Tivoli ➠ ViewStar Integration

Preliminary

Created:	Bill Plein	June	1997
Revised:	Tedy Shalev	July	1997

TABLE OF CONTENTS

Page

1.0 INTRODUCTION

Tivoli is Halifax's choice for Enterprise Management Software. Halifax has requested that Mosaix investigate the potential monitoring the ViewStar system resources, by Tivoli. The Tivoli software can, among others, remotely monitor systems, deliver software, schedule and activate tasks, compile data, etc.

This integration project attempts to resolve and define those areas of the ViewStar product that can be integrated with the Tivoli management solution, and to provide a mechanism for that integration.

The challenge was to provide Tivoli access to ViewStar resource information, without a detrimental affect on the routine operation of the Business Centres production.

2.0 FEASIBILITY STUDY

The following possible information sources were considered:

1. ViewStar System Reports - available through the System Tools module.

2. ViewStar Queue Monitor Report - available through the Queue Monitor Utility.

3. ViewStar Custom Reports - developed using ViewStar Script.
    3.1. Supplants online lookups in Configure, Document Locate, System Tools.
    3.2. Adds reporting for resources that are not easily administered online.

4. ViewStar Log files - tracked by the LogWatch Utility.

5. Operating Environment Resources (non-ViewStar)
    5.1. Native NT monitoring of disk space
    5.2. Native Oracle tools for RDBMS management
    5.3. Network monitoring of LAN/WAN (SNMP, etc.)

In general, a System Administrator will use information from all the above resources to manage a single ViewStar system. In the specific case of the Halifax Business Centres installations, there is a need to leverage the administration across multiple systems such that performance monitoring can be administered from a central site.

Resource depletion is a known common cause of any workflow system failures. Tivoli's scheduled remote monitoring could provide early warnings and send alerts upstream, such that proactive action can be taken. However, such implementation requires the integration of Tivoli with ViewStar.

The integration solution had to address the following:

- **ViewStar Reports** - through a Reporting/Tasking mechanism. This would allow Tivoli to launch a request for a specific report or sets of reports. The request would be interpreted by ViewStar and the appropriate task, generating the required report would be launched. The resulting report would then be posted to a predefined directory on the File Server. The reports will be Tivoli readable.
- **ViewStar Log files** - a LogWatch Messaging mechanism, providing Tivoli with messages that can be defined through the standard LogWatch filters and alerts. This messaging mechanism will be implemented through a Tivoli supplied DLL.
- **Native Monitoring** - it is assumed that Tivoli will handle the Operating Environment resources natively, with no required input from ViewStar.

This document concentrates on the subject of ViewStar Reports.

3.0 THE METHOD

Many ViewStar resources can be administered through standard reports, custom reports or, where all these fall short, by using the application interface of the standard ViewStar modules. Examples of typical resources that a system administrator may take interest in are:

- Queue Report (count all, or count with status busy, locked, error etc.)
- Data Directory Allocation and Remaining space, includes File Server volume space and internal ViewStar thresholds.
- Cache Directory Allocation and Usage.
- Optical Volumes information, eg. Volumes names, access and space remaining.
- Changes in ViewStar Users and Profiles.

When examining methods for Tivoli to monitor the state of these ViewStar resources, we rejected the "Direct Inspection Method", described below:

> *Direct inspection of the ViewStar system is reading files and tables directly in order to measure ViewStar resources. Tivoli, with its ability to call command line files, could launch just about anything (command line SQL tools, TESTIT.EXE, etc.)*

This method was abandoned due to the potential for blocking-actions which could adversely affect the ViewStar system. In addition, this method was bound to overlook long term supportability, since ViewStar internals, database schema and file formats could change, with future software releases.

We chose an integration method through a "Reporting/Tasking" mechanism. This has been pursued as a "proof of concept", using the file system to emulate an In Box, Out Box messaging style communication.

4.0 FILE-BASED MESSAGING

4.1 OVERVIEW

One benefit of using an “indirect” file based messaging system, using Input / Output mail-boxes, is that it provides the ViewStar system with full control over those activities that may be triggered at Tivoli’s request. Furthermore, Tivoli’s interference with the routine operation of the production system is limited to a pre-determined set of reports that may be delivered at a frequency that is determined by a ViewStar Process Agent configuration.

As the system and business administrators find areas that they wish to automate through scheduled tasks, those tasks could be invoked by Tivoli, rather than the Process Agent Schedule directly. This would allow all the systems to ensure mirroring of their scheduled tasks *where such mirroring is appropriate.*

4.2 OPERATION

The “Tivoli Input Box” is a pre-defined sub-directory, under ViewStar’s dataroot, namely <dataroot>\Tivoli\INPUT. The only entry in this mail box that ViewStar will be monitoring is an ASCII text file named Tiv_Task.INP. This file is created and/or exclusively updated by Tivoli. A ViewStar task is scheduled to run on a Process Agent (dedicated to Tivoli?). This task probes for existence of this input file. Upon detecting this file the ViewStar task copies it to a work area and deletes the original file. It then interprets each line in the file as a request to generate a report. Each line is expected to contain a list of two strings (or more, if in future there is a need to pass more arguments to the ViewStar task). The first string nominates the ViewStar task (the file name) that will be invoked. The task name is limited to a maximum of eight characters, also bound by the traditional DOS file naming convention. The second string is the name of the expected report output file. The report file name is in 8.3 format, ie. the full file name may be defined, including extension.

Appendix B describes the Tivoli input file (Tiv_Task.INP) format and contents.

The “Tivoli Output Box” is assigned as a pre-defined sub-directory, under ViewStar’s dataroot, namely <dataroot>\Tivoli\OUTPUT. Tivoli expects the resulting report file(s) to be posted to its “Out Mailbox. It is the Tivoli software responsibility to manage the naming conventions of these report output files and to appropriately administer archiving and distribution of these files.

4.3 VIEWSTAR SCHEDULED TASK

The Tiv_Task.VFX or Tiv_Task.FSL script file must exist in a special Tivoli task directory. This script includes a generic task spawner, that will invoke the tasks that were nominated in the Tiv_Task.INP file, by Tivoli.

The concept here is that Tivoli appends to the Tiv_Task.INP file for each report requested, and the Tiv_Task script scrapes this file and runs the scripts. The script tiv_Main found in Tiv_Task is set to run on a Process Agent, at appropriate intervals (once every minute ?)

> *Note: Triggering a request to run a task through an SQL database could provide an alternative that could reduce potential file locking problems.*

5.0 LEVERAGE

The real strength in the Reporting/Tasking is not in the simple reports themselves. The strength comes with the ability of the Tivoli system to store this information, and process it to compile further details such as historical and statistical comparisons. This could be taken a step further, where day-to-day changes in the system could be tracked, and the “delta” or difference in a particular resource could be tracked where applicable. An example would be locked documents. If a rapid change in numbers of locked documents were to occur, this could raise a flag for further investigation by the administrator.

So certain reports will be passed on to the administrators without further analysis by Tivoli, while great benefit could be had by having Tivoli do further analysis on certain resources, like disk space, allowing extrapolation of the resources state for some period in the future.

To take this one level higher, comparisons between systems could also be enlightening. This could highlight differences in the way each Business Centre processes its workflow, perhaps highlighting problem areas or uncovering user efficiencies that could be spread or taught to users on other systems.

6.0 TRIGGERS AND THRESHOLDS

Any given ViewStar system can be thought of as a living, evolving system. While the Business Centres, might have an identical workflow map and library structures, their rate and cycles of workpackets will be unique. For example, Business Centre “A” might typically process 2000 items per day, while Business Centre “B” might only do 500. This clearly indicates that alerts on any thresholds would have to consider local volumes. Similarly, the two systems, despite the huge difference in absolute size, may have similar absolute fluctuations in resources, i.e., the peaks and valleys in queue counts may be 100 workpackets in absolute measures.

As a result, each system will require individual triggers and thresholds for each resource that one may want to oversee. These thresholds are a matter of “taste” for the local System Administrators, and could need tuning and development through experience and over time.

7.0 SYSTEM TOOLS REPORTS

There are a number of standard ViewStar reports available through the System Tools module. Many of these could be triggered from script and integrated into this system. Some of these reports require user intervention and will therefore not run as a task on a Process Agent. If needed these will have to be developed using Script.

The following examples are standard ViewStar reports that are activated directly from a Script run by a Process Agent.

7.1 USERS

Creates a USERS report (refer to System Tools), containing a list of users that were defined through ViewStar’s Security module.

7.2 CACHEDIRS

Creates a report on the ViewStar’s Cache Directories (refer to System Tools), containing a list of Sites, Directories, Maximum and Minimum threshold levels and the Filled status.

7.3 DATADIRS

Creates a report on the ViewStar's Data Directories (refer to System Tools), containing a list of Sites, Directories, Max Files and current number of files.

8.0 CUSTOM REPORTS

Resulting from discussions with the System Administrator at Lovell Park, the following custom reports were developed:

8.1 OFFXLOCK

The Halifax Business Centre workflow makes continuous use of a number of "offload" queues (currently 4 queues BC_OFFX01, BC_OFFX02, BC_OFFX03, BC_OFFX04) serviced by (currently 10?) ViewStar Process Agents. Monitoring these queues is most critical to maintaining the workflow in good health. On occasions workpackets that for whatever reason fail while being routed to their next destination in the workflow, remain locked in one of these offload queues. Regular monitoring of these queues using the conventional Queue Utility Search function from the Configure module is time consuming. Furthermore, workpackets may orderly be locked (STATUS = 1) for short periods, in these queues by tasks that are executed by Process Agents. The challenge is to note which of the currently locked workpackets have not moved for a longer period. Under normal conditions workpackets should not be locked by Process Agents for longer than a minute. The "Last Updated" field of the Status (REQUEST) table, found in the Workflow Tracking database is the only source that contains an indication as to when (Date and Time) last a workpacket was locked or updated in any ways. Combined monitoring of these tables, through Queue Utility Search (Configure) and Document.Locate… (Desktop module or System Admin application) is a laborious task. This report automates and streamlines this procedure.

This report searches the offload queues for any workpackets with a STATUS = 3 (error) and reports their details: Queue Name, Last update Date and Time, Status, WF_ReqID and Roll Number. Also reported are all those workpackets in the offload queues that are unavailable (ie. STATUS > 0) and their last update date and time is longer than 15 minutes, before the time the report was created.

If any workpackets are reported a message is sent to the Process Agent's log (when in logging level 1). This message may be trapped by any conventional methods offered by the LogWatch utility.

Example OFFLOCK report

Offload Queues Exception Report - created: 08-Jul-1997 09:44:35
==========

Queue	Last update	Status	ReqID	RollNo
BC_OFFX01	08-Jul-1997 07:28:31	3	226898	A/ 9760349- 4
BC_OFFX01	08-Jul-1997 08:39:42	1	206602	A/ 3300779- 3
BC_OFFX02	08-Jul-1997 07:24:51	3	107537	A/ 9593863- 4
BC_OFFX02	02-Jul-1997 12:36:04	2	225340	000000
BC_OFFX03	08-Jul-1997 07:24:51	3	70542	D/35327899- 9

---------- E N D ----------

8.2 OFFXSCAN

The BC_OFFX04 queue is a special offload queue, dedicated to offloading scanned workpackets. The OffXScan report monitors this queue and reports the number of locked documents (STATUS > 1). If any such workpackets are detected, a message is sent to the Process Agent's log (when in logging level 1). This message may be trapped by any conventional methods offered by the LogWatch utility.

Example OFFXSCAN report

Scanner Offload Queue Count Report - created: 11-Jul-1997 11:30:15
==========

Queue	Count (Status > 1)
BC_OFFX04	10

---------- E N D ----------

8.3 PRODFAX1

When a user attempts to send a fax to an invalid number, following the usual retries attempts, the workpacket containing the fax document remains locked with an error status (STATUS = 3) in PRODFAX1, the fax out queue. This report monitors this queue and reports the Date and Time, Telephone number, From and To details that are found on this workpacket.

If any workpackets are reported a message is sent to the Process Agent's log (when in logging level 1). This message may be trapped by any conventional methods offered by the LogWatch utility.

Example PRODFAX1 report

```
Fax Output Queue Errors Report - created: 11-Jul-1997  10:06:09
============================================================

Date            Time            Telephone       From              To
-------------------------------------------------------------------------------------------------------
11/07/97        09:34:37        01618320833     RAYT Nerrinder    NW Property Unit

------------------------------------------------- E N D -----------------------------------------
```

8.4 OFFCOUNT

Counts the total number of workpackets in the offload queues and reports the result. A high number of workpackets is generally an indication of some malfunction. This report does not send an error message to the Process Agent's log file.

Example OFFCOUNT report

```
Offload Queues Count Report - created: 11-Jul-1997  10:33:15
===========================================

Queue           Count
-----------------------------------------------------------
BC_OFFX01       11
BC_OFFX02       5
BC_OFFX03       5
BC_OFFX04       146

---------------------------------- E N D -------------------------------
```

8.5 ARCHVOLS

This report delivers information on the optical volumes. The Volume Names, Access (Read Only or read/Write), the Online/Offline state and MB remaining for each volume are displayed in a tabulated format. This report does not send an error message to the Process Agent's log file.

Example ARCHVOLS report

BC Archive Volumes Report - created: 10-Jul-1997 17:42:44

Volume	Access	Online	MB remaining
BC96A01	Read Only	Yes	1114.21
BC96B01	Read Only	Yes	911.45
BC96A02	Read Only	Yes	-
BC96B02	Read Only	Yes	-
BC96A03	---???---	No	1182.66
BC96B03	---???---	No	1182.66
BC96A04	Read Only	Yes	-
BC96B04	Read Only	Yes	-
BC97A05	Read/Write	Yes	461.23
BC97B05	Read/Write	Yes	402.10

---------- E N D ----------

9.0 LOG MONITORING

Generally, the typical logs that a System Administrator is interested in are:

- Process Agent Log
- Optical Log
- Archive Log
- Fax Server Log

Tivoli ships with its own log file adapter that scans log files, and can be used to trigger on errors, etc. Their adapter, however, was not designed to cope with the unique ViewStar logging, which rolls over to other file names (wfserve.log, wfserve.001, wfserve.002, etc.). Since their adapter is not customisable to handle ViewStar specific logging, and also due to the potential for blocking actions should a process open the file in an ill-behaved way, we will not use Tivoli to monitor the log files directly.

It is envisaged that the LogWatch session will run on a server, that will also run the Tivoli Process Agent. A Tivoli communications process will be triggered through the LogWatch capability of executing a DLL function on a pre-set event, passing the message on as a Tivoli Alert. Given LogWatch's ability to pass the name of the log file and the line number to the DLL function, it is possible that a Tivoli messaging agent could scrape the line and its context lines above it and send that up to the Tivoli system. The possibilities are limitless. However, the required DLL has yet to be developed, so a proof-of-concept test is not available at the time or writing.

10.0 SUMMARY

The project has so far developed the methods of integration with Tivoli. An initial set of critical reports where developed using Script, based on experience gained from the pilot installation at Leeds - Lovell Park. Additional such reports may be added on, following the guidelines set down by those already developed.

The outstanding task is that of setting up the specific LogWatch triggers, where such are applicable. This cannot be done in a vacuum, from an abstract sense. Any implementation MUST be developed by local System Administrators familiar with the particular system. Their knowledge of the system specifics is important, and equally important is that the administrators understand exactly what is being monitored, as that knowledge empowers them to be wary for those resources that are NOT being monitored. Only a working relationship between ViewStar System Administrators and Tivoli integrators can finalise a successful implementation of such management system.

APPENDIX A - DIRECTORY TREE SPECIFICATION

The inclusion of the Tivoli integration within the existing ViewStar File Server directory structure is described in Figure 1 below. All necessary scripts and sources (on the development system only) are stored under the readroot. The dynamic part of this system, ie. the Tivoli requests and ViewStar's resulting reports are stored under the readroot directory structure.

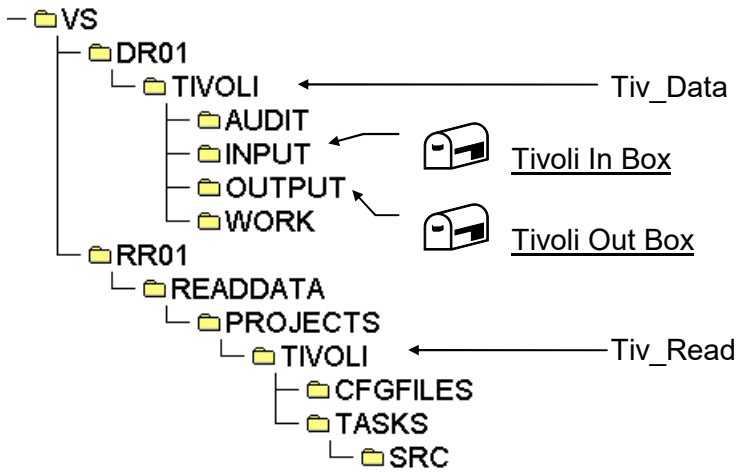

Figure 1 - Tivoli Integration directory tree

The files and directories that control the messaging are as follows:

<Tiv_Read>	Abbreviated form for: ..\VS\RR01\READDATA\PROJECTS\TIVOLI
<Tiv_Data>	Abbreviated form for: ..\VS\DR01\TIVOLI
<Tiv_Read>\Tiv_Task.FSL	The compiled version of a script file that contains the function tiv_Main, scheduled to run at a pre-set interval on a Tivoli Process Agent.

<Tiv_Read>\CFGFILES\	Sub-directory containing configuration (.CFG) files used by ViewStar tasks invoked at Tivoli's requests. Note that the CFGFILES may be written to by the tasks, which strictly speaking would indicate that they should be moved under <Tiv_Data>.
<Tiv_Read>\TASKS\	Sub-directory containing compiled ViewStar script files (.VFX or .FSL). These scripts are invoked by the tiv_Main ViewStar function, as per Tivoli's requests that were found in Tiv_Task.INP. The following custom scripts should be placed in this sub-directory: OffLock.FSL, OffScan.FSL, OffCount.FSL, ProdFax1.FSL, ArchVols.FSL. These files generate ViewStar reports stored under <Tiv_Data>\OUTPUT\.
<Tiv_Read>\TASKS\SRC\	Sub-directory containing ViewStar scripts source code (.VSS). The compiled version of these files is stored in <Tiv_Read>\TASKS\. This directory should exist only on the Development System.
<Tiv_Data>\AUDIT\	Location for any necessary logging by ViewStar tasks, operating at Tivoli's requests.
<Tiv_Data>\INPUT\	Input directory for Tivoli's requests. Tivoli places a text file named Tiv_Task.INP in this directory. If the file exists Tivoli appends a new request to the end of this file.
<Tiv_Data>\INPUT\index.lock	Counter for filename extensions, used by Tivoli.
<Tiv_Data>\OUTPUT\	Output directory for ViewStar tasks. Tivoli expects all ViewStar report files to be stored in this sub-directory.
<Tiv_Data>\WORK\	Working directory for use by ViewStar.
<Tiv_Data>\WORK\Tiv_Task.WRK	Working copy of the Tiv_Task.INP file. This file is created by the tiv_Main function of Tiv_Task.FSL.

APPENDIX B - TIV_TASK.INP FILE FORMAT

The file is a series of lines. Each line consists of a ViewStar list of two strings. The first string is the name of the task, and the second string is the name of the output file.

Example:

```
------Tiv_Task.INP----------------------------------------------------
["qmon" "qmon.019"]
["users" "users.01a"]
["cachdir" "cachdir.01b"]
["offxload" "offxload.001"]
["offxscan" "offxscan.001"]
["offcount" "offcount.001"]
------ End of file ----------------------------------------------------
```

Tivoli appends new lines to Tiv_Task.INP as requests are made from its scheduler, console, etc. On each scheduled pass, the ViewStar process agent will move this file to the <Tiv_Data>\WORK directory, then read each line and run the task as applicable. The output filename appended to the output directory path, is passed as an argument to the report script.

Tivoli is responsible for unique output file naming conventions. If the filename already exists, the ViewStar output will either append or overwrite, as determined by the report (Queue Monitor and other system reports always append, but custom reports could be written otherwise). Tivoli uses index.lock to supply a new extension to every request. Tivoli is responsible for erasing old reports as necessary.

APPENDIX C - RULES FOR CUSTOM REPORT TASKS

The function tiv_Main of Tiv_Task.VSS passes the fully qualified output file name, as the one and only argument, to the report generating task. Tasks that generate ViewStar custom reports must be written so as to require no user intervention and display no messages, other than to the Process Agent's log file (:WFM_MESSAGE). Following this rule ensures that the tasks will run cleanly on a Process Agent. Documentation for these tasks should include minimum run cycles (i.e. "Run as often as necessary" or "Database Intensive - run no more than once an hour", etc.).

APPENDIX D - VIEWSTAR LOGS

The following is a list of ViewStar generated log files:

VIEWSTAR.LOG:
Logs information and errors discovered by startup.exe and vstart32.exe shows names of Lisp applications that have started and stopped. Also logs all Lisp errors discovered by each Lisp environment and reported through VSError that have occurred at the workstation. Written to %windir%.

VSSETUP.LOG:
Used by installation to list information and errors. Written to c:\vstemp (sub-directory created by installation program). Default name set in mods\vssetup\vssetup.inf, but changeable by user dialogue.

IMPTERR.LOG
Used by the Importer module, located by default in <dataroot>\imptfile\impterr.log. Name can be set by an environment variable

C:\STKTRACE.TXT
Default name for a Dr. Watson type log created by Lisp when it faults. Configurable in viewstar.ini, Allegro CL section.

VSSQL.LOG
Used by idba to log all database information both from it and from the vendor. found in %windir% or as defined in VIEWSTAR.INI

VSSYSTEM.LOG
Used for critical system administration issues by the LogWatch module, shared by workstations, located in <dataroot>.

PROCESS AGENT LOGS
Monitor the state of the server. Process Agent log name set by agent name, but changeable by user dialogue. Increments to 100 logs of 100k each and then kills the oldest. Logs extensions are renamed so the *.log is always current, *.L00 next, and *.L99 oldest.

OPTICAL.LOG
Monitor the state of the server. Optical name set in a local optical.ini file.

APPTRACE LOGS
Tracks the flow of documents through applications written to <username>.log under the <dataroot>\userlog subdirectory.

FWDERROR.LOG
Formatted log generated when documents fail to forward; uses an automatic recovery script that can process errors, written to <dataroot>.

ISAM FILE LOGS
Created when certain corruptions are detected and (sometimes) corrected in .ISM files name is path and base name of isam file, with the extension ".ILG". Records corruptions along with the date, time, user name, and (in the case of partial records) the contents of the deleted partial record by setting (as env variable or in viewstar.ini) the variable ISAM_SYS_LOG to a path system logging is turned on for global logging of ISAM corruptions. All that is written to this file is a single record indicating the error and the path to the .ILG file containing details.

COMMON SERVER (FAX) LOGS.
Path and name settable in the fax server configuration. Increments to 100 logs of 100k each and then kills the oldest. Logs extensions are renamed so the *.LOG is always current, *.L00 next and *.L99 oldest.

OCR & DATABASE LOGS
Process Architect creates a file OUTPUT.TXT during builds.

SCHEDARC.LOG
Written to the dataroot

ARCHIVE.LOG
Written to dataroot

CKINONLY.LOG
Written to dataroot.

RECVARCH.LOG
Written to dataroot.

RECOVER.LOG
In the dataroot seems to contain optical recovery info.

WFSERVER.LOG
Process Replicator log file. This file is created during the upgrade process.