

## **DTRA Next Generation METOC Data Server (MDS)**

### **OVERVIEW**

The [Defense Threat Reduction Agency \(DTRA\)](#) is currently undertaking a mission to provide tools to support hazard assessment from releases of chemical, biological, radiological, nuclear and high-explosive (CBRNE) materials. A large part of this effort is the ability to accurately predict and incorporate weather intelligence into threat assessment. Because the nature of meteorology applied to the transport and dispersion problem is specialized, special care must be taken when applying general weather tools. Much of the science and technology related to this field is immature; the products needed to fulfill DTRA's unique weather requirements for highly detailed meteorological data do not exist in a mature form. By further developing delivered technologies and modifying existing weather systems, DTRA can achieve the goal of providing its customers with the highest quality weather data available.

DTRA's [Hazard Prediction and Assessment Capability \(HPAC\)](#) is an advanced tool that can accurately calculate hazardous areas resulting from atmospheric CBRN releases. In order to minimize collateral effects for target planning and incident reaction/response, HPAC users require the highest fidelity of available meteorological data. Due to the global nature of HPAC customers, this data must be made available to every user in a simple and timely fashion. The HPAC software package must also have the ability to ingest various types and forms of meteorological data from numerous sources. In addition to HPAC, the Department of Defense (DoD) Joint Program Manager for Information Systems (JPM-IS) is currently developing the replacement for the HPAC system, the Joint Effects Model (JEM). JEM will be the single accredited DoD Program of Record for hazard prediction and assessment. Both of these systems have standing requirements for real time access to environmental datasets, including meteorological information.

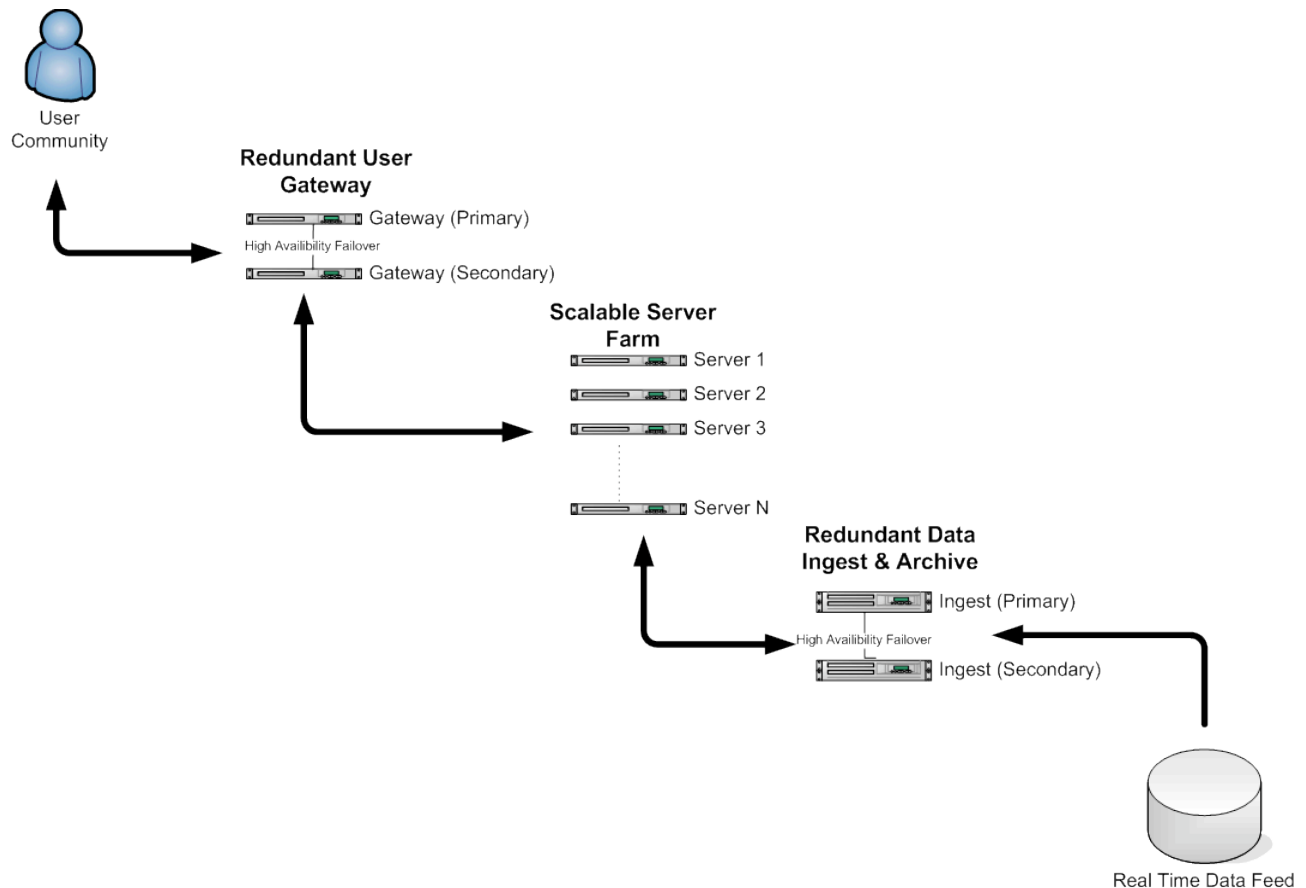
To fulfill these requirements, DTRA currently maintains and operates a redundant set of METOC Data Server (MDS) systems. Each of these systems is continuously populated with real time meteorological forecast and observational products from the [National Center for Environmental Prediction \(NCEP\)](#), [National Weather Service \(NWS\)](#), [Fleet Numerical Meteorology and Oceanography Center \(FNMOC\)](#), [Air Force Weather Agency \(AFWA\)](#), DTRA, and others. The MDS system responds to requests made by an HPAC/JEM user via the Internet, NIPRNET, or SIPRNET. Once a request is received, the server parses out the appropriate data and returns a file only containing the meteorological data necessary to calculate the hazard prediction for the area and time of interest. In addition to the MDS systems, DTRA has included several meteorological utilities with the HPAC/JEM software package, allowing a user to view and manipulate various weather products from his/her own personal computer.

Unfortunately, the current MDS system architecture has not changed substantially since its inception in the 1990's and is in dire need of a system wide overhaul. Subsequently, the systems are struggling to keep up with rising user demands and associated system loads. In addition, the associated MDS interface is also dated, using the File Transfer Protocol (FTP) as its primary client/server interface. Due to these circumstances, DTRA has contracted the National Center for Atmospheric Research / Research Applications Laboratory (NCAR/RAL) to perform a complete rebuild of the MDS system architecture and interface, based on NCAR/RAL's experience developing proven meteorological data management capabilities and related ongoing research and development activities.

### **SYSTEM ARCHITECTURE**

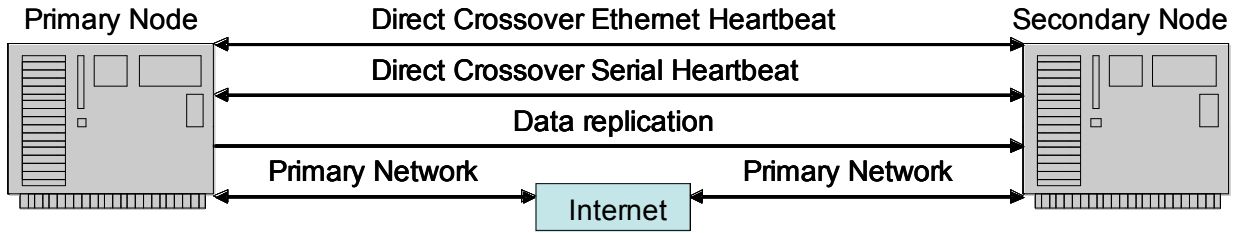
The Next Generation MDS system architecture is based on a proven high availability, linearly scalable, framework. As shown in Figure 1, the system is composed of three distinct components:

1. User Gateway
2. Server Farm
3. Data Ingest/Archive



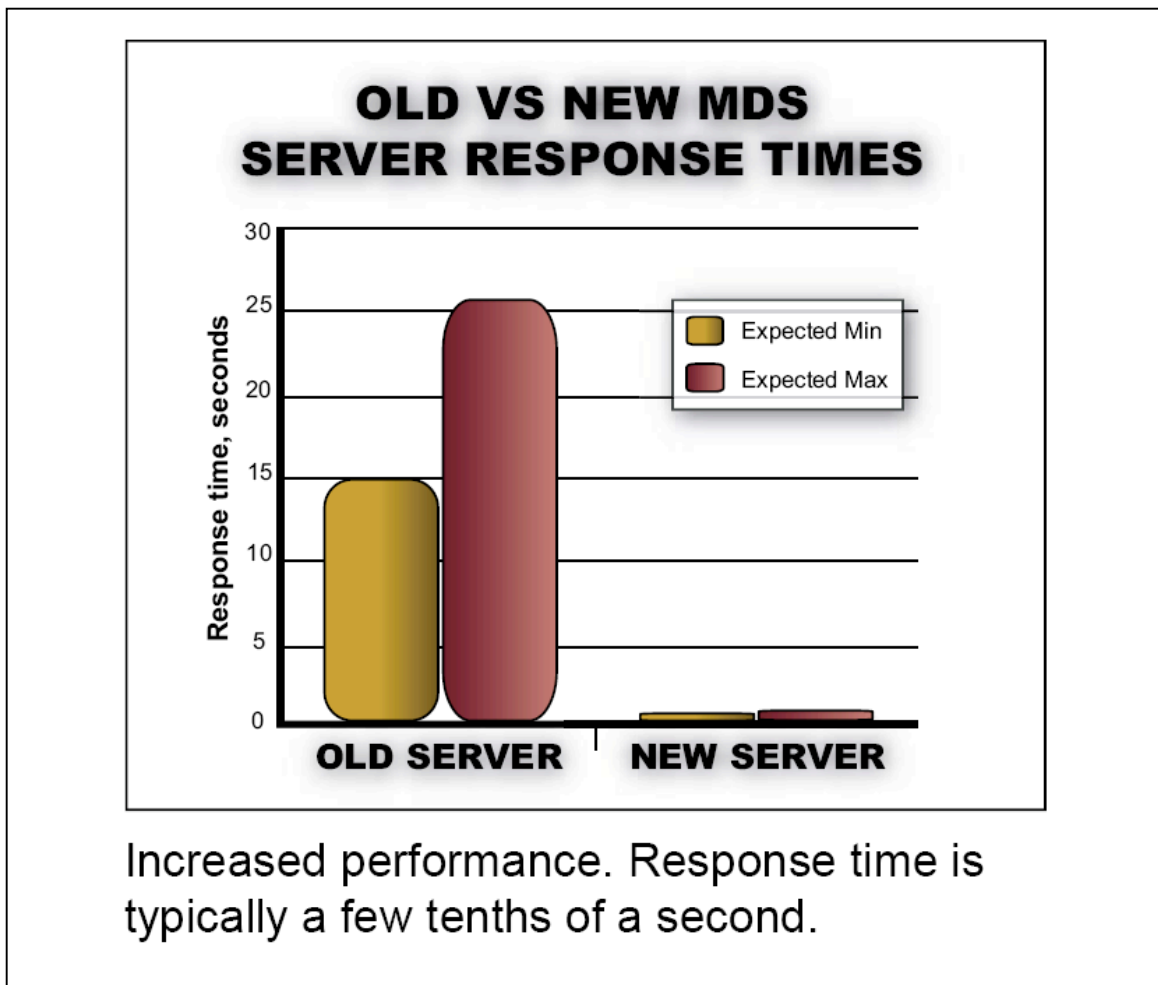
**Figure 1. Next Generation MDS System Architecture**

The User Gateway's primary function is to route incoming user requests to one of several data servers, contained within the Server Farm. Incoming requests are distributed among the server farm, using a simple round robin protocol, to efficiently distribute the user request load among the expandable server system. In addition, the Gateway serves as a firewall between the exposed internet and the rest of the MDS system, providing an extra layer of protection to the core system. The User Gateway is comprised of two distinct gateway nodes, a Primary and Secondary, connected via a high availability failover system (Figure 2). In the event that the primary gateway fails, the secondary system immediately takes full control of all User Gateway functions, ensuring uninterrupted service to the user community. During this time, the failed gateway can either be repaired or replaced without interrupting operational system performance.



**Figure 2. High Availability Failover Configuration**

The Server Farm is the workhouse of the entire MDS system. It receives incoming user requests, extracts the requested products, either from locally stored real time data or from archived datasets, stored on the Data Ingest system, and then returns the resulting data to the requesting entity, all in a matter of seconds. Figure 3 shows server response times between the Legacy and Next Generation MDS systems.



**Figure 3. Comparison of expected minimum and maximum server response times between the Legacy and Next Generation MDS system.**

The number of server nodes, within the Server Farm, is infinitely expandable, although only three nodes are used for the current system configuration.

Continuous ingest and archival of all incoming raw data feeds is performed by the Data Ingest/Archival component. For maximum retrieval efficiency, raw meteorological gridded model datasets are first converted into a NCAR/RAL proprietary Meteorological Data Volume (MDV) format, while raw observational data is stored within a MySQL relational database. The resulting data is then stored on the Data Ingest/Archival system for up to 7 days, depending on available disk storage, before being permanently scrubbed from the system. This data is then distributed to each server node for more efficient retrieval and to maintain optimal system performance during heavy request loading periods. Similar to the User Gateway component, the Data Ingest/Archival system is also configured in a high availability failover configuration to ensure uninterrupted operation of the system.

The MySQL database is also used to store user account authentication information, which can be queried by system administrators to provide user information and associated usage statistics.

## **SYSTEM SUBCOMPONENTS**

### *MDV*

MDV was designed to provide quick access to time-based data. A MDV data repository consists of a directory structure indexed by date and time, containing files that store a series of met variables sub-indexed by vertical level. The MDV software library also includes a large number of C++ routines to handle different mapping projections, unit conversion, and output to other data types.

### *DsMdvServer*

The DsMdvServer application provides an efficient and flexible interface to the MDV data repository. It allows user applications to request data in a variety of ways, with primary concern for fast retrieval of the appropriate data.

### *MySQL*

The MDS system uses a relational database management system (RDBMS) to store data online. The decision to use a relational database was based on several factors:

- A location was needed to store the system data.
- A flexible mechanism was needed to handle retrieval requests from researchers for climatology, forecasting, and model validation.
- There was a need to be able to store and retrieve data using a variety of programming languages, such as C/C++, Perl and Java, without using a proprietary or legacy interface.
- One of the system requirements is to be able to efficiently handle a large amount of data.

A relational database meets all of these needs. Relational databases store data, using tables to hold the data and relations to link data from one table to another. Most relational databases use SQL, a standardized language for querying databases. There are also standard interfaces for database programming, such as ODBC and JDBC. Databases can handle large amount of data, and by using indexes, can retrieve individual records at high speeds.

MySQL is a very fast, efficient and reliable database server, and is relatively simple to maintain compared to most other database systems. As the name implies, MySQL uses SQL for its query language. Drivers are available for both ODBC and JDBC, and perl modules are available as well.

More information on MySQL can be found at <http://www.mysql.org>

### *FIDO*

The 4DWX Interface for Data Operations (FIDO) is an API and a data server that 4DWX applications use to retrieve data from several sources. FIDO clients can retrieve surface observations, forecast model grids and satellite images using the same server interface. FIDO has been used by the 4DWX system for several years and has been useful and reliable.

The MDS system uses the FIDO data components to retrieve observation data from the MySQL database. These components have been integrated into the MDS system. The system translates observation data requests into FIDO requests, then uses the FIDO components to retrieve the data. Configuring the FIDO components for the MDS system was not a challenge, since the database schema for the 4DWX system and the MDS system are similar. Several enhancements were added to FIDO to support MDS requirements, such as merging of similar data sets and resorting.

In addition to integrating FIDO technology into the system, a stand-alone FIDO server has been installed on the MDS. This server can provide data to FIDO client applications. The FIDO server is intended to provide visualization capabilities to MDS administrators, and will not be available to the MDS end-users.

### *MDS API*

The MDS API is a Java based interface to access MDS services, using the Web Services Simple Object Access Protocol (SOAP) for transport protocol. This flexible API allows requests for a specific model source, output variable, vertical level, horizontal domain, and temporal domain. In addition, the API includes an advanced query capability to extract related metadata for all MDS data holdings. A snippet from the API javadoc is shown in Figure 4.

edu.ucar.rap.mds.client

## Class WxDataRequest

java.lang.Object  
└ edu.ucar.rap.mds.client.WxDataRequest

All Implemented Interfaces:  
java.lang.Cloneable

```
public class WxDataRequest  
extends java.lang.Object  
implements java.lang.Cloneable
```

Represents a single request for weather related data. Sample usage: see [TestDemo](#)

Field Summary	
java.lang.String()	<u>fieldNames</u> Names of requested fields or null.
int	<u>fieldSet</u> field selection type: one of FIELDSET*.
double	<u>latMax</u> max latitude, decimal degrees (negative for S of the equator)
double	<u>latMin</u> min latitude, decimal degrees (negative for S of the equator)
double	<u>lonMax</u> max longitude, decimal deg (negative for W of Greenwich UK)
double	<u>lonMin</u> min longitude, decimal deg (negative for W of Greenwich UK)
java.lang.String	<u>modelName</u> Preferred model: One of "best", "NOGAPS", "GFS", "ETA", etc, or null.
java.lang.String	<u>outFile</u> Name of the output file.
int	<u>outFormat</u> Requested format of output file.
int	<u>outType</u>

Figure 4. MDS API javadoc describing the WxDataRequest java class.

## Sysview

SysView is a web-based display for the system monitor. It allows a user to easily see which components of the system are functioning properly and which are not. Troubleshooting information can be found within the SysView web pages. This will give a user an idea of how to fix the problem when it occurs.

SysView is very simple to use since it is read-only and there are no configurations to make. The main page (Figure 5) shows a map of the system and how the different components interact with each other. All of the components will be colored **red**, **yellow**, or **green**. **Red** means that there is a severe problem with the component. **Yellow** means that the component is functioning at a reduced level. **Green** means that the component is functioning properly. The map is organized in a hierarchical structure so that as the components are clicked on, it will show more detail until finally getting down to the level of an individual component. When viewing a sub-diagram as a single component, its color will reflect the most severe state from any of the components below it. This does not mean that all components below are the same state.

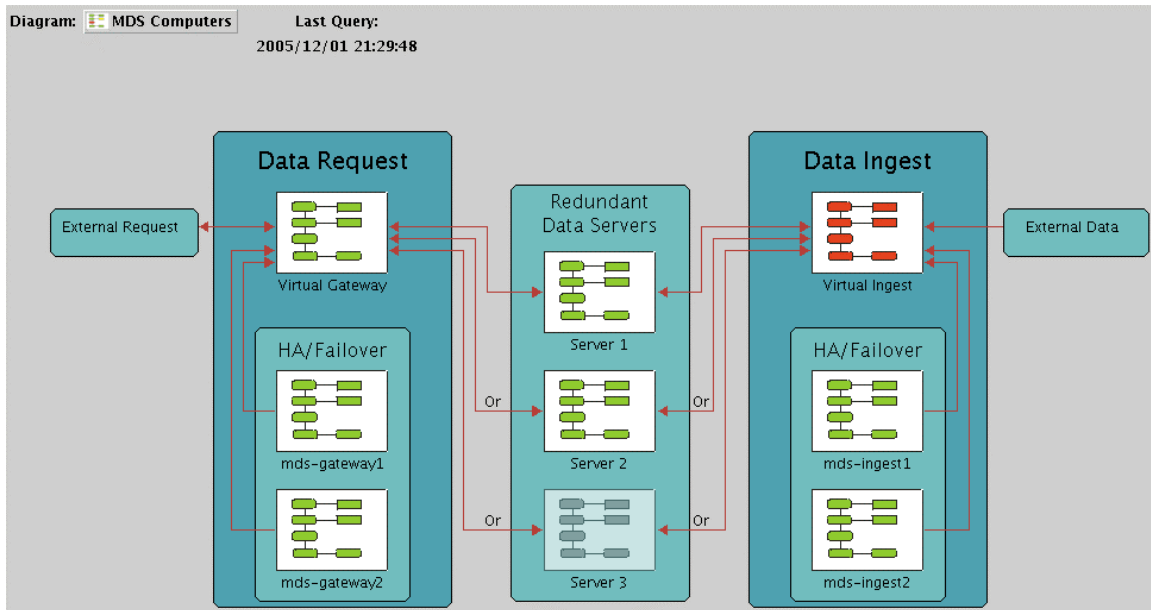


Figure 5. SysView's main page for the MDS System.

There are three different component types in the diagrams. Throughout SysView, the colors **red**, **yellow**, **green**, and **gray** mean severe problems, minor problems, no problems, and no information respectively.

One component represents the state of a sub-diagram. These are the boxes that appear to have other diagrams drawn within them. When these are clicked on, another diagram will appear to show the details of that component. See Figure 6 for an example.

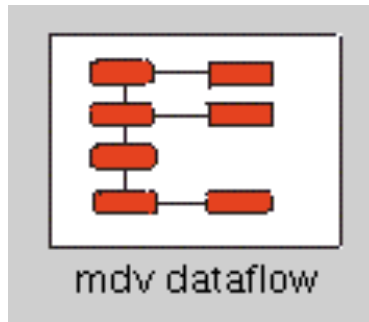


Figure 6. Example of a SysView component, representing a sub-diagram.

Another type of component represents the state of an individual item. These boxes will be filled with a solid color to show their states. Clicking on an individual item will show more details about the item. The details will include a [Troubleshooting](#) link. This link will give possible solutions for fixing a problem with the component. See Figure 7 for an example.

The 'Processing Agents' window shows a grid of 14 green rounded rectangles, each representing a Grib2Mdv process. The processes are arranged in two columns of seven. The left column contains processes with instance names: afwa\_t01k, afwa\_t02q, afwa\_t04q, fnmoc\_0032, fnmoc\_0158, fnmoc\_0187, and fnmoc\_0240. The right column contains processes with instance names: afwa\_t02p, afwa\_t03k, afwa\_t06k, fnmoc\_0033, fnmoc\_0186, fnmoc\_0188, and fnmoc\_0245. Red arrows point from the left edge of the 'Processing Agents' window to the left edge of the 'Grib2Mdv afwa\_t03k' and 'Grib2Mdv fnmoc\_0186' components.

The detailed status window for 'Grib2Mdv afwa\_t03k' shows the following information:

Status: PROCESS_STATUS_OKAY	
Process Name	Grib2Mdv
Process Instance	afwa_t03k
Process Host	mds-ingest
Priority	PRIORITY_HIGH
Info:	
Process Name	Grib2Mdv
Process Instance	afwa_t03k
Host	mds-ingest1
User	4dwx
PID	3526
Start Time	Mon Nov 28 09:41:49 MST 2005
Heartbeat Time	Thu Dec 01 15:01:08 MST 2005
Last Registration Time	Thu Dec 01 15:01:08 MST 2005
Max Register Interval	120000
Num Registrations	22593
Status Value	0
Status String	Checking for data

Figure 7. SysView Components, highlighting the state of individual MDS processes. This example shows the status of several GRIB to MDV ingest conversion routines.

The boxes that are blue-green in color cannot be clicked on, and usually represent something outside of the system. These may represent things such as a data source. This makes it easier to see where the data are coming from and may expose problems outside of the *MDS* system.