Inside Citrix chapter eighteen – The one with all the FMA core services

The FlexCast Management Architecture is the true foundation of Citrix's applications and desktop (virtualisation) platforms and probably will be for many years to come.

It has always been the next generation architecture for VDI, at least from Citrix's point of view, and it has evolved over the years to now also include and support XenApp / RDSH workloads: in fact we are up to version 7.8 of XenDesktop and XenApp already. It is robust, scalable and flexible at the same time and of course it (almost) goes without saying that its main goal is to securely manage and deliver or broker applications and desktops to our users all over the world from every device thinkable.

The FMA takes care of itself. Think about this. Most people have no idea how many services and components are involved when installing and configuring XenDesktop for example, and still they are capable of running huge environments without to much trouble. Often consisting of thousands of desktops and users.

The FlexCast Management Architecture is built up around thriteen core services. The FMA, like other platforms and technologies, is constantly being improved and enhanced and as such is undergoing constant change.

As a result, if we look back over the past five to six years or so we can see that the FMA evolved from six services back in 2010 up to thirteen services today, an impressive ride to say the least. However, the problem that we often see with rapid evolving platforms, is that over time as features and functionalities are being added they become much harder to manage and maintain.

That is why the FMA has been designed using a set of clearly defined standards where new functionality can be introduced by adding new services, or build on top of existing ones if needed.

The Delivery Controller is often referred to as the heart of the FMA: this is mainly because the mentioned services, all thirteen of them, live or reside on your Delivery Controllers, making it an extremely important component. Next to these thirteen services, which we will have a closer look at in just a minute, both VDAs (desktop and server) also host several services, which communicate with your Delivery Controllers as well. These will be discussed in more detail as we progress throughout this chapter.

Internal communication

The release of XenApp/XenDesktop version 7.12 introduced couple of new FMA services (primarily used by LHC). As you might be aware, I have written multiple articles on the FlexCast Management Architecture in the past, talking about its core services, their responsibilities, capabilities, communication channels/interfaces and so on.

Throughout the past two years I also came up with a nice graphical overview (at least I like to think so) representing a Delivery Controller including all main FMA services. This chapter is

meant to provide you with an update on the FMA and its primary services, graphical overview included.

Although FMA services run completely isolated from each other, internal communications between the different services takes place using so-called WCF (Windows Communication Foundation) end points (also referred to as service interfaces) over port 80 by default. Though, here I would like to note that port 80 can be changed into any port number you might prefer and that encryption is supported as well.

The endpoint address is represented by the EndpointAddress class, which contains a Uniform Resource Identifier (URI) that represents the address of the service (a secure identity including a collection of optional Headers). Each service has the ability to interact with all other services in order to carry out important tasks and actions throughout the FMA. As a best practice, you will deploy two or more Delivery Controllers, avoiding a single point of failure and providing scalability where and when needed.

As we will shortly see, there are four services that have a special place within the FMA and take on a more prominent role than the others (a.k.a. core services). These are the Broker, Configuration, Delegated Administration and the Configuration Logging service.

> **FMA fact**: While all services closely interact with and depend on each other, at the same time they are also completely separated from each other. Each service is configured to communicate to the Central Site database using its own individual DB connection string. This way, if one service fails it will not affect any the other services.

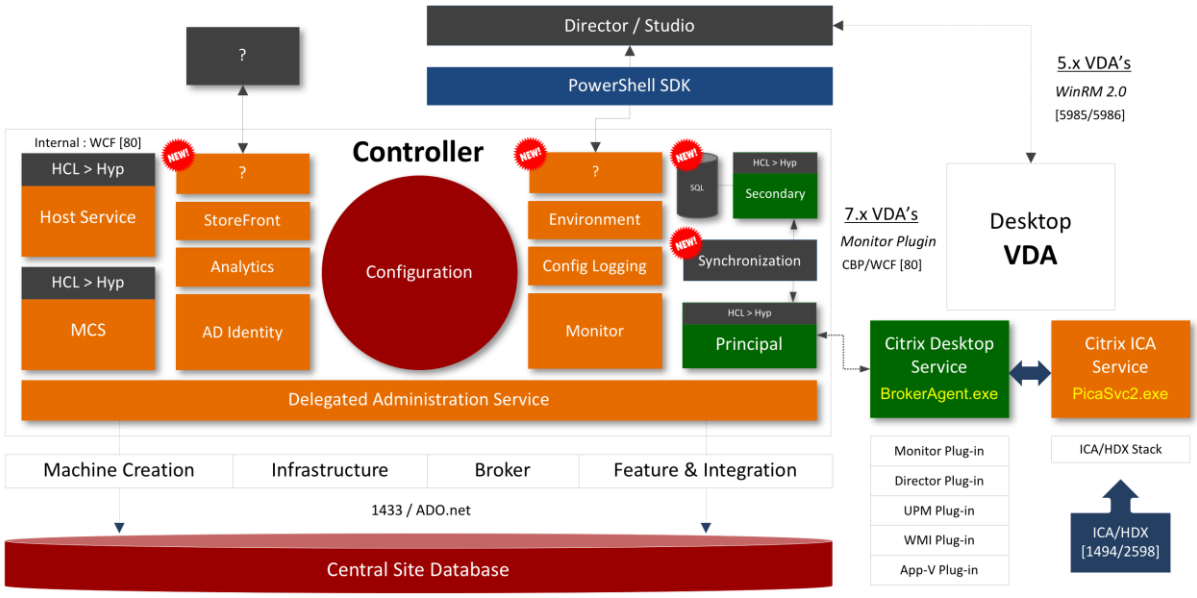
One of the biggest differences between the IMA and FMA is that with the FMA the Delivery Controller is only responsible for managing and brokering connections to managed as well as unmanaged VDAs installed on your server and desktop machines. It doesn't host any sessions of its own.

Also, all the ICA / HDX bits, bytes and related services (controlling the HDX functionality throughout a session) reside on the VDA itself, again relieving the Delivery Controller from any additional tasks. Because of this approach, it is also easier to maintain the code and different operating systems (server and desktop) can be deployed within the same site.

> **FMA fact**: Keep in mind that if you change something for one specific service, like the DB connection string, for example you must do this for all the other FMA services as well

The FMA's thirteen, and then some

Let's go over each of the services one at the time. Where applicable I'll elaborate a bit more on the inner workings, any special considerations and/or relations to other services like the XML / STA Service, for example.



> **FMA fact**: All FMA services run under the NT AUTHORITY\Network service account. Also, when authenticating to the Central Site database (this is where the Configuration Service plays an important role as well) all services use the local computer account of the machine that they are currently running on.

1. Broker Service

The broker (XML) Service is probably the best-known one. By the way, it's indicated as Principal and Secondary (in green) on the image above, with the Synchronisation Service in between. From a Delivery Controller point of view, it brokers new and manages existing sessions, handles resource enumeration, the creation and verification of STA tickets, user validation, disconnected sessions, and more (see the list under 1.1 as well). From a VDA point of view it takes care of all communication to and from the Delivery Controller. It does this by communicating with the VDA Citrix Desktop Service, a.k.a. BrokerAgent.exe, which is part of the desktop as well as server VDA.

Note that the STA (Service) is also part of the Broker Service, and has been as of Presentation Server 4.0. Before that it was written as an ISAPI extension for Microsoft Internet Information Services, or IIS. As of XenDesktop 4.x the XML service (ctxxmlss.exe) has been rewritten in .NET and became part of the Broker Service as well. T

he Broker Service is built up of three separate services (or four if you also count the newly introduced Principal Broker Service, see below) all handling different tasks: it brokers connections, it enumerates resources, it can take care of authentication (token) and it acts as the Secure Ticket Authority, generating and validating STA tickets; however, this only applies to resources launched through a NetScaler, and last but not least, as of version 7.12 it is also involved in managing the Local Host Cache.

As of XenApp/XenDesktop version 7.12 the Local Host Cache (LHC) got re-introduced (though it's new to the FMA). One of the main LHC components is the Broker Service, however, when the LHC is involved it is also referred to as the Principal Broker Service (as you will find out shortly there is also a High Availability Service a.k.a. the Secondary Broker Service).

Think of it as a sub-service, just like the XML Service, for example. The Principal Broker Service will accept connection requests from StoreFront and it communicates with the Central Site Database just like before — brokering connections, taking care of load balancing and so on, while it also (continuously) interacts with the Configuration Synchronisation Service as well as the High Availability Service when the LHC becomes active.

1.1 Broker Service main responsibilities

As mentioned, the Broker Service has some huge responsibilities within the FMA. Besides some of the tasks already highlighted, one of its most important tasks is the registration of all VDAs, including ongoing management from a Delivery Controller perspective. To give you a complete overview of the main tasks and responsibilities of the Broker Service, have a look below:

1. As already mentioned it takes care of VDA registration, resource allocation, connection brokering, licensing enforcement, amongst other tasks.
2. During the initial user logon process it will validate the end-user's identity, based on credentials received through StoreFront.
3. Based on your configuration, it can take care of XML based user authentication (token creation).
4. Functions as the Principal Broker Service when LHC is enabled.
5. It continuously interacts with Configuration Synchronisation Service.
6. It is involved in HDX policy management.
7. It manages the overall power state of desktops and server machines when run virtually, starting and stopping VMs based on usage and on administrator configuration, including idle pool management.
8. It temporarily stores user credentials to allow users to be logged into virtual desktops without having to re-enter credentials (single sign-on).
9. It keeps track of virtual desktop state, based on information received from virtual desktops. As such, it will take appropriate action when needed.
10. It will participate in the initial load-balancing process, deciding which desktop or server to connect to. This information will eventually end up in an ICA launch file.
11. Administrators will use the Broker Service, although they may not actually know it to log off sessions, define Machine Catalogs and publish virtual desktops based on computer identity.
12. It exposes the Hypervisor state and alert information.
13. It will also handle all power management features, including but not limited to: power policy rules, reboot schedules and cycles, pool/buffer size management, and remote PC wake on LAN.

1.2 The Broker Service Site services

The Broker Service is somewhat special in that it also houses multiple so-called Site Services, let me explain. Site services provide Site-wide maintenance and housekeeping functionalities within and a XenDesktop Site. They take care of things like managing connections to your Host Connections, checking up on session idle times, managing reboot schedules, cache maintenance (refresh) and more.

Before I go any further you need to know that there are eighteen Site services in total, with each having its own responsibility, or multiple in some cases and that they are part the Broker Service. More specifically, each individual Site service will only run on one of your Delivery Controllers (within a Broker Service), creating a distributed model. As soon as a Delivery Controller misses a *heartbeat with the Central Site database, all Site services running on that Delivery Controller will be transferred to one of the other active and still considered healthy Delivery Controllers. Again, they will be moved from one Broker Service to another.

*A heartbeat message is exchanged between a Delivery Controller and the database every 20 seconds with a default timeout of 40 seconds.

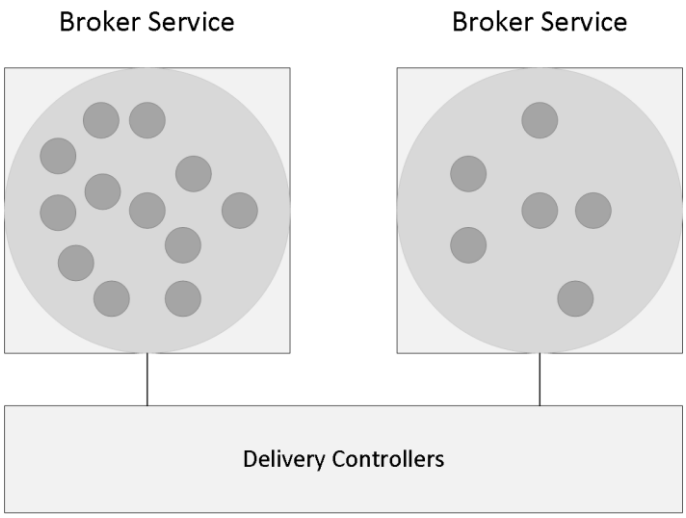
> **FMA fact**: While it is considered a best practice to keep all Delivery Controllers equally configured, Site services are the exception to the rule, so to speak.

At runtime, when a Delivery Controller becomes active the Site services will automatically be divided between all active Delivery Controllers within your Site. Do note that although you as an Administrator can assign certain Site services to a Delivery Controller if you want or need to – this is not a recommended or supported approach. The election mechanism is controlled by the contents of the Central Site database, the FMA will take care of this for you.

The eighteen Site services are:

1. ControllerReaper.
2. ControllerNameCacheRefresh.
3. Licensing.
4. BrokerReaper.
5. RegistrationHardening.
6. WorkerNameCacheRefresh.
7. AccountNameCacheRefresh.
8. PowerPolicy.
9. GroupUsage.
10. AddressNameResolver.
11. RebootScheduleManager.
12. RebootCycleManager.
13. ScopeNamesRefresh.
14. FeatureChecks.
15. RemotePC.
16. IdleSessionManager.

17. LeaseReaper.
18. Hypervisor connection.



Broker Service Site Services

Hypervisor Connection Site service

This one is different from the other Site services. The Hypervisor Connection Site service is the only service which can exist on multiple Delivery Controllers within the same Site and can be controlled, or configured using the PowerShell SDK. As the name implies it manages your Host Connection configured through Studio. It allows you to configure which Delivery Controller manages and takes ownership of a specific Host Connection, or multiple when desired.

Multiple controllers can host and manage multiple Host Connections, either assigned by you (through the Hypervisor Connection Site service) or they can be automatically load balanced over your Delivery Controllers, which will happen by default, again, the FMA is that smart. Technically you could also assign all of your Host Connections to be managed and owned by a single Delivery Controller, which might be better equipped for the task at hand, compute and resource-wise. However, you might want to check with Citrix if this is a supported setup.

2. The Citrix High Availability Service A.k.a. Secondary Broker Service

I already briefly mentioned the Secondary Broker Service/High Availability Service while referring to the Principal Broker Service a couple of paragraphs back. Together with the Configuration Synchronisation Service (see below) all three services reside on every Delivery Controller in your environment — assuming you are running version 7.12 or later.

When an outage occurs, the Principal Broker Service will no longer be able to communicate with the Central Site Database, as a result it will stop listening for any incoming StoreFront and/or VDA information and it will instruct the High Availability Service/Secondary Broker Service to start listening for incoming connection requests and handle them accordingly.

As soon as a VDA communicates with the High Availability Service/Secondary Broker Service a VDA re-registration will be triggered. This way the High Availability Service/Secondary Broker Service will receive the most current session information related to that specific VDA (who is connected to which machine, for example). In the meantime, while the High Availability Service/Secondary Broker Service is handling new and existing connections/sessions, the (Principal) Broker Service will continue to monitor the connection to the Central Site Database.

As soon as it notices that the connection to the Central Site Database has been restored it will instruct the High Availability Service/Secondary Broker Service to stop listening for, and handle new and existing connections/sessions. From this point on it will resume brokering operations as before, basically repeating the abovementioned steps of VDA registration to get up to speed with the latest connection/session information.

Finally, the High Availability Service/Secondary Broker Service will remove any remaining VDA registrations and will again continue to update the local SQL Express database (together with the help of the CSS) with any configuration changes from that point on, as highlighted before.

3. Configuration Synchronisation Service (LHC)

Every two minutes the Principal Broker Service will be checked for configuration changes. If a configuration change has been detected it will be copied over, or synchronised to the High Availability Service/Secondary Broker Service. This is the primary task of the Configuration Synchronisation Service (CSS). These configuration changes include but are not limited to published icons, changes to Delivery Groups and Catalogs, certain Citrix policies and so on. It will not include information about who is connected to which server (Load Balancing), using what application (s) etc. this is referred to as the current state of the Site/Farm, which is not considered a configuration change.

All (synchronised) data is stored in a Microsoft SQL Server Express LocalDB database (notice the LocalDB part, this is different from SQL Express), which resides on the same Delivery Controller. In fact, each time new information is copied over, the database will be re-created entirely. This way the CSS can, and will ensure that all configuration data stored in the Central Site Database will match that of the data stored in the local SQL Express database keeping the LHC current.

As a second prime responsibility, the Configuration Synchroniser Service will provide the High Availability Service/Secondary Broker Service (s) with information on all other controllers within your Site (Primary Zone), including any additional Zones you might have configured. This way each High Availability Service/Secondary Broker Service will know about all the other available High Availability Services/Secondary Broker Services within your (entire) Site.

Communication between the various High Availability Services/Secondary Broker Services takes place over a separate channel based on an alphabetical list containing the FQDN's of the machines they (the services) currently run. This information is used to elect which High Availability Service/Secondary Broker Service (read: Delivery Controller) will take over within that specific Zone when the LHC becomes active because of a DB failure or another outage of some sort.

4. Configuration service

All FMA services need to register with the Configuration Service on start-up so that it knows they are all good to go. This is one of the main reasons why the Configuration Service has such a prominent role: it handles all inter-service communication within the FMA. The Configuration Service is the glue holding the FMA together.

Located at the centre of the FlexCast Management Architecture, it holds and manages a list of all FMA services, allowing them to advertise their WCF addresses, or end points (service interfaces), including the functionality that they provide. Only after a service successfully registers with the Configuration Service, when adding in more Controllers, after a reboot or during Site creation, for example, will it become active and able to communicate with other FMA services.

Once all services have successfully registered themselves, the Configuration Service will share a listing of all active and registered services as being active Site members, including their main responsibilities, or capabilities and service (communication) interfaces.

As soon as an individual FMA service needs to communicate with one of the other FMA services it will first (need to) contact the Configuration Service to get a copy of the services listing mentioned earlier.

After an FMA service successfully queries the Configuration Service and the listing has been received, this information will be cached for five minutes. This is mainly to ensure that the system isn't being overwhelmed with service listing requests, preventing the Configuration Service from becoming a potential bottleneck. At this point the requesting service knows where to find the other services, what they are capable of (responsibilities) and how to communicate with them (end points/service interfaces).

As a side note, the Machine Creation Service and the Machine Identity Service both communicate through the Host Service to find out about the configuration and connections of the underlying Hypervisor/Cloud connection (Host Connections) if any, including the storage and network configurations needed for virtual machine provisioning. This information will be cached for one minute as opposed to the five minutes mentioned earlier.

> **FMA fact:** Each FMA service can query the Configuration Service to look up other services using the listing mentioned earlier. In short, service registration and communication are both reliant on the Configuration Service. It will also store configuration metadata for all services, relieving Active Directory.

4.1 Permissions

The Configuration Service directory stores the Active Directory machine account identifier (SID) for each service that has successfully registered with it. At the same time, this information will also be stored in the Central Site database where it will be accessible to all Delivery Controllers including the services that they host. Only when the machine SID of the

accompanying FMA service is listed, and known by the Configuration Service will communication between FMA services be possible.

When a service with an unregistered machine account contacts the Configuration Service for the services listing it will receive an access denied. The only exception to this is the 'Network service' account: it is always allowed. Viewing and validation the successful registration of FMA services can be done through the PowerShell SDK. Use the following syntax:

Get-ConfigRegisteredServiceInstance -InterfaceType sdk | select serviceaccount, interfacetype, servicetype | format-table

> **FMA fact:** If you would like to refresh the cache of one of the FMA services (remember the five minutes), all you have to do is restart the accompanying Windows Service. The cache (services listing) is retrieved during service start-up.

5. Configuration Logging Service

Monitors and logs all configuration changes made within a XenDesktop Site, including all Administrator activity. Depending on its configuration, no Site changes are possible when its database is unreachable, making it one of the four core services as mentioned earlier. The data itself can be stored within the Central Site Database or a separate database can be created, which would be the recommended approach. As of XenDesktop version 7.7 a separate location / database can be selected during the initial installation configuration process.

6. Delegated Administration Service

The Delegated Administration Service is also considered to be one of the FMA's more critical services. All other FMA services will need to communicate with the Delegated Administration Service to validate if they have all the proper permissions and/or rights needed to make the necessary changes to the Central Site Database. Next to this, it manages the configuration and administration of all delegated administrative permissions. Thus, if this service becomes unresponsive or unavailable site-wide configuration changes will not be possible.

7. AD Identity Service

Handles all Active Directory computer accounts (identities) related to XenApp / XenDesktop virtual and physical machines.

8. Machine Creation Services

Handles the creation of new virtual machines. When this service is unavailable no additional virtual machines can be created, at least not using MCS. Also note that MCS is only capable of provisioning/creating virtual machines, not physical. If you want to be able to 'service' physical machines you will need to use Provisioning Services.

> **FMA fact:** If you do not configure a Host Connection within Studio, when creating a new Device Catalog, the option to use MCS as a provisioning mechanism will not be available (greyed out).

9. Host Service

Manages all connections between the physical hosts, the Delivery Controllers and the underlying Hypervisor (s) or Cloud platform (s), both referred to as Host Connections. As far as the Hypervisor goes this can be either vSphere, XenServer, Hyper-V or the Nutanix Acropolis Hypervisor. This is where your virtual server and/or desktop VMs live. Physical machines are still optional as well, but again you'll use PVS instead of MCS.

The following Cloud platforms are supported: Amazon Web Service, Microsoft Azure and CloudPlatform. As highlighted earlier, the Host Service is also responsible for discovering and managing the connections and configurations of the Hypervisor, network and storage that are required and used by machine provisioning operations. It does the same for any cloud platforms/connections you may have set up.

9.1 Hypervisor Communications Library

The HCL is used by several FMA services, the Broker, Host and MCS to be a bit more precise to provide an abstract Application Programming Interface, or API for interacting with the underlying Host Connection, or multiple. This will ensure a consistent and consequent representation of the configured Host Connection including network and storage resources.

As an example, while multiple Hypervisors are supported this also adds to the complexity of the code, this is where the HCL comes in, it functions as an abstraction layer. As a result, when a new version of a Hypervisor is released or an existing one needs to altered, Citrix can quickly add support without the need to replace the code in multiple places. Again, the same applies to Cloud platforms as well.

10. Environment Test Service

Takes care of all Site-wide tests, initiated from Studio. You can run tests on your Delivery Groups, Machine Catalogs or even on your entire Site configuration, and more. New tests are introduced with every new version of XenDesktop/XenApp going forward.

11. Monitor Service

Monitors the overall FMA architecture and produces alerts and warnings when it finds something potentially wrong. These will pop up in Studio or Director. Note, however, that, although these alerts will tell us that something is potentially wrong, they won't tell us what is wrong or where to look for answers. Therefore, FMA services are best checked/monitored using PowerShell code typed in directly from a PowerShell Command Prompt.

For example, try using the Get-BrokerServiceStatus and/or Get-ConfigServiceStatus cmdlets to view the status of the Broker and Configuration service. There is a 'Get-' command for each FMA service.

12. StoreFront Service

This takes care of your StoreFront deployment, which can be added as well as managed directly from Studio. Note that your StoreFront can and probably will appear twice within Studio. Once under the console root as an integral part of Studio (it will be there by default) which is needed to be able to configure the Citrix Receiver as part of your published hosted shared desktop (s) images directly from a Delivery Group. And a second time (as a separate node) enabling you to manage your StoreFront deployment as if you were using the separate MMC based StoreFront management console. The latter gives you the option (s) to change its look and feel, authentication methods, adding and removing Stores, and more.

13. Analytics Service

The Analytics Service does as the name implies: it collects analytical data used by Director/Studio (custom reports, to name one). It is also leveraged by the Citrix Customer Experience Improvement Program (CEIP), which will be enabled by default — this applies to the Citrix Call Home functionality as well by the way. All data will be shared anonymously, encrypted and, as always, will be used for the greater good.

More to come

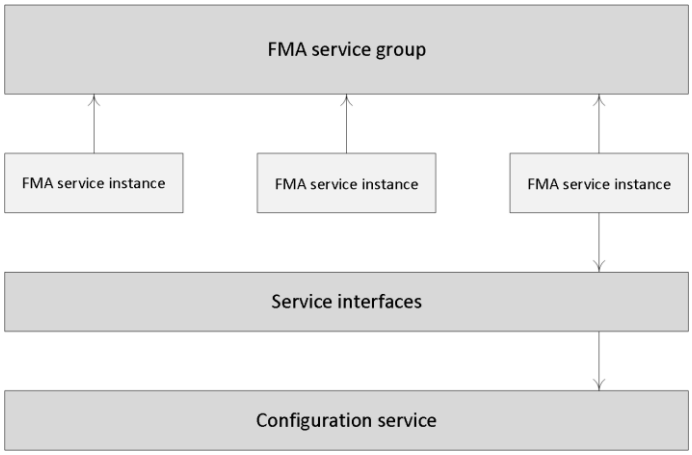
As you can see on the graphical overview, more services are on their way. Unfortunately, I can't share any additional information as all this is still under NDA. Just know that the services are already there, just not activated, usable and/or viewable.

Making the FMA services highly available

As we've established earlier, a XenDesktop Site consists of thirteen main services and each individual service, also referred to as a service instance, will run on every active Delivery Controller within your Site.

As a way to make these FMA services highly available, each service will need to register itself with a so-called peer service group, which contains all registered service instances of the same type. For example, if you have four Delivery Controllers you will also have four Broker services, of which only one will be active at a time.

All four Broker services will register themselves with the above mentioned service group (the Broker service, service group), and they will do so at startup. After registration is successful this information is written to, and stored in the Central Site Database so that all active Delivery Controllers including their services will have access to this information whenever needed. This way, if the active Broker service fails for some reason, one of the other Broker services can and will take over. This basically means that there will be a service group for every main FMA service, all thirteen of them, the same principle applies.



FMA Services high availability

> **FMA fact**: Each service group has a unique identifier, which can be queried using the PowerShell SDK if and when needed.

The Desktop VDA

As mentioned in the 'The Virtual Delivery Agent' chapter, the VDA is a relatively small piece of software that gets installed on all virtual and physical machines running a Windows server and/or desktop operating system within a XenDesktop Site (note that there is also a Linux based VDA). It serves multiple purposes, as we will find out shortly, but only after a VDA is able to successfully register itself with one of your Delivery Controllers.

VDA registration

As soon as a Virtual Delivery Agent starts up, meaning the desktop or server Operating System boots, it will try and register itself with one of the Delivery Controllers known within the Site. For this to happen there needs to be a mechanism in place that tells the VDAs which Delivery Controllers are part of the same Site and how they can contact or reach them. For this, Citrix introduced the 'auto-update' feature, which will be enabled by default. It will keep all VDAs updated when Delivery Controllers are added or removed (go offline) from the Site. Each VDA maintains a persistent storage location to store this information. Also see the section named 'Desktop VDA ICA stack services' over at page 286 for some more detailed information regarding the VDA registration process.

When the auto-update feature is disabled, or does not supply the correct information, the VDA will check the following locations (in this order):

- Through configured policies.
- The ListOfDDCs Registry Key.
- OU-based discovery (legacy).
- The Personality.ini file created by MCS.

If a VDA is unable to register itself with a Delivery Controller or communication between the VDA and the Delivery Controller fails for any reason, you will not be able to connect to it.

Here I would also like to note that VDA registration is the Nr. one issue reported over at Citrix support, and as such deserves some extra attention. Make sure to also have a look at the 'Troubleshooting the FMA' chapter for some more details around the potential troubleshooting steps involved when dealing with VDA registration issues.

This is also where the Citrix Desktop service, as part of installed VDA plays an important role. The Desktop service communicates directly with the Broker service over at the Delivery Controller and takes care of the initial VDA registration process through the Connection Brokering Protocol (CBP). The CBP is a collection of WCF (Windows Communication Foundation) end points defined to exchange information and handle the registration process.

> **FMA fact**: Restarting the Citrix Desktop service on the VDA triggers the registration process and can be used to force re-registration when needed.

Launching a VDI desktop

Let's have a closer look and see what happens when a VDI-based virtual machine is launched from a trusted, internal network. The XML service will again play an important role throughout the whole process.

- Let's assume that the VM is pre-subscribed and already present on the user's (StoreFront) home screen. Here it does not matter how we are connected: using a locally installed Receiver or using the Receiver for Web sites.
- After the user clicks the desktop icon the StoreFront server will contact the Broker (XML/STA) service to check if any registered VDAs are available. It (the Broker service) does this by communicating with underlying Hypervisor platform through the Host service on the Delivery Controller.
- If needed it will first start / boot a VM. It's not uncommon to pre-boot a few VMs, since, as you can probably imagine, this will positively influence the overall user experience. Understanding the usage patterns of your users allows you to boot enough machines before they're needed.
- In between the VDA will register itself with the Delivery Controller handling the initial request. It will do so by leveraging the Connection Brokering Protocol (CBP) and communicating with the Broker service over at the Delivery Controller, as highlighted earlier.
- Next the Delivery Controller, or Broker (XML/STA) service, will contact one of the VDAs and send a StartListening request. By default, the VDA isn't listening for any new connections on port 494 or 2595 until it gets notified that a user wants to connect.
- As soon as the VDA is listening, the Broker (XML/STA) service will send this information back to the StoreFront server in the form of an XML file.
- Based on this information, the StoreFront server will then generate a launch.ica file (it uses the default.ica file as a template) containing the IP address of the VDA and a whole

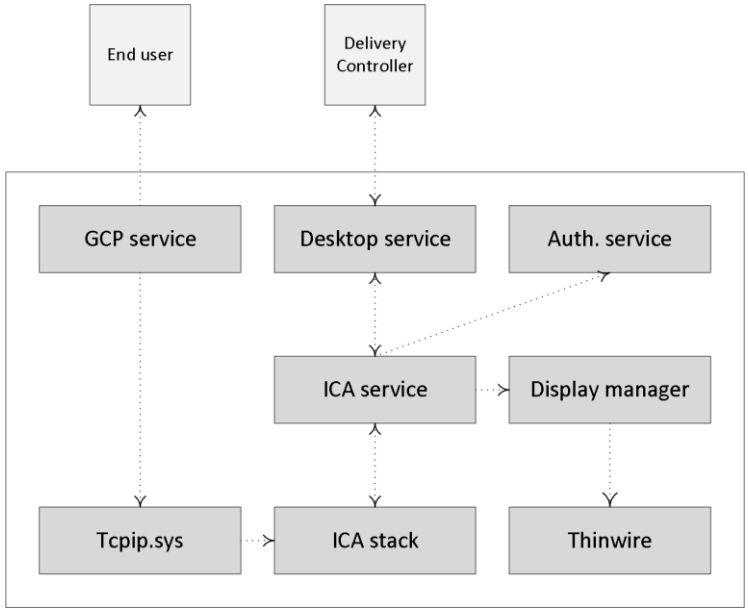
bunch of other connection properties that are or might be needed. This is sent down to the user.

- The locally installed Receiver (or HTML5-based Receiver) will read and autolaunch the launch.ica file, initiating a direct connection from the user's end point to the VDA.
- The installed VDA will verify its license file with the Delivery Controller.
- The Delivery Controller checks with the Citrix License Server to verify that the end-user has a valid ticket. A big change when compared to the IMA where every Session Host would communicate with the license server.
- At this time any applicable session policies will be passed on to the VDA and the session is launched.

What happens inside the VDA

Let's take it one step further and see what happens inside the VDA during launch time. The process below assumes that Session Reliability is enabled and that a desktop OS VDA gets launched as we've seen in the previous section. Remember that as of XenDesktop 7 there is also a server OS-based VDA, which we will have a closer look at in the next section.

- The CGP service will receive the connection and sends this information on to the tcpip.sys, which will forward it to the ICA stack.
- The ICA stack will notify the ICA Service a.k.a. the PortICA service (picaSvc) that a connection has been made after which the picaSvc will accept the connection.
- Then the ICA Service will lock the workstation because the user needs to be authenticated to ensure that the user is allowed access to that particular machine.
- As soon as the user logs onto the workstation, the PortICA service will communicate with the display manager to change the display mode to remote ICA, this request will be forwarded to the ThinWire driver.
- In the meantime, the PortICA service will hand over the 'pre-logon' ticket data, which it received from the ICA stack, up to the Desktop service and from there back to the Delivery Controller in exchange for 'real' credentials.
- The Desktop service receives the user's credentials, which are sent back to the PortICA service.
- The PortICA service contacts the authentication service to log on the user.



VDA: what happens during launch

Desktop VDA ICA stack services

If we have a closer at the Desktop VDA (check the main overview image) we can see that it consists of multiple FMA core services and plug-ins. The two main services are the Citrix Desktop Services a.k.a. BrokerAgent.exe and the Citrix ICA Service a.k.a. PicaSvc2.exe; together they control all HDX functionality for the duration of the session.

The Citrix Desktop service

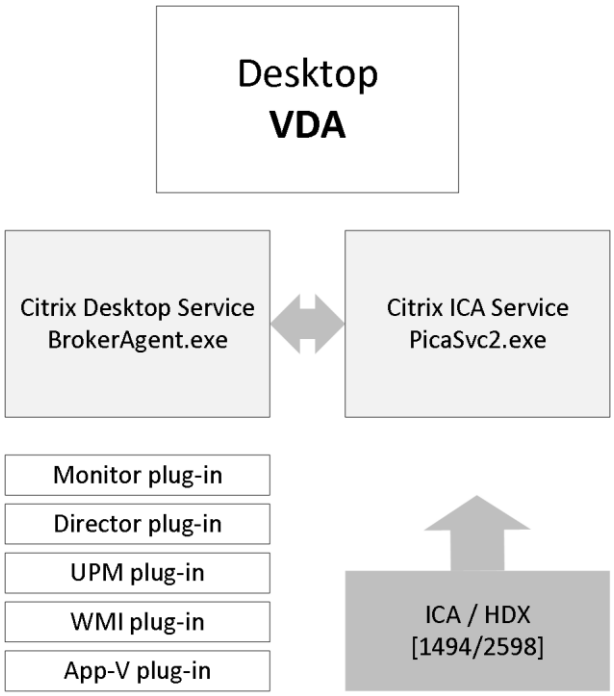
The Citrix Desktop service communicates directly with the Broker service over at the Delivery Controller and, as highlighted earlier, takes care of the initial VDA registration process through the Connection Brokering Protocol (CBP), which is a collection of WCF (Windows Communication Foundation) end points defined to exchange information and handle the registration process. Once the VDA has been registered successfully the Citrix Desktop service will continue to communicate regularly with the Broker service on the Delivery Controller, also including a number of different feature plug-ins.

The Citrix ICA service

The Citrix ICA service implements the actual ICA protocol bits and bytes within the VDA, or the biggest part of it anyway. It will receive instructions through the Citrix Desktop service protocol coming from the Broker service over at the Delivery Controller. As soon as it is notified of a new connection request, and after AD authentication has taken place, and ticketing, licensing and HDX policy information has been successfully exchanged, the ICA stack will start listening for incoming connections allowing the launch request to complete.

Both Director and Studio also communicate with the underlying FMA services through WCF over port 80. Here it is also worth noting that when working with older VDAs (5.x and below) these will require WinRM listening to be enabled on port 5985 or 5986 to be able to

communicate with Director and Studio. With 7.x VDAs this information is provided by the monitor plugin through the CBP protocol highlighted earlier.



Desktop VDA services

The Server VDA

Ever since the introduction of XenDesktop 7, where the FMA took over and XenApp was integrated, a lot has been written with regards to its components, services, agents and so on. What surprises me though, is that the Server VDA is (almost) never mentioned, while this is, or at least was a brand-new component. Never before was it optional to install a (relatively) lightweight agent onto a XenApp server, it was basically all or nothing.

Although the (new(ish) FMA based Server VDA has been built from the ground up it still has a lot of similarities when compared to the 'old' ICA protocol stack deployed with XenApp 6.5 and earlier versions. However, unlike XenApp, the VDA (Virtual Delivery Agent) directly communicates with the Delivery Controller, it does this through the Broker Agent, basically the same way as we are used to with the desktop VDA (PortICA).

Before we move on…

While not directly related, the PortICA service, now referred to as PicaSvc2.exe as of XenDesktop 7, does not represent the whole ICA stack within a Desktop VDA, it 'just' controls it. The ICA stack, and this goes for the RDP protocol stack as well, is made up out of a whole bunch of different components all running and interacting within kernel mode.

> **FMA fact**: As opposed to the Desktop VDA, which has been around for a couple of years now, there is no PortICA service within a Server VDA, it simply does not exist.

Head to head

One of the biggest differences between the Server and the Desktop VDA is its ability to accept and manage multiple user sessions at once, hence the RDSH (XenApp) model, whereas the Desktop VDA, also referred to as PortICA, can only handle one ICA session at a time.

Both Server and Desktop VDAs communicate directly and exclusively with the Delivery Controller, and as such they do not need access to the Central Site (SQL) Database or license server.

Also, the underlying OS of an RDSH / XenApp server does not have to be the same as that of the Delivery Controller. And of course we can use multiple Operating Systems throughout our Site if needed or desired. As mentioned, for server machines Citrix now includes a multi-user ICA stack, which extends the Windows Remote Desktop Services with the HDX protocol.

This is the same ICA protocol stack developed for Citrix XenApp 6.5, just with a different management interface to make it compatible with XenDesktop 7.x controllers. Have a look at the table below for some of the biggest differences between the two agents.

Server VDA vs. desktop VDA

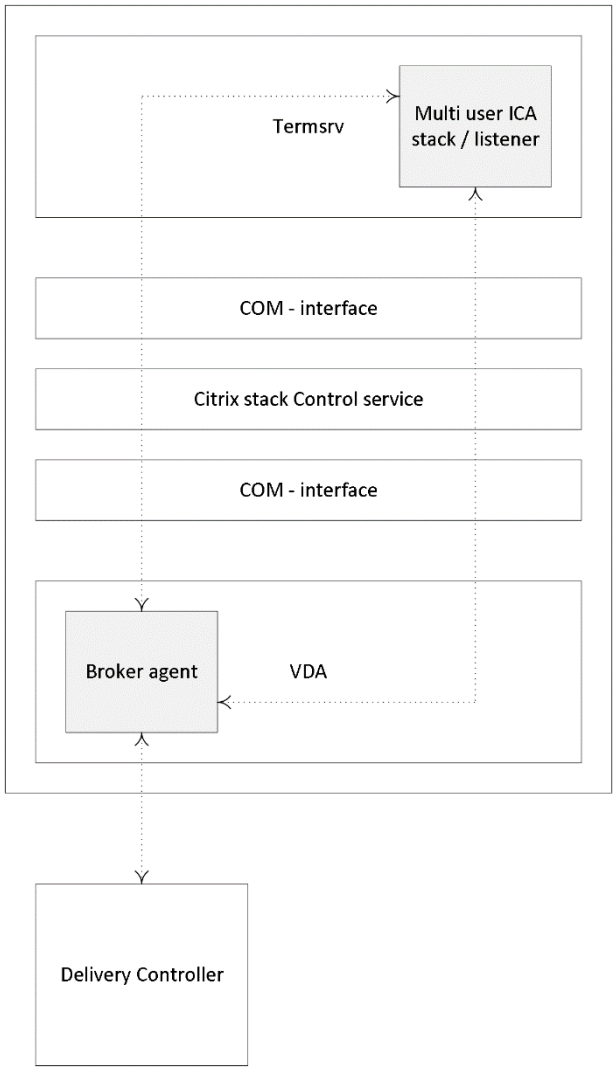
Server VDA	Desktop VDA
Build from ground up	Existing / updated
Multiple sessions/users	One user per session
Desktops and application.	Desktops and applications
CTX Stack Control service / ICA stack	PicaSvc2.exe a.k.a. PortICA
Desktop service a.k.a. BrokerAgent.exe	Desktop service a.k.a. BrokerAgent.exe
Server OS only	Desktop OS only
Non-brokered RDP and ICA connections allowed	Non-brokered RDP connections are allowed; non-brokered ICA connections are not allowed, except in HA mode
Server OS can be hosted using a server VDI configuration / set-up (niche)	

What happens during installation

During Server VDA installation one of the things it will do is register the Broker Agent Service (used for direct communication with the Delivery Controller), which is similar to the Desktop VDA process. Next it will install the multi-user ICA stack, as it does with earlier XenApp versions, which will then become part of Termsrv, creating the ICA stack listener waiting for new ICA connections.

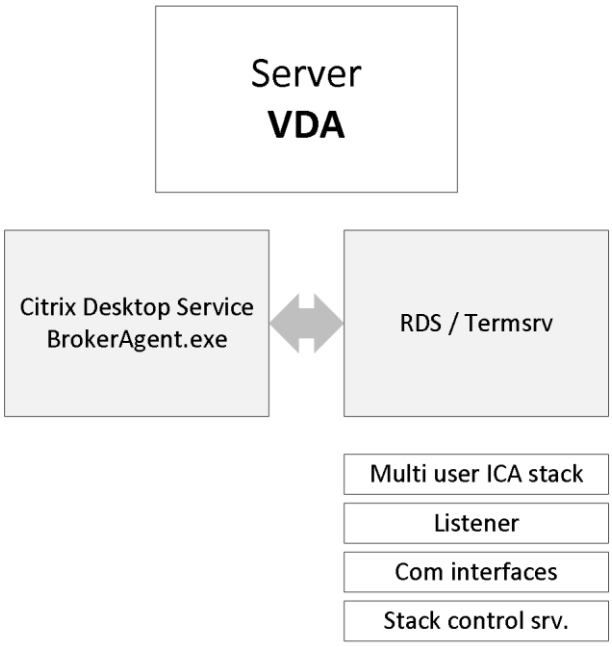
The ICA stack itself has changed very little with the introduction of the FMA: one of its biggest changes is to be found in its communication interface, which is now better known as the earlier mentioned Broker Agent.

Last but not least, it will install and configure the Citrix Stack Control Service: this is its display name within the Windows services overview. As you will see in the graphical overview on the next page, the abovementioned Citrix Stack Control Service, a.k.a. SCService64, will act as an interface between the Broker Agent (a.k.a. BrokerAgent.exe and part of the new FMA) and the ICA stack running in Termsrv, mapping a direct COM interface between the two.



Server VDA internals

So to a certain extent you could say that the SCService64.exe takes on a few of the responsibilities similar to those of the PortICA service in a Desktop VDA. Obviously this is new behaviour and was first introduced with XenDesktop 7 (FMA 2.0). The same can be said for the PortICAsvc.exe as part of XenDesktop 5.x: as of XenDesktop 7 it has been slightly altered and renamed as picaSvc2.exe.



Server VDA services

> **FMA fact**: Each Terminal Server protocol (like Citrix's ICA) will have a protocol stack instance loaded (a listener stack awaiting a connection request). When installed, the Server VDA basically extends Microsoft's RDS protocol with the ICA/HDX feature set / protocol.

Although similar, still different

Although the above might show a lot of similarities to the way the ICA stack and communications functioned with earlier XA versions, the Server VDA is much more simplified and light weight when compared to earlier XA/ICA installations (although I believe it's still over 300 MB in total).

It now solely consists out of the components needed to host sessions and as such it doesn't share any of the other components and services installed on the Delivery Controllers, which wasn't the case with 6.5 and before.

Conclusion

As you can see there is a lot going on under the covers when it comes to the FlexCast Management Architecture and its services. Hopefully this chapter has answered some of the questions that you might have. For this chapter a special thank-you goes out to my FMA partner in crime, Mick Glover, and Martin Zugec as they were able to answer just about every question I had.

Key takeaways

- FMA stands for FlexCast Management Architecture and, as of XenDesktop version 7, includes a Desktop as well as a Server VDA.
- It is the next generation architecture for XenDesktop and XenApp VDI and/or RDSH-based deployments.
- Over the years it has evolved from six up to thirteen main services in total.
- Internal communication takes place over port 80 using Windows Communication Foundation end points.
- Each service runs complete separated from the other services, as a result each service also has its own separate database connection string: if one service fails it will not directly affect any of the other services.
- There is a distinct difference in architecture when compared to the IMA. All of the HDX / ICA bits and bytes are installed as part of the VDA on the Session Host and VDI based VMs while the Delivery Controllers primarily concerns itself with brokering, maintaining and optimizing existing sessions.
- All services run under the NT AUTHORITY \ Network account and use the local computer account for database authentication purposes. One of the benefits this brings is that password are automatically changed every 30 days. This is a big deal, as service accounts are usually very dangerous.
- The Broker service includes the XML as well as the STA service.
- There are 18 active (sub) site services in total, all running within the Broker services, taking care of various Site housekeeping tasks.
- There needs to be a way that VDAs can track and contact the various Delivery Controllers within a Site to be able to register themselves. Citrix uses the auto-update feature for this.
- As we have seen, the PortICA, or picaSvc2.exe, service is an important one during the VDA launch and user login process.
- The PortICA a.k.a. PicaSvc2.exe and the Citrix Desktop Service a.k.a. BrokerAgent.exe services are the two main FMA services within the Desktop VDA.
- The Connection Brokering Protocol (CPB) plays an important role in the VDA registration process. It is basically a collection of WCF end points.
- The Server VDA does not have the PortICA service; however, it does have a Broker service.
- It basically uses the same ICA stack as with XenApp 6.5, but with a different management interface to make it compatible with the 7.x Delivery Controllers.
- Service groups make FMA services highly available.