Object – User** wizard and click **Next**. Note that we specified this user ID at the start of this document as part of our planning.



© Copyright IBM Corporation 2018

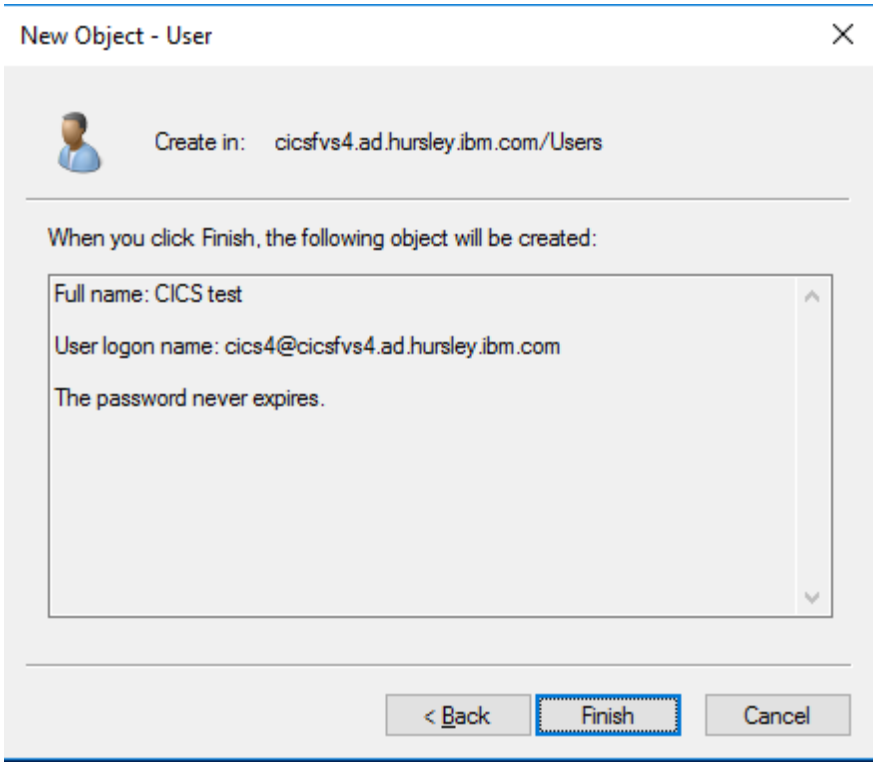
4. Enter a suitable password on the next page and keep a note of it. This password will be needed later. As this is a test environment, we can keep things simple for ourselves by setting the password to never expire. This is not recommended in a production system.



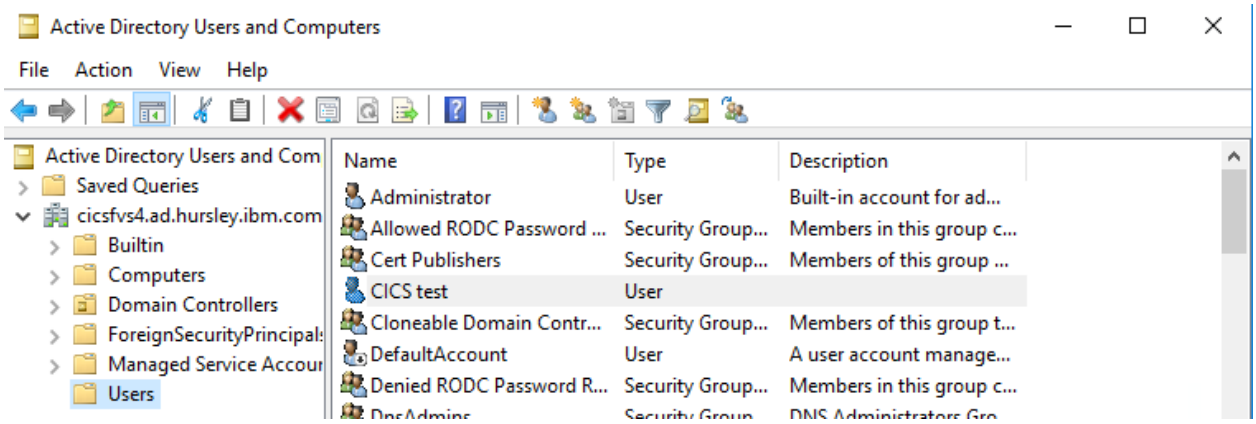
Windows will typically disallow some obvious passwords. For this example, we used the password ‘k1e2r3b!’. This will also be required later. Click **Next**.

5. Finally, the wizard displays a summary of the user ID that we are about to create. Click **Finish**.

© Copyright IBM Corporation 2018



6. We can now see our new user ID listed in the **Active Directory Users and Computers** window



7. We can also confirm that the Kerberos principal has been created by issuing a simple command:

```
setspn cics4
```

This command gives the following response:

```
PS C:\Windows\system32> setspn cics4
Registered ServicePrincipalNames for CN=CICS test,CN=Users,DC=cicsfvs4,DC=ad,DC=hursley,DC=ibm,DC=com:
```

This principal is now ready to be used.

© Copyright IBM Corporation 2018

Configuring trust between Microsoft Active Directory and RACF

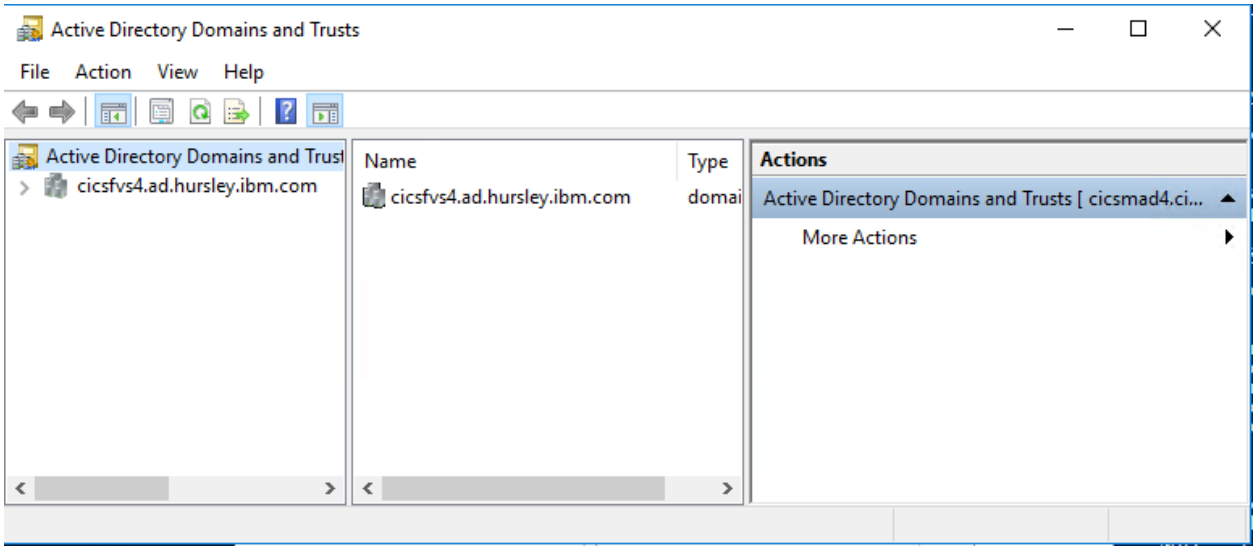
Configuring trust between the realms requires corresponding trust definitions to be created in both realms. The same password must be specified on z/OS and Microsoft Active Directory. Remember that z/OS folds passwords to upper case by default, but passwords on Windows are case-sensitive. Unless you have the SETROPTS PASSWORD(MIXEDCASE) option specified for your z/OS system, an upper-case only password must be used on both systems.

Microsoft Active Directory changes

On the Windows server, we need to create a cross-realm trust with our z/OS realm, and make some DNS changes.

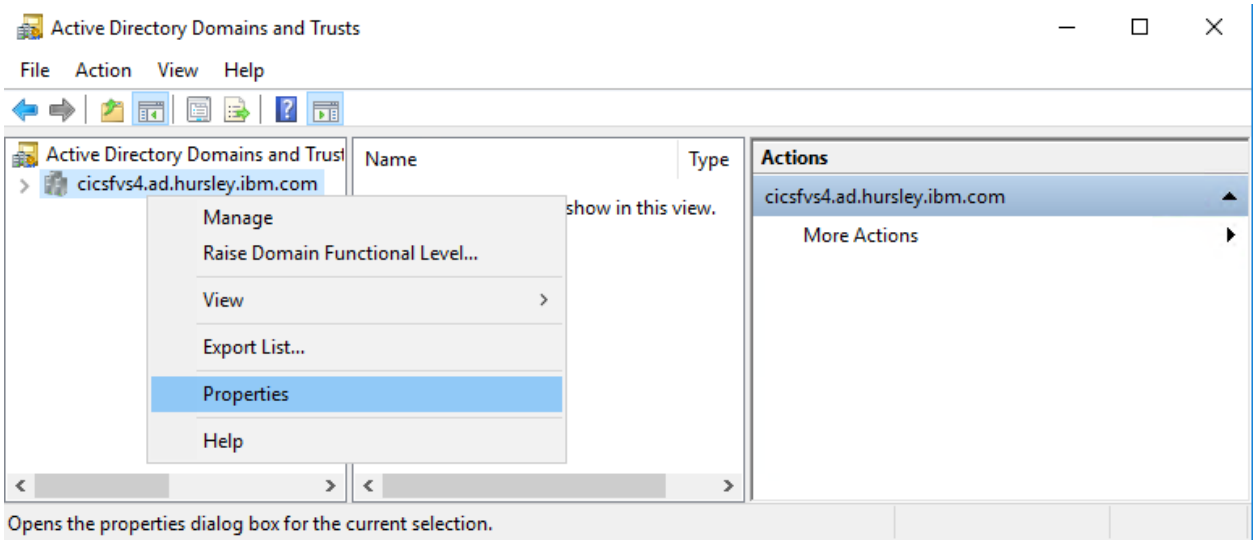
Configure cross-realm trust on Microsoft Active Directory

1. Open Active Directory Domains and Trusts by clicking **Start -> Programs -> Windows Administrative Tools -> Active Directory Domains and Trusts** (Note: don't forget to run this as Administrator).

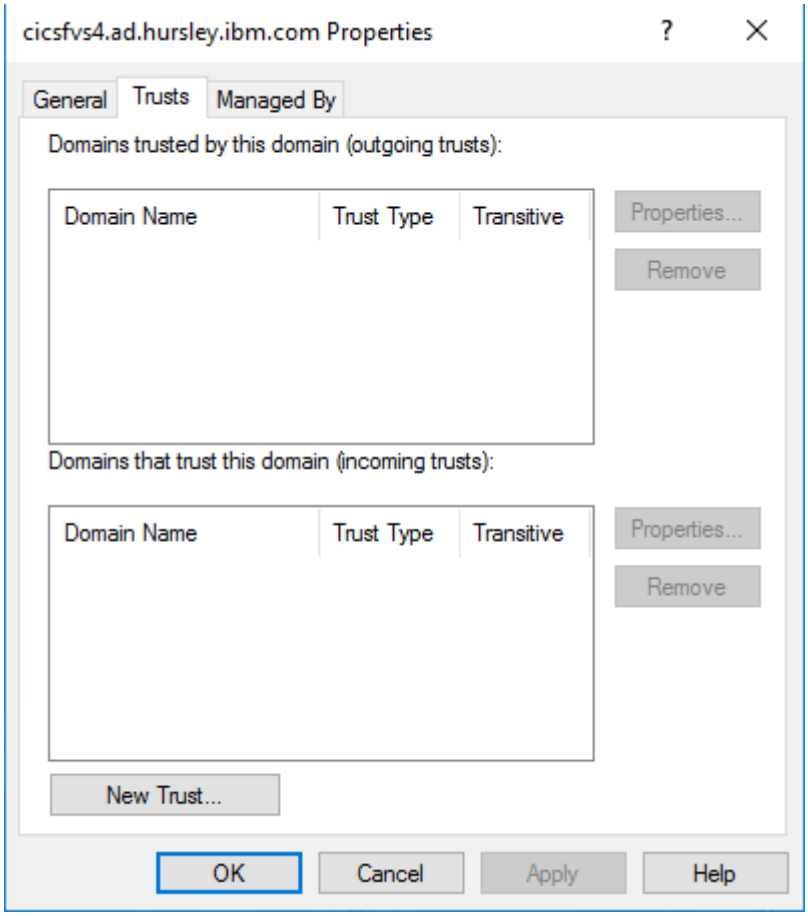


2. Right click on the domain and select **Properties**.

© Copyright IBM Corporation 2018

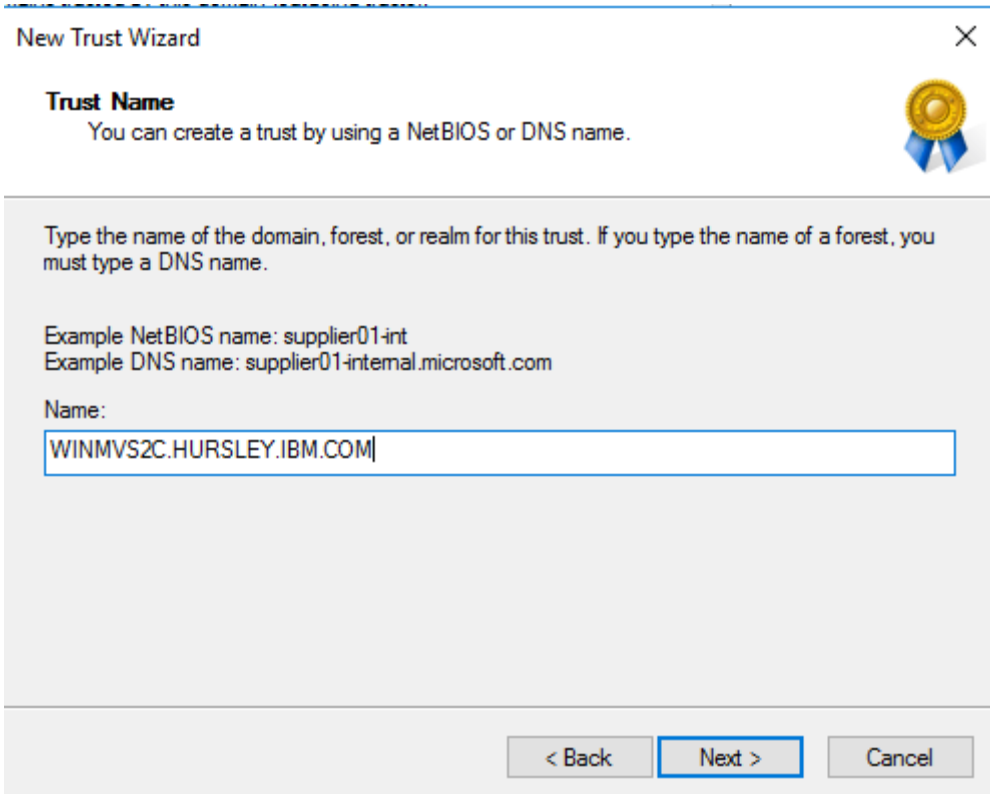


3. In the properties dialog, switch to the **Trusts** tab and click on **New Trust** to add a realm to trust.

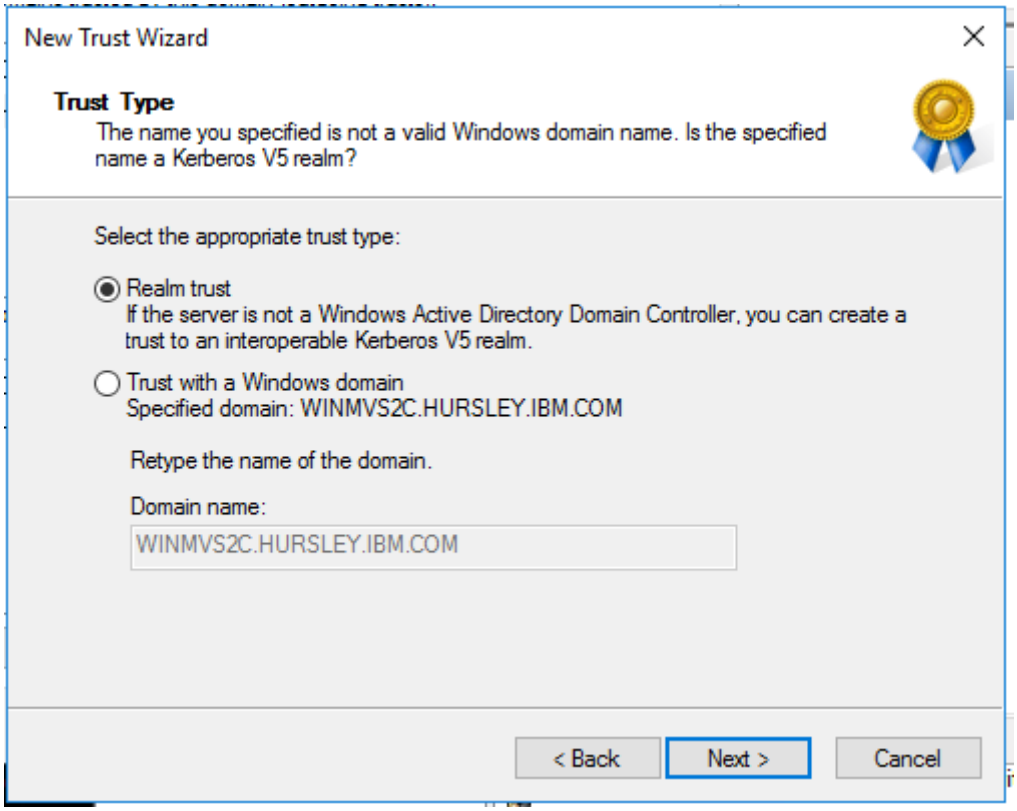


4. The **New Trust** wizard launches. Click **Next** on the wizard first page. Enter the z/OS realm name as the trust realm name, and click **Next**.

© Copyright IBM Corporation 2018

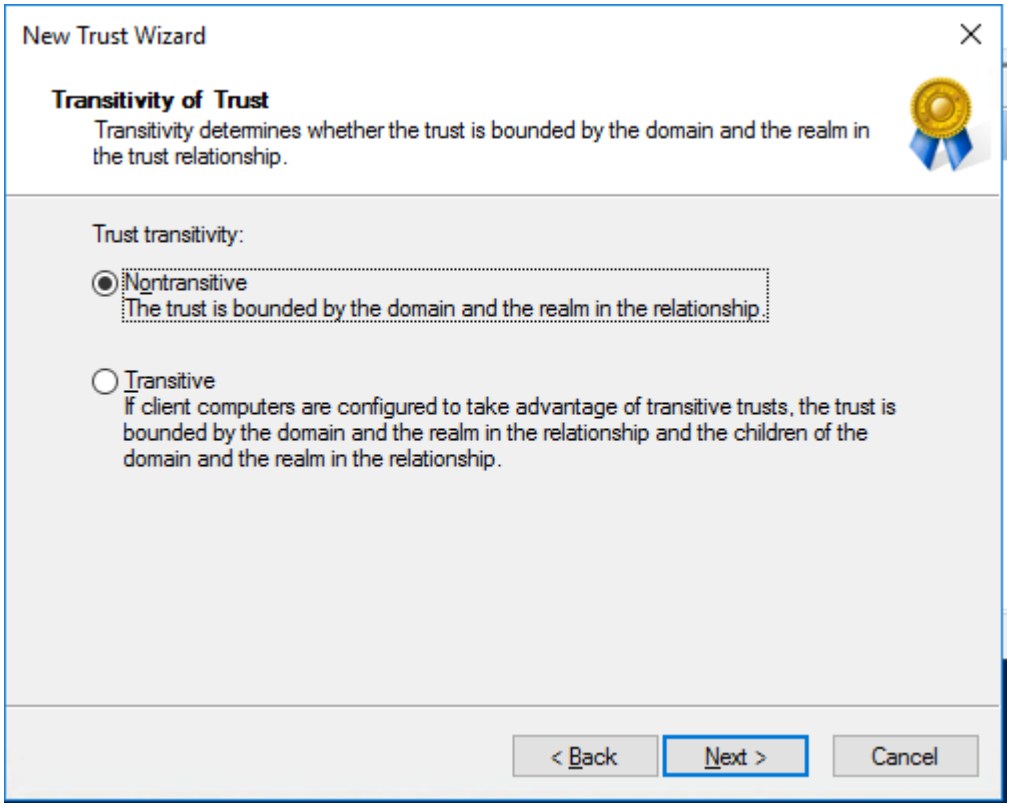


5. For **Trust Type**, select **Realm trust** (should already be selected), and click **Next**:

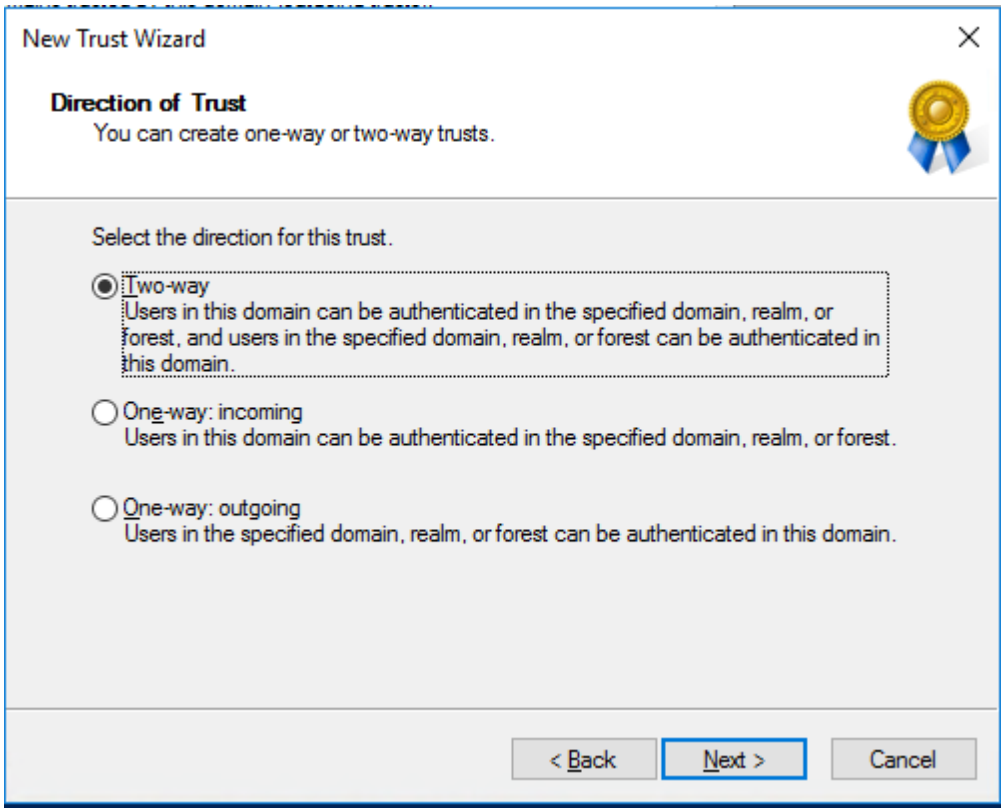


6. Select **Nontransitive** trust and click **Next**.

© Copyright IBM Corporation 2018

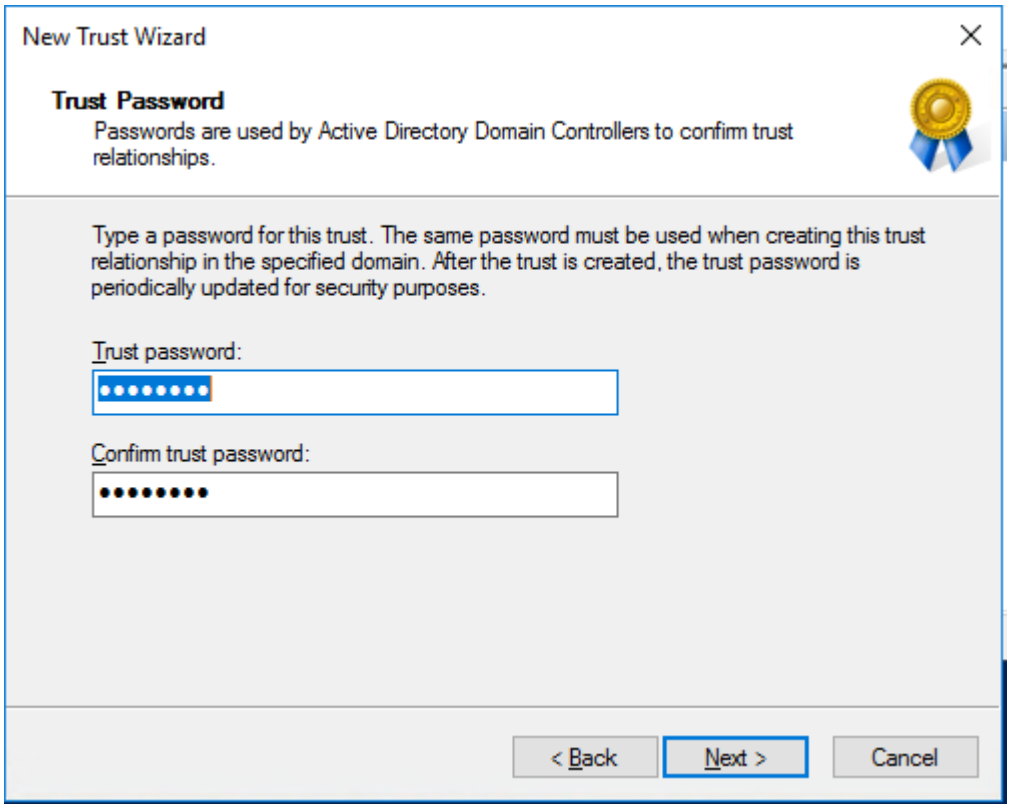


7. Select **Two-way** trust and click **Next**.



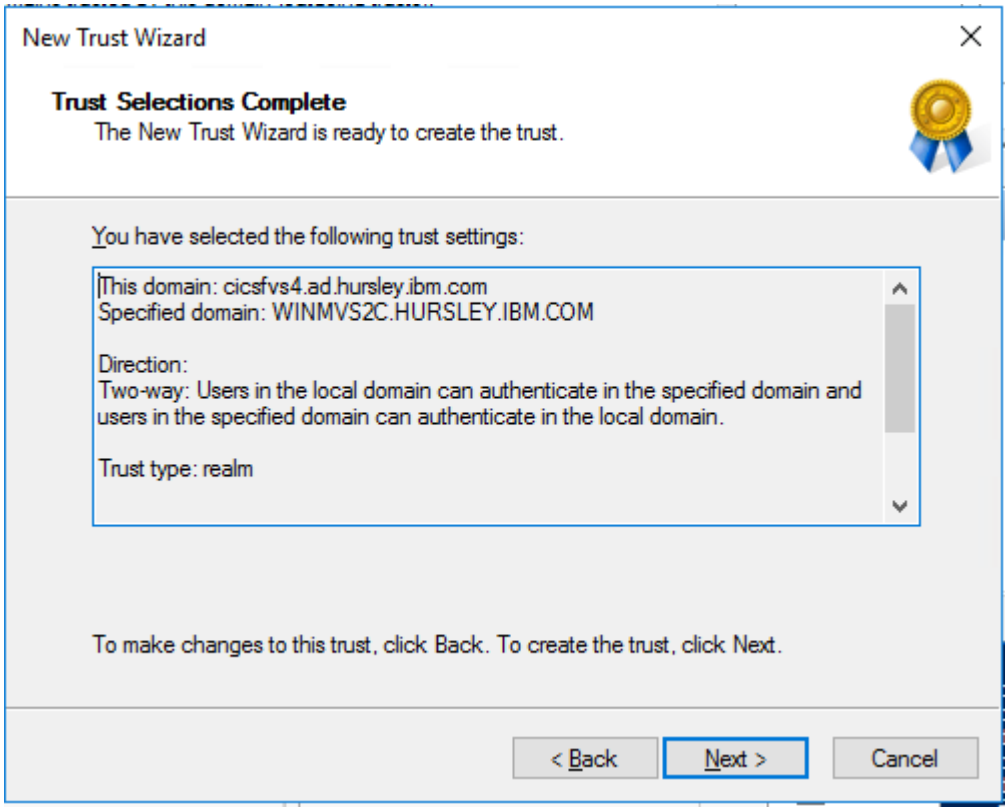
© Copyright IBM Corporation 2018

8. The trust password must match the password in the corresponding RACF RDEFINE commands. If you do not have SETROPTS PASSWORD(MIXEDCASE) set on z/OS to allow lowercase characters then this password must not contain lowercase characters. To keep things simple here, we use an uppercase only password. We used K1E2R3B! here. Click **Next**.

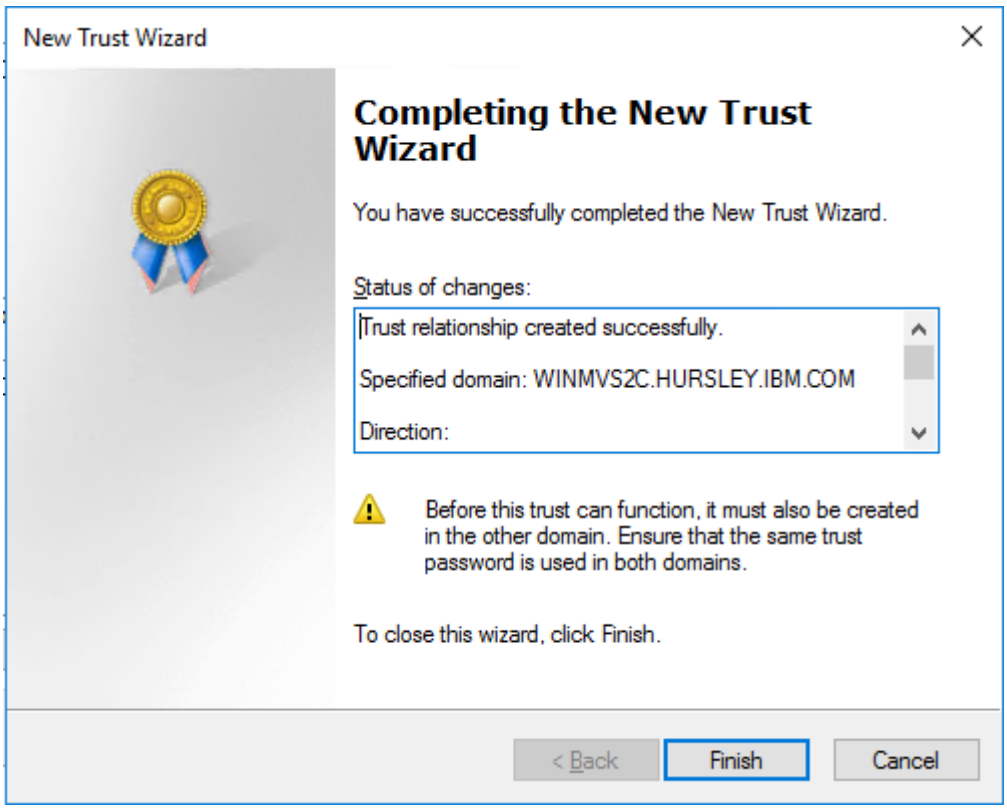


9. Review the trust selection summary and click **Next**.

© Copyright IBM Corporation 2018

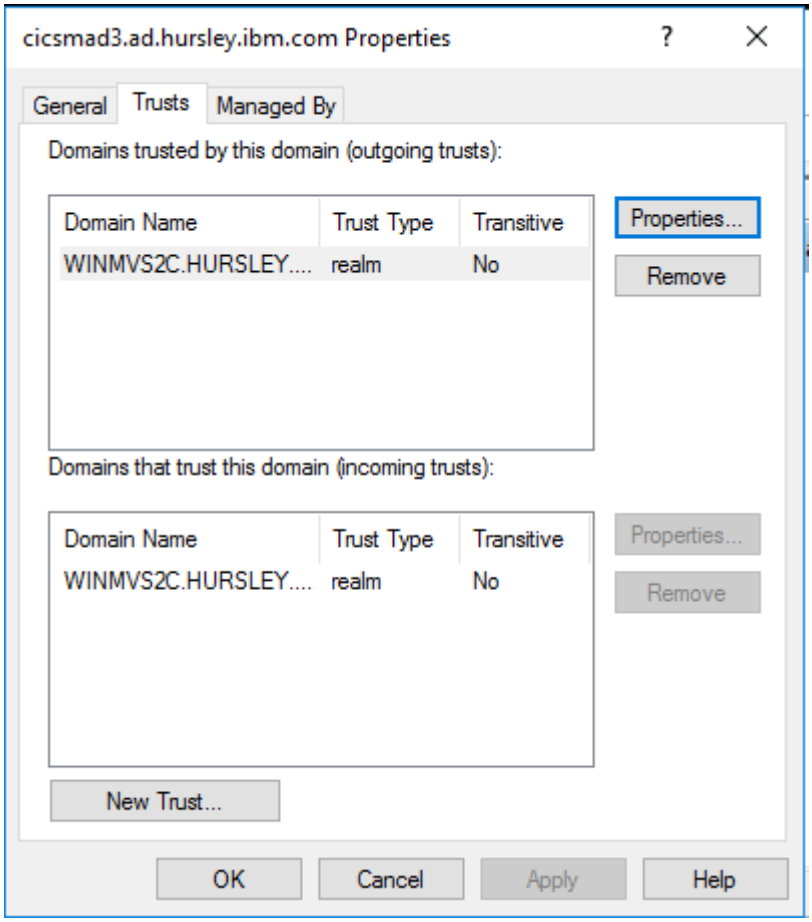


10. On the final page, click **Finish** to close the New Trust Wizard and return to the properties dialog.



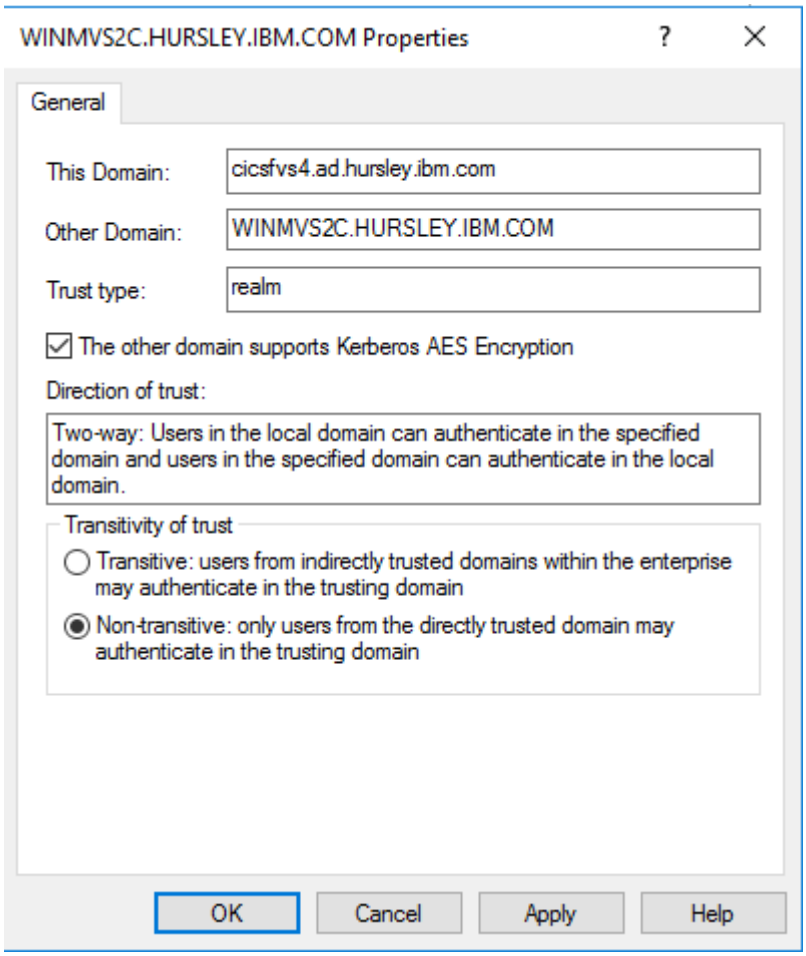
11. Back in the **Trusts** tab, we see our z/OS realm listed for both incoming and outgoing trust. Click **OK**.

© Copyright IBM Corporation 2018



12. Now click on on one of the trusts you have just defined, and click on **Properties**. The Properties dialog includes a check box labelled **The other domain supports Kerberos AES Encryption**. Select this check box and click **OK**.

© Copyright IBM Corporation 2018

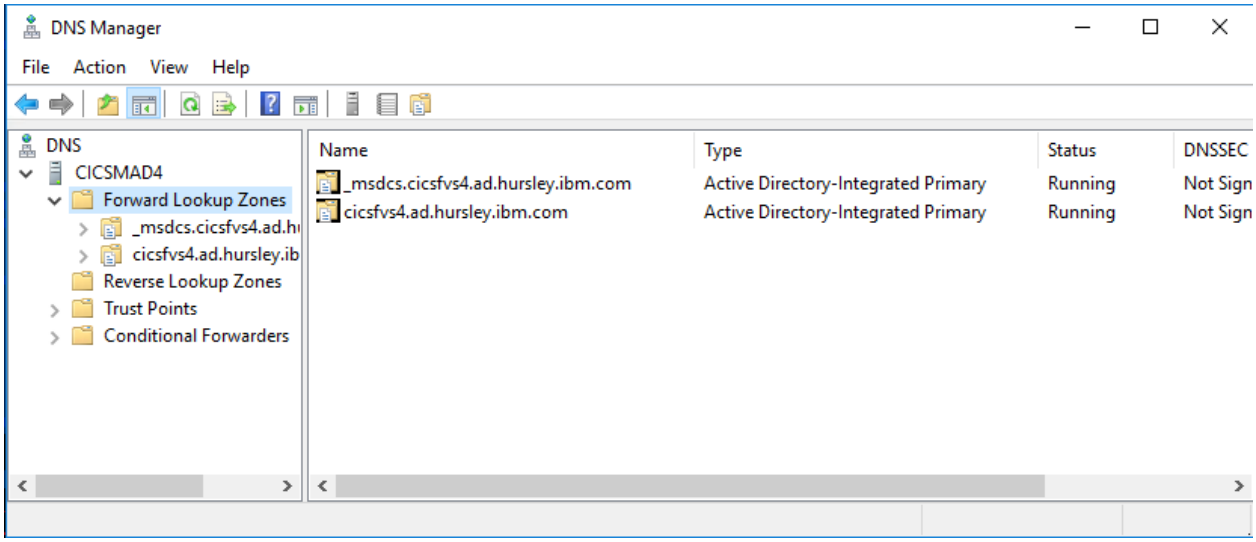


You now have the trust set up on Microsoft Active Directory, but the trust will not work until the corresponding changes have been applied to RACF.

DNS updates for the cross-realm trust support

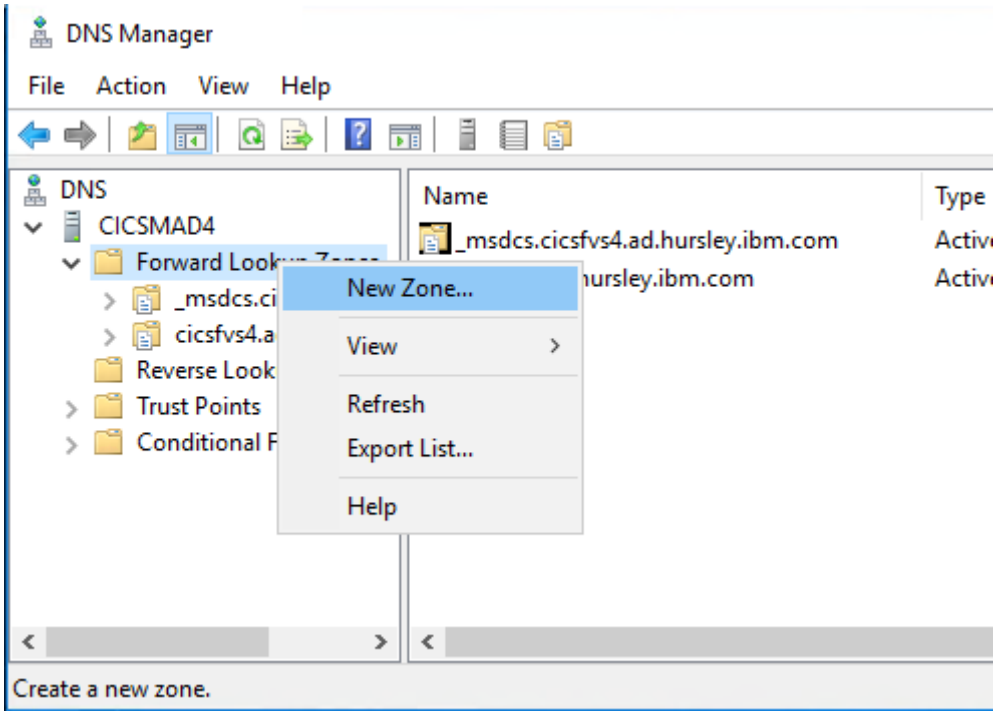
1. For DNS on the Windows system to resolve the z/OS KDC host, you need to configure the forward lookup zone and the reverse lookup zone. Updates to the DNS are done through the DNS manager. To open the DNS manager, click **Start -> Programs -> Windows Administrative Tools -> DNS**.

© Copyright IBM Corporation 2018



2. Configure the forward lookup zone:

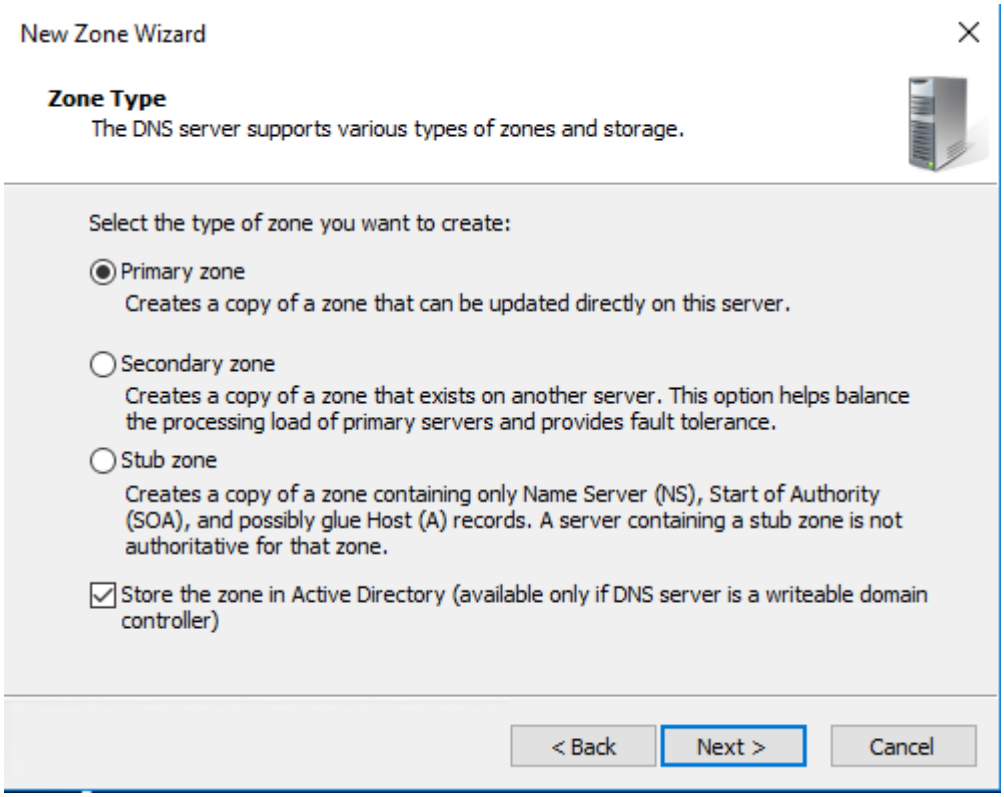
   a. In the DNS manager window, right-click **Forward Lookup Zone** and click **New Zone**.



   b. Click **Next** on the first page of the **New Zone Wizard**.

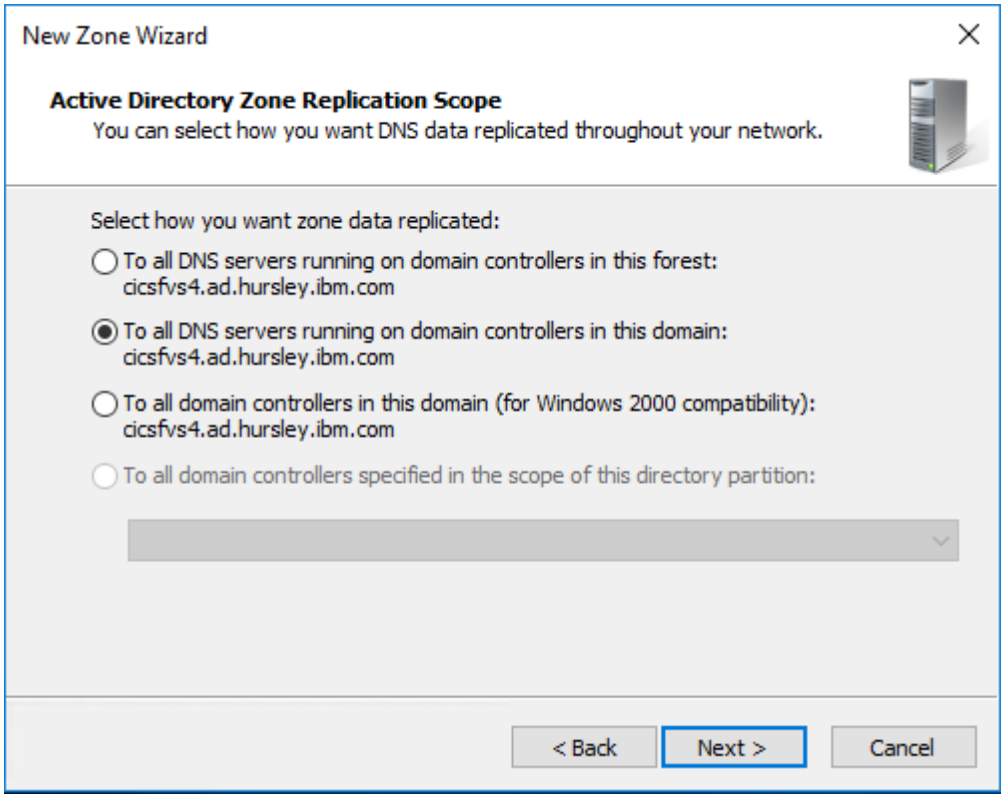
© Copyright IBM Corporation 2018

c. Select **Primary Zone** and **Store the zone in Active Directory**, and then click **Next**.

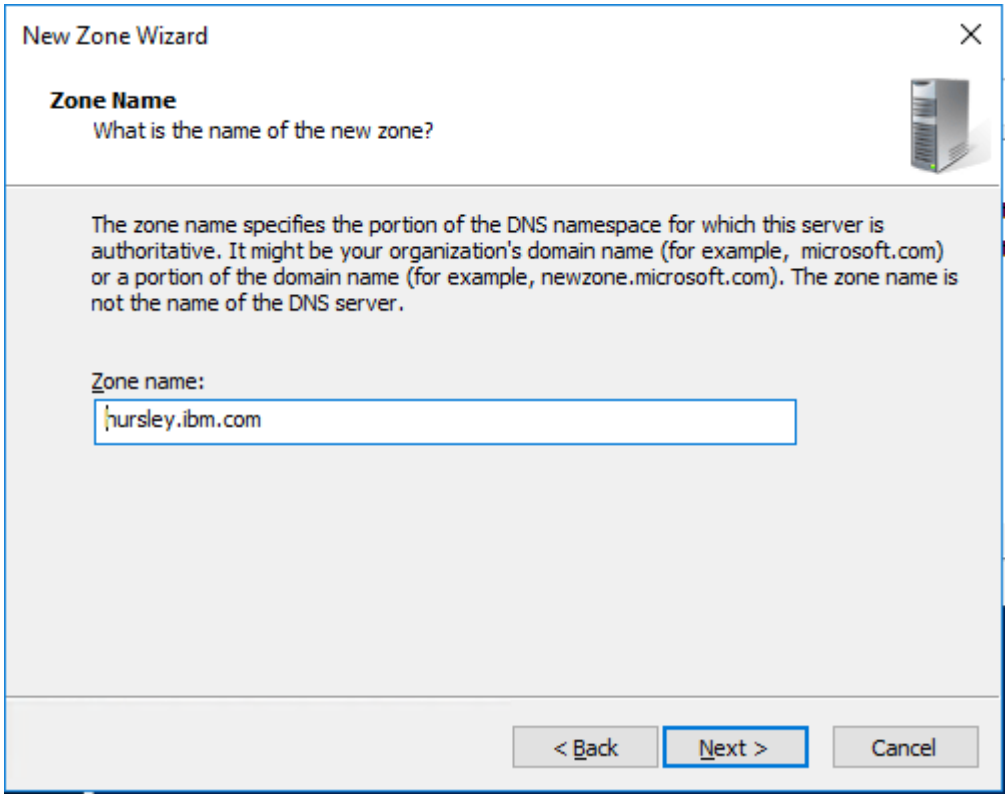


d. For the replication scope, select **To all DNS Servers running on domain controllers in this domain**, and then click **Next**.

© Copyright IBM Corporation 2018

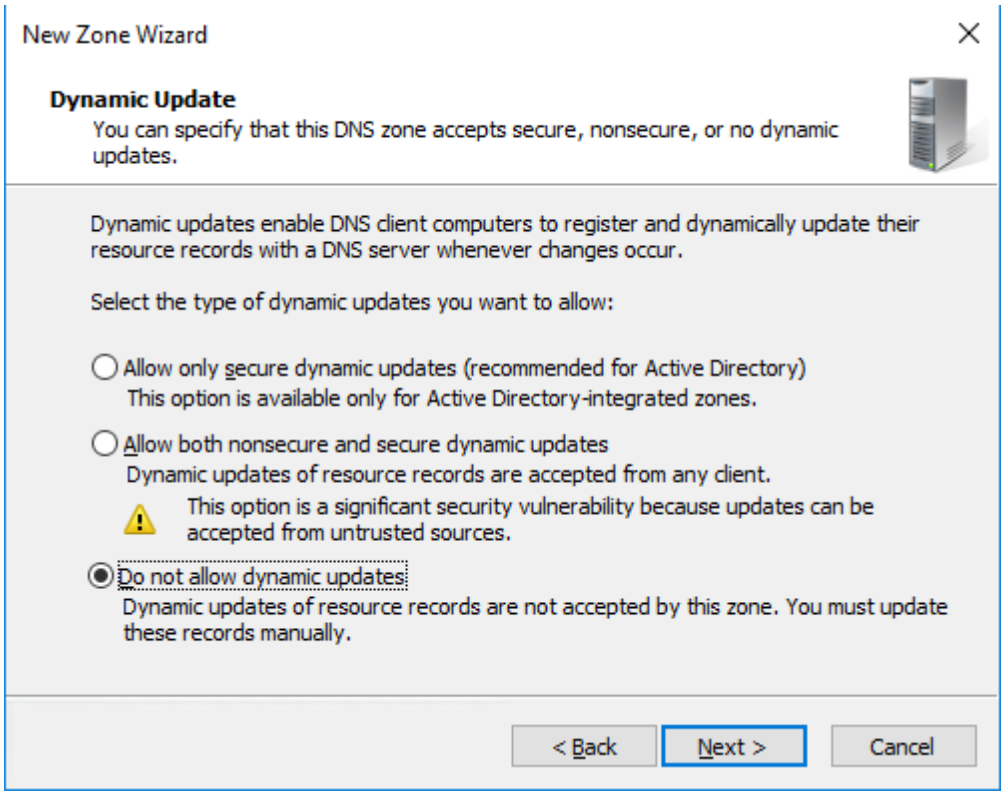


e. For the **Zone name**, specify the z/OS system domain name, and then click **Next**. In this example, the z/OS system host name is winmvs2c.hursley.ibm.com, so the domain name is hursley.ibm.com. Click **Next**.

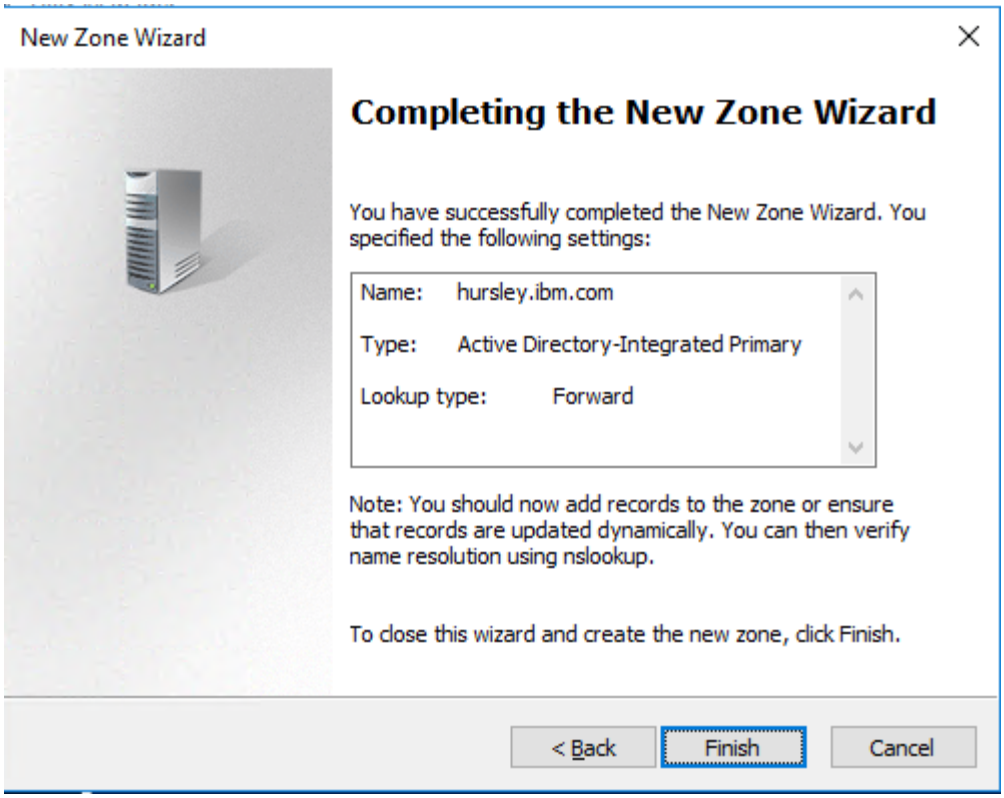


© Copyright IBM Corporation 2018

f. For the **Dynamic Update** setting, select **Do not allow dynamic updates** and then click **Next**.

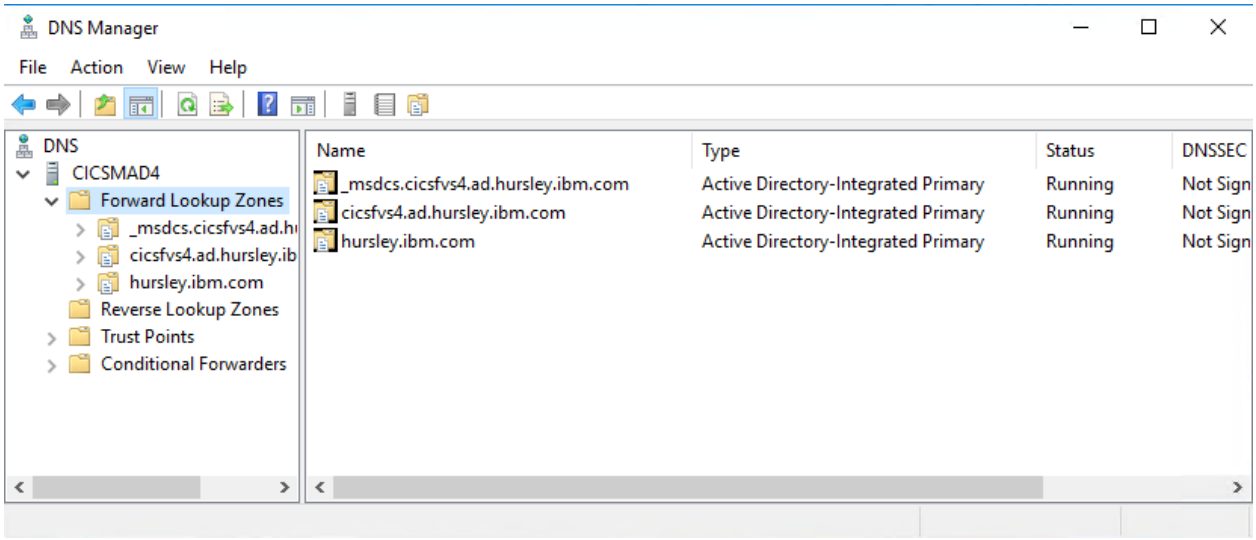


g. Review the new forward zone summary and then click **Finish**.



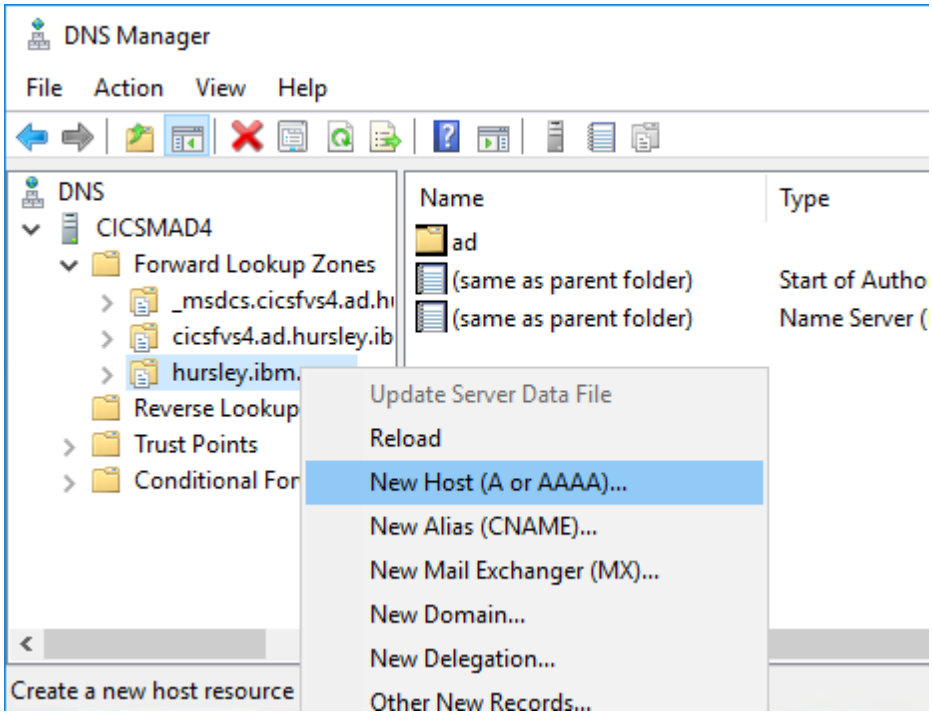
© Copyright IBM Corporation 2018

h. In the **DNS Manager** window, we see that an entry has been created for the domain containing the z/OS Kerberos system.



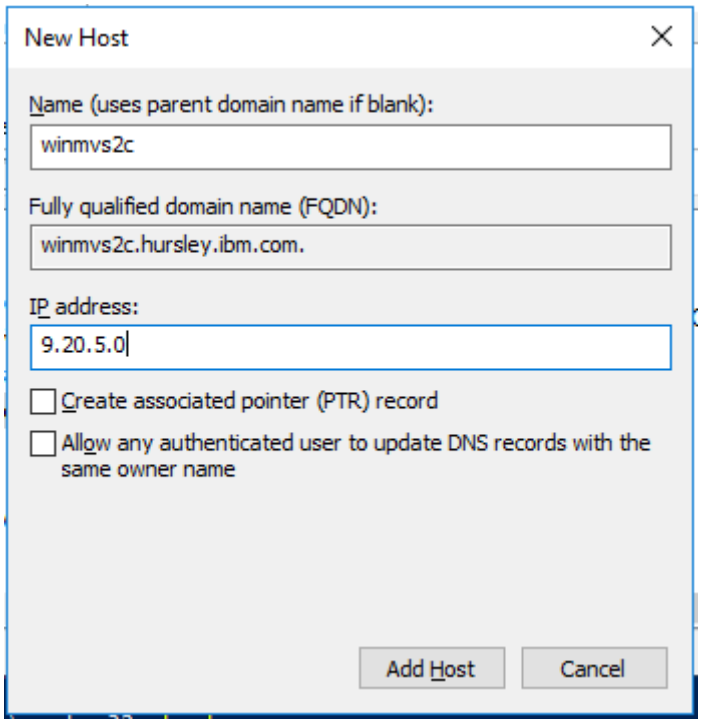
3. Create a new host entry for the zone. We need to create a new host entry for the new zone by allowing these steps in the DNS Manager:

a. Right-click the z/OS domain and select **New Host (A or AAAA)**.

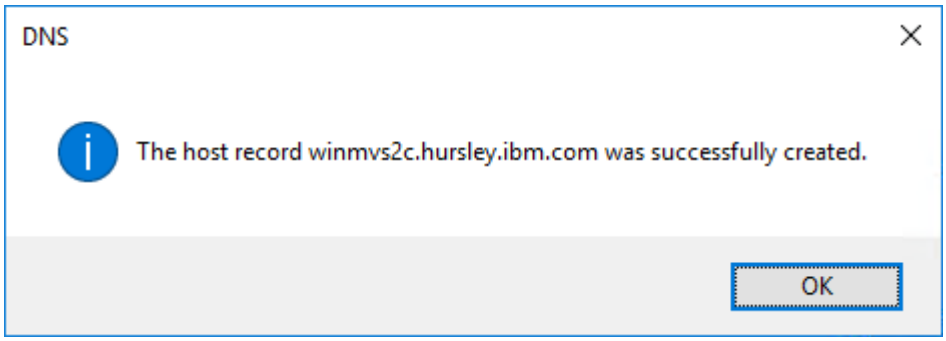


b. Specify the host name and IP address of the z/OS system, and click **Add Host.** The host name is the short name of the z/OS system. After entering this name, the full z/OS system name displays in the **Fully qualified domain name (FQDN)** field.

© Copyright IBM Corporation 2018

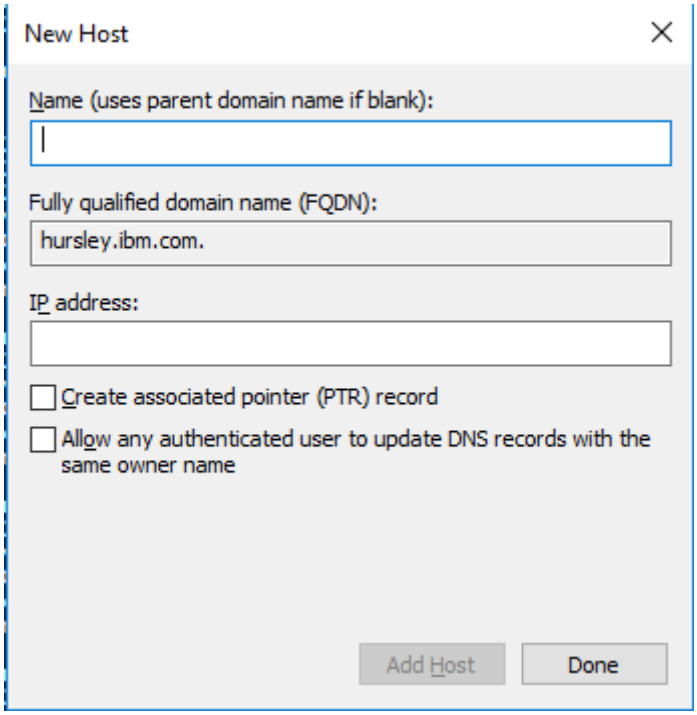


c. A confirmation dialog is displayed. Click **OK**.



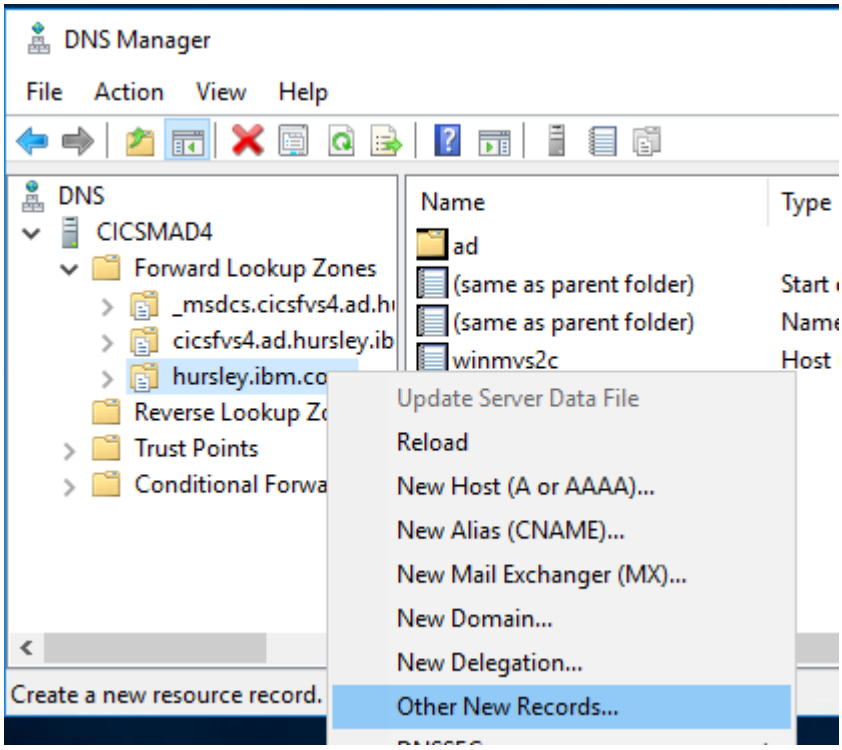
d. Click **Done** to close the new host dialog.

© Copyright IBM Corporation 2018



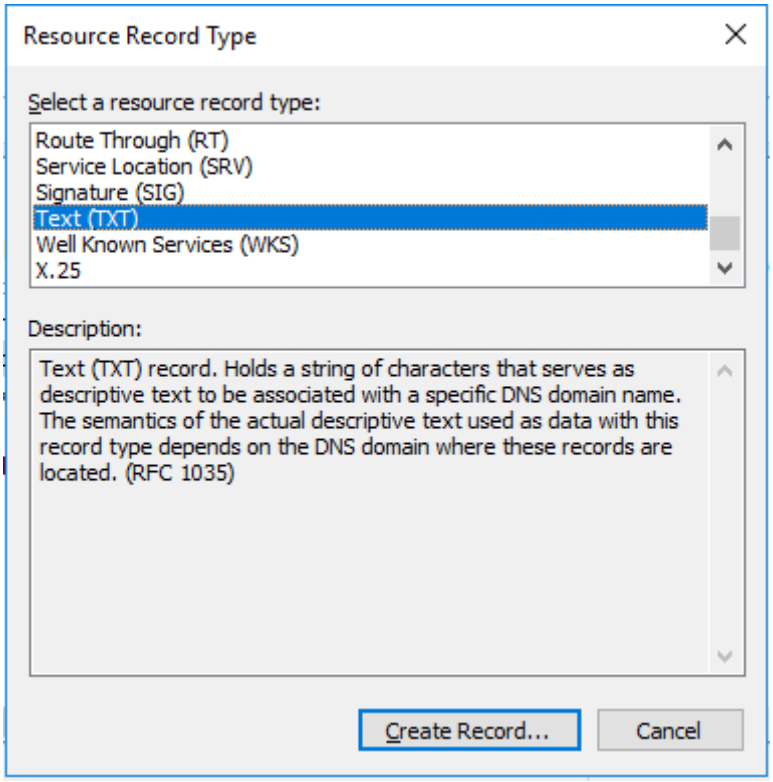
4. Create additional records for the z/OS realm.

    a. Right-click the z/OS domain, and select **Other New Records**.

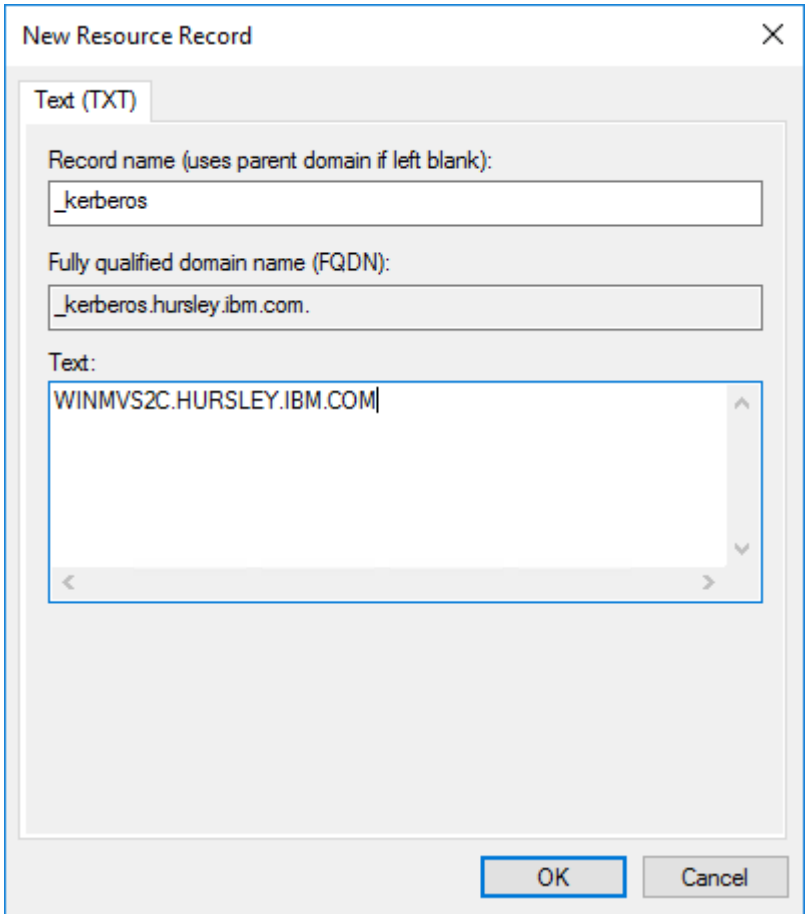


    b. Firstly, we create a text record to define the realm name. For the **Resource Record Type**, select **TEXT (TXT)**, and click **Create Record**.

© Copyright IBM Corporation 2018



c. Specify the record name as _kerberos, and specify the text as the z/OS realm name, in this case WINMVS2C.HURSLEY.IBM.COM and click **OK**.



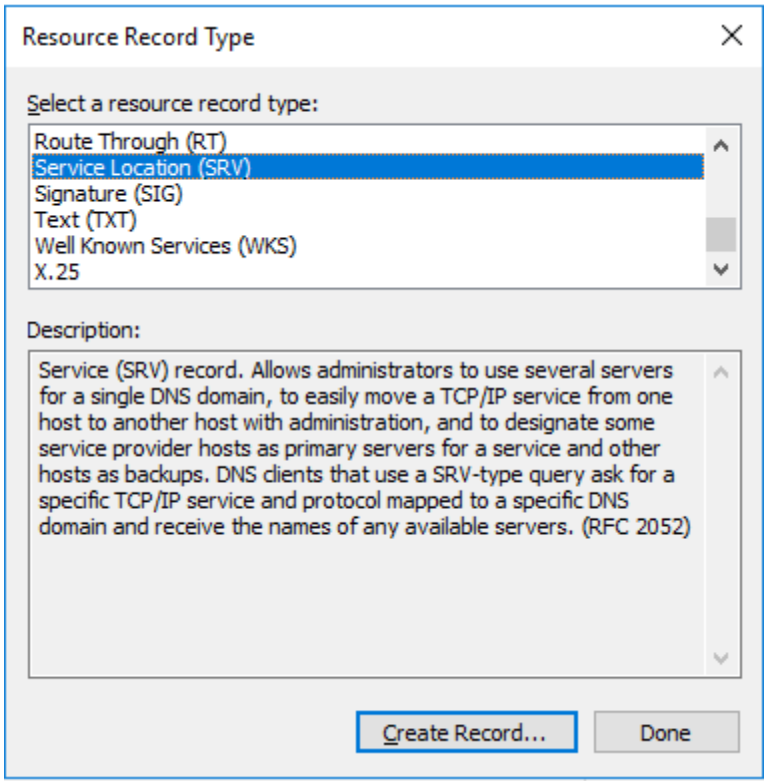
© Copyright IBM Corporation 2018

5. Next, we create the service location records. The z/OS Kerberos realm is a foreign realm to the Microsoft realm. The Microsoft DNS server can locate the z/OS Kerberos realm if you add a service location record to the Microsoft DNS server. In this section, we are going to create four similar service location records. This table lists the details of the records that will be created:

Service	_kerberos	_kerberos	_kpasswd	_kpasswd.
Protocol	_tcp	_udp	_tcp	_udp
Port number	88	88	464	464
Host offering this service	Host offering this service should be set to the fully qualified name of the z/OS system (in this case, winmvs2c.hursley.ibm.com)			

Below, we show the process to add the first of these service location records. Then repeat the process (with the values from the table) to create the next three records.

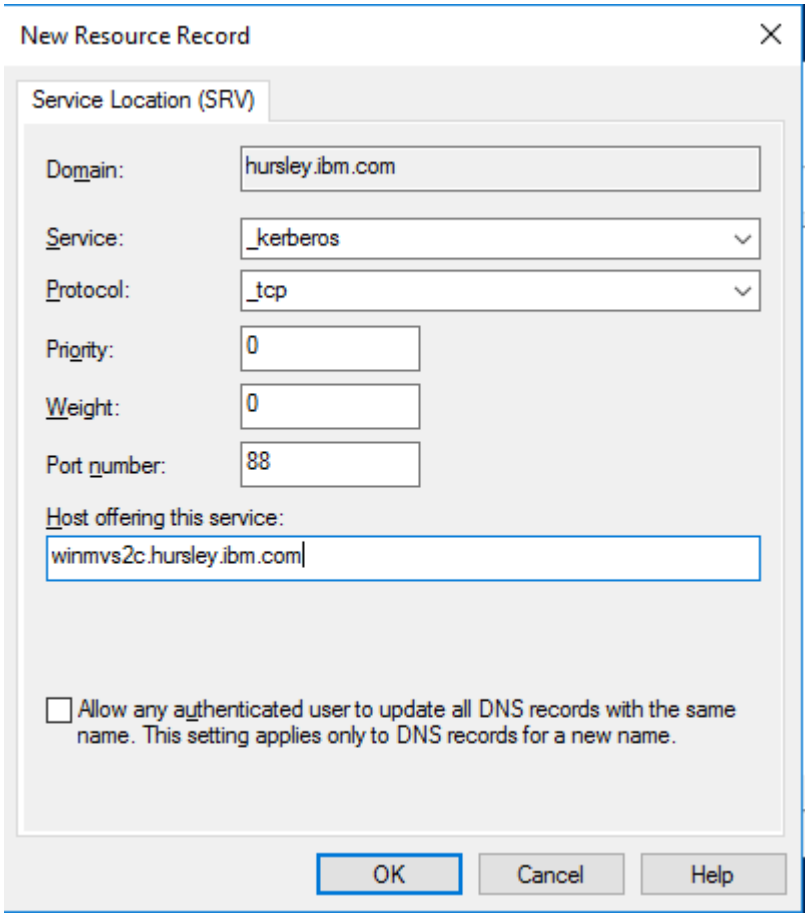
a. To create the first service location record, in the create resource record type dialog, select **Service Location (SRV)**, and click **Create Record**.



b. In the **New Resource Record** dialog, complete the following fields (first column of the table above) and click **OK**:

   i. Service name = _kerberos
   ii. Protocol = _tcp
   iii. Port number = 88
   iv. Host offering this service should be set to the fully qualified name of the z/OS system

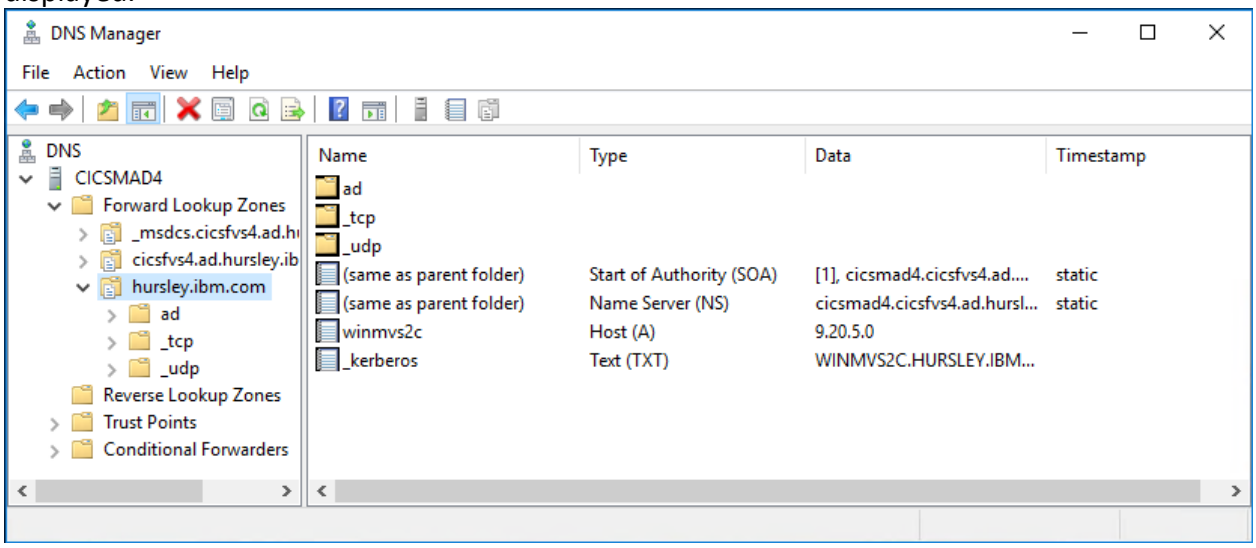
© Copyright IBM Corporation 2018



c. Repeat the process for the values in the three remaining columns of the table to create four records in total.

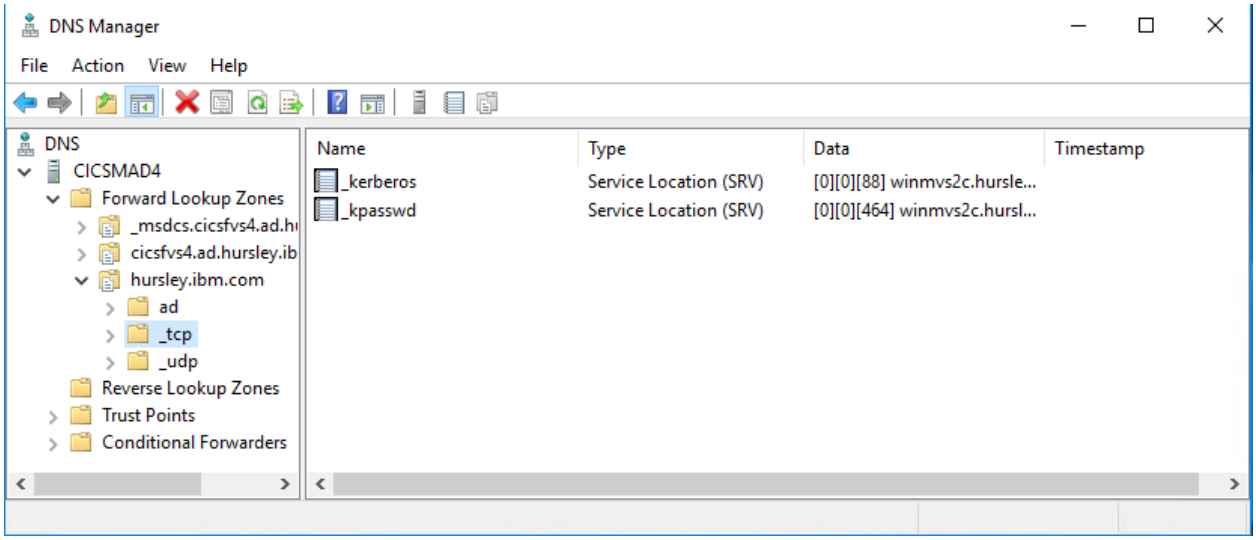
d. When all four service location records are created, we are finished, so click **Done** on the Resource Record Type dialog.

6. We can now ensure that the records have been created. Select the z/OS domain resource in the DNS Manager window. A text record for _kerberos and a host record for the z/OS system should be displayed.

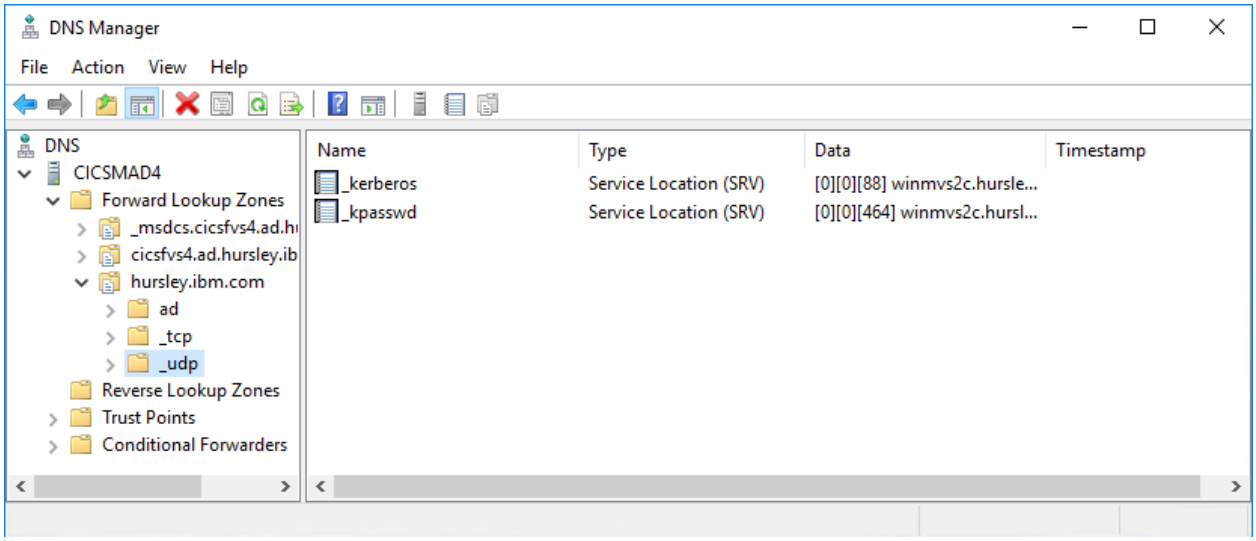


© Copyright IBM Corporation 2018

a. Select _tcp under the z/OS domain. A service location record for _kerberos and _kpasswd with the z/OS fully-qualified host name should be displayed.



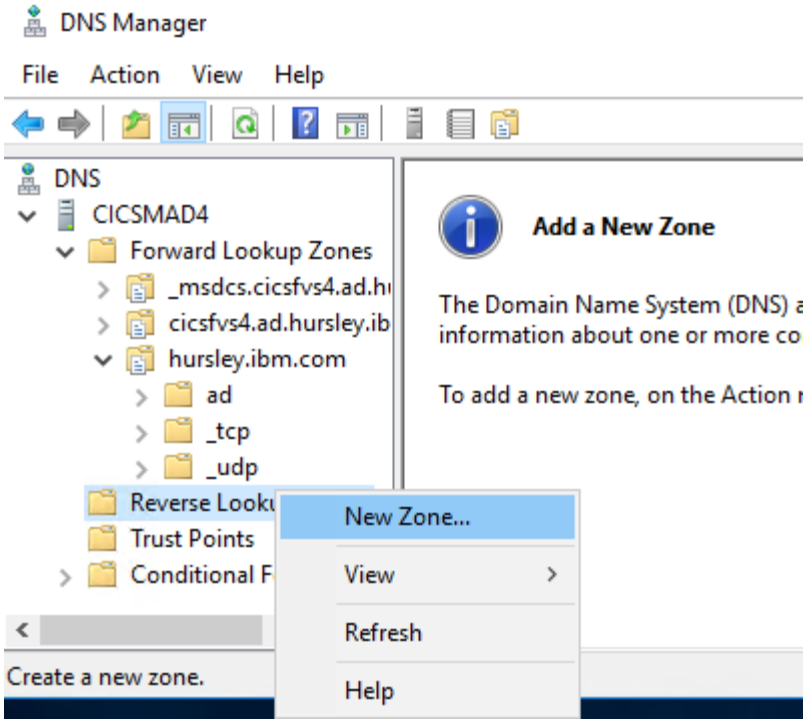
b. Select _udp under the z/OS domain. A service location record for _kerberos and _kpasswd with the z/OS fully-qualified host name should be displayed.



Create a reverse lookup zone

1. We now need to create a reverse lookup zone for the z/OS system. Right-click the **Reverse Lookup Zone** folder, and select **New Zone…**

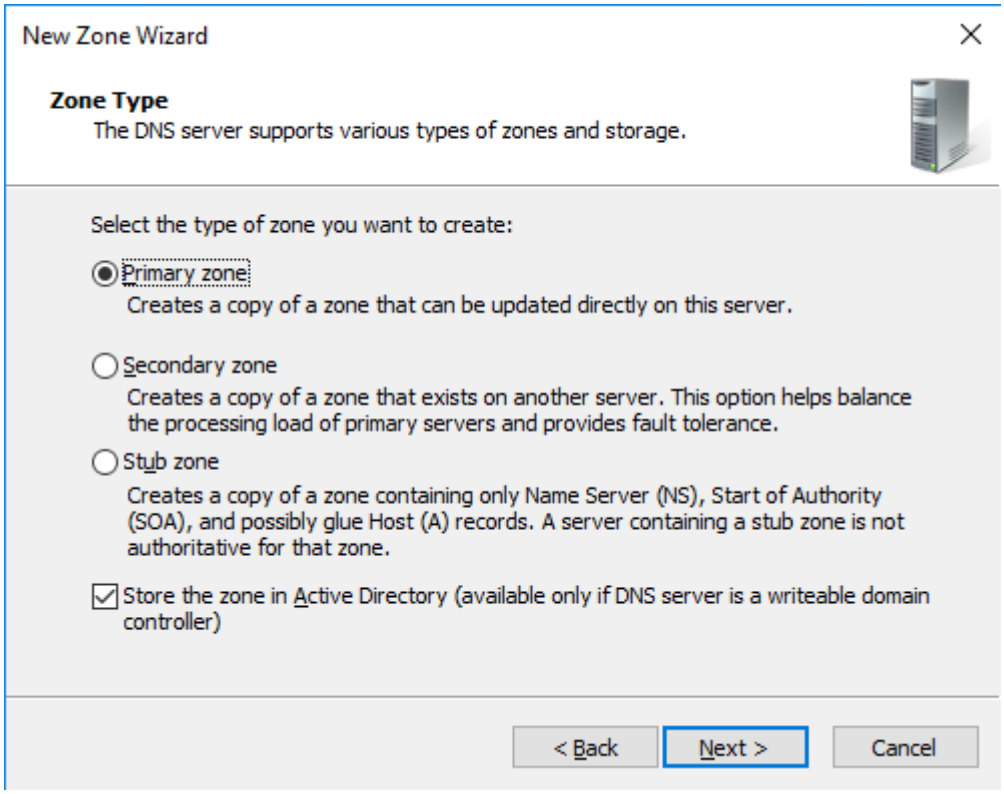
© Copyright IBM Corporation 2018



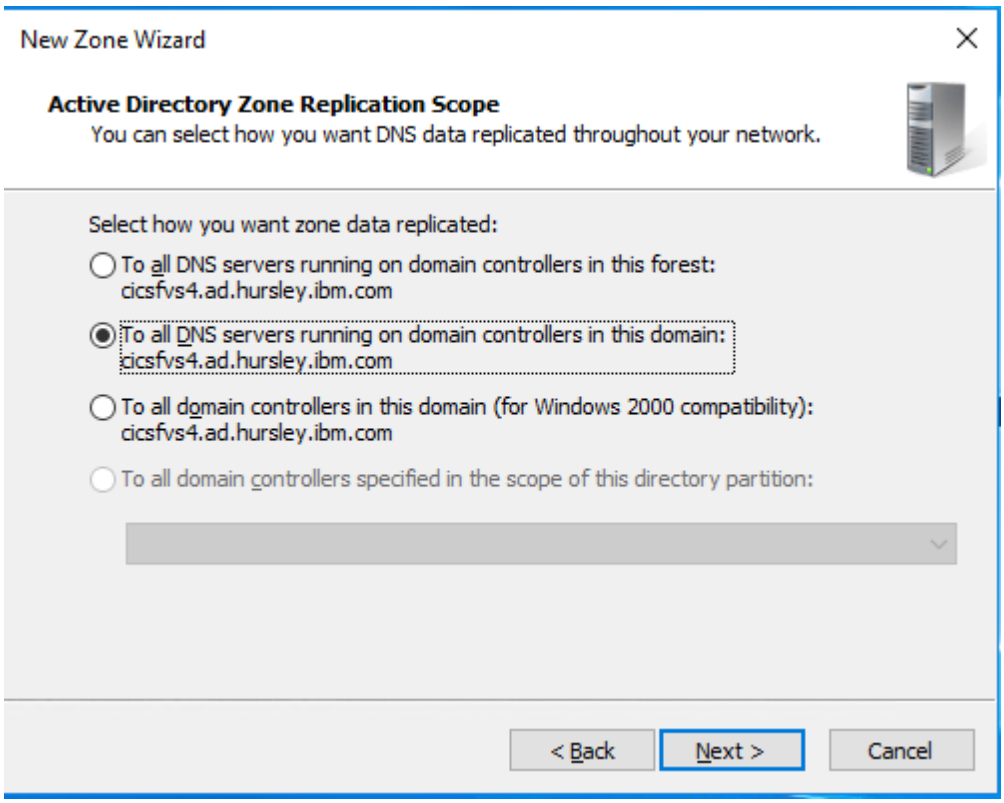
2. Click **Next** on the first pane of the **New Zone Wizard**.

3. In the **Zone Type** window, select **Primary zone**, and select **Store the zone in Active Directory**. Then click **Next**.

© Copyright IBM Corporation 2018

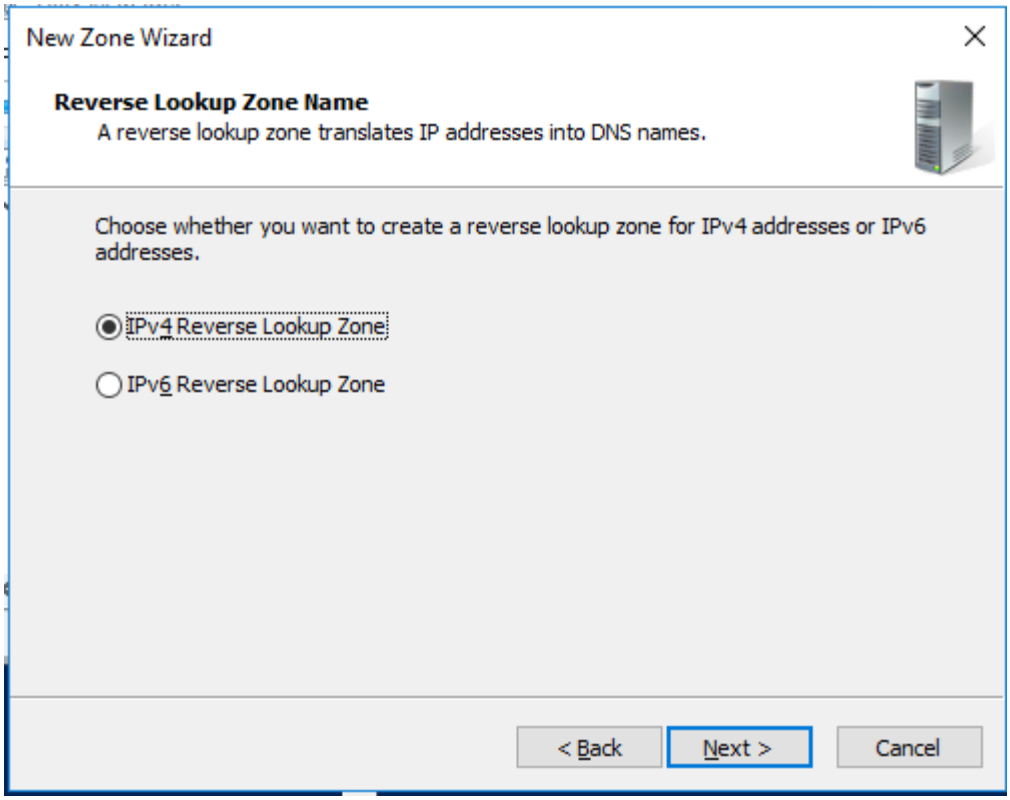


4. For the replication scopes, select **To all DNS servers running on domain controllers in this domain**, and then click **Next**.

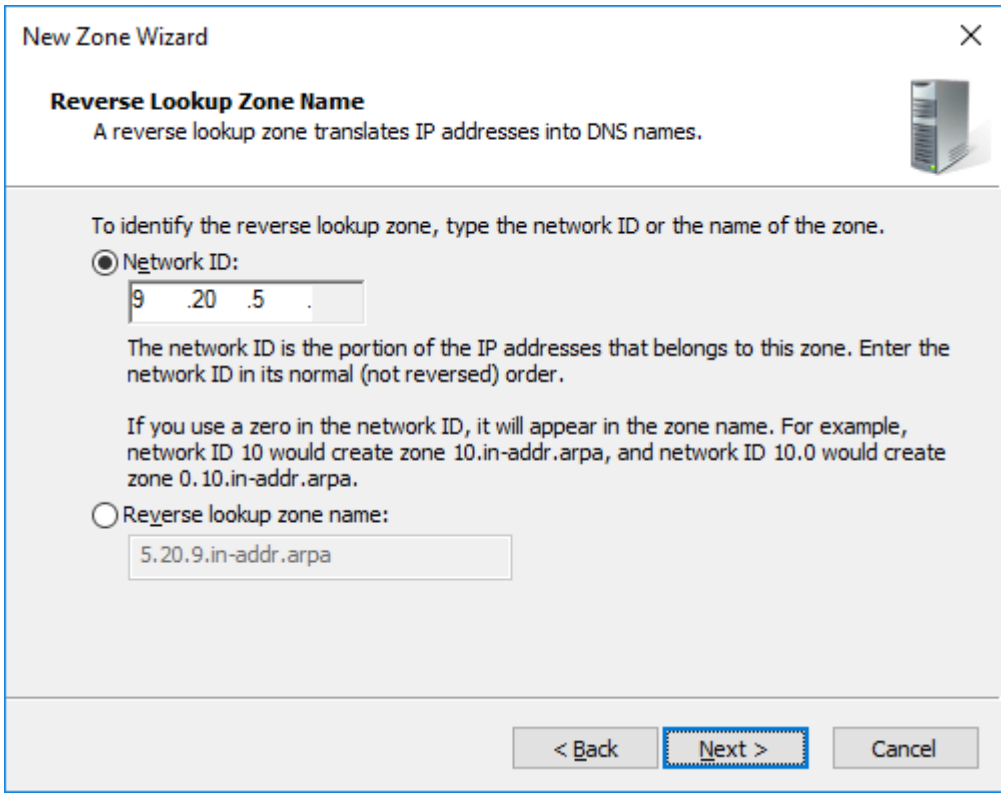


5. If prompted, select the appropriate IP zone and then click **Next**.

© Copyright IBM Corporation 2018

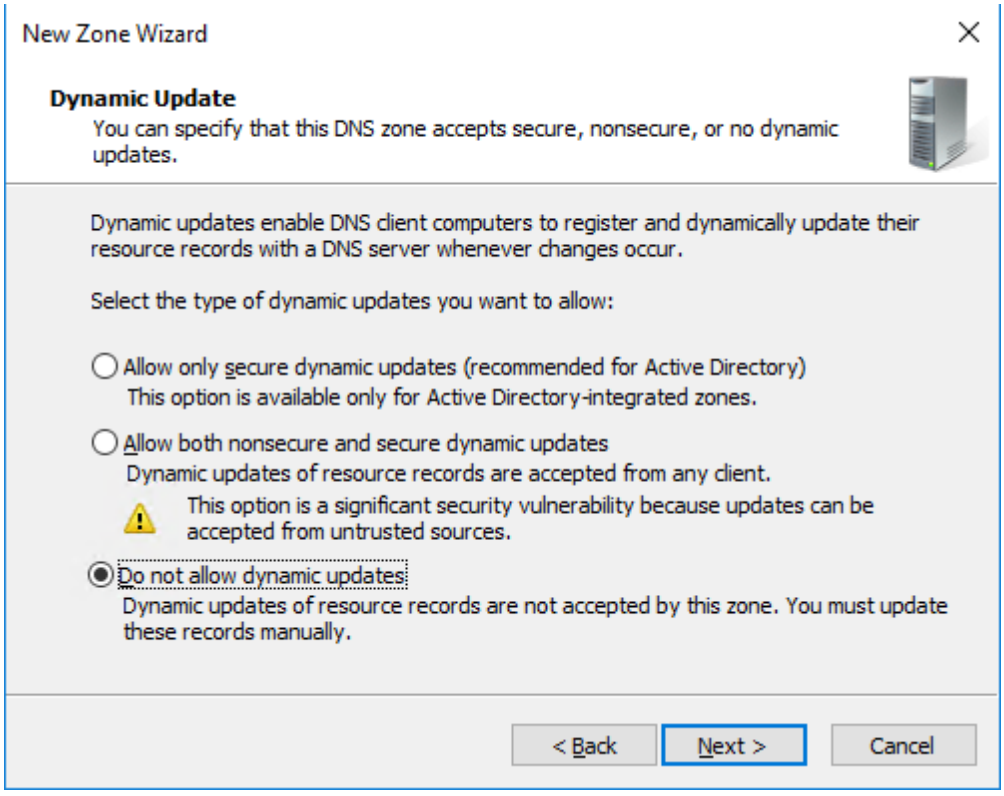


6. Specify the **Network ID** and then click **Next.** The network ID is the first three octets of the z/OS system's IP address.

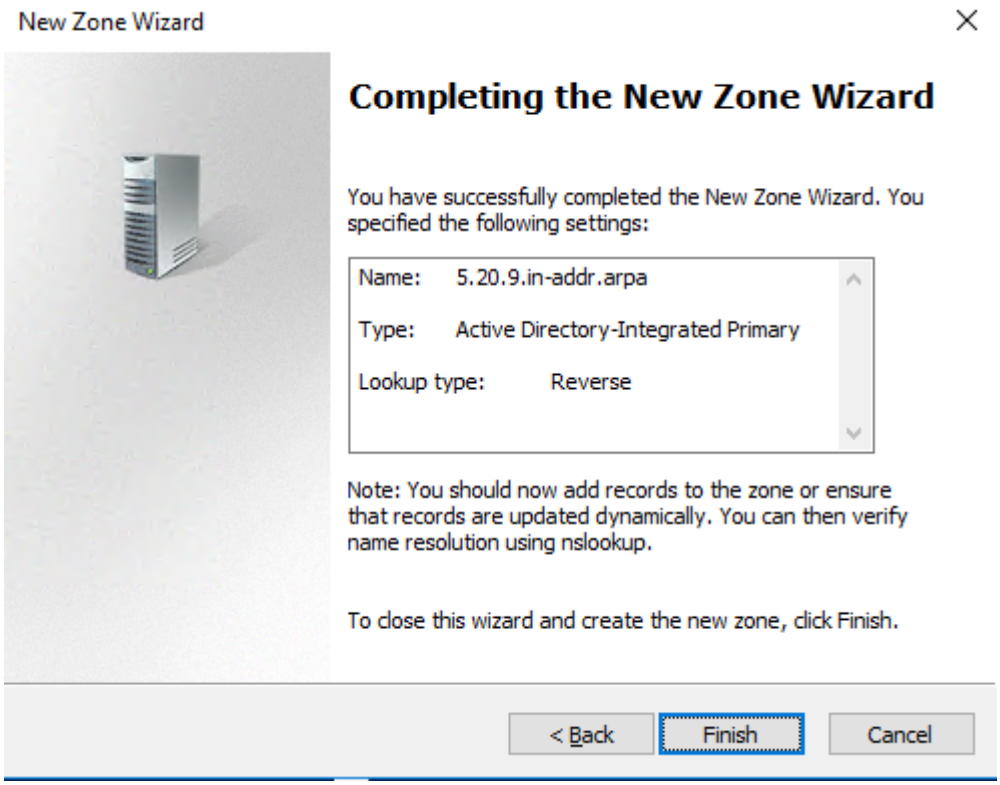


7. Select **Do not allow dynamic updates**, and then click **Next**.

© Copyright IBM Corporation 2018



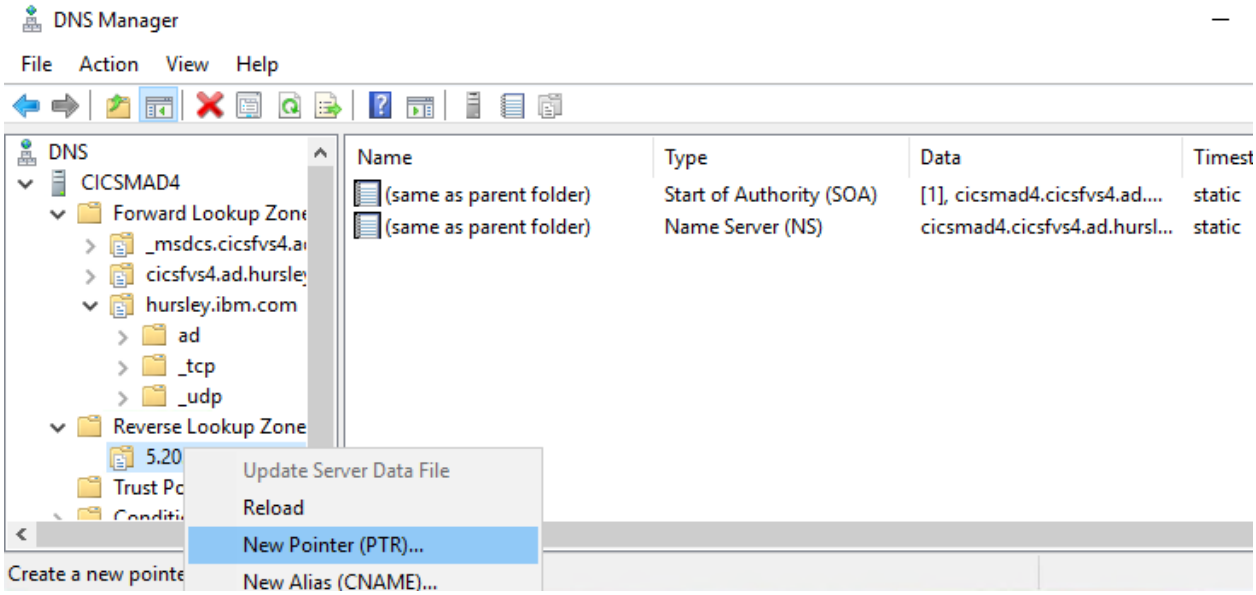
8. Verify the new reverse lookup zone summary and then click **Finish**.



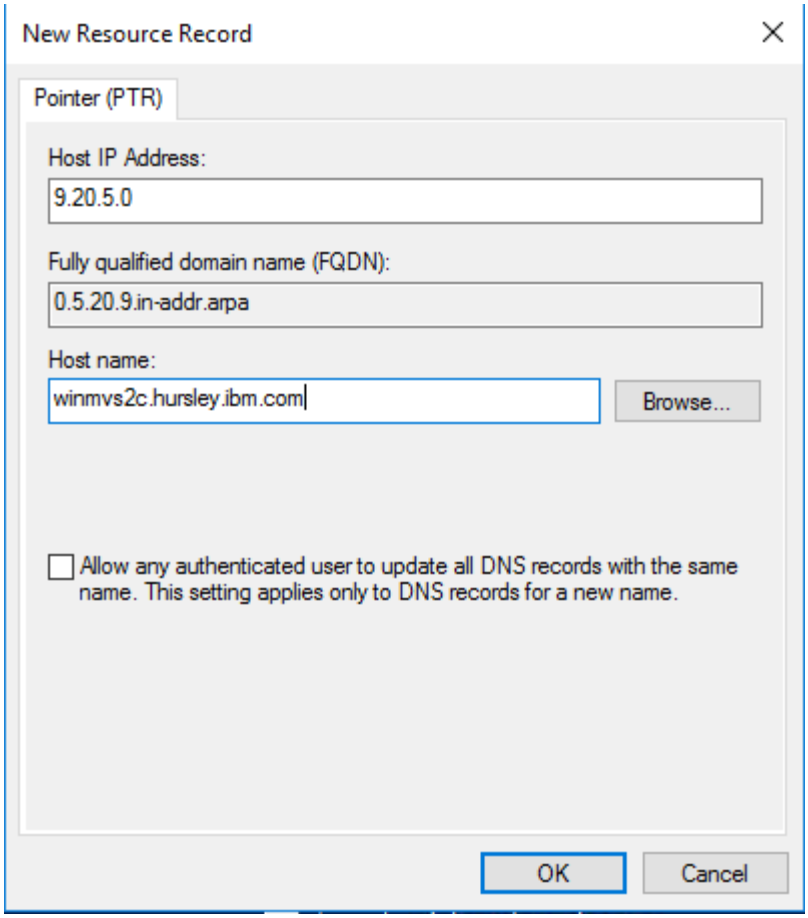
Create a pointer record

© Copyright IBM Corporation 2018

1. Next we need to create a new pointer record for the new reverse lookup zone. Right-click the **Reverse Lookup Zone**, and select **New Pointer (PTR)**.

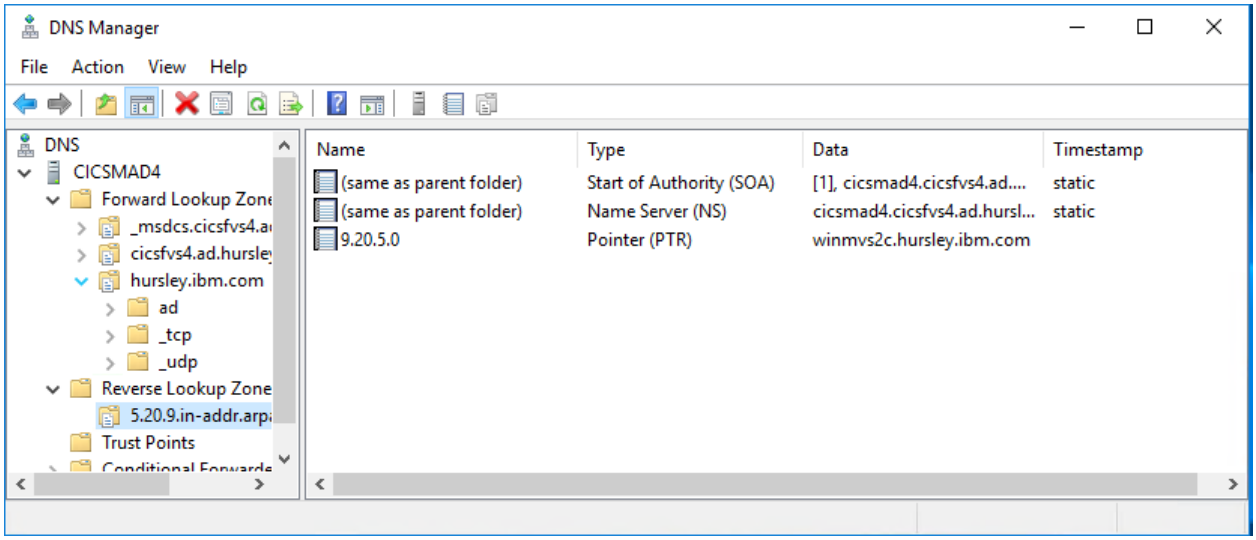


2. For the **Pointer** record, specify the z/OS system IP address and fully- qualified host name and click **OK**.



3. In the DNS window, select the z/OS **Reverse Lookup Zone**. A pointer record for the z/OS system is now displayed.

© Copyright IBM Corporation 2018



That is the end of the required DNS updates.

z/OS changes

On z/OS, we need to define the cross-realm trust in RACF, update the /etc/skrb/krb5.conf file and define user ID mappings for the users defined in the Active Directory realm.

Configuring cross realm trust in RACF

In this section, we issue RACF commands to configure trust on the z/OS KDC. The values used need to be consistent with those used for the Active Directory configuration above.

Note that while Kerberos principals usually have the format 'Primary/Instance@Realm', RACF uses a special format for entering and displaying fully qualified principal names due to the '@' being a variant character. This format is /.../Realm/Primary/Instance.

Run the RDEFINE REALM commands to define the incoming and outgoing cross realm trust between the MAD KDC and z/OS KDC, as follows:

RDEFINE REALM /.../<*MAD realm*>/krbtgt/<*zOS realm*> KERB(PASSWORD(<*trust password*>) ENCRYPT(<*encrypt type*>))
RDEFINE REALM /.../<*zOS realm*>/krbtgt/<*MAD realm*> KERB(PASSWORD(<*trust password*>) ENCRYPT(<*encrypt type*>))

Specify the values for <*MAD realm*>, <*zOS realm*>, <*trust password*>, and <*encrypt type*>. For the incoming cross trust realm, you do not need to specify the ENCRYPT argument if you do not want to restrict incoming encryption types. However, it is useful to explicitly set the encryption types for the outgoing cross trust realm to only types supported by both KDCs because we do not want to encrypt a ticket that the remote KDC cannot decrypt.

Note: The same trust password ***must*** be used here as that which was used when creating the trust on Microsoft Active Directory.

In our case, the commands are:

© Copyright IBM Corporation 2018

```
RDEFINE REALM /.../WINMVS2C.HURSLEY.IBM.COM/krbtgt/ CICSFVS4.AD.HURSLEY.IBM.COM
KERB(PASSWORD(K1E2R3B!)) ENCRYPT(NODES NODES3 NODESD AES128 AES256)

RDEFINE REALM /.../ CICSFVS4.AD.HURSLEY.IBM.COM/krbtgt/WINMVS2C.HURSLEY.IBM.COM
KERB(PASSWORD(K1E2R3B!)) ENCRYPT(NODES NODES3 NODESD AES128 AES256)
```

Again, we can confirm that the commands have had the desired effect with simple commands:

```
RLIST REALM /.../WINMVS2C.HURSLEY.IBM.COM/KRBTGT/CICSFVS4.AD.HURSLEY.IBM.COM
KERB NORACF

RLIST REALM /.../CICSFVS4.AD.HURSLEY.IBM.COM/KRBTGT/WINMVS2C.HURSLEY.IBM.COM
KERB NORACF
```

…and we get the following output:

```
RL REALM    /.../WINMVS2C.HURSLEY.IBM.COM/KRBTGT/CICSFVS4.AD.HURSLEY.IBM.COM    KERB NORACF
CLASS     NAME
-----     ----
REALM     /.../WINMVS2C.HURSLEY.IBM.COM/KRBTGT/CICSFVS4.AD.HURSLEY.IBM.COM

KERB INFORMATION
----------------
KEY VERSION= 001
KEY ENCRYPTION TYPE= NODES NODES3 NODESD AES128 AES256
CHECK ADDRESSES= NO
READY

RL REALM    /.../CICSFVS4.AD.HURSLEY.IBM.COM/KRBTGT/WINMVS2C.HURSLEY.IBM.COM    KERB NORACF
CLASS     NAME
-----     ----
REALM     /.../CICSFVS4.AD.HURSLEY.IBM.COM/KRBTGT/WINMVS2C.HURSLEY.IBM.COM

KERB INFORMATION
----------------
KEY VERSION= 001
KEY ENCRYPTION TYPE= NODES NODES3 DESD AES128 AES256
CHECK ADDRESSES= NO
READY
```

Mapping Active Directory Principals to RACF users

The Active Directory principals must be mapped to RACF user IDs. Multiple principals can be mapped to the same RACF user ID, or each principal can be mapped to its own RACF user ID. Each principal in a foreign realm can be mapped to its own user ID in RACF, or multiple principals in a foreign realm can be mapped to the same user ID in RACF.

To map a foreign Kerberos principal to a RACF user, define a general resource profile in the KERBLINK class. Each mapping is defined and modified using the RDEFINE and RALTER commands. The format of the command is:

```
RDEFINE KERBLINK /.../FOREIGN.REALM.COM/foreignKerberosUser APPLDATA('USERID')
```

© Copyright IBM Corporation 2018

Note that the characters of the profile name of the KERBLINK class are not converted to upper case, so be sure to enter the realm portion of the profile name in upper case and the foreign principal name in the appropriate case.
So, in our case, we issued this command to map the Active Directory user ‘cics1’ to RACF user ‘JT1D’:

```
RDEFINE KERBLINK /.../CICSFVS4.AD.HURSLEY.IBM.COM/cics4 APPLDATA('JT1D')
```

To verify that the previous command completed successfully, we issue the following RACF command

```
RLIST KERBLINK /.../CICSFVS4.AD.HURSLEY.IBM.COM/cics4
```

and we get the following output:

```
CLASS     NAME
-----     ----
KERBLINK   /.../ CICSFVS4.AD.HURSLEY.IBM.COM/cics4

LEVEL  OWNER     UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------  ----------------  -----------  -------
00    IBMUSER       NONE            ALTER    NO

INSTALLATION DATA
-----------------
NONE

APPLICATION DATA
----------------
JT1D

AUDITING
--------
FAILURES(READ)

NOTIFY
------
NO USER TO BE NOTIFIED
```

© Copyright IBM Corporation 2018

Testing the cross-realm configuration

We now have both Kerberos realms configured and trusting each other. We can now use our test application to ensure that everything is configured correctly. We will use the same test approach that we used earlier to test the z/OS-only Kerberos configuration.

If you used your own application to test the single-realm Kerberos configuration above then you should be able to use the same application here. If you used the web service defined in SupportPac CA1P as described earlier in this paper, you can use the same CICS definitions and client program here. We merely need to change a couple of the command-line parameters used to invoke the client program so that they specify the client principal that we have defined in Microsoft Active Directory, rather than a client principal defined in RACF.

You might remember that the client should be invoked with a command such as:
java -cp cicstest.jar cics.Requester <host:port> <echoString> <clientPrincipalName> <clientPassword> <servicePrincipalName>

For this test, based on the two-realm configuration that we are using here, we use the following command:
java -cp cicstest.jar cics.Requester winmvs2c.hursley.ibm.com:12414 HelloWorld WINMVS2C.HURSLEY.IBM.COM winmvs2c.hursley.ibm.com cics4@CICSFVS4.AD.HURSLEY.IBM.COM k1e2r3b! jt1b_service

If you compare this command with the command used in the z/OS-only Kerberos test above, you see that the only parameters that have changed are the client principal and the client password, as this command is referencing the user ID and password that are defined on Windows, not the user defined in RACF.

```
Web service requester for CICS Echo service
-------------------------------------------
Parameters
Host:Port = winmvs2c.hursley.ibm.com:12414
EchoString = HelloWorld
realm =  WINMVS2C.HURSLEY.IBM.COM
kdc = winmvs2c.hursley.ibm.com
clientPrincipalName = cics4@CICSFVS4.AD.HURSLEY.IBM.COM
clientPassword = k1e2r3b!
servicePrincipalName = jt1b_service

Target URL is : http://winmvs2c.hursley.ibm.com:12414/ca1p/echoProgProviderBatch


[JGSS_DBG_CRED]  main JAAS config: debug=true
[JGSS_DBG_CRED]  main JAAS config: credsType=initiate only (default)
[JGSS_DBG_CRED]  main config: useDefaultCcache=false
[JGSS_DBG_CRED]  main config: useCcache=null
[JGSS_DBG_CRED]  main config: useDefaultKeytab=false (default)
[JGSS_DBG_CRED]  main config: useKeytab=null
[JGSS_DBG_CRED]  main JAAS config: forwardable=false (default)
[JGSS_DBG_CRED]  main JAAS config: renewable=false (default)
```

© Copyright IBM Corporation 2018

[JGSS_DBG_CRED] main JAAS config: proxiable=false (default)
[JGSS_DBG_CRED] main JAAS config: tryFirstPass=false (default)
[JGSS_DBG_CRED] main JAAS config: useFirstPass=false (default)
[JGSS_DBG_CRED] main JAAS config: moduleBanner=false (default)
[JGSS_DBG_CRED] main JAAS config: interactive login? yes
[JGSS_DBG_CRED] main JAAS config: refreshKrb5Config = true
[JGSS_DBG_CRED] main Retrieving Kerberos creds from cache for principal=null
[JGSS_DBG_CRED] main No Kerberos creds in cache for principal cics4@CICSFVS4.AD.HURSLEY.IBM.COM
[JGSS_DBG_CRED] main Doing Kerberos login for principal cics4@CICSFVS4.AD.HURSLEY.IBM.COM
[JGSS_DBG_CRED] main Doing Kerberos login for principal: cics4@CICSFVS4.AD.HURSLEY.IBM.COM
[JGSS_DBG_CRED] main Kerberos login complete
[JGSS_DBG_CRED] main Login successful
[JGSS_DBG_CRED] main kprincipal : cics4@CICSFVS4.AD.HURSLEY.IBM.COM
[JGSS_DBG_CRED] main cics4@CICSFVS4.AD.HURSLEY.IBM.COM added to Subject
[JGSS_DBG_CRED] main Kerberos ticket added to Subject
[JGSS_DBG_CRED] main added key of type aes128-cts-hmac-sha1-96
[JGSS_DBG_CRED] main added key of type des3-cbc-sha1
[JGSS_DBG_CRED] main added key of type rc4-hmac
Request message

<?xml version="1.0" standalone="no"?> <soapenv:Envelope
xmlns:q0="http://www.ECHOPROG.ECHOCOMM.Request.com"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP-ENV:Header xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"> <wsse:Security SOAP-ENV:mustUnderstand="0"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
<wss:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-
profile-1.1#Kerberosv5_AP_REQ" Id="uuide7eafbbf-0143-1ffd-8da2-b17ce09c1b4d"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">YIICFgYJKoZIhvcSAQICAQBuggIFMIICAaADAgEFoQMCAQ6iBwMFAAAAAACjggEbYYIBFzCCARogAwIBB
aEaGxhXSU5NVlMyQy5IVVJTTEVZLklCTS5DT02iGTAXoAMCAQGhEDAOGwxqdDFiX3NlcnZpY2WjgdQwgdGgAwI
BEqEDAgFEooHEBIHBAA1dkl6oI4CaXfa398khi1Mu31T8elwOmalsFEPHTXPSloHLR0lFTFBEckKLEyvGIfb+R5rSyk+
XbgUQOzdMsb4tdQX7avHtk79uXaJAjUJVYASSgIFy0nhRN12Nfi+wvCF/o1WET0lTb+ddcl8dUcbcGPtUU9N8FVTC
AHzG3BnvwdpqyqVLh1qU7Bzv+Ojsx3J3BEXkWmN82IGUwVNj4t+LMNehwTFeXJVrAQ9OfNwTzbAWbrmL4HW
GCBJlc1/N5KSBzDCByaADAgERooHBBIG+WgOboyM0e+lu/kVpf9K5U6TN/aFr+weWdpSE3WWjJt6TbyYaidsG1S
OVSf/wnwhPxPoweAqaZrh3HVDRuzdfZg6+TRCqPyS6TxPYE7VkobTtvshvP6btNNEfGzYTnEfKJdd0vPr/lUbAiaKhK
hXpbNb6MUO8DJwrrQ2JIdiMkHAKrcYKIMQkFn2wUprPWhwM3CzzBAZw+w7KwMwj+CPV0L7Y1ZtyHUiM9ws4
CpK/pVpkvKsSm26wXMCKIq52sg==</wss:BinarySecurityToken> </wsse:Security> </SOAP-ENV:Header>
<soapenv:Body> <q0:ECHOPROGOperation> <q0:echo_string> HelloWorld</q0:echo_string>
</q0:ECHOPROGOperation> </soapenv:Body> </soapenv:Envelope>

Attempting to read response

<SOAP-ENV:Envelope xmlns:q0="http://www.ECHOPROG.ECHOCOMM.Request.com"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Body><ECHOPROGOperationResponse
xmlns="http://www.ECHOPROG.ECHOCOMM.Response.com"><echo_string>CICS--> HelloWorld
</echo_string></ECHOPROGOperationResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>

End of response

© Copyright IBM Corporation 2018

If you get similar output, congratulations! You have successfully configured Kerberos in Microsoft Active Directory and RACF, with trust between these two realms and invoked a CICS web service using your Kerberos configuration. If you did not see similar output, first re-run the z/OS-only configuration test described above as that should still work. If that test also fails, refer to the debugging advice covered in the CA1P SupportPac documentation to address any web service issues. If you still cannot resolve the problem then refer to the Troubleshooting section for advice about debugging Kerberos configurations.

Troubleshooting

If your Kerberos request fails, it can be challenging to find the cause of the problem. Some Kerberos-related error messages don't explicitly point out the exact cause of a failure, but there are debugging techniques and sources of additional information that can be of use.

Problems are often caused by mistyped names and passwords, so a good place to start is by checking the configuration and test program for incorrect values.

Sources of diagnostic information

If you are using the sample client program and CICS web service described above then the client program includes several print statements that show progress through the process of logging into the KDC, requesting a Kerberos token, issuing the request to CICS and receiving a reply.

If your client program is written in Java™ (such as the client used here), then there are command-line options that can be used when invoking the client to generate a lot more diagnostic information. The options are dependent on the JRE you are using. If you are using an IBM JRE (e.g. running the client on z/OS), add the following options to the java command:

```
-Dcom.ibm.security.jgss.debug=all -Dcom.ibm.security.krb5.Krb5Debug=all
```

If you are using an Oracle JRE, add the following option to the java command:

```
-Dsun.security.krb5.debug=true
```

Common errors

The causes of some of the common errors are not immediately obvious from the error messages issued. This section shows some of these error messages and suggests some possible causes.

Failure to connect to KDC

If you fail to connect to the KDC, then firstly check that the KDC host name is correctly specified and that the KDC is running. The next step is to check for network issues, and in particular firewall settings.

It is worth ensuring that the file /etc/services (or equivalent data set) contains udp and tcp entries for the KDC (default is port 88). You can see this on lines such as:

```
kerberos    88/udp      kdc         # Kerberos V5 KDC
kerberos    88/tcp      kdc         # Kerberos V5 KDC
```

© Copyright IBM Corporation 2018

Failure to log into KDC

These errors occur once we have connected to the KDC. Errors at this stage are typically related to authenticating the client.

Client not found in Kerberos database

The start of the stack trace (and the preceding trace output) is shown below:

```
[JGSS_DBG_CRED]  main Doing Kerberos login for principal: cics4x@CICSFVS4.AD.HURSLEY.IBM.COM
com.ibm.security.krb5.KrbException, status code: 6
        message: Client not found in Kerberos database
```

This error can be caused by an incorrect client principal name.

Integrity check on decrypted field failed

The start of the stack trace (and the preceding trace output) is shown below:

```
[JGSS_DBG_CRED]  main Doing Kerberos login for principal: cics4@CICSFVS4.AD.HURSLEY.IBM.COM
com.ibm.security.krb5.internal.crypto.KrbCryptoException, status code: 0
        message: com.ibm.security.krb5.internal.KrbException, status code: 31
        message: Integrity check on decrypted field failed
```

This error can be caused by an incorrect client password (remember that passwords are case-sensitive).

Failure requesting a Kerberos token

These errors occur after the client has successfully logged into the KDC, so we can assume that the client principal and password are correct. Some of these scenarios have error messages that are reasonably easy to understand, while others can be more challenging to diagnose.

Server principal is not found in security registry

The start of the stack trace is shown below:

```
com.ibm.security.krb5.KrbException, status code: 7
        message: Server principal is not found in security
registry:jt1b_serviceX@WINMVS2C.HURSLEY.IBM.COM
        at com.ibm.security.krb5.KrbTgsRep.<init>(Unknown Source)
```

© Copyright IBM Corporation 2018

This error can be caused by an incorrect service principal. Ensure that the service principal specified in the client request matches that defined to the KDC.

No credential

The stack trace (edited) is shown below:

```
java.lang.Exception: No credential
        at com.ibm.security.jgss.i18n.I18NException.throwException(Unknown Source)
.
.
.
Exception in thread "main" org.ietf.jgss.GSSException, major code: 11, minor code: 0
        major string: General failure, unspecified at GSSAPI level
        minor string: Error: java.lang.Exception: Error: java.lang.Exception: No credential
        at com.ibm.security.jgss.i18n.I18NException.throwGSSException(Unknown Source)
.
.
.
```

Errors such as this one can be caused by a number of different problems. Additional information can be gained by using the debug flags described above. In the rather extensive additional output generated with debug enabled in the client, we see the following lines:

```
[KRB_DBG_KDC] KRBError:main:   >>>KRBError:
[KRB_DBG_KDC] KRBError:main:     sTime is Thu Feb 22 17:42:48 GMT 2018 1519321368000
[KRB_DBG_KDC] KRBError:main:     suSec is 484814
[KRB_DBG_KDC] KRBError:main:     error code is 14
[KRB_DBG_KDC] KRBError:main:     error Message is KDC has no support for encryption type
[KRB_DBG_KDC] KRBError:main:     sname is
krbtgt/WINMVS2C.HURSLEY.IBM.COM@CICSFVS4.AD.HURSLEY.IBM.COM
[KRB_DBG_KDC] KRBError:main:     msgType is 30
```

Here we see the more helpful error message “KDC has no support for encryption type”. When the cross-realm trust is specified, both KDCs must support at least one common encryption type. In RACF, this can be specified when issuing the RDEFINE REALM … commands to set up the trust. Note that in Microsoft Active Directory, when the cross realm trust has been defined, you can select the properties of the trust you have specified and there is a check box labelled **The other domain supports Kerberos AES Encryption** which you must select for AES encryption to be supported.

© Copyright IBM Corporation 2018

Integrity check fails

Another example of an issue reported as “General failure, unspecified at GSSAPI level”:

```
com.ibm.security.krb5.KrbException, status code: 31
        message: Integrity check fails
        at com.ibm.security.krb5.KrbTgsRep.<init>(Unknown Source)
        at com.ibm.security.krb5.KrbTgsReq.getReply(Unknown Source)
.
.
.
java.lang.Exception: No credential
        at com.ibm.security.jgss.i18n.I18NException.throwException(Unknown Source)
        at com.ibm.security.krb5.internal.l.a(Unknown Source)
.
.
.
Exception in thread "main" org.ietf.jgss.GSSException, major code: 11, minor code: 0
        major string: General failure, unspecified at GSSAPI level
        minor string: Error: java.lang.Exception: Error: java.lang.Exception: No credential
        at com.ibm.security.jgss.i18n.I18NException.throwGSSException(Unknown Source)
.
.
.
```

Note that the rather lengthy concatenation of stack traces includes the generic and quite common error message “General failure, unspecified at GSSAPI level”, but the most interesting message can be found right at the start of the stack trace.

This error can be caused by having the trusts defined between the two realms, but where the passwords for the trust definitions do not exactly match. Remember that passwords are case sensitive.

Service key not available

This is one of the less obvious error messages:

```
com.ibm.security.krb5.KrbException, status code: 45
        message: Service key is not available
```

© Copyright IBM Corporation 2018

Typically this error is caused by a mismatch of supported encryption types. The types of encryption supported are specified in multiple locations, and if the various parties involved cannot agree on an encryption type, then you can get this error message. It can be challenging to identify which specifications don't match. The following is a list of places that the supported encryption types can be specified:

- /etc/skrb/home/kdc/envar
- /etc/skrb/krb5.conf
- The 'KERBDFLT' realm
- The client principal
- The server principal

Check all of these and ensure that they are consistent.

Pre-authentication information was invalid

This is another error message that can be misleading at first sight.

```
[KRB_DBG_KDC] KRBError:main:   >>>KRBError:
[KRB_DBG_KDC] KRBError:main:      cTime is Fri Mar 09 12:12:22 GMT 2018 1520597542000
[KRB_DBG_KDC] KRBError:main:      sTime is Fri Mar 09 12:12:18 GMT 2018 1520597538000
[KRB_DBG_KDC] KRBError:main:      suSec is 83024
[KRB_DBG_KDC] KRBError:main:      error code is 24
[KRB_DBG_KDC] KRBError:main:      error Message is Pre-authentication information was invalid
[KRB_DBG_KDC] KRBError:main:      cname is jt1c_client@WINMVS2C.HURSLEY.IBM.COM
[KRB_DBG_KDC] KRBError:main:      sname is
krbtgt/WINMVS2C.HURSLEY.IBM.COM@WINMVS2C.HURSLEY.IBM.COM
[KRB_DBG_KDC] KRBError:main:      etext is Preauthentication failed
[KRB_DBG_KDC] KRBError:main:      msgType is 30
[KRB_DBG_AS] KrbAsRep:main:   KRBError received: Preauthentication failed
com.ibm.security.krb5.KrbException, status code: 24
        message: Pre-authentication information was invalid
```

This error can be generated by passing the incorrect password for the client principal.

Kerberos token rejected by CICS

When we have a token from the KDC, it is passed to the CICS region. To get this far, we know that the client principal and service principal are both known to the KDC, we passed the correct password for the client principal, and in the case of a client principal defined in MAD being used with a service principal defined in RACF, the trust between the two realms has been set up correctly.

There are numerous reasons why a token can be rejected by the CICS region, such as invalid client principal name, invalid service principal name, or an expired certificate. Such errors typically result in the client program receiving a 500 server error (in the case of a web service), and more usefully, CICS issuing a DFHXS1402 error message which contains details of the specific problem, such as this message:

```
DFHXS1402 02/22/2018 13:23:57 IYK2ZGVC A request to inquire the client principal of a kerberos token
obtained from a Security Token Service has failed, reason = R_USERMAP service responded no userid for client
principal. SAF codes are (X'00000008',X'00000000') ESM codes are (X'00000008',X'00000010') Taskid
(0000054) Tranid (CPIH) Task userid (CICSUSER)
```

© Copyright IBM Corporation 2018

As you can see from the message, in this particular case, the client principal (defined in Microsoft Active Directory) has not been associated with a RACF user ID. These messages tend to be reasonably self-explanatory.

© Copyright IBM Corporation 2018

Find more information

The following is a list of resources that might be useful. There are lots of resources available that cover aspects of the subject material in this document, but this list contains a few references we considered to be particularly useful or relevant.

RFC 4120 – The Kerberos Network Authentication Service (V5): https://tools.ietf.org/html/rfc4120

Windows Server Kerberos Authentication Overview: https://docs.microsoft.com/en-us/windows-server/security/kerberos/kerberos-authentication-overview

z/OS Integrated Security Services Network Authentication Service Administration: https://www.ibm.com/support/knowledgecenter/SSLTBW_2.3.0/com.ibm.zos.v2r3.euvfa00/toc.htm

CICS Kerberos support: https://www.ibm.com/support/knowledgecenter/SSGMCP_5.4.0/security/kerberos/kerberos_support.html

© Copyright IBM Corporation 2018

© Copyright IBM Corporation 2018

Notices

Notices applicable to this publication are available here:
https://developer.ibm.com/cics/legal-notices/

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

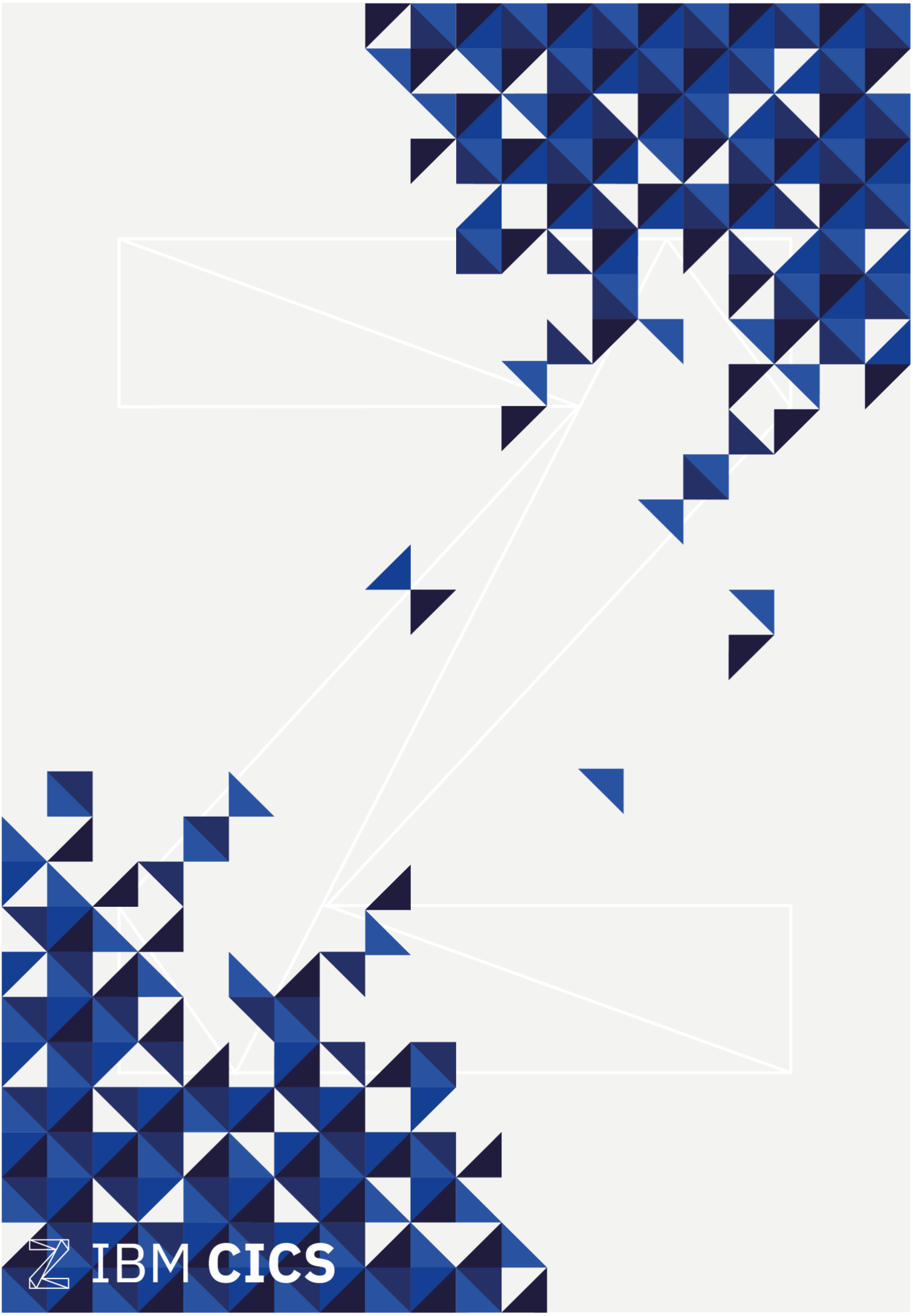
Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft, Windows and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

© Copyright IBM Corporation 2018


IBM CICS