

## The Interim, Until You Achieve an Operationally Responsive Ground System

Bob Wendlandt, Kelly Clarke, Charles Miyamoto, Jordan Lei, Kyran Owen-Mankovich

Jet Propulsion Laboratory  
California Institute of Technology  
4800 Oak Grove Drive, M/S 301-240  
Pasadena, CA 91109-8099

{Bob.Wendlandt, Kelly.Clarke, Charles.Miyamoto, Jordan.Lei, Kyran} @jpl.nasa.gov  
JPL Approval # \_\_\_\_\_

### Introduction

Everyone wants to achieve a "Responsive" Ground Data System (GDS), but that takes time. What do you do in the interim? Our group, called the Integration, Test and Deployment Team (ITD), is a group of responsive engineers whose primary focus is to assist JPL projects to successfully adapt, test, integrate and deploy their ground data system.

The team configures and adapts the GDS for a project, so that analysts, engineers and scientist do not need to be experts in the GDS to operate it. The team has developed a human interface to accommodate all types of users. It provides Graphical User Interfaces (GUI's) for those that want GUI's, command line interfaces for those that want control, and selection button interfaces for other users.

The cornerstone of a responsive Ground Data System is responsive people. Without individuals who can be aware of a project's changing needs and requirements, how can the GDS become responsive?

### Ground Data System

JPL's GDS is a UNIX based system running primarily on Sun workstations. It is currently moving toward Linux. Development began in 1986 to generate a single generic Multi-mission Ground Data System that would not need to be rewritten for each new spacecraft project. Today much of the core GDS is table driven so that when a new mission is under development it is not a major effort to adapt the GDS to it. Users view the GDS system in many ways and with different abilities.

The GDS is an exceptional piece of architecture that has been designed with a great deal of flexibility. But, have you ever thought for a minute about what flexibility entails? Flexibility means options. Options mean variability. Variability means more to remember. More to remember means forgetting something. Forgetting something means possible mistakes.

The ITD team works in cooperation with the mission ground data system engineers and the JPL institutional

software development organizations. The team's goal is to provide GDS system solutions to help alleviate the possibility of mistakes by users of the GDS. By designing and adapting ground solution processes and environments it is possible to mitigate user error.

The following discussion will present various vignettes or real life examples that demonstrate how we have been responsive in attempting to alleviate this situation and still give the flexibility to the user that they want and need.

### The Nature of Our Team: Responding to Dynamic Environments

GDS users work in a very dynamic environment. One moment everything is quiet, no data is arriving, then on schedule, data begins to arrive and everyone is actively reviewing the latest data for the health and welfare status of the spacecraft or rover. When a Ground Data System user experiences unexpected results and they are not sure what is going on, they call the ITD Team. The team attempts to resolve any user problem in an expedient manner. If it is an immediate issue (e.g., no one is seeing any data) then the team has to act quickly. Most of the time each team member is working on or investigating other project issues, so when a problem call comes in, the appropriate team member for that project will set aside what they are currently working on and respond to the problem. Each team member constantly has to juggle between immediate and long-term tasks. The long-term tasks are essentially queue driven, when one is finished the next task in the queue moves to the top.

The short term or immediate tasks operate more like a stack. The stack is empty, and then within a very short time span there can be several urgent tasks pushed onto it. This requires each team member to be flexible in their work schedule to set aside what they are working on and change gears to a different more important activity. Responding to this ever changing environment of immediate verses long term tasks and activities is what challenges the ITD Team.

When an issue comes up and a team member does not know the answer to a problem, they are able to utilize the resources of the team. This is where having the team co-located in a lab environment benefits the team and ultimately the projects. Any time a team member has a question, they need only turn around and ask other members in the lab. If the individual is in a project area, they can call the central number at the lab. Chances are someone has encountered a similar issue before. If one person is stuck, another is likely to have a new idea, a different perspective, an alternate thought, or a different track to take on how to approach the problem. An incredible amount of time is saved by this interaction. Through this lab environment cross training is accomplished on a daily basis, and customer issues are resolved much faster than sitting in a cube alone, banging our head against a wall for hours. The result of this is that the individual can now utilize the resources of the entire team when they encounter a new or unusual situation. It is the individual characteristics of each person that comprise the team. Or to state it another way: the team is greater than the sum of all its members.

While the team also formally cross-trains, the spontaneous bouts of informal learning make it a lot easier for individuals on the team to cover one another when someone leaves on vacation or is out sick. Each team member has a core set of skills and knowledge base that makes it easy for each one to learn from the other.

This teaming comes into play not only for learning and growing, but also in more concrete terms. We are able to find out about innovative GDS solutions, usually in the form of tools, which have been provided to other missions. We can then take and adapt some of these tools for other projects. These tools often expand to meet the dynamically changing needs of new missions.

Teaming has been about what and how the ITD team has accomplished what it does. Now let us take a closer look at some of the GDS issues the team needs to respond to and find solutions for.

### What Time Is It?

What time is it? A simple enough question when we deal with it in the home or the office, but what about when our office is a Space Flight Operations Area and our home is Mars? On Earth we have PST and UTC time differences. Dealing with “what time is it?” on the Mars Exploration Rover (MER) Project, time takes on an even wider perspective.

At the beginning of the MER Surface Operations, the team had a lot to learn what others had in mind when they specified a time. We would get a phone call or someone would walk by and say: “data is coming down at 3.”

After they left, we would realize that we did not know what time that really was. We did not know what their perspective of time was. This does not just pertain to data, but, also, for meetings, meals, etc. The following table of “times” demonstrates the different times that we regularly had to deal with:

Pacific Standard Time	= 3:00am PST = 3:0 0am PST
Universal Time Coordinated	= 0300 UTC = 8:00pm PST
For Mars time:	
Spacecraft2 /MER-A/Spirit	= 3 LMST = 6:40pm PST
Or	
Spacecraft1/MER-B/Opportunity	= 3 LMST = 6:58am PST
<i>Note: The two rovers, Spirit and Opportunity, are on opposite sides of Mars. One would be AM and one would be PM.</i>	

Table 1. Example of Different Time Perspectives

In Table 1 each of these times is different! And hopefully you can see the confusion that could arise because of the differences. The confusion is expanded when each rover can and is referred to by its original spacecraft designation number (1 or 2), by its launch identifier (MER-A or MER-B), or by its assigned name (Spirit or Opportunity). Note that Spacecraft 2 is MER-A, because it was the second spacecraft to begin construction, but it became the first spacecraft to be launched.

Initially, MER Operations was run on Mars time. Each Rover had its own team that lived on a Mars day that was about 39 minutes longer than an Earth day. Therefore, each day they started their workday, 39 minutes later than the day before. But this is a whole different topic for another time. For more information about Martian time, please visit: [www.giss.nasa.gov/tools/mars24/](http://www.giss.nasa.gov/tools/mars24/), Goddard’s web site about Time on Mars. Or go to Mars and look at your watch...

Our team comprised MER’s Mission Data Operations Team (MDOT) and dealt with both rovers. For this reason our team did not have to live on Mars time, but was able to live and work on Earth time. Our Team still worked 24 hours a day, 7 days a week, like the rest of the teams, but we were rotating through on a standard 8-hour schedule. Working on the MDOT team, we constantly had to translate time and rover references depending on who stated the time value. It generally depended on which rover team and point of reference they were working and living on. Briefly, the MDOT Team was responsible for keeping an eye on the Ground Data System for MER. The MDOT team is the primary focus for resolving any GDS questions or problems.

Our team for many years has utilized a standard Mission Clock that we deliver to each mission we support. It is present on each Unix computer screen and contains both Pacific Standard Time (PST) and Universal Time Coordinated (UTC), in addition to other useful project

information. For the MER mission we expanded this clock to also include the unique local Mars time for each rover. These times started ticking at the moment that each rover landed. Each rover's time is color-coded, Spirit is **Blue** and Opportunity is **Green**.

The primary length of the MER mission was 90 Solar Days or sols. To be on the safe side, a three-digit sol time was included on the project clock. The rovers are running so well that we have had to update the clocks and several other scripts to contain a four-digit sol instead of the original three-digit sol. This was similar to what the computer industry went through back in the year 2000, remember it was called Y2K? We called our situation sol2k. We are currently up to 1509 sols for Spirit, and 1489 sols for Opportunity. That is over 2 Martian years or 4 Earth years and the rovers are still going. Figure 1 is an image of the MER project clock.

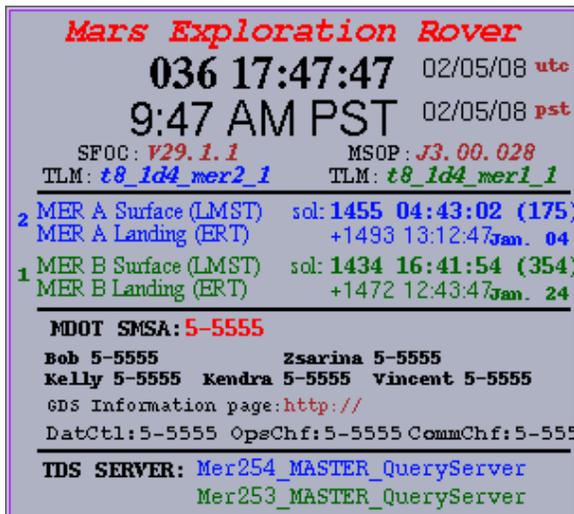


Figure 1. Mars Exploration Rovers Mission Clock

What about the situation where someone says that data is coming down from Spirit at 3:20 LMST, what time is that here on Earth? We created our own simple conversion tool that would convert from Mars time to UTC and vice-versa. Now we would be ready at the correct time for the data flow. Learning which people were on which rover team also helped a lot.

This is just one example of how we have responded to a need, which enabled the Ground Data System to be a more responsive ground system.

### “Wrapping” the Multi-Mission GDS

GDS software should be flexible and have various kinds of options or built-in hooks that allow users to manipulate/morph/transform data as they need to.

Every flight project has users that run the entire range from experienced to non-experienced. We have three types of users. First, those who are experienced users of our GDS. Second, is the repetitive user who uses the GDS everyday, but only to perform one particular repetitive task. The third type is the occasional user that only uses the GDS from time to time. Our approach has been to setup our GDS interfaces for this occasional user. They are the users we are concerned about most. They are the user that may end up continually calling our group for help.

The experienced user is not a quiet bunch, they will let us know what they need, or they will figure it out themselves. If the GDS is simple enough to be operated by the occasional user, the rest of the varied users should have no trouble. But, you can't just focus on one level of user group; you need to provide a GDS that satisfies all the various levels of expertise.

The technique that has provided the most benefit to assisting users interface with the GDS is by wrapping the GDS. The fact is that if a user says, “The GDS should be able to do xyz,” they are generally right. As an example, the GDS should be able to manipulate and represent data in many many many different ways.

The GDS is not infinitely configurable, but this should be the goal of the multi-mission GDS developers. However, due to limited requirements, time, and funding, the developers are unable to implement all the options or hooks that users would want. Therefore, our job is to mold the user interfaces into a form that is focused on a project's particular requirement.

It is our responsibility to; 1) Make sure the multi-mission GDS software correctly supports the data formats for a project and 2) provide small wrapper scripts that allow users to conveniently view their data and get their job done. They should be able to do this without knowing the many options and combinations that the various GDS components are streamed together with.

The goal is not to hide the details; rather the goal is to push the details to the back so the user can focus on the necessary parts (querying data, viewing telemetry) so they can get their job done. The users job is not to be a GDS expert, that job is our job. Users want to see their data; they don't want to know how the data arrives or how the GDS works. Imagine that someone asks: “what time is it?” No one would ever respond with: “Dozens of little gears are hidden underneath the clock's covering that slowly rotate. Each gear drives another gear that moves the next gear.” All the person wanted to know was “what time it is”, not how the clock works, or how the GDS works. They just want to see their data.

This is where the GDS wrapper script comes in. So what is a GDS wrapper script?

On the Dawn project, as an example, we have many small scripts containing several hundred lines of code each, usually written in Perl. Sometimes we use Python, or simply C-shell, the language does not really matter. What matters is that a user can execute these wrapper scripts, utilizing a minimum of options, and up pops their Telemetry Display, or their queried data, or their Spacecraft Event Messages all color-coded and nicely displayed in their format of choice. Remember, our goal is to streamline (aka simplify) the GDS.

Using wrapper scripts makes the project GDS more straightforward for the average user. The expert user can ignore the wrapper scripts we provided and utilize the core options themselves if they desire. The ultimate goal is for the GDS to work for the user, whether they are an expert or an occasional user. The core GDS with all of its options is always available to the user.

We can assist the users in two ways. The first is to make it easy for someone to understand what the wrapper script does. Second, we can anticipate future user needs by keeping the GDS, and the wrappers, flexible. One method to accomplish this is via a hidden option on all wrapper scripts. No matter how many options the wrapper script has, always have one extra option called, -debug.

As users transition from beginner to experienced, they tend to start asking more in depth questions and want to know exactly what the wrapper scripts are doing. This is knowledge we want to share! The easiest way to achieve this goal is to allow the user to execute the wrapper in a debug (or verbose) mode. In this mode the wrapper would print everything it is doing to the screen, such that the user can learn how the multi-mission GDS components are being used.

The delivered multi-mission GDS may be written in many languages and much of it delivered as compiled binaries. This makes sense for the core GDS. However, the wrapper scripts are not core. They belong to the project and are simply tools that build up the arguments for a particular project environment, for a particular capability, and execute it. By keeping them open, we fulfill an earlier goal of not hiding the details.

Another way to assist the experienced user is to be flexible in our GDS deployment, where we listen to the experienced users (they are usually our biggest clients) and add options to our wrappers that give them access to the hooks they need. Whenever possible, new options should be defaulted to the original value, so that a user

who does not specify the new option gets the same behavior as they did before the option was added. We need to be thoughtful as we add options to our wrappers, because if we add too many, then we are back where we started, and the occasional user is lost in a sea of options. If this happens then we have wasted our time by duplicating the same hooks the core GDS already had.

With these wrapper scripts we have streamlined the ease with which a user can get what they need out of the system. In addition, these wrapper scripts prevent us from duplicating simple GDS syntax in multiple locations. If we have one script to query data, and another script also needs to query data, then one can call the other. If we need to be able to perform some GDS action from a pull-down or from a button on a GUI, simply have that pull-down or button call the wrapper script. The next time the multi-mission software is upgraded and a new required argument is added to the Spacecraft Event Viewer, for example, we do not need to add it every place a user can invoke the tool. We just change our one project wrapper script that calls the Event Viewer.

The payoff is that the use of wrapper scripts allows for a consistent way for all users to have only the visibility they want and/or need, which ultimately leads to less user errors and a more responsive project GDS.

The reality of a GDS for any mission is that it is complex, flexible and has many unique parts. We have taken these multi-mission masterpieces and adapted them with scripts and wrappers to enable the most common functions to be non-repetitive. After all, why should a user have to specify their mission (e.g. browser -mission=Galileo) every time they invoke a command? Why should an occasional user be presented with the opportunity to have typos if we can configure it so they do not? We have customized the environment to make things project specific – again to reduce repetitive steps, but more importantly to avoid mistakes.

From the wrapper scripts we now go one level further in simplifying a user's interface with the GDS. We do this via the drop down menu and the remote menu.

### **The Beauty of the Project-Specific Menu**

Most users need a small subset of available GDS capabilities to do their everyday jobs. In addition they do not want to be bothered with an infinite number of options. Sometimes a user will say, "I just want to see my data." Why make them remember a command line syntax? We can take these command line syntaxes and put them into a drop down menu that utilizes our customized environment and wrapper scripts. In that vein, we then put together a drop down menu with these tools

already configured for the user on their project. Figure 2 below is an example from the Mars Exploration Rovers drop-down menu.

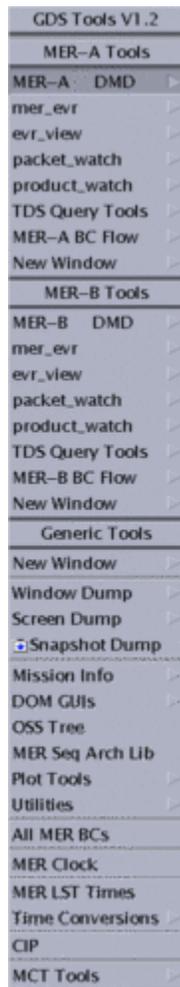


Figure 2. Mars Exploration Rover's drop-down menu.

We have also tried simplifying things to enable key teams to do their jobs quickly, easily, and with fewer errors and shorter training times. For example, the people who command the spacecraft are interested in sending commands, rather than analyzing data. They are permitted to do things not everyone else is able to do. The Spitzer Space Telescope project has a special tool that allows the people commanding the spacecraft to have all their tools brought up properly configured and setup for them to focus on doing their jobs accurately.

Some of our users log in remotely, but the drop-down menu is not available to remote users because the window manager controls the drop down menu. Whether they are logging in from home or from a university half way around the world, we wanted a tool that would provide them the same capability as the drop down menu. We

have created a tool that enables users to bring up most of the same functionality of the drop down menu remotely.

This tool makes available the most frequently used tools that are already customized for the user's project. Everything is x-hosted back to the user's workstation or laptop. Figure 3 is an example of Mars Exploration Rovers remote menu for Spirit.

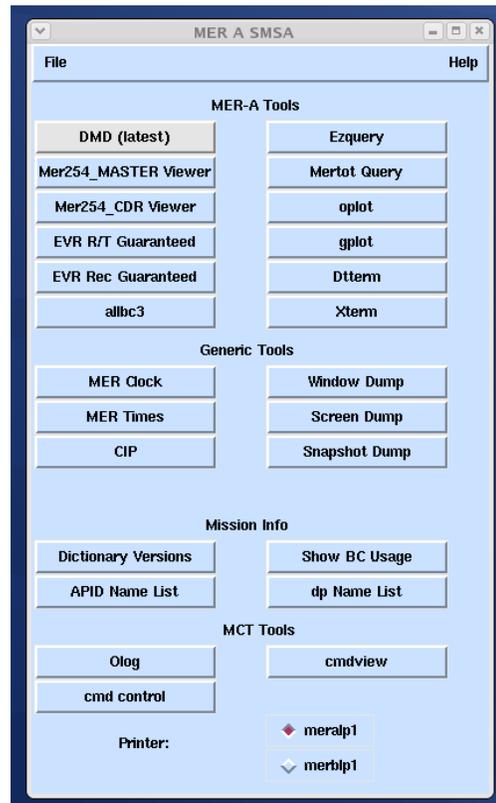


Figure 3. Mars Exploration Rovers remote menu for Spirit

The remote menu for Spirit (also known as MER-A) is blue and lists a group of tools that are setup and customized for Spirit. Opportunity (also known as MER-B) has a similar menu that is green. The coloring was done by choice; so that our users would know which spacecraft they were looking at immediately, without having to read a small text box hidden in a corner. When invoking the vast majority of the buttons from these menus, they in turn bring up tools that we have customized for the specific rover and are color coded to help eliminate confusion. At 3:00 am in the morning anyone can make mistakes. By customizing the commands and color-coding the tools the remote menus make it more difficult for users to make certain errors.

These menus enable users to quickly start the necessary tools. These menus also eliminate some aspects of human error since the wrapper scripts are dynamically invoked

by the menu and are customized and tailored for particular uses.

We are quickly able to diagnose user problems with any tool activated from one of these menus because we know exactly how the tool is implemented. Less training is required because everything is visually displayed and easy to invoke. Besides eliminating a lot of repetitive typing it also reduces the confusion of the user thinking:

- “What was that command-line tool again?”
- “What is that argument?”
- “What is the value of that argument?”
- “Oops, I didn’t mean that value”

And thus saves time, energy, and prevents some errors. The end result is GDS users can focus their time on being responsive to the data the spacecraft sends down instead of thinking about the details of the GDS tools.

### Victor’s Magic Button

Taking this progression one step further, from a drop-down menu to a GUI based remote menu, leads us to “Victor’s Magic Button”, a single “do-everything” button.

A primary focus of our team is to assist in the configuration and setup of the GDS in a project’s testbed environment. A project testbed is an environment where various parts of a project’s hardware and software can be tested. Before testbed testing can begin, the GDS system needs to be configured and activated. This includes starting parts of the GDS that normally a project would not start in flight operations. However, in a testbed environment you do not connect to the Deep Space Network (DSN), you do not have flight operations personnel available to run and maintain your GDS. You have to run it with an ever-rotating group of test engineers. To put it simply, this is not just a single system process to be initiated, but multiple system processes, each with their own special configuration requirements.

Historically, each system was brought up manually, then as time permitted, wrapper scripts were written to help automate activation of each system. This still took time and some manual effort. A testbed lead, Victor, did not want to waste test time doing this initialization for every test every day. “Why can’t you just give me a single button that I can press to start the whole system?” Victor asked. That button or the concept has become known to our team as “Victor’s Magic Button”. Today, that idea has continued to expand and been improved upon. Many of the tools and scripts we use today have undergone similar improvements and enhancements, as users have suggested innovations that they would like incorporated to make their jobs, and lives more productive.

In JPL’s Multi-mission System Architecture Platform (MSAP) Testbed we have what is called the “msap\_menu”. It is more of a GUI than a menu. It allows a user of this unique testbed to start the GDS with a single button push, accepting all the current predefined default configuration values. Alternatively, the user may change a dozen different defaults to values that are unique for their specific testing criteria. For example, this can include using an older command dictionary, or even a new, non-released command dictionary they have just built themselves via the menu.

Figure 4 is an example of today’s magic button on the MSAP project. It’s the button titled: Start DOWNLINK

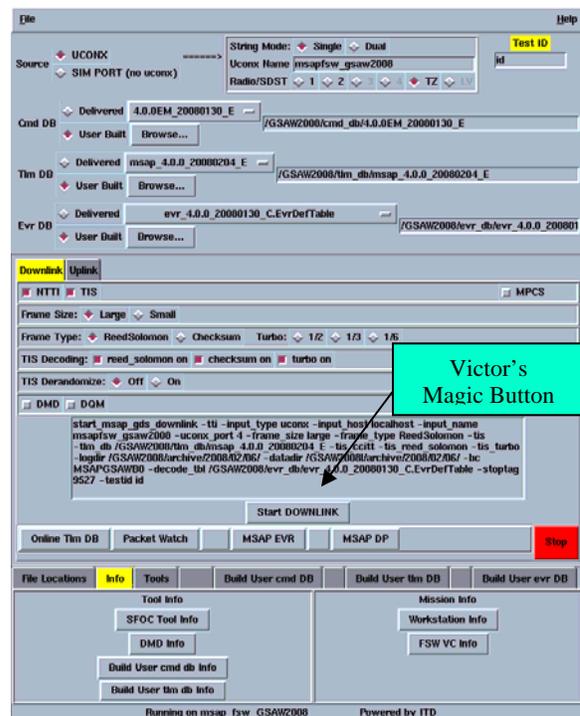


Figure 4. Victor’s Magic Button: Start DOWNLINK

The underlying feature of MSAP’s menu is that the button selection display and associated functionality for each button has been separated from each other into separate processes. This separation of function from button selection has allowed the addition of an entirely new GDS architecture to the menu, to be a simple effort. The user still pushes the same single magic button, but now they interact with a new GDS system.

At this point we have given users: clocks, wrapper tools, menus, and even a magic button. What could be next? How about automating the automation that is already built into the GDS?

## **Automating the Automation**

Many GDS's have some automation features already built in. It is these automation features that allow a GDS to be operationally responsive. One way to create a more responsive GDS is to leverage off these automation features and streamline the current human-in-the-loop processes to be faster and more efficient.

An example of a pre-existing automation feature at JPL is the GDS's common file cataloging system that many projects use to store their flight related files. The designers of this cataloging system decided to add a messaging server that monitors the transactions it performs and distributes messages on certain events. What if we build additional automation leveraging on top of this already automated messaging service?

That is exactly what the early GDS Engineers on the Dawn project did. They thought about the numerous versions of Alarm files, Telemetry Dictionaries, and other required project files. Combining this with the numerous independent operational environments spread across the country and they realized it would be tedious to propagate manually updated files into each environment throughout the life of the project.

Therefore, they implemented software for the propagation of Alarm files and Telemetry Dictionaries. In the case of the Alarm file propagation, the result is that every time a new file is pushed to the Alarm file collection on the common file system, one project workstation in each of the independently different environments sees the updated file and automatically copies it to the local systems. The time from when a team member builds and publishes a new file, until it is available on all project workstations is a matter of minutes. It is the best way to guarantee minimum turn around time from the user triggering the process until the GDS has the new file available for use in operations.

The workflow for the automated propagation of Telemetry Dictionaries is even more involved. The actual Telemetry Dictionary builds are not published at JPL, but at an external contractor site on the other side of the country. The contractor team member takes the latest Telemetry Dictionary files and publishes it to this common file system. However, the format they publish needs modification for it to be usable by the JPL project workstations. The first set of automation sees the newly published files, copies them down, and builds the JPL formatted files, compiles them, and publishes them back up to a different collection in the common file system. Now that the files are usable by the JPL project workstations, one project workstation in each of the different locations sees the updated file and copies it

down. Ultimately a user will be able to sit down at any project workstation and is able to choose this newest Telemetry Dictionary when they start their display or query tools.

Both of these automation processes took some time for the project to implement, but leveraging off the core GDS capabilities made the Dawn project's GDS very responsive, enabling users to receive their project files faster. Much faster now that the human interaction has been removed.

As you have read we have done many things to ease the life of a user. Although, they still do have issues as they interact with the GDS. Here are some issues and how we respond to them.

### **I Can't See My Data. That's OK, No One Else Can Either**

Three user issues we frequently respond to are:

1. "I can't see my data."
2. "Something is wrong."
3. "The Ground Data System isn't working!"

We try hard to be responsive to user questions and concerns. Many times however, the user has done something incorrect and it is easy to figure out why something is not working correctly. However, sometimes it is hard to figure out what they have done. In this case it usually means that the user has really done something unique this time. It is not always the system that actually has a problem. Our GDS at JPL is actually a very reliable system; we know that, so going into a problem, we are quite sure it is not the GDS that has a problem. We know the strengths and limitations of our GDS, and are willing to defend it.

We did not build the ground system, that was done by developers in another section. We are however, the ones left to defend it, or if not defend it, at least get it to work the way the user would like it to work. Sometimes we can respond immediately to the user's question: "I can't see my data, the system isn't working!" With the simple statement: "That's OK, no one else can either, because right now there is no data coming down". In addition to responding to user problems we are also able reassure them about the health of their GDS, and that it is working fine. If we could not respond quickly to their concern about the ground system, then their concern that "the GDS is not working correctly", turns to a reality within their mind. And from there, they tell everyone how bad your GDS system is. Not because it is bad, but because they perceived that it was bad. Part of our accepted task, although it is not in our job description, is to be encouragers of the ground system. Of course sometimes

they cannot see their data because there really is a problem with the data flow or sometimes simply because they have done something wrong in their setup.

Most of the time, the ground system is working correctly. However, sometime, something or someone may not be functioning correctly. The more we can ascertain or demonstrate that the ground system is working correctly, the more people will have confidence in their GDS.

The real challenge is the third statement: “The Ground System isn’t working”. We have come to realize that not all that is deemed wrong is wrong with the ground system. The challenge is to figure out what actually is wrong. It may be the GDS, the user, or something entirely different. Many times when we try and duplicate the user’s problem, everything works fine for us. One example was on our MSAP project. MSAP stands for Multi-mission System Architecture Platform (MSAP), in a nutshell it’s a generic Multi-mission testbed, which allows a new project to walk in the door and immediately begin to be productive in their testing. As part of the processing a user needs to remotely log into another machine and at the prompt, they need to enter their password. This has been done many times and every time it works fine. What this user was experiencing however was rather strange. When they got to the password login prompt, the prompt instead of sitting silently in one place waiting for input from its user, this prompt, became rebellious and continuously scrolled off the screen preventing the user from completing their login. There it was scrolling past on the screen, but what was causing it? It turned out it was not the ground system doing this but it was the operating system. The simple fix was that when the user had started the process, they had put it into the background. Background processes on a Unix System don’t really have a concept of having their own terminal window for displays, so in this case the password prompt kept flying past on the screen.

Once the user learned not to begin the process in the background, everything worked fine. It would have been impossible to determine what was going on over the phone. Many times we will meet with the user to observe their interactions with the system. The extra five minutes it takes for us to walk over to their office or lab, is a lot quicker than spending thirty minutes on the phone doing question and answer dialogs.

How do we manage all of this button, selection, automation, and user issues? Here is how we manage everything.

### **Using a Ticket-less Support System**

We all know that a support system, electronic or not, is widely prevalent in industry to keep track of problems, complaints, requests and many other subjects. The methods and applications used vary greatly, but the main purpose of a ticketed system is to track, open and close requests by users.

The challenging aspect of a ticketed support system is the fact that users must take the time to fill out fields and describe the problem they are experiencing. Not only is this time consuming, but also many times the user is already frustrated and does not wish to spend time typing out details only to be placed in a queue on a ticket system.

Our team uses a form of a ticket-less support system, which has proven to be not only a more efficient system, but also user accepted. In many ticket systems you fill out the form and are lucky if you hear back from them the same day. We generally need to fix a problem as soon as possible. Waiting hours to fix a problem in a testbed that has half a dozen engineers standing around is just not acceptable.

This ticket-less system still allows for internal accountability of calls and issues from users, but also removes the dependency of the user to fill out forms or answer pre-defined questions asked by a standard ticketed system. A common scenario is when a user is attempting to provide details of a problem, but provides incorrect information which leads to longer troubleshooting time. Another common occurrence is when a user will simply state, “The System is not working” or write what they perceive is the problem, but in reality is only a symptom.

A ticket-less support system consists of a user calling in a problem, our team member assessing the situation and determining what course of action may be needed to resolve the problem in real-time. If the recipient of the call cannot recommend a solution over the phone, they commonly will obtain the users information, including name, location and extension. At which point they will then go to the user’s office or location and sit down in front of the workstation to resolve the problem with the user.

We try to always keep in mind that these users are spacecraft or science data analyst’s trying to obtain data from their spacecraft. If they cannot perform their job, bigger problems can arise. This method makes troubleshooting much easier and also gives the user a sense of connection between themselves and those who are there to help them. Meeting with the user gives the user the feeling that they can count on our team to assist them any time a problem should arise. Additionally they know that they will be attended to and not just placed in a support system as a number, which is an annoyance to

most users. This promotes the user to call more often for any problems they may have without any hesitation. These are usually simple questions that can be easily answered, which further reinforces assurance to the user that the GDS is doing what it is supposed to be doing.

When the problem has been resolved, the engineer who fixed it can now enter all data of interest into their own internal problem tracking system. This data could contain time spent on the problem, resolution, symptoms, and any other information. This information may be useful to display the work performed, and as a future reference on similar problems which may be called in.

This system keeps the users happy since they do not have to spend time writing out details of a problem and can talk directly to a support engineer. We do not have an automated answering system; the project personnel do however have our cell phone numbers. Responsiveness is expected and it is provided. This system has proven to be very efficient and cuts down on troubleshooting time for certain issues. The volume of calls may increase due to a caller's lack of hesitation in placing a call. That is because they have a trusting relationship with the support engineers and know that their problems will be resolved in a timely manner without any hassle or annoyances, which is our goal.

That is how we handle tracking. Now let us turn to how do we coordinate and train our team? We have formal cross training on all of our missions, but to give you a clearer understanding of how it works we'll focus on MER.

### **You Are Certified**

During the second extended period of the Mars Exploration Rover (MER) mission in 2005, experienced project engineers were leaving for new projects. This included the GDS analysts (GDSA) in the Mission Data Operation Team (MDOT). New hires and new members were joining MDOT but needed training quickly in order to support the mission effectively. The question was, how?

This on the job training process required time and effort, but time was something we did not have much of. After some brain storming, a certification program was created to accelerate the process. The certification program did not intend to fail anyone but rather to assist new team members in adapting to the project environment and learn the GDSA roll as quickly as possible. All new team members who completed the certification program should be capable of supporting daily mission operations.

The MER MDOT certification program covers three categories: data flow monitoring, MDOT process operations and general user support. In each area, it contains questions related to hardware and software.

There are about two hundred questions of review material. The certification process can take several days to complete. The examinee is required to take some action or give verbal response to each question. The examinee could look for references from either documentations or other MDOT members within a period of time. The examinee that did not answer ninety percent of the questions correctly was required to take the test again. The test was performed by the GDSAs on the MDOT team. It is taken in the MER operations area.

Five new MDOT members have been through this program. They all passed the test within two months. They had learned where to look for procedures and instructions within the MER environment during their training. And their technical GDSA related skills have been sharpened. As a result, they are qualified to support the mission. As we know, personnel rotation is a natural process on a lengthy mission. Effective training enables our team to keep up with the pace of change, hence to become more responsive to this dynamic environment.

With all of this activity it may seem like we are doing a lot and are in more than one location at a time. Truth is we are in more than one place at a time. Here is how we accomplish this.

### **Walking Laptop**

One typical day at work, while being preoccupied in reading project e-mail, a loud ring tone from a deployment engineer's cell phone startled him. It was a member of the testbed crew, "We are doing a hardware compatibility test and can not see the command echo on the ground. Can you come over?" he asked. "I will be there in about five minutes," our deployment engineer replied as he closed the monitor screen of his laptop, grabbing it he ran out of the meeting room...

This is a familiar scenario for a member of our team. We are meant to be everywhere primarily because we need to: A) address GDS issues during real time operations; B) attend formal and informal meetings; C) help project users and subsystem developers.

#### **A. Addressing GDS issues**

In the testbed, ATLO and operations, we represent GDS. Any GDS real time and near real time anomalies, come to us first. We gather first hand evidence of failures, perform trouble-shooting, and make decisions to provide a work

around, or pass a problem along to the subsystem developers.

### **B. Attending meetings**

We attend project GDS meetings to provide technical details to ensure GDS design, processes and schedules make sense. At the same time, we acquire essential information from those meetings. The Mars Science Laboratory (MSL) mission is in the stage of integrating spacecraft instruments in the system testbed, as well as early ATLO preparation and deployment. A new and developing GDS is being deployed to the testbed for operation. Because using these new services and subsystems there is a learning process. Not only do we have to go to GDS system level meetings, but subsystem level meetings also. These meetings can be anywhere. It might be in the MSL project area, which is about a seven-minute walk from our offices. It might be in a public conference room in our building, or somewhere else. It might even be at a remote site, such as at the Cape.

### **C. Helping project users and subsystem developers**

Currently for MSL, project users and subsystem developers are scattered around the JPL campus. Sometimes, we visit their offices to help project users solve GDS related problems, or provide subsystem developers project specific information. This face-to-face interaction in general can speed up the problem solving/development process.

Constantly multi-tasking and working from place to place, we must be skillful in time management as it is a big challenge for us. To save time, laptops become our best friends. We bring our laptops everywhere we go. During a meeting, we can take notes on our laptops; update our process documents and procedures if any changes occur;

during a long design discussion, when the topic is irrelevant to us, we sometimes utilize the time for a script development or modification. Since conference rooms have wireless network coverage, we can also respond to users' requests via email during the meeting. For instance, the Simulation Support Equipment (SSE) team has just delivered a new command and telemetry dictionary set and needed us to make it available for testbed use. They notify us via email. Upon reading it, we remotely logged onto a project workstation via our laptop to complete the deployment process. When we are in a testbed in support of a test, if all the workstations are occupied, we can use our laptop to read online the testbed activity reports, fill out online anomaly reports, even remotely log onto a testbed workstation to monitor a test run.

As deployment engineers, we need to be wherever we are needed, fast. Providing timely responses and solutions, enables us to form close relationships with the GDS, and with its users and developers. A laptop cannot walk itself. We make it a walking laptop.

A key component for a responsive Ground Data System is responsive people that are able to hear the heartbeat of a project's needs and respond to its ever-changing requirements.

### **Acknowledgements:**

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.