#21 zNibbler (zOS' HourGlass)
zTidBits Series

Many data processing installations have applications that contain time-sensitive logic—special processing that executes at certain intervals such as at the end of a week, month, or quarter. Until now, multiple system Power-On-Reset (POR) procedures were required to reset the MVS™ system clock to a test date and time, and then back to the current date and time. Setting the system clock in this manner affects not only the application being tested, but also other applications and components running on the system. Another drawback to the POR method is that only one test date and time can be in effect at a time.

HourGlass facilitates the testing of your time-sensitive applications by eliminating the POR procedures. Additionally, unlike the POR method, you may set as many altered date and time values as needed, and the altered values affect only the applications you specify. In other words, all other applications and system components such as your tape management system and system catalog are unaffected. With HourGlass, the testing of time-sensitive applications becomes simpler and less disruptive to your operation. So simple, in fact, that such testing becomes a normal day-to-day development activity.

Specifically, HourGlass allows you to temporarily alter the system date or time for one or more application programs to any date and time value supported by the MVS 64-bit time-of-day clock: 1900-01-01, 00:00:00.000000 (midnight) UTC through 2042-09-17, 23:53:47.370495 UTC.

The HourGlass product includes an Audit Trail Facility that aids in the identification of applications that request the system date and time. Audit trail reports display the frequency of requests, organized by job step and showing the date and time actually returned. The Audit Trail Facility can be used to track the job steps that must be tested for date or time compatibility. Later, it can be utilized to report on job steps making use of altered dates via the HourGlass product.

Incorporating HourGlass into your testing procedures can also help to ensure that your test environment is consistent regarding the dimension of time from one testing cycle to the next. This can make regression testing and the auditing of system changes easier and more reliable.

With HourGlass, the customer can adjust the time of day forward or backward from the current time by as much as 23 hours and 59 minutes. This facility can be used to run time-sensitive applications for users located in a time zone other than that of the data center in which the application runs. You can also simulate start times for specific processing steps as well as force the time to remain constant throughout the life of an application process.

Note: Setting date and time values in the past (that is, to a value prior to the current date and time), while extremely useful in a number of situations, it may be undesirable at certain sites due to production control and auditing considerations.

HourGlass intercepts all system time-of-day requests that internally issue an **SVC 11 instruction** (equivalent to the TIME macro with the LINKAGE=SVC parameter) or requests to the PC Time system service (equivalent to the TIME macro with the LINKAGE=SYSTEM parameter). These application-oriented interfaces are commonly used by most software products and programming languages to retrieve the date and time from the operating system.

HourGlass does not automatically intercept time-of-day requests made via the Store Clock (STCK) instruction since the STCK instruction is purely a hardware function. Nevertheless, HourGlass is able to intercept time-of-day requests when made from one of the several products that normally use the STCK instruction, for which a product-specific interface has been developed. Such interfaces are available for several products including: DB2, Enterprise Cobol for z/OS® , Enterprise PL/I for z/OS, NATURAL, IDEAL, IEF, and others. For other products containing the STCK instruction for which you have the source, you can reassemble the module using a supplied STCK macro. If the module source is not available, or a module reassembly is undesirable, the STCK instructions in the load module can be patched to allow interception and processing by HourGlass.

HourGlass can be used in both the batch and online environments. Full support for IMS message regions (online IMS), CICS, IDMS and COM-PLETE regions is provided. In an IMS online environment, all transactions running in a message region or group of message regions can receive an altered date and time value that is common to all transactions in the region or group. This is called region-level or global-level control.

Alternately, individual users can override the region-level date and time value and receive a date and time value unique to their userid. This is called user-level control. Both global-level and user-level control are supported as well in the CICS Transaction Server product, and in the COM-PLETE, IDMS/CV, and ADS/O and DC COBOL environments.

In the IMS online environment, HourGlass does not provide support for altering the MFS date keywords, but does provide support for setting the HourGlass-altered date and time value in the IMS IOPCB.

In the DB2 environment, HourGlass provides altered date and time values for time requests made via the DB2 special registers CURRENT DATE, CURRENT TIME and CURRENT TIMESTAMP as well as for inserts of DATE, TIME, or TIMESTAMP columns for which no value is specified, and are defined with the ″NOT NULL WITH DEFAULT″ attribute.

Setting the date and time values

You can set the HourGlass date and time values for a batch job in one of four ways. In the most straightforward method, you add one or two DD statements, specifying the date and/or time, to the JCL step. You can also change the job statement to affect the entire job.

Using the second method, you can set the dates interactively, using a series of ISPF

dialogs, with no JCL changes required. This is the best method for system testing; it is very flexible and provides wildcard capabilities. However, this method does not require a deliberate action to alter a date as the JCL change does. You can use the HourGlass Control Center to set dates using ISPF dialogs.

The third method allows for system testing of one or more multi-step jobs to create a continuous 'rolling time' between jobs and job steps. The Scheduling Facility allows a 'schedule' to start at any requested date and time. All date and time requests for any job in the schedule receive a date and time relative in time to this requested date and time.

Lastly, for sites with remote users, the hard-coded date and time method may be useful. In this method, jobs with names matching the listed prefixes automatically receive altered dates and times. This technique is implemented via the HourGlass customization process.

Cost and Time Benefits

- Eliminates the difficulty testing complex, multi-step enterprise applications under multiple date / time scenarios
- Reduces the time recoding or in-house work around solutions
- HourGlass intercepts requests for the system date and time and does not require you to change application code, re-IPL your CPU or LPAR, or make runtime JCL changes
- Developers can run simultaneous tests – each using a different system date if desired.
- Recently updated error and informational messages can help to reduce the time and effort required to resolve application errors.

Using HourGlass

Once you activate HourGlass, it begins intercepting all requests for the system date and time which are made through the interfaces supported by HourGlass. Each date and time request is checked to determine whether it is eligible to receive an HourGlass-altered date and time value.

In order to minimize system overhead, this eligibility checking code is highly optimized and designed to use as few system resources as possible. If a date and time request is eligible for an HourGlass-altered date and time value, HourGlass calculates the altered date and time value according to the control information that corresponds to that request. Otherwise, the request is simply handed off to the operating system service that would normally handle that type of request.

– – –