Top 10 Best Practices for Date/Time Simulation Testing
Top 10 Best Practices for Date/Time Simulation Testing

James Phillips
Design and Development Manager
DDV Technologies

Scott Garrett
Director of Sales
SoftBase
Today’s Agenda

1. Why Date/Time Simulation (DTS) Test?
2. Typical DTS Testing Methods
3. Top 10 Best Practices for DTS Testing
4. Considerations in Selecting a DTS Plan
5. DTS Done Right – Case Study
6. Q & A Session
Top 10 Best Practices for Date/Time Simulation Testing

Why DTS test?

- **Ensure functionality**
  - 1 error can propagate exponentially, especially in SOA

- **Errors can be highly visible and affect business**
  - Customers and financials can be directly affected!

- **Maintain user uptime**

- **Testing = compliance**
  - Sarbanes Oxley, Basel 2, etc…
  - Most applications can’t be certified without date testing (*31- & 64-bit date fields*)
Top 10 Best Practices for Date/Time Simulation Testing

Why DTS test?

Date and time errors in critical applications will happen if you fail to properly DTS test!

Can you afford these errors?
Ways to perform DTS testing

Multiple LPARs

Set up multiple LPARs for each date/time you need to test

- Expensive
- Inflexible
- High maintenance
Ways to perform DTS testing

IPL Test Environment

Manually set test environment system date/time with an IPL

- Risks early expiration of your testing tools or other software functionality, e.g.
  - HSM may delete datasets it thinks have expired
  - Database logs will be out of sequence

- Inflexible and time-consuming to maintain

- All applications and users must use same date/time. Any change means another IPL.
Ways to perform DTS testing

System-Wide Simulation Tools

Third party tools with system-wide impact

- Inflexible – all jobs on system must use modified system date/time routines, adding significant overhead.
- System upgrades become more difficult, with the risk of “unpredictable errors” or abends if not done properly
  - Time and effort to maintain
- Limited platform support
- A few are unsupported and NONE are being actively enhanced.
Ways to perform DTS testing

Dynamic Tool – the Ideal Solution

Look for flexibility, support for as many platforms as possible, and ability to dynamically target which job steps require DTS.

A dynamic DTS tool allows for:

- Lower TCO than IPL, LPAR and hardcoding solutions
- Enables faster, more efficient and more flexible DTS testing
- Holistic DTS testing across major mainframe platforms, including: z/OS, USS, WebSphere, Java, etc
- NO permanent changes to system date/time routines

More to follow after best practices…
Poll Question
Top 10 Best Practices for Date/Time Simulation Testing

DTS test all z/OS apps (especially SOA related apps) #1

Issues and Challenges

- When z/OS is a main SOA hub it is usually because of the wealth of information and logic that resides in CICS, DB2, IMS and other databases, a major heart of the organization
- WebSphere Application Servers (WAS) and Java language apps are in an open world with sometimes millions of tablet, smartphone and PC users logging in to collaborate with aforementioned critical information
- In an SOA environment, 1 stored procedure with bad date/time logic could affect 100’s of apps
- Often the Java apps are integrated with batch, CICS, WAS for z/OS and must be treated with the same, zealous testing as has been the case for years with Cobol, PL1, etc.
- Exponential date and time errors can occur far outside of z/OS in online apps being used by clients
Top 10 Best Practices for Date/Time Simulation Testing

DTS test all z/OS apps (especially SOA related apps)

Resolution

Look for a solution that:

- Covers the majority of your platforms (single tool reduces learning curve and increases adoption rate)
- Enables system-wide DTS rules definitions down to the job step level
- Allows users to document the purpose of each DTS rule
- Has an easy to use and consistent interface
DTS test z/OS 64-bit support

Issues and Challenges

- zSeries provides for data above the 2-gig bar
- Home grown and 3rd party apps are using data above the 2-gig bar in combination with IMS, DB2, Java, USS, SAP and more
- Performance and TCO are paramount
- Once again, exponential date and time errors can occur far outside of z/OS in online apps being used by clients
DTS test z/OS 64-bit support

Resolution

- Thoroughly DTS test all 24, 31 and 64-bit apps whatever code or storage is being used
- Choose a solution that doesn’t limit exploitation of modern zSeries and z/OS features
Poll Question
Top 10 Best Practices for Date/Time Simulation Testing

DTS test DB2 Stored Procedures coming from any SOA component to z/OS

Issues and Challenges

- DB2 Stored Procedures are now essential with or without SOA
- Stored Procedures (SP’s):
  - are often embedded components in one or more distributed DB2 databases across the SOA environment
  - reduce network traffic where the SP performs the database accessing without returning unnecessary data across the network
  - are re-usable SOA components, reducing costs
  - are known as “repeatable services” to be shared between apps
  - define business rules that are common to several applications. This is another way to define business rules, in addition to using constraints and triggers
- Because SP’s have such a wide reach, a single date/time error can easily proliferate out of control and disrupt customer services
DTS test DB2 Stored Procedures coming from any SOA component to z/OS

Resolution

- Ensure your DTS solution can correctly handle date or time calls from within SP’s that are nested to any level (that is, SP’s that call other SP’s)
- Make sure you shift CURRENT DATE, TIME or TIMESTAMP calls from Stored Procedures consistently with whatever virtual clock the caller of that Stored Procedure is running with.
- Make sure different callers of the same Stored Procedure in the same DB2 subsystem can be running with different simulated clocks (or none at all). These different callers could be CICS or IMS transactions, batch programs, Java programs running under WebSphere Application Server for z/OS, and so on.
DTS Test Event-Driven SOA Triggers

Issues and Challenges

- Event-driven SOA lets business users monitor, analyze, and enrich events to make the connections among disparate events.
- Triggered SQL and Stored Procedures arriving from any SOA component to z/OS are specifically event-driven.
- The triggers create high-level business events from numerous lower level events.
- Triggered SQL support is essential with or without SOA.
- Like SP’s, SOA triggers have a wide reach - a single error can have far reaching effects.
DTS Test Event-driven SOA Triggers

Resolution

- Ensure your DTS solution can handle triggered SQL and SP’s
- Triggers can fire at any stage before or after an SP and/or SQL. Your DTS solution must handle date and time calls from these no matter what context they are executed in so as to remove the risk of exponential data errors.
DTS test IMS as Much as Possible

Issues and Challenges

- Under SOA, IMS (and DB2) is no longer used only within self-contained applications, but is also accessed from loosely-coupled apps both from within zSeries systems and from outside.
- Your DTS testing must be able to seamlessly manage dynamic IMS region configurations.
DTS test IMS as Much as Possible

Resolution

- Your DTS solution should flexibly cater for sharing date rules amongst all possible configurations of IMS regions, e.g.:
  - all regions in one IMS subsystem
  - selected regions from multiple IMS subsystems
  - all regions on the same LPAR
  - multiple regions across multiple LPARs in an IMSplex

- If a user changes a simulated clock in one region then that change should automatically propagate to all other grouped regions
DTS test CICS (especially within SOA)

Issues and Challenges

- CICS Transaction Server serves as key components in SOA
- In combination, WebSphere Application Server and CICS support almost any mission-critical SOA solution.
- CICS connects to applications affecting customer services via WebSphere Application Server for z/OS (WAS for z/OS) and Java

- Errors can easily get out of control!
DTS test CICS (especially within SOA)

Resolution

- DTS test all combined SOA logic
  - High quality QA is essential
- Date and Time errors are not an oversight, they are a strategic mistake, letting down many thousands of hours of work
- SOA results deserve more
Issues and Challenges

- CICS MRO (Multi-Region Operation) allows application components to execute on multiple interconnected CICS regions.
- Regions in the MRO group may be spread across more than one z/OS image, and regions in the group may be dynamically added or removed under control of CICSplex SM.
- MRO configurations are the norm for all but the smallest systems.
- Application transactions may be dynamically routed amongst different regions without the knowledge or control of the end user.
- Applications must account for regional time-zone differences.
- Your DTS solution must be able to seamlessly propagate virtual clock settings across all regions in the MRO group.
#7

DTS test CICS MRO under SOA

Resolution

- In an SOA world your applications may execute across multiple dynamically configured regions. Ensure you DTS test the entire application no matter what the MRO configuration.
- Your DTS solution should allow easy mapping of your MRO configuration.
- It should automatically propagate any virtual clock changes across all regions so that users see the same virtual date no matter where their transactions are routed.
- Use a tool that supports CICS MRO and complete global time-zone virtualization!
DTS test CICSPlex under SOA

Issues and Challenges

- Most z/OS-based enterprises are running many tens or hundreds of CICS regions under the SOA banner, and many run their MRO configurations in a “CICSPlex” distributed across more than one z/OS system
- A CICSPlex supports their needs for capacity, resilience, and workload management
- Integration with WAS for z/OS and Java are highly strategic
- Manually keeping DTS rules synchronized across the CICSPlex is very difficult and error-prone
Top 10 Best Practices for Date/Time Simulation Testing

DTS test CICSPlex under SOA

Resolution

- DTS testing needs to seamlessly cater for MRO and CICSpelix configurations in an integrated manner, without needing manual intervention
- Use a tool that automatically ensures that DTS rules apply and are kept synchronized across all CICS regions
DTS test with Parallel Sysplex

Issues and Challenges

- Parallel Sysplex allows a cluster of mainframes to act together as a single logical entity under z/OS
- A well-designed high-availability infrastructure can provide a solution to these issues by building on the unique technology provided in a System z Parallel Sysplex
- Unfortunately, this can add complexity to accurate DTS testing
DTS test with Parallel Sysplex

Resolution

- DTS test Parallel Sysplex with automatic synchronization of shared rules and clocks across the Sysplex
- For example if a CICS user sets a virtual date in one region of a CICS MRO that date should be propagated automatically to all other participating regions, even if spread across multiple LPARs
DTS test z/OS UNIX System Services (USS)

Issues and Challenges

- USS (technically OMVS) is now tightly integrated under z/OS
  - “USS is built for the enterprise where you can prioritize workloads for high performance when running with a mixed workload”
  - “Applications can work with data in both the z/OS UNIX file systems and traditional MVS data sets”
  - “MVS programs can access UNIX files, and UNIX programs can access MVS data sets”
  - WebSphere Application Server, CICS, DB2, IMS, Java and more, all use z/OS USS.
  - “There is a broad range of ISV applications ported to z/OS UNIX, such as SAP
  - USS provides UNIX under z/OS
  - UNIX apps are sometimes ported from UNIX boxes to USS under partial or full SOA

- ...DTS testing of the 2013 USS is complex!
DTS test z/OS UNIX System Services (USS)

Resolution

- As a DTS tester you must be able to thoroughly test at a strategically high SOA level but tactically drill down and test the inter-connected apps that may well be in association with several, if not many components of SOA.
- Use an SOA z/OS-based DTS tool that can test USS, Parallel Sysplex, CICSPlex, CICS MRO, CICS, IMS (including FastPath), fully nested DB2 Stored Procedures/SQL/Triggers, and full support of 64-bit.
- Make sure the tool can synchronize the date and time environment, at the platform level of all your SOA systems.
Poll Question
Considerations in Selecting a DTS Solution

- **Platform Support**
  - z/OS, WebSphere, WAS, Java, USS

- **Low cost of ownership**
  - No IPLs, no extra LPARs, no recoding for upgrades

- **Flexibility and low impact**
  - Dynamic intercepts
  - 100’s of job rules with easy grouping capability
DTS Testing Success Story

Background and Challenges

- Large US Federal Agency
- Needed to simulate legitimate (no Feb 30th) past or future dates within range of SOA environment
  - COBOL apps that ran in executed batch, CICS and DB2
- Apps needed integration with their WAS servers, driven by Java-based programs used by clients
- Needed a single DTS tool supporting Unix System Services, z/OS 64 bit, maximum granularity in activation of job step via JCL, with low system impact and no permanent code changes
DTS Testing Success Story

Benefits

- Able to fully test all SOA app suites
- Virtually no impact on jobs not using DTS
- Now using a single tool for all platforms
- All environments are synched with rule sets
- Enjoying accelerated development and lower TCO
SoftDate by DDV

A single solution that:

- Enables you to DTS test across all mission critical customer-centric applications
- Ties together DTS testing across all critical environments using advanced rule sets
- Coexists with incumbent DTS tools and backward compatible with ALL of them
- Avoids system code changes, reducing overall costs for system upgrades
- Provides a lower TCO than other DTS solutions
SoftDate by DDV

SoftDate enables you to DTS test:

- Faster and more easily,
- While testing more thoroughly and consistently,
- With minimal impact on system resources,
- Across ALL major z/OS platforms!
How SoftDate Works

1. When a new job step starts
2. When SoftDate activates for a new job step
3. When a program requests date or time
Program date and time requests

Various ways for programs to get date/time:

**Cobol**
- ACCEPT ... DATE (and others)

**LE (Cobol, PL/I, C/C++, Java):**
- FUNCTION CURRENT-DATE (and others)

**SQL:**
- SELECT CURRENT DATE (or TIME, TIMESTAMP)

**CICS**
- EXEC CICS ASKTIME
- EIBDATE

**Assembler**
- TIME macro (SVC or PC linkage)
- STCK (or STCKE) “Store Clock” machine instruction
1. When SoftDate has been requested for a new job step ...
   - SoftDate dynamically inserts intercepts for the system TIME service and selected STCK instructions into the target address space
   - These hooks are *only* inserted for address spaces that need SoftDate

2. When a program in that address space requests the date or time, SoftDate’s dynamic intercepts ...
   - pass the request on to the appropriate system routine
   - intercept the return from the system routine and adjust the value appropriately before returning it to the caller

*With two very minor exceptions, SoftDate has zero impact on any other work running in the z/OS system.*
SoftDate dynamic intercepts

Whenever any new job step starts

A SoftDate exit checks if SoftDate is needed for this step ...

1. $$TFNPRM DD in the step JCL?
2. Prior TFNPSET step in the job?
3. Matching job rule?

When SoftDate activates for a new job step

SoftDate inserts dynamic intercepts for date and time requests into the target address space

When a user program requests date or time

A SoftDate intercept sees the request and passes it on to the standard system date/time routine

SoftDate adjusts the returned result
Using SoftDate

To use SoftDate you need to do just two things:

Tell it what jobs or job steps need it

- In this context a job could be a batch job, TSO user, CICS region, IMS T/M region, WebSphere A/S region, other started task...

Tell it what to set the “virtual system clock” to

- DATE=(+|-)yyyy-mm-dd
- TIME=(+|-)hh:mm

The value is set when the job (step) is started by z/OS

For regions like CICS, the clock is usually not set when the region starts. Users normally set their own clocks within the environment.
Q&A Session; Available Resources

Top 10 Best Practices for Date/Time Simulation Testing

Additional resources (email sgarrett@softbase.com to request free copies)

- Top 10 DTS Best Practices Spreadsheet
- DTS Testing for SOA
- SoftDate Case Study
- SoftDate TCO Study

Email Scott at sgarrett@softbase.com