
The Citrix Server Farm: Functionality and Optimizations

By Mike Stringer

Citrix Systems, Inc.



CITRIX[®]

Notice

The information in this publication is subject to change without notice.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. CITRIX SYSTEMS, INC. ("CITRIX"), SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR DIRECT, INCIDENTAL, CONSEQUENTIAL, OR ANY OTHER DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS PUBLICATION, EVEN IF CITRIX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES IN ADVANCE.

This publication contains information protected by copyright. Except for internal distribution, no part of this publication may be photocopied or reproduced in any form without prior written consent from Citrix.

The exclusive warranty for any Citrix products discussed in this publication, if any, is stated in the product documentation accompanying such products. Citrix does not warrant products other than its own.

Product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

© 2000–2002 Citrix Systems, Inc.

All rights reserved. Printed in the U.S.A.

Version History		
May 5, 2000	Mike Stringer, Citrix Systems, Inc.	Version 1.0
April 26, 2002	Mike Stringer, Citrix Systems, Inc.	Update



TABLE OF CONTENTS

Overview	1
Server-Side Perspective	2
Details of the Process	4
Steps 1 and 2	4
ICA Gateway Functionality and Optimization Techniques	5
Steps 3 and 4	7
Optimizations	7
Citrix Server Farm Limitations	10
Performance Issues Due to WAN and Load Balancing Issues over the WAN	10
Functionality Limitation	10
Built-in Tools for Troubleshooting Program Neighborhood Service Problems	10
Appendix A	13
Multiple Domain MetaFrame Server Farm Functionality	13

Overview

The information in this article applies to Citrix System's MetaFrame 1.8, Service Pack 1.

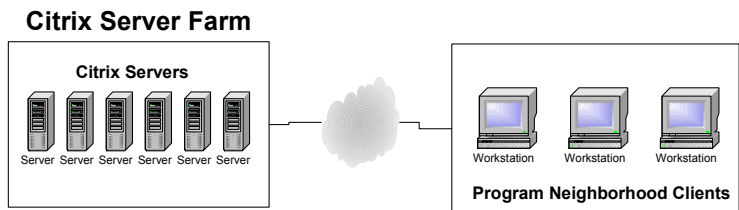
Program Neighborhood is a set of services and APIs that provide the Program Neighborhood ICA Client with a filtered view of all of the published applications (managed applications) in a given enterprise that a particular user can access. The introduction of Program Neighborhood in MetaFrame 1.8 enables the implementation of the Citrix server application farm. Most enterprise customers have already implemented, or are currently in the process of implementing, this configuration. This article discusses this configuration, including functionality and optimization of the services and protocols utilized.

Two main configurations exist:

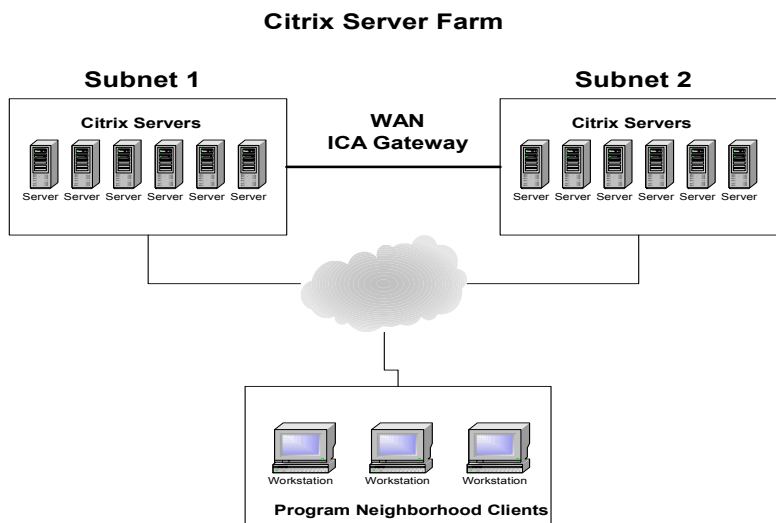
The single subnet Citrix server farm

The multi-subnet Citrix server farm

The single subnet Citrix server farm



The multi-subnet Citrix server farm

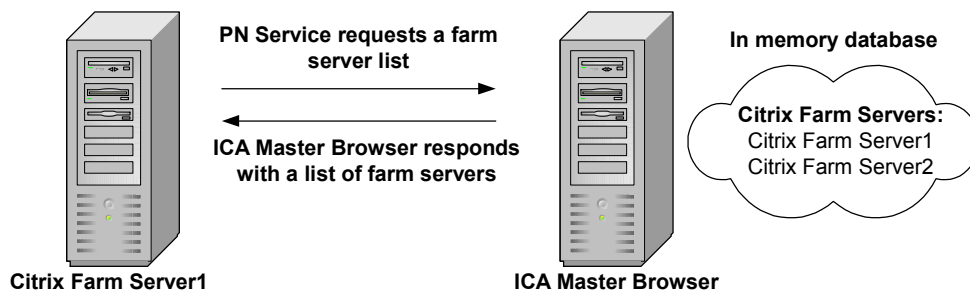


The two examples above represent the Citrix server farm in single and multiple subnet configurations and are typical of what our customers are implementing. The MetaFrame servers within these farms can run multiple, often different, managed applications in a load balanced environment. This scenario can span multiple Windows NT domains (see Appendix A) while still providing a single point of administration. Specific applications can be located at any site in the Citrix server farm. Users, utilizing the Program Neighborhood Client, can access all applications, regardless of where those applications actually reside, by simply logging onto the Citrix server farm. For example; an enterprise site has a Citrix farm server configured with Microsoft Office. Its users require access to a database application that resides in another office across the WAN. The Citrix server farm provides users access to all the required applications with a *single* logon to the farm.

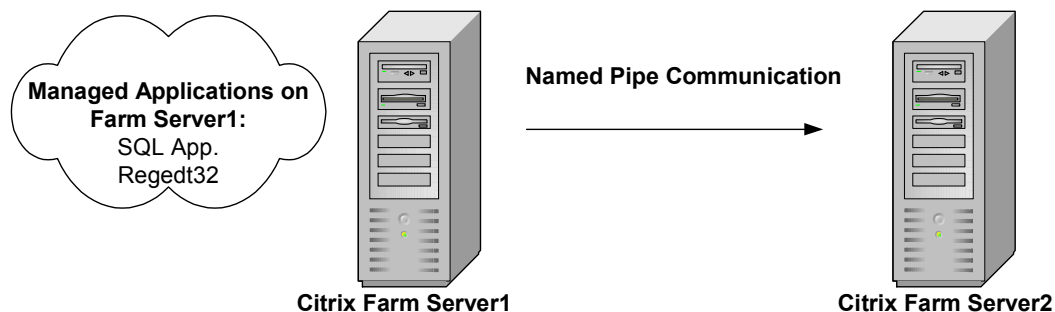
The two main elements of this process are the ICA Browser service and the Program Neighborhood (PN) service. The ICA Browser (with regard to the PN service) ensures that the Program Neighborhood service is aware of all the MetaFrame servers in a server farm. The Pnsvc.exe (Program Neighborhood service) is the server-side process responsible for implementing most of the Program Neighborhood functionality. The Program Neighborhood service (PN service) ensures that all Program Neighborhood Clients receive a complete, filtered list of all managed applications available in the farm.

Server-Side Perspective

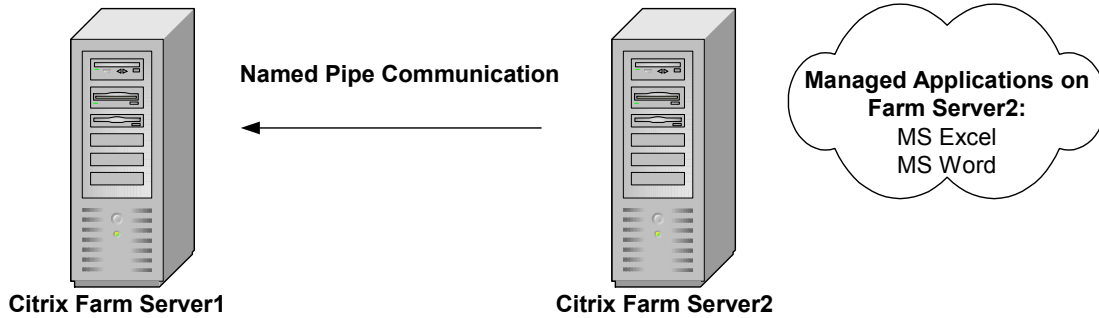
1. When contacted with an updated managed application list, all farm servers respond to the initiating server with an update of their managed application lists.
2. The resulting managed application list shown below represents the available applications for Program Neighborhood Clients.



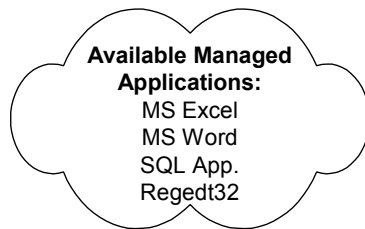
3. The PN service builds a list of servers it needs to update with its contribution to the available managed applications. The update is performed by named pipe connections.



4. When contacted with an updated managed application list, all farm servers respond back to the initiating server with an update of their managed application lists.



5. The resulting managed application list shown below represents the available applications for Program Neighborhood Clients.



Details of the Process

Steps 1 and 2

The PN Service contacts the master ICA Browser in two situations.

- The PN service is on a farm server that has just been rebooted or the server has just been added into the server farm.
- The master ICA Browser check parameter is enabled (must be at MetaFrame Service Pack 1 or later) in the registry. (This is enabled by default.)

The master ICA Browser check parameter is a new feature added to the PN service in Service Pack 1. The PN service now asks the master browser for a list of servers in the farm. In two seconds (the default is two), the browser requests this information again and compares the two queries. If there is a change in the list (for example; a new server is added to the farm), the browser continues to request updates every two seconds. When the list stabilizes, the PN service increases its “ask time” (two times by default) after each successful comparison. These settings add robustness to the management of applications available for PN Clients.

The master ICA Browser check parameters are located at –

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ProgNeighborhood\Parameters

The following values can be added:

- **BrowserThreadMinTimeout**

The **BrowserThreadMinimumTimeout** is the minimum time, in seconds, the new thread sleeps before contacting the browser looking for server farm changes. The default setting is two seconds.

- **BrowserThreadMaxTimeout**

The **BrowserThreadMaximumTimeout** is the maximum time, in seconds, the new thread sleeps before contacting the browser looking for server farm changes. The default is 600 seconds.

- **BrowserThreadTimeoutFactor**

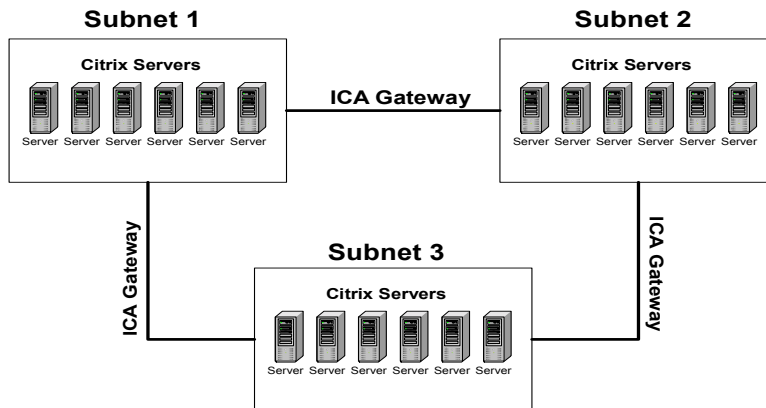
The **BrowserThreadTimeoutIncreaseFactor** is the multiplication factor used on time-out when no changes from the master browser are found in the server farm. The default is two times.

Note: Increasing the frequency of these parameters increases network traffic.

These parameters can be modified to optimize the availability of managed applications. They become even more significant in multiple subnet scenarios. The multi-subnet scenario requires ICA gateways between all farm servers. These ICA gateways communicate farm server availability between subnet master ICA Browsers. To ensure reliable availability of managed applications, the PN service must be aware of all farm servers. ICA gateway optimization plays a crucial role in this configuration.

ICA Gateway Functionality and Optimization Techniques

ICA gateways need to be configured **only** on one server per **one** side of each subnet. The master browser of each subnet sends all ICA gateway updates. Redundancy is the only reason to configure an ICA gateway on multiple servers or from both subnets. Additionally, ICA gateways are not transitive. Every MetaFrame server subnet must be connected to every other MetaFrame server subnet through an ICA gateway. In the following example, all the subnets are connected through ICA gateways.



The amount of data transferred across an ICA gateway is large; however, it includes only data update information (server name, server information, server load level, network address, application name, application information, disconnected session information, and user session information). Queries, browser elections, network acknowledgements, and so on are not included.

Each time a member browser experiences a change in user load, it reports its entire browser database to the master browser. This database includes the full list of all published and managed applications, all current user sessions, and all disconnected sessions.

Pertinent Registry Entries

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ICABrowser\Parameters

AgeDatabaseTime REG_DWORD 0 to 0xffffffff seconds (300 = default)

This indicates how frequently the master browser checks the “time to live” value associated with browser data. If the browser data is not being updated, it is deleted. This value can be set to zero to disable aging of browser data. Lower this value to increase the accuracy of the master browser’s database.

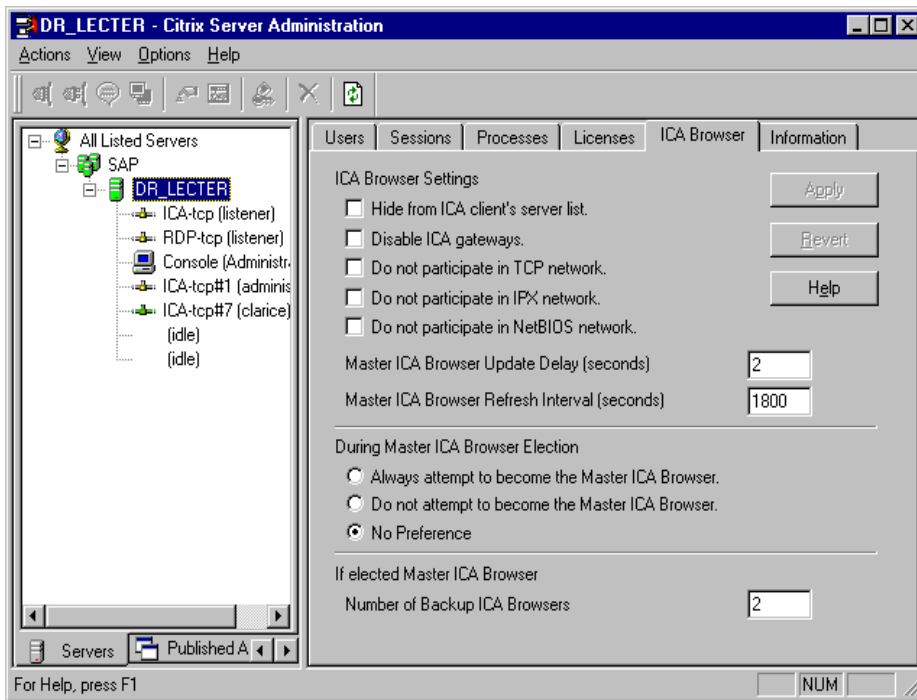
GatewayAddTime REG_DWORD 0 to 0xffffffff seconds (1800 = default)

This indicates the frequency at which the browser sends the **gateway add** command. The gateway add command configures all gateways that are specified in the registry. This does not affect how frequently browser updates are sent across the ICA gateway and should not be changed.

UpdateTime REG_DWORD 0 to 0xffffffff seconds (1800 = default)

This indicates the frequency at which the ICA browser updates the master browser. After an election, all ICA browsers know the address of the master browser. After a random delay (4–6 seconds) each browser sends an update datagram to the master browser. After this initial update, the ICA browsers update the master ICA browser every *x* seconds. When the master browser receives data from an ICA browser, the master browser sends an acknowledgement (ACK). The master browser updates are also sent whenever a client connects or disconnects from a MetaFrame server in that master browser's subnet. Lowering this time makes Browser data more accurate, but increases the CPU and network load.

Lower this value when load balancing is used. This becomes even more important in a WAN scenario. This forces the master browser to send more frequent updates across any configured ICA gateway. You can set the **Master ICA Browser Refresh Interval (seconds)** in the Citrix Server Administration dialog box.

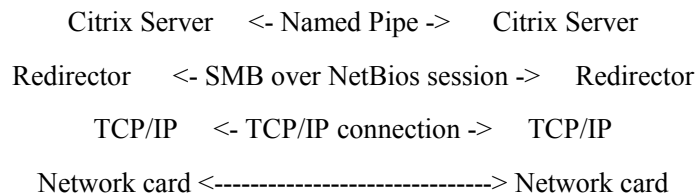


Steps 3 and 4

1. The PN service builds a list of servers it updates with its contribution to the available managed applications. The update is performed by named pipe connections.
2. When contacted with an updated managed application list, all MetaFrame servers respond to the initiating server with an update of their managed application lists.

When the PN service receives the list of available MetaFrame servers, it initiates communication to the \\MFServer\Citrix_NHWatch named pipe on each of the servers. The servers communicate their managed application lists through the named pipe. For the pipe connection to be allowed by Windows NT security, the pipe must be listed in the NullSessionPipes key of the LanmanServer services (HKEY_LOCAL_MACHINE\SYSTEM\Services\lanmanserver\parameters).

Named pipes use the NetBIOS protocol. Each NetBIOS session requires “KeepAlive” messages to flow between the redirectors responsible for maintaining the named pipes. NetBIOS sessions over TCP/IP use NetBIOS-to-IP address name resolution, requiring (in a routed environment) either LMHOSTS file entries for each MetaFrame server or a WINS server to provide this name resolution.



A typical message packet includes the following information:

```
[ IP | TCP [ NetBIOS [ SMB ( PN Managed Application Data) ] ] ]
```

Note: This is key to maintaining a reliable managed application list for Program Neighborhood Client users. If named pipe connectivity or NetBIOS resolution fail, the servers will not have an accurate managed applications list.

Optimizations

Program Neighborhood Service

Named pipe connectivity is essential for this process to function correctly. The following values can be modified to aid this connectivity:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ProgNeighborhood\Parameters

1. ConnectionRetries

The **ConnectionRetries** value is the number of times the PN service tries to connect (through named pipes) to a particular server. The default is three tries.

2. ConnectionSleepBetweenRetries

The **ConnectionSleepBetweenRetries** value is the number of seconds the PN service waits between retries. The default is five seconds.

3. SDKEY

The **SDKEY** (security descriptor) acts as the cache of the Windows NT authorized user and groups information for all of the published applications in a domain (or group of trusted domains). The information is cached in Security Descriptor format for fast retrieval and authentication when given a particular user's SID. Limiting the size of this value by using only groups (in the place of a long user list) can increase authentication time for PN Client updates.

NetBIOS and TCP/IP

NetBIOS and TCP/IP optimizations are included below.

1. Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\lanmanworkstation\parameters

Value: KeepConn REG_DWORD 1 to 65535 seconds

Default: 600 (10 minutes)

This value specifies the maximum amount of time that a connection can be left dormant. Changing KeepConn may generate significant SMB overhead.

2. Set the NetBIOS name resolution mode to m-node (broadcasts followed by name server) on all WAN servers.

This setting ensures that a local (for example; on the same subnet) domain controller is always contacted first for name resolution. The NetBIOS name resolution mode can be set to m-node with the modification of the following key:

Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters

Value: NodeType REG_DWORD 4 (4 is for m-node)

3. Make sure a backup domain controller (BDC) is at each WAN site.

To ensure the authentication requests from the MetaFrame servers do not go across the WAN, place the BDC's name in the LMHOST file on the MetaFrame servers. Specify the #PRE and #DOM parameters.

4. NetBIOS KeepAlives

Default Settings and Behavior: Any frame sent by NetBT (including a NetBIOS KeepAlive) is considered to be data by TCP. Because the NetBT KeepAlive parameter is set lower than TcpKeepCnt, one NetBT KeepAlive is sent per minute in an idle NetBIOS session. No TCP KeepAlives are sent. In an idle non-NetBIOS TCP session, such as that used by a Windows Sockets program, TCP KeepAlives are sent, with their frequency controlled by the TcpKeepCnt parameter.

For NetBT KeepAlive:

Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters

Add a value for the NetBT KeepAlive parameter as a REG_DWORD with the desired value in seconds. Zero (0) disables NetBT KeepAlives.

For TcpKeepCnt:

Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters

Add a value for the TcpKeepCnt parameter as a REG_DWORD with the desired value in seconds. Zero (0) disables TCP keepalives.

Defaults: NbtKeepAlive = 60 seconds
TcpKeepCnt = 120 seconds

All pertinent TCP/IP parameters are values located under:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:

Tcpip\Parameters

The values are:

- **KeepAliveInterval**

Key: Tcpip\Parameters

Value Type: REG_DWORD – Time in milliseconds

Valid Range: 1 – 0xFFFFFFFF

Default: 1000 (one second)

This parameter determines the interval separating KeepAlive retransmissions until a response is received. After a response is received, the delay until the next KeepAlive transmission is again controlled by the value of KeepAliveTime. The connection is aborted after the number of retransmissions specified by TcpMaxDataRetransmissions have gone unanswered.

- **KeepAliveTime**

Key: Tcpip\Parameters

Value Type: REG_DWORD – Time in milliseconds

Valid Range: 1 – 0xFFFFFFFF

Default: 7,200,000 (two hours)

This parameter controls how often TCP attempts to verify that an idle connection is still intact by sending a KeepAlive packet. If the remote system is still functioning and reachable, it will acknowledge the KeepAlive transmission. KeepAlive packets are not sent by default. This feature can be enabled on a connection by an application.

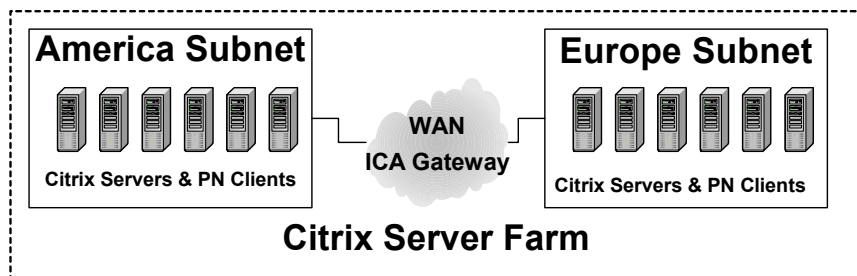
Citrix Server Farm Limitations

Performance Issues Due to WAN and Load Balancing Issues over the WAN

ICA gateways produce a large amount of traffic on the WAN. A MetaFrame server farm with a high amount of activity (logons and logoffs) increases this traffic. Increasing the update time in the registry allows more up-to-date data to be transmitted to the master browsers across a WAN. However, this causes additional traffic as well.

Functionality Limitation

Program Neighborhood Clients make a load-balanced connection to the least busy server in the farm for authentication and to obtain a list of managed applications. If the least busy server is located on the other side of the WAN, the client must traverse the WAN to get this information. When the client receives its managed application list, it is possible it could be load balanced to an application located across the WAN, even though that same application is published on the local LAN servers.



In this scenario, two subnets are separated by a long distance. The time difference between Europe and America could cause the European servers to show up as the least busy servers for a load balance query initiated during the American workday. Even though the same applications reside on the American servers, you could get load balanced to a European server.

Built-in Tools for Troubleshooting Program Neighborhood Service Problems

The most common problems occur when PN Client users do not have the correct Program Neighborhood application icons or when they have an incorrect list of the icons.

Three command line tools are available for troubleshooting PN service problems. These are:

1. **qserver /serverfarm**

Qserver /serverfarm lists all servers in a particular server farm.

2. **qserver /resetserverfarm <farm name>**

Qserver /resetfarm *farm name* forces every server in the specified farm name to stop and start its Program Neighborhood service, therefore causing a re-enumeration of all servers and applications in the farm.

3. `qservr /debugnwatch <servername>`

The third tool, `qserver /debugnwatch servername` is by far the most useful troubleshooting tool. It lists the following data:


```
C:\>qserver /debugnwatch "ServerName1"
Calling _DebugProgramNeighborhoodService()
    Client Protocol Detected
    Client Protocol Version will be 1
    I thought Peer name would be "ServerName1"
    Peer name is "ServerName1"
    Service debug initiated on : "ServerName1"
```

```
Debugging Program Neighborhood Service in farm "FarmName"
    There are 0 PN clients connected to this service.
    There are 3 other known servers in this farm:
    This service knows about 6 managed applications on this farm:
    1 : "Application 1" (Servers: "ServerName1, ServerName2")
    2 : "Application 2" (Servers: "ServerName1")
    3 : "Application 3" (Servers: "ServerName1")
    4 : "Application 4" (Servers: "ServerName1, ServerName3")
    5 : "Application 5" (Servers: "ServerName1")
    6 : "Application 6" (Servers: "ServerName1")
```

Three servers are in this server farm (called **FarmName**). This particular server knows about six available managed applications. **ServerName1** has all six applications published on it. **ServerName2** has Application 1 published, and **ServerName3** has Application 4 published. When `qservr /debugnwatch "ServerName"` is executed for each of these servers, they all provide the same output; that is, three servers and six managed applications available in the farm.

This becomes important when a PN Client requests an update of managed applications. The PN Client's request is sent to the master browser. The following events occur:

1. The PN Client requests a server from the master browser (this request is *load balanced* to any available server in the farm).
2. The client initiates a connection to this farm server. This is a partial ICA connection but a complete authentication to the MetaFrame server (therefore local logon rights must be given on all farm servers).
3. The update is completed and a "filtered" list of applications is displayed for the PN Client user.



If the PN client requests an update from a server that has an incorrect managed application list, that PN Client will have an incorrect “filtered” managed application list.

When ICA gateways are involved, it is important to check the output of **qserver**, which enumerates all servers from all ICA gateway subnets. Additionally, UDP port 1604 must be open for ICA gateways.

NetBIOS resolution is important and can be checked by testing whether or not Microsoft’s Network Neighborhood can resolve MetaFrame servers. You can also type **net view** or **net use** at a command prompt to determine if the MetaFrame servers are included in the list that appears.

PN Client resolution and connectivity to MetaFrame farm servers can be aided by placing a server’s IP address in the client’s server locator entry.

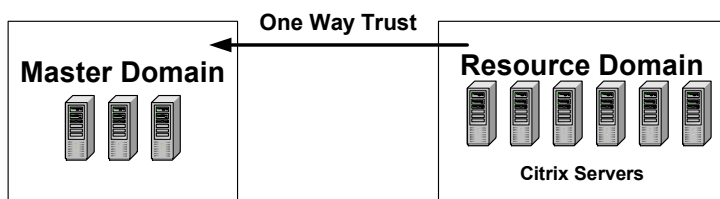
Appendix A

Multiple Domain MetaFrame Server Farm Functionality

Multiple domain MetaFrame server farms function correctly if you follow the rules below:

1. All domains must trust a common domain (example: the master/resource domain model).
2. Users must be able to access all domains in the server farm. The key to this rule is that all farm servers must be accessible to the clients because all servers in a farm have the capability of providing the PN Client with updated managed application data. This update requires a partial logon (local logon rights) to the updating server.

The illustration below demonstrates these rules. This functions correctly if all farm servers are in the Resource domain. However, if farm servers were part of the Master domain, you would need a two way trust.



When a user logs on to a farm, the user's security token is requested from the domain controller. This token is used to check the user's access rights to managed applications. This is performed by comparing the access token to each managed application's security descriptor. The Program Neighborhood services on the MetaFrame servers synchronize this data (the security descriptors of each server's managed applications) at startup and when changes are made to the server farm. Therefore, each server has a complete replicated database of each other server's managed application rights. When a PN Client requests a refresh, another token is created and you are prompted to log on again.

Domain Scenario

When a domain scenario is involved, you must consider the authentication process as well. When a client attempts to authenticate to Program Neighborhood, the following occurs:

1. A user opens the Program Neighborhood Client.
2. The server to which the client connects needs to authenticate this user against the domain controller.
3. The server sends a session setup SMB to the domain controller. This contains the Logon domain, User Account, and Password. With this computer name, Netlogon uses Remote Procedure Call (RPC) to forward the credentials to the Netlogon service on a domain controller
4. The domain controller checks its SAM database and authenticates the user when the correct logon credentials are passed.
5. The domain controller sends the access token to the MetaFrame server.



851 W. Cypress Creek Road Fort Lauderdale, FL 33309 954-267-3000



<http://www.citrix.com>