A custom programming service

Active Storage Management with Operations Manager/DFM

Programming services to extend Operations Manager capabilities for proactive identification, reporting, and lights-out event handling

Product conceived & developed by Nolan Milton Vega (PRISMATICS, LLC)

KEY FEATURES

Eliminate human error
Human error causes 32% of application downtime – eliminate it. Eliminate repetitive storage administration tasks by automating the uniform handling of responses to events: meet SLAs; reduce downtime; enable business compliance

Get more from your investment
Operations Manager can be an ACTIVE storage framework for your NetApp environment – NetApp Professional Services can unlock its power

Improve efficiency
Increase productivity and reduce administrative overhead by minimizing disruptions, maximizing business efficiency, and optimizing system administration efficiency

Lower risk
Enable customized enterprise-wide configuration management, distributed policy setting, custom reporting and auditing, intelligent provisioning, and rapid and reliable event response

MAXIMIZE YOUR MANAGEMENT INVESTMENT

Out of the box, Operations Manager provides an extensive array of pre-defined choices for alerts and reports:

- Hundreds of event types
- Hundreds of reports

These cover the most common requirements, *however*, Operations Manager can provide much more.

The DFM database and the archive stores that support Operations Manager contain data on every aspect of your NetApp storage environment. In customizing solutions for you, NetApp Professional Services utilizes all of the performance counters and status flags maintained by Data ONTAP. By using a collection of APIs and other interfaces, we can integrate scripts, and even other applications.

All of this can be used to create specialized information stores, alarms and reports.

- Build your own DFM data warehouse
- Develop views to enable fast problem-solving for maximum availability and improved resource and capital utilization
- Create targeted responses to events that take remedial action instantly by adding customized “event handlers”

LET US CUSTOMIZE YOUR MANAGEMENT EXPERIENCE

We tap into Operations Manager to monitor those key performance indicators that are important to *you*:

- Extend monitoring to a wide range of custom performance and threshold counters.
- Generate detailed reports specific to your needs and on your schedule through custom programs.
- Initiate custom alerts and thresholds to meet your SLAs.
- Develop a framework for actionable and automated response to common problems and requests.

Using OM as a base, we build a personalized development framework, enabling our developers to create custom monitoring, event handling and reporting programs:

- Access to over 100,000 performance and threshold counters maintained within ONTAP.
- Extensive list of built-in OM/DFM commands usable in scripts.
- *Event handlers* – not only view incoming events, but also *ACT* on them the way you want, without manual intervention.

THE CHALLENGE

OM can access virtually everything that affects your storage systems. Any task that can be described as a process or workflow can be automated with carefully written plug-in programs. We can make “lights out” management a reality for you.

PROTECT PRODUCTION PERFORMANCE

Typical problem: DEV and PROD volumes are contained within the same aggregate. How can you protect the performance of the PROD environment volumes if the DEV volumes exceed an I/O limit?

- Create custom events and map them to customized event handlers. These handlers can initiate policy-driven actions without user intervention.
- Create an auditable log file of any changes initiated by the event handler.

HOW WE DO IT

THE RAW MATERIAL FOR PROACTIVE MANAGEMENT

OM provides many different interfaces into the wealth of data collected from NetApp storage systems – counters, APIs, standard and custom fields. We use any number of these to provide the actions you require:

- **Event-driven:** trigger action-taking scripts based on incoming event types.
- **Scheduler:** trigger scripts *ad hoc*, or at scheduled times.
- **Direct ONTAP (remote) command interface:** initiate remote ONTAP commands to natively query and/or modify your storage environment.
- **DFM ONTAP counter view/container:** gather and store ONTAP counter values at pre-defined intervals on a per filer/per counter basis. Make a searchable resource for historical (trending) data.
- **Custom performance and threshold definition:** extend the many predefined event types to create custom events, triggering actions based on very specific threshold conditions. Tie the event type to an alarm, report, or action.
- **Pre-configured and custom DFM reports:** use preconfigured or custom reports as actionable input to custom scripts.
- **Protection Manager:** manage critical SnapMirror & SnapVault relationships – use event handler scripts to ensure business continuity.

EXAMPLE

THE PROBLEM

A multimedia company uses NetApp storage resources to protect a growing number of video clips. The storage system must support the accelerated increase in the number of these very small files.

The growth in these file systems cause inodes to be exhausted in volumes on a regular basis. From a kernel perspective, it is inefficient to pre-allocate inodes. However, the manual work to add the inodes just-in-time taxes the company's administrative resources.

THE SOLUTION

Inode status (Almost Full, Full, Normal) is an event type monitored by OM/DFM. We use this event type as a trigger for a customized script that automatically increases number of available inodes.

(The following sequence corresponds to the figure below).

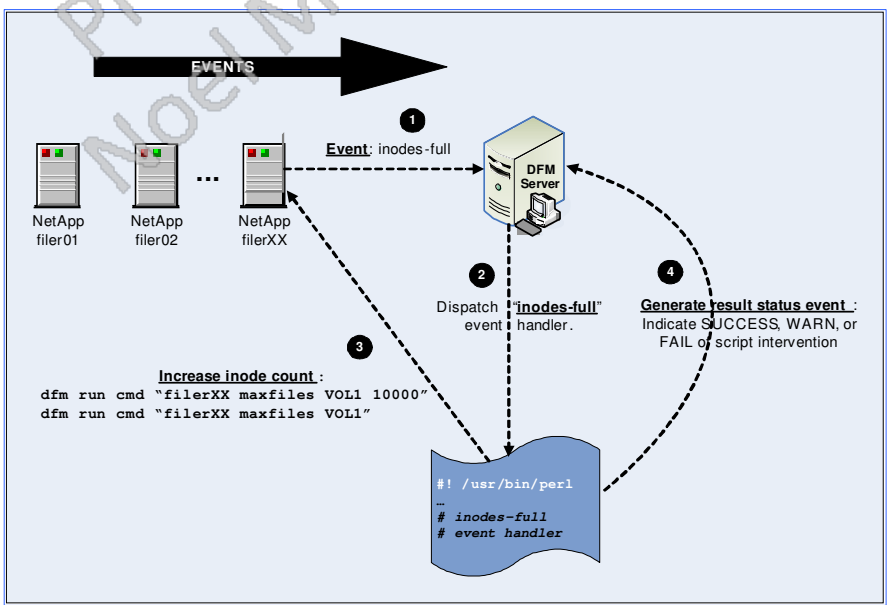
(1) Event: **VOL1** on **filerXX** experiences an exhausted inode condition.

(2) The condition triggers an "inodes-full" (or "inodes-almost-full") DFM event.

(3) The event triggers the handler script developed for this condition. The script will attempt to increase the number of inodes for **filerXX:VOL1**.

(4) Finally, the script verifies the attempted inode increase (SUCCESS, WARN, FAIL), and generates its own event to notify the NOC of this status.

All event types can be handled in a similar fashion.



STRATEGIC PLANNING AND ANALYSIS IS CRITICAL TO SUCCESS

NetApp Professional Services leverages proven methodologies, tools and best practices to assess, consult, deploy and manage the right storage management solution for your data center.

ASSESS

We work with you to identify and characterize your data center and your business environment to determine your pain points, priorities, policies and processes, including discovery, identification and mapping to your business requirements and objectives.

CONSULT

The NetApp **OM Customization Service** provides a detailed blueprint for the OM customizations identified for implementation. The design defines the capabilities to be developed and deployed into your management environment.

DEPLOY

NetApp Professional Services develops any scripts or programs needed to automate OM actions for your environment, without disruption to production activities. You agree on the timeline and priority for each new development.

MANAGE

Storage management is a journey. Your first customization engagement will spark ideas for automation of other functions. Organizational priorities may change. New goals may cause your infrastructure to change.

NetApp Professional Services can be re-engaged at any time to provide on-going consulting services to aid you in modifying and adding to your custom management structure.

Contact us to see how NetApp Operations Manager Enhancement Services can help you address the challenges of managing an increasing amount of storage with diminishing resources, while improving efficiency, lowering risk, and ensuring compliance with your goals. **E-mail** nmvega@prismalytics.io

Example – proactive event handling: Inodes exhausted on a volume. Customized script triggers an automatic increase in inodes available.

Product conceived & developed by Noel Milton Vega, PRISMALYTICS, LLC

Ops Manager / DFM

Customization through programming

Noel Milton Vega
Consultant
PRISMALYTICS, LLC.
(formerly Rensselaer Technology Group, Ltd.)

Product conceived & developed by
Noel Milton Vega (PRISMALYTICS, LLC)

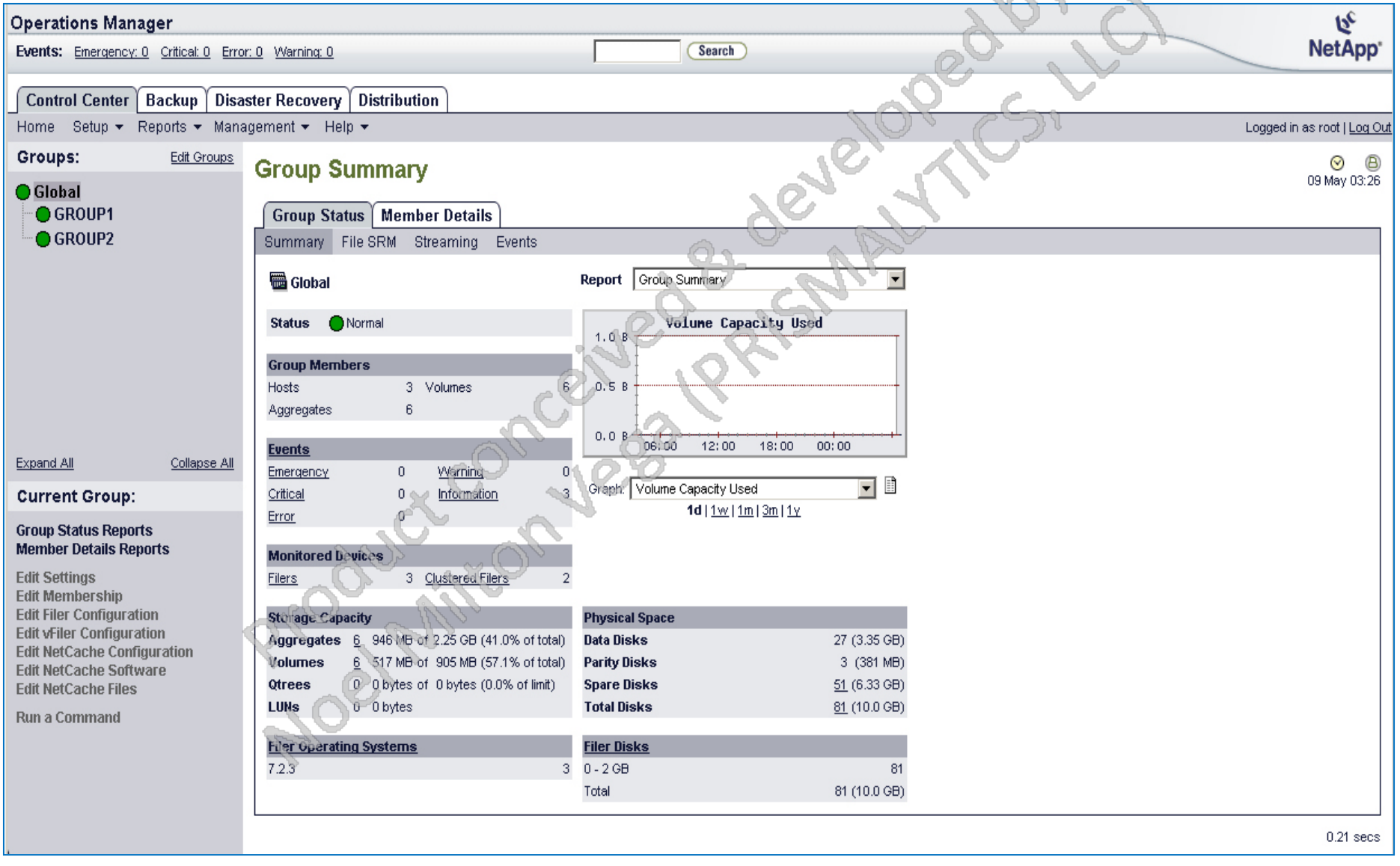
Motivation: Typical Customer Question/Scenario

- **Customer requirement**: A DEVELOPMENT volume and several PRODUCTION volumes are contained within **the same Aggregate**. To ensure the performance of the PRODUCTION environment volume, we want to "offline" the DEVELOPMENT volume if it exceeds 86,000 I/O's per second. Can I use OM/DFM to accomplish this programmatically? YES.

Out of the box, OM/DFM is a passive web based data collection, reporting, and notification tool. Advanced use of DFM interfaces to hands-off actively manage an environment, requires custom programming against OM/DFM. Various OM/DFM CLI interfaces (described later), enable customers to do this.

Product conceived & developed by
Noel Mitton (era/PRISMATECH, LLC)

Limitations in the OM/DFM Web User Interface (UI)
CLI enables more fine-tuned/automated control



Product Conceived & developed by Noel Milton Vega (PRISMALYTICS, LLC)

OM/DFM Customization Framework

- DFM provides a framework of **CLI based interfaces** allowing developers to tap data stored in DFM to create custom **monitoring**, **event handling**, and **reporting** programs.

- DFM has access to hundreds-of-thousands of ONTAP performance and threshold counters, against which scripts can base intelligent action taking decisions. A typical counter is defined by the following sex-tuple:

filerName:objType:instance:counter:label1-name:label2-name

Example:

nas01:volume:vol5:nfsv3_read_ops:label1-name:label2-name

Product conceived & developed by Noel Milton Vega (mNEMALYTICS, LLC)

OM/DFM CLI COMMANDS (a partial list)

- **dfm aggr** adjust the list of managed aggregates
- **dfm alarm** create, modify, delete, and list alarms
- **dfm autosupport** view and send autosupport transmissions
- **dfm backup** manage backup copies of the database
- **dfm config** manage configuration of Filer and NetCache appliances
- **dfm database** configure the embedded database
- **dfm detail** get details about a particular database item
- **dfm diag** get diagnostic information
- **dfm event** acknowledge, delete, generate and query details of events
- **dfm eventType** list, modify event types
- **dfm filer** edit and list the managed filers
- **dfm graph** create graphs of data over time
- **dfm option** adjust global options
- **dfm perf** gets performance information
- **dfm qtree** edit and list the managed qtrees
- **dfm report** manage reports
- **dfm run** manage the execution of remote commands
- **dfm schedule** create and manage schedules
- **dfm script** install and manage scripts
- **dfm search** search the database for matching items
- **dfm service** set up, start, and stop embedded services
- and more ...

Product conceived & developed by Noel Milton Vega (PRISMALYTICS, LLC)

Script triggered by:

(1) A OM/DFM alarmed event
(2) Scheduled job
(3) Manually run by user

```
#! /usr/bin/perl -w

[ ... ]

 > dfm event detail eventID

[ ... ]

 > dfm perf view retrieve "NFSv3_Latencies" "2008-4-6 6:55:00" \
                                             "2008-4-6 7:00:00"

 > dfm report view volumes-capacity

 > dfm run cmd -t 20 filer01 "some OnTAP command"

 > dfdrm mirror [break|quiesce|update|...]  …

 > dfm event generate ...

[ ... ]

exit
```

Product conceived & developed by Noel Nitton Vega (PRISMALYTICS, LLC)

The seven data CLI interfaces into OM/DFM

Product conceived & developed by
Noel Milton Vega (PRISMALYTICS, LLC)

The CLI based interfaces

- **Event driven interface.**
  - Incoming events trigger action taking scripts based on the event type.
  - This method is EVENT driven versus POLL or SCHEDULE driven.
    - `dfm alarm …`
    - `dfm eventType list`
- **Scheduler interface.**
  - The DFM scheduler can be used to schedule the running of scripts (like cron).
  - DFM imposes no limitation on what the scripts do (it is arbitrary).
    - `dfm script …`
- **Direct filer OnTAP (remote) command interface.**
    - `dfm run command …`

```
root# dfm run cmd -t 20 filer01 version
Stdout: NetApp Release 7.2.3: Thu Jul  5 08:54:40 PDT 2007
```

Product conceived & developed by Noel Milton Vega (PRISMALYTICS, LLC)

OM/DFM Interface types

- **DFM OnTAP Counter view/container interface.**
  - Can be configured to gather and store ONTAP counters values at pre-defined intervals on a per-filer / per-counter basis.
  - Counters can be subsequently queried and to monitor the environment, generate reports (e.g. trending), or take corrective action.
  - `appliance-name:obj-name:instance-name:counter-name:label1:label2`
    - `filer01:volume:vol0:nfsv3_read_ops:label1-name:label2-name`
    - `filer02:nfsv3:nfs:nfsv3_read_ops:label1-name:label2-name`

    - ❑ `dfm perf view [ … ]`
    - ❑ `dfm perf counter list filer:*:*:*:*:*`

Product conceived & developed by Noel Milton Vega (PRISMALYTICS, LLC)

The CLI based interfaces (cont.)

- **Custom Performance & Threshold definition interface.**

Over 360 pre-defined event types out of the box. However, customers can also define custom performance thresholds to trigger events for very specific conditions.

- **dfm perf threshold create […]**

The following example creates a threshold that will trigger an event if the average I/O latency of vol0 on filer1 exceeds 200us for 2 seconds.

```
root# dfm perf threshold create -o filer1:/vol0 -c volume:avg_latency \
                                -b upper -t 200 -d 2 -u microsec

root# dfm perf threshold list
ID  ObjId  OjbName       Counter              Type   Value  Unit  Intrvl Status
--  -----  ------------  -------------------  -----  -----  ----  ------ --------
•  72    filer1:/vol0 volume:avg_latency  upper  1      usec   2      enabled

root# dfm eventType list | egrep "^perf:|^Event "
Event Name                        Severity  Class
perf:volume:avg_latency:high      Error     perf:volume:avg_latency
perf:volume:avg_latency:low       Error     perf:volume:avg_latency
perf:volume:avg_latency:normal    Normal    perf:volume:avg_latency
```

eventType Name format: **perf:<counterGroupname>:<counterName>:high|low|normal**

This new eventType can be tied to an alarm/script to take corrective action.

Product conceived & developed by Noel Wilton Vega (PRISMALYTICS, LLC)

The CLI based interfaces (cont.)

- **Pre-configured & Custom DFM reports interface.**

  261 Pre-configured reports; and the ability to define custom reports that can be used as actionable input to custom scripts.

  - `dfm report list`
  - `dfm report [view | create | modify] …`

  Example: "volumes-capacity" report.

```
root# dfm report view volumes-capacity
Object ID Volume  Aggregate Filer/vFiler          Used   Total  Used (%)
--------- ------- --------- --------------------- ------ ------ --------
72        vol0    aggr0     simNode1.localdomain 152060 247696     61.4
88        vol0    aggr0     simNode3.localdomain 148464 247696     59.9
80        vol0    aggr0     simNode2.localdomain  90504 164608     55.0
116       rootVOL AGGR1     simNode3.localdomain  11832 246784      4.8
Totals                                          402860 906784     44.4
```

Product conceived & developed by Noel Milton Vega (PRISMALYTICS, LLC)

The CLI based interfaces (cont.)

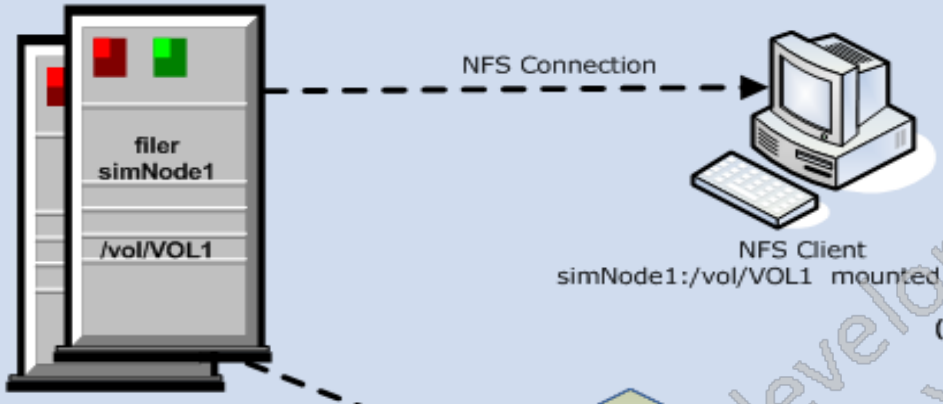
- **Snapmirror Interface (Protection Manager).**
  - Dedicated dfpm(1M) protection manager commands available to create and manage snapmirror relationships.
  - This command can, for example, be integrated into an event handler script to update a mirror if the LAG time between source and destination becomes too large.
    - `dfdrm policy create -t [template|async|sync|failover] name`
    - `dfdrm mirror [initialize|list|break|quiesce|update|...] …`

Product conceived & developed by Noel Milton Vega (PRISMATICS, LLC)

A real world example:

Programmatic (automated), just-in-time, resolution of a problem

Product conceived & developed by
Noel Milton Vega (PRISMALYTICS, LLC)



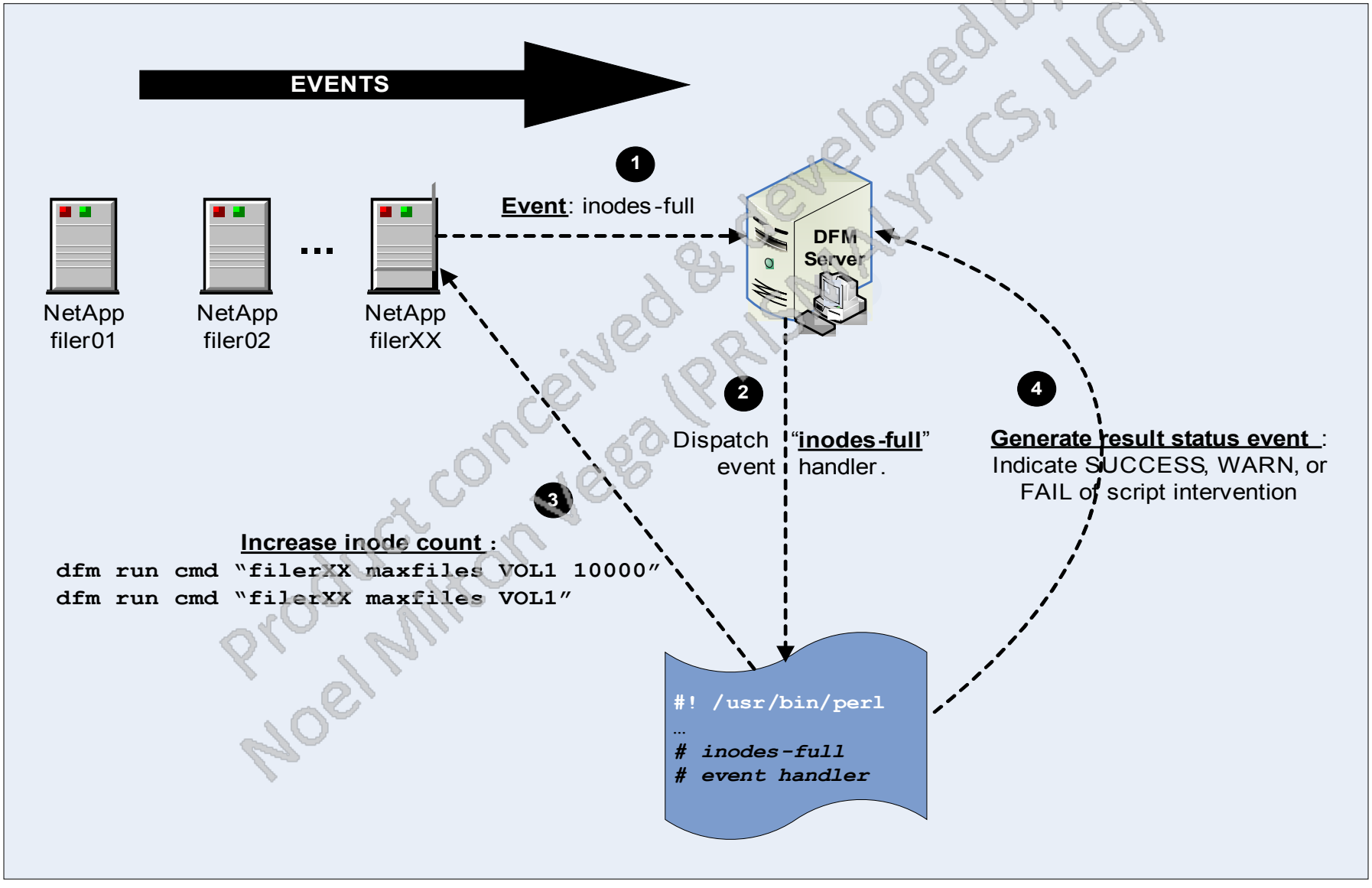

```
(Application WRITES)

  touch /mnt/file01
  touch /mnt/file02
  touch /mnt/file03
  ...
  touch /mnt/file3272

(Application PAUSE)
```

```
START
simNode1> df -i
Filesystem              iused      ifree  %iused  Mounted on
/vol/vol0/               7331      12659     37%  /vol/vol0
/vol/VOL1/                100       3272      3%  /vol/VOL1

EXAUSTED INODES
simNode1> df -i
Filesystem              iused      ifree  %iused  Mounted on
/vol/vol0/               7331      12659     37%  /vol/vol0
/vol/VOL1/               3273       3272    100%  /vol/VOL1

RESOLVED
simNode1> df -i
Filesystem              iused      ifree  %iused  Mounted on
/vol/vol0/               7331      12659     37%  /vol/vol0
/vol/VOL1/               3372       1676     67%  /vol/VOL1
```

Product conceived & developed by Noel Milton Vega (RISMALYTICS, LLC)

Real customer example

inodes exhausted on a volume, triggers an automated increase of inodes.



Product conceived & developed by Noel Milton Vega (PRIMALYTICS, LLC)

Real customer example (Cont.)

inodes exhausted on a volume, triggers an automated increase of inodes.

- (1) Event: VOL1 on filerXX experiences an exhausted inode condition.
- (2) That condition triggers an "inodes-full" and/or an "inodes-almost-full" DFM event.
- (3) DFM, being previously configured to run a particular script when this event type occurs, run's the handler script for the "inodes-full" condition. The script will attempt to increase the number of inodes for "filerXX:VOL1".
- (4) Finally, the script verifies if the attempted inode count increase succeeded (SUCCESS); partially succeeded (WARN); or was unsuccessful (FAIL). It subsequently generates its own event to notify the NOC of this status.

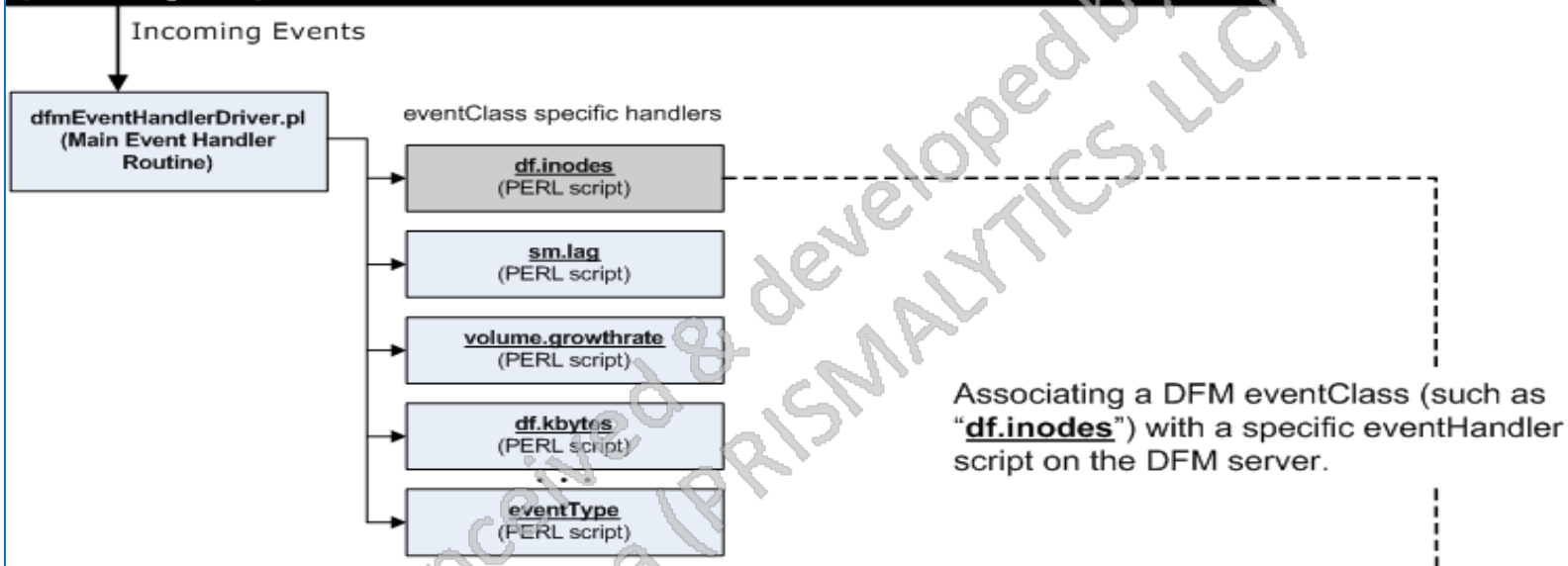
That completes the handling of the event. All incoming events, irrespective of type, are handled similarly.

Product conceived & developed by Noel Wilson Vega (PRISMALYTICS, LLC)

Example of a OM/DFM Event Handler Development Programming Framework

Product conceived & developed by
Noel Milton Vega (PRISMALYTICS, LLC)

```
root# dfm eventType list
Event Name                          Severity     Class
----------------------------------- ------------ ------------------
inodes-almost-full                  Warning      df.inodes
inodes-full                         Error        df.inodes
[ ... snip ...]
```



A Unix directory representation of the above plug-in structure.

```
root# find /opt/NTAPdfm/script-plugins/eventDrivenScripts.d
/opt/NTAPdfm/script-plugins/eventDrivenScripts.d
/opt/NTAPdfm/script-plugins/eventDrivenScripts.d/dfmEventHandlerDriver.pl

/opt/NTAPdfm/script-plugins/eventDrivenScripts.d/HANDLERS.d/
/opt/NTAPdfm/script-plugins/eventDrivenScripts.d/HANDLERS.d/sm.lag
/opt/NTAPdfm/script-plugins/eventDrivenScripts.d/HANDLERS.d/df.inodes
/opt/NTAPdfm/script-plugins/eventDrivenScripts.d/HANDLERS.d/df.kbytes
[ ... and so on ... ]
```

```
root# dfm alarm create -C df.inodes -s /pathTo/dfmEventHandlerDriver.pl \
                       -u root -h inodes-full -r yes -i 5

root# dfm alarm create -C df.inodes -s /pathTo/dfmEventHandlerDriver.pl \
                       -u root -h inodes-almost-full -r yes -i 5
```

Product conceived & developed by Noel Wilson (PRISMALYTICS, LLC)

A DFM sample program: Auto-grow exhausted inodes

```
#! /usr/bin/perl -w
#
##############################################################################
# Script Name.....: dfmMakeInodes.pl
# Author..........: Noel Milton Vega (
##############################################################################
use strict;
use warnings;

# ========================================================
# Define Global variables.
# ========================================================
my $increaseFactor = "1.5";
my $DFM = "dfm";
my $scriptName = "dfmMakeInodes.pl";
my ($eventID, $dfmHostName, $dfmHostID, $dfmSourceID, $dfmSourceName, $dfmSourceType, $dfmEventSeverity);
my ($eventLine, $eventType, $dfInodesPercent, $dfInodesTotal, $dfInodesUsed, $eventArguments, $ojbFullName);
my ($newInodeCnt, $volname, $result, $eventAcked);
# ========================================================

$eventID       = $ENV{DFM_EVENT_ID};
$dfmHostName   = $ENV{DFM_HOST_NAME};
$dfmSourceName = $ENV{DFM_SOURCE_NAME};
$volname = (split(/\//, $dfmSourceName))[1];

# ================================================
# Loop through the event detail output, and store
# only details we'll need to determine what to do.
# Note: I'll create a global way of handling
# errors.
# ================================================
open (PIPE, "${DFM} event detail $eventID |") or
  eventGen ($dfmSourceName,
        "$scriptName: Script runtime error encountered during handling of eventID $eventID: $!",
        "error");

while (defined ($eventLine = <PIPE>))
{
   next if $eventLine =~ /^$/;
   chomp $eventLine;
   ($eventLine =~ /^eventType\s.*/)       && ($eventType       = (split(/\s\s*/, $eventLine))[1]);
   ($eventLine =~ /^dfInodesPercent\s.*/) && ($dfInodesPercent = (split(/\s\s*/, $eventLine))[1]);
   ($eventLine =~ /^dfInodesTotal\s.*/)   && ($dfInodesTotal   = (split(/\s\s*/, $eventLine))[1]);
   ($eventLine =~ /^dfInodesUsed\s.*/)    && ($dfInodesUsed    = (split(/\s\s*/, $eventLine))[1]);
   ($eventLine =~ /^eventArguments\s.*/)  && ($eventArguments  = (split(/\&/,    $eventLine))[1]);
   ($eventLine =~ /^ojbFullName\s.*/)     && ($ojbFullName     = (split(/\s\s*/, $eventLine))[1]);
```

Product conceived & developed by Noel Milton Vega (PRISMALYTICS, LLC)

Thank You

Noel Milton Vega
Consultant
PRISMALYTICS, LLC.
(formerly Rensselaer Technology Group, Ltd.)
eMail: nmvega@prismalytics.io
Web: www.prismalytics.com

Product conceived & developed by
Noel Milton Vega (PRISMALYTICS, LLC)