THE EPICS PROCESS VARIABLE GATEWAY -- VERSION 2*

K. Evans
Argonne National Laboratory, Argonne, Illinois USA

ABSTRACT

The EPICS Process Variable Gateway is both a Channel Access Server and Channel Access Client that provides a means for many clients, typically on different subnets, to access a process variable while making only one connection to the server that owns the process variable. It also provides additional access security beyond that implemented on the server. It thus protects critical servers while providing suitably restricted access to needed process variables. The original version of the Gateway worked with EPICS Base 3.13 but required a special version, since the changes necessary for its operation were never incorporated into EPICS Base. Version 2 works with any standard EPICS Base 3.14.6 or later and has many improvements in both performance and features over the older version. The Gateway is now used at many institutions and has become a stable, high-performance application. It is capable of handling tens of thousands of process variables with hundreds of thousands of events per second. It has run for over three months in a production environment without having to be restarted. It has many internal process variables that can be used to monitor its state using standard EPICS client tools, such as MEDM [1] and StripTool [2]. Other internal process variables can be used to stop the Gateway, make several kinds of reports, or change the access security without stopping the Gateway. It can even be started on remote workstations from MEDM by using a Secure Shell script. This paper will describe the new Gateway and how it is used.

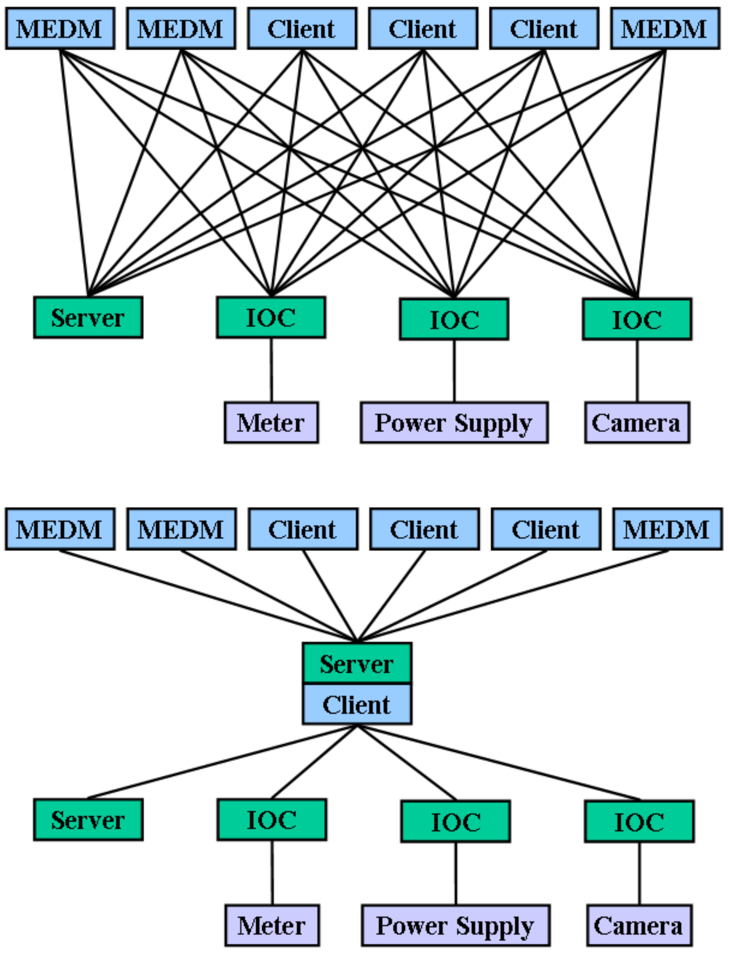

Figure 1: Several MEDMs connected through the Gateway to process variables in IOCs, with and without the Gateway. The Gateway reduces the impact on the IOCs, facilitates bridging subnets, and can provide additional access security.

INTRODUCTION

The Gateway is both a server (like an EPICS Input/Output Controller (IOC)) and a client (like the EPICS Motif Editor and Display Manager (MEDM), StripTool, and others). Clients connect to the server side, and the client side connects to IOCs and other servers, possibly other Gateways. See Fig. 1. There are perhaps three principal reasons for using the Gateway: (1) it allows many clients to access a process variable while making only one

* Work supported by U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

connection to the remote server, thus reducing the load on critical IOCs or other servers; (2) it provides convenient access from one subnet to another; and (3) it provides extensive additional access security. It also has other capabilities, such as providing aliases or renaming for process variables.

In addition to providing access to process variables in other servers, the Gateway publishes its own process variables, which allow it to be controlled and monitored using EPICS tools, such as MEDM and StripTool. Owing to these process variables, you can have a very intimate knowledge of what is happening internally in the Gateway.

Version 1 of the Gateway (currently version 1.3) works with EPICS Base 3.13, and needs a special version of that base to work correctly. Version 2 should work with unmodified EPICS Base 3.14.6 or later. In addition, it has a substantial number of improvements over the first Gateway.

The Gateway is in use at many EPICS facilities and has become reasonably robust and powerful, handling thousands of process variables at high event rates. Reference [3] describes its capability and performance in use at the Advanced Photon Source (APS), where there are currently a total of 40 Gateways running. The main APS Gateway handles on the order of 10,000 – 20,000 process variables on its client side, and its server event rates are typically on the order of 2 – 10 kHz. It is typically connected to over 200 IOCs at a time. It does this using a moderately small amount of CPU, on the order of 10%, on a reasonably powerful, dual 2.8-GHz processor, Linux workstation. It has run as long as three months without having to be restarted. It runs on Solaris, Linux, other versions of UNIX, and Microsoft Windows. It has been extensively used and tested on Solaris and Linux.

Additional information about the Gateway, including the manual, relevant papers and presentations, and a download link, can be found on its web page [4].

GATEWAY INTERNALS

In this paper we will often refer to the clients as MEDM and the server as an IOC. This is to avoid confusion with the client and server parts of the Gateway and because MEDMs and IOCs are the most typical and most used client and server. Keep in mind, however, that this is for convenience and clarity only and that there are many other clients than MEDM and there are other types of servers than an IOC. The Gateway itself is, in fact, another type of server than an IOC.

Internally the Gateway maintains two objects for each process variable. There is a Process Variable (PV) object that handles the Gateway client connections to the IOC, and there is a Virtual Connection (VC) object that handles the Gateway server connections. Each VC has a list of Channel (Chan) objects that represent connections to the MEDMs. The PV object implements the client side of the Gateway, and the VC object and its Chans implement the server side.

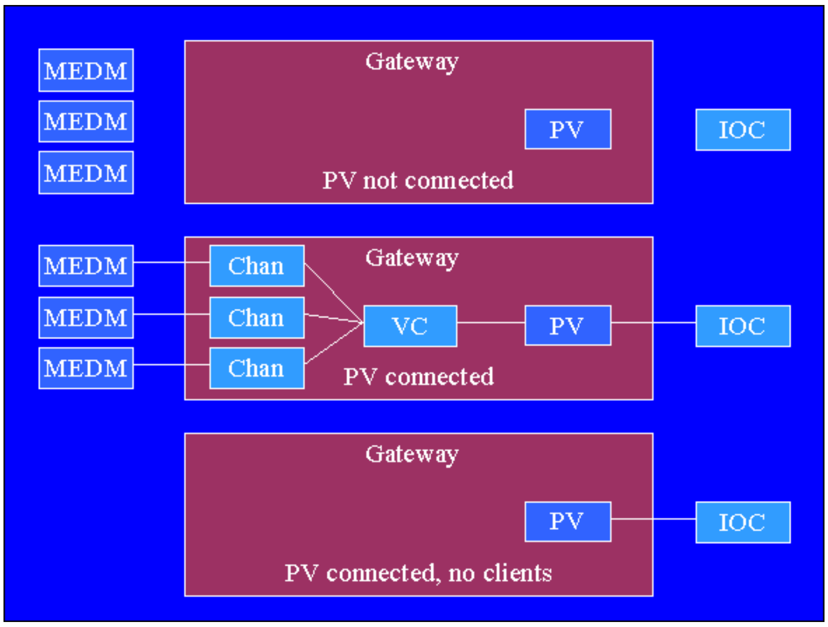

Figure 2: Typical states for the Gateway internal objects.

Figure 2 shows three typical states for these objects. In the top state there is no connection to the IOC, either because it is still trying to connect or because the connection has been lost. In that case there is no VC, and the MEDMs see the process variable as not existing. In this case the PV object is destroyed after two minutes unless it connects. The middle state is most typical. There is a connection to the IOC, and MEDMs are connected via the VC. In the lower state the PV is connected to the IOC, but there are no MEDMs currently interested in the process variable. The PV object and connection to the IOC are maintained for two hours after the last time they are used in case another MEDM or other client wants to use it again. The other client could be a command-line program, like Caget, that requests the value and then exits. In that case the PV object will still stay around for at least two hours.

PV States

When a Channel Access Client wants to connect to a process variable, it send out a series of search requests, which are UDP packets, on the network. These packets are initially sent at a very fast rate with the time between them doubling at each try until 100 are sent or there is an affirmative reply. Each server that gets such a packet is required to determine if it has the process variable; that is, perform an **Exist Test**. When the Gateway gets such a request, it creates an unconnected PV object that itself sends a search request to the IOCs. This PV starts out as **Connecting**, and the Gateway responds to the search request as postponed. If it does not connect in a very short time, it becomes **Dead** and the Gateway responds to subsequent search requests as not having it. If it is not found by the Gateway after two minutes, the PV is destroyed. If it becomes connected to an IOC, its new state depends on its old state. If it was **Disconnected** or **Dead**, then it becomes **Inactive** and the next **Exist Test** will return that it was found. If it was **Connecting**, postponed **Exist Tests** are processed, and if a VC is created because an MEDM is still interested, it becomes **Active**. Soon afterward a Chan is created, and the MEDM is connected to the VC and through it to the PV. Once the PV is **Active** or **Inactive**, the Gateway responds to subsequent search requests as having the process variable. If the connection is then lost, the VC and its Chans are destroyed, and the PV becomes **Disconnected**.

PVs are either **Connected** or **Unconnected**. **Connected** consists of either **Active** or **Inactive**. **Unconnected** consists of **Connecting**, **Dead**, or **Disconnected**. The top state in Fig. 2 is **Unconnected,** but it is not clear whether it is **Connecting**, **Dead**, or **Disconnected**. The middle state is **Connected** and **Active**. The bottom state is **Connected** and **Inactive**.

Saying this another way: the PVs for which the Gateway has an established client connection with an IOC are **Connected**. Otherwise they are **Unconnected**. For the **Connected** ones, if there is a connection to an MEDM, then they are **Active**. Otherwise they are **Inactive**. If they are **Unconnected**, it may because they are **Disconnected** (meaning formerly connected), **Connecting**, or **Dead**.

The possible states are shown in Fig. 3.

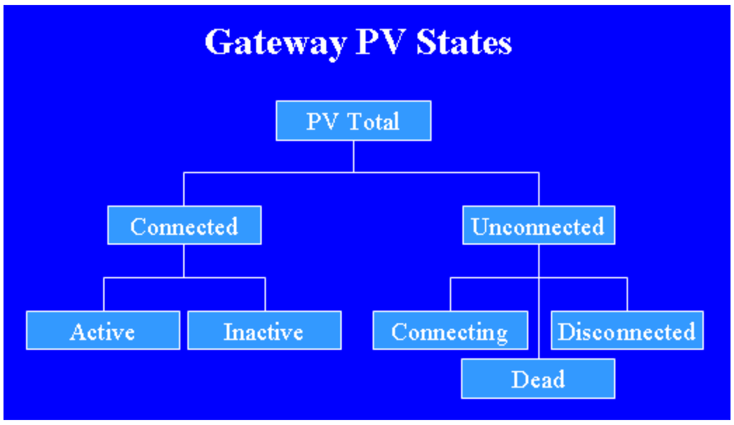

Figure 3: Possible PV States:
PvTotal = Connected + Unconnected
Connected = Active + Inactive
Unconnected = Connecting + Disconnected + Dead

FEATURES

Access Security

The Gateway applies access security in addition to any access security that may be implemented in the IOCs or other servers to which it connects. It supplements but cannot override IOC access security. The access security is specified in two files, **gateway.pvlist** and **gateway.access**, by default. The first file, **gateway.pvlist**, specifies what process variable name patterns are allowed and denied. The second, **gateway.access**, specifies the access security rules. These two files are read at startup and can be reread later on the fly.

The access security can be changed on demand by setting the internal process variable **gateway:newAsFlag** or by other means. This causes the two files to be reread and the access security to be changed accordingly. After rereading access security, the Gateway generates a beacon anomaly, which will cause MEDMs to reissue search requests for unfound PVs.

The access security rules specified in **gateway.access** are the same as used for access security in IOCs. In fact, both IOCs and the Gateway use much of the same access-security code. Specifying access security can be complicated, and you should see the *Application Developers Guide* for EPICS Base for more information on specifying the access security rules in **gateway.access**. It can be found on the EPICS web pages [5] under each version of EPICS Base.

The **gateway.pvlist** file is specific to the Gateway. Its function is to specify what process variable name patterns are allowed or denied and to optionally associate patterns with access security groups and security levels in the access file. The patterns are GNU-style regular expressions. In addition, aliases can be specified so that the Gateway supplies the affected process variables under a different name than used in the IOC. There are examples of both files in the Gateway source distribution, and there is much more information in the Gateway Reference Manual [4].

Gateway Process Variables

The Gateway publishes several process variables, allowing you to control and monitor it. By default these have a prefix, which is the name of the host machine, but this can be changed by the -prefix command-line option. The Gateway internal process variables are not record based as in an IOC. They do have a DESC field, however. This avoids delays when using PvInfo in MEDM and StripTool and allows StripTool to put a description in the legend. There are three groups of process variables: (1) ones giving the number of process variables in the PV states described above as well as the number of VC connections, (2) ones that give event rates in the server and client or show CPU usage, and (3) ones that allow you to control the Gateway and cause it to print reports or quit.

The internal process variables include:

gateway:vctotal
gateway:pvtotal
gateway:connected
gateway:active
gateway:inactive
gateway:unconnected
gateway:connecting
gateway:disconnected
gateway:dead

gateway:clientEventRate
gateway:clientPostRate
gateway:existTestRate
gateway:loopRate
gateway:cpuFract
gateway:load
gateway:serverEventRate
gateway:serverPostRate

gateway:commandFlag
gateway:report1Flag
gateway:report2Flag
gateway:report3Flag
gateway:newAsFlag
gateway:quitFlag
gateway:quitServerFlag

Server Mode

The Gateway can be operated in server mode by specifying -server on the command line. In this mode a server process is started that in turn starts the regular Gateway process and then watches it and automatically restarts it if it dies. Using this mode the Gateway can recover from crashes. The running Gateway can effectively be reset by killing it via one of the ways to stop the Gateway, or the server process can be killed, in which case no more regular Gateway processes will be automatically started. The server feature is not available on Microsoft Windows.

Put Logging

It is possible to log whenever someone writes to a Gateway process variable. To do this you need to specify -putlog on the command line when the Gateway is started and specify a filename for the put logging. There needs to be an access security group in the **gateway.access** file that has a rule with WRITE and TRAPWRITE specified, for example: "RULE(1,WRITE,TRAPWRITE)". (See the *Application Developers Guide* for EPICS Base in Ref. [5] for more information about the syntax of these rules.) Then whenever there is a write (or put) to any process variable in that group, it will be logged in the specified putlog file. As with the log file, this file will be automatically renamed and saved whenever the Gateway restarts.

Reports

The Gateway prints three kinds of reports: (1) the Virtual Connection Report, which includes all MEDM or other client connections to all process variables; (2) the Process Variable Report, which includes all process variables grouped by state; and (3) the Access Security Report, which includes the allowed and denied process-variable patterns from the pvlist file, **gateway.pvlist**, by default. The Gateway prints its reports to the report file, **gateway.report**, by default, appending them if the file already exists. These reports can be long.

TYPICAL CONFIGURATIONS

Symmetric Gateways

Figure 4 shows a configuration in which each of three networks has a Gateway that finds process variables in IOCs in the other two networks.

However, note that if you are on say, Net A, and using the Gateway whose server is on Net A, then the client side of this Gateway sees the server side of the other two Gateways. However, these Gateways each see servers on Net A and consequently see the Net A Gateway. Therefore, the Net A Gateway can see process variables on IOCs on Net B and Net C directly and also through the other two servers. To prevent this loop, each Gateway must be configured to not allow process variables from the other two Gateways. This can be done through the -ignore command-line option.

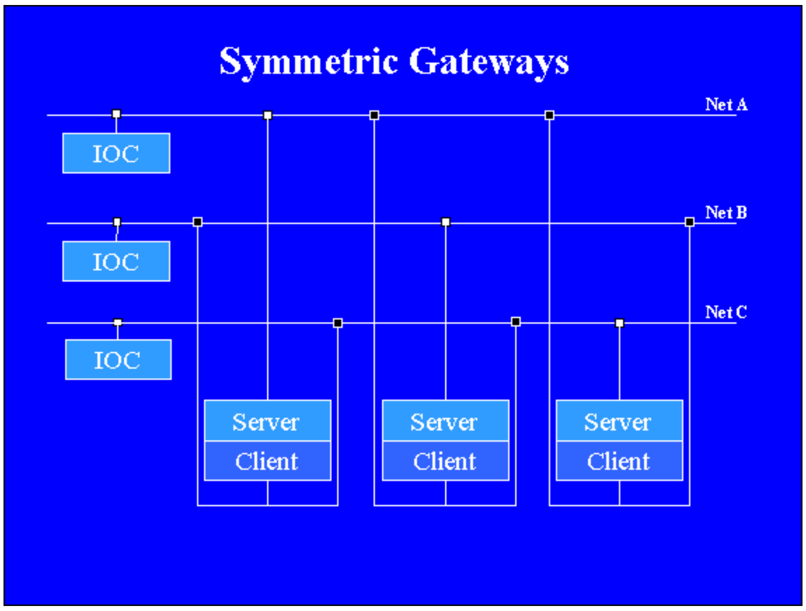

Figure 4: A symmetric Gateway configuration. Used to find process variables on one subnet from the other two (or more).

Reverse Gateway

Figure 5 shows a Gateway that gets process variables from Net Z. This is a configuration used at Argonne. Net Z is the machine network, and the other networks belong to individual experimental teams. These teams have read access to the machine network and whatever access they want to the IOCs on their own network. Argonne at present has eight of these configurations, most with four Gateways plus a reverse Gateway. Owing to the reverse Gateways, the internal process variables for all these Gateways can be seen on one MEDM screen (see Ref. [3]) through a Gateway that sees process

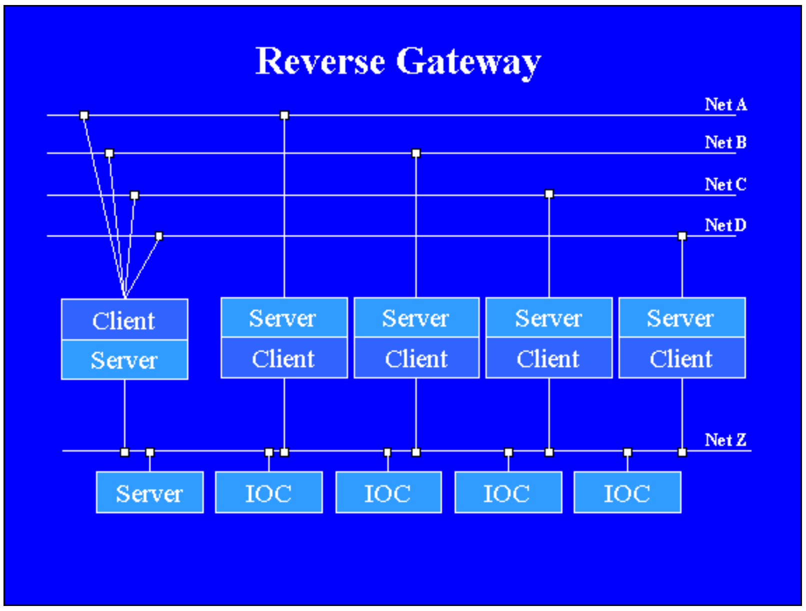

Figure 5: A reverse Gateway configuration. Used to access internal process variables from another Gateway.

variables on the machine network. One does not need (and typically does not have) access to each of these private networks. This leads to a great convenience in managing such a large number of Gateways.

To prevent looping, the reverse Gateway only allows the internal process variables from the other four Gateways and itself, and each of the other Gateways denies the internal process variables from the other three.

Alias Gateway

It is possible to use the Gateway just to provide aliases for process variables, and Fig. 6 shows an alias Gateway. Only one subnet is needed. If an MEDM asks for the alias name, it is found through the Gateway. If it uses the real name, it is found through the IOC. The two names must be different to avoid finding the same process variable in two different servers. This type of Gateway has been used during the transition period of a large-scale renaming of process variables.

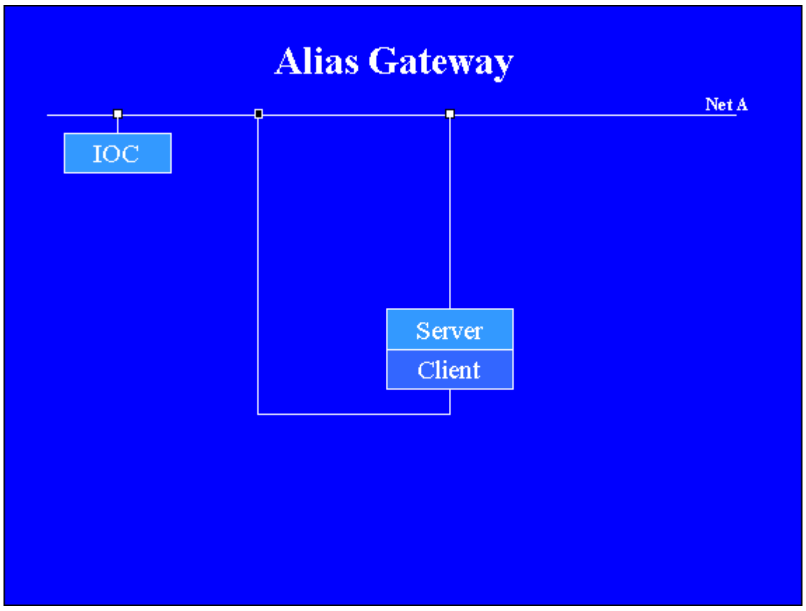

Figure 6: An alias Gateway configuration. Used to provide renaming for process variables.

SUMMARY

The Gateway provides a means of providing access to critical IOCs to many users while minimizing the impact on the IOCs. It provides additional, highly configurable access security, usually used to provide read-only access. The current version of the Gateway is quite stable and capable of handling a large number of process variables with high update rates, and has run for long periods of time in a production environment without having to be restarted. Its internal process variables allow it to be easily monitored and controlled. It is an essential tool for large installations.

REFERENCES

[1] MEDM: Motif Editor and Display Manager, http://www.aps.anl.gov/epics/extensions/medm/index.php.
[2] StripTool, http://www.aps.anl.gov/epics/extensions/StripTool/index.php.
[3] Kenneth Evans and Martin Smith, "Experience with the EPICS PV Gateway at the APS," Proc. 2005 Particle Accelerator Conf., Knoxville, TN, May 16 - 20, 2005, to be published.
[4] Gateway: The Process Variable Gateway, http://www.aps.anl.gov/epics/extensions/gateway/index.php.
[5] Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics/index.php.