ASAP Extension 2.0

Joe Davis
Software Designer
NonStop™ Enterprise Division

May 17, 2002

What is ASAPX

An application interface to ASAP

A set of API procedures used by your application

A calculation and collection service

An alerting mechanism

An availability tracker

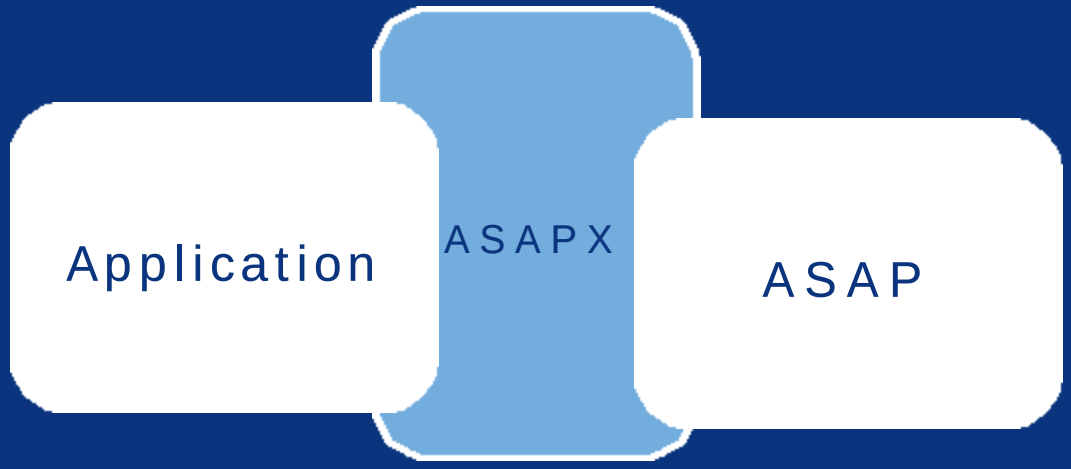
How ASAPX Works

Applications register by calling the ASAPX API

ASAPX allocates shared memory in each processor for application data

Applications store and update counters and state values in shared memory

ASAPX samples shared memory to compute metrics for an interval



ASAP Entities

Entities are Applications or other logical groupings

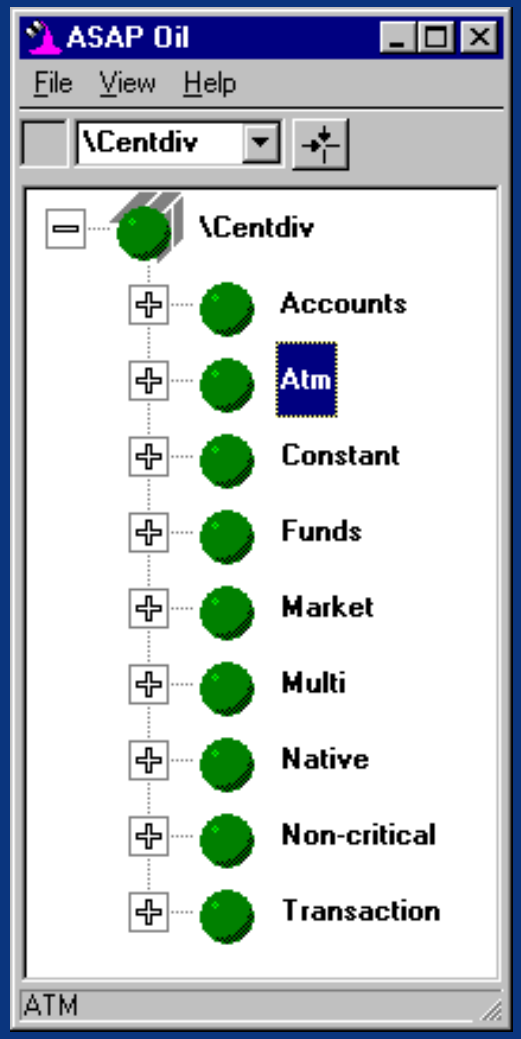
All components of an Entity share unique ASAPX definitions

Entities are described using Entity Definition Language (EDL)

Approximately 88 user entities

- Depends on System entities
- ASAP 2.0 maximum is 100 entities

Generic APP entity applies when no entity is defined



Entity-Specific ASAPX Definitions

DataItems EDL entity property

- Up to 12 items (0-11) in shared memory
- Three classes of DataItems
  - Counters or Constants - Types I or C
  - Time Units - Types S, M, and U
- Example: DataItems “0 I, 1 M, 2 C”

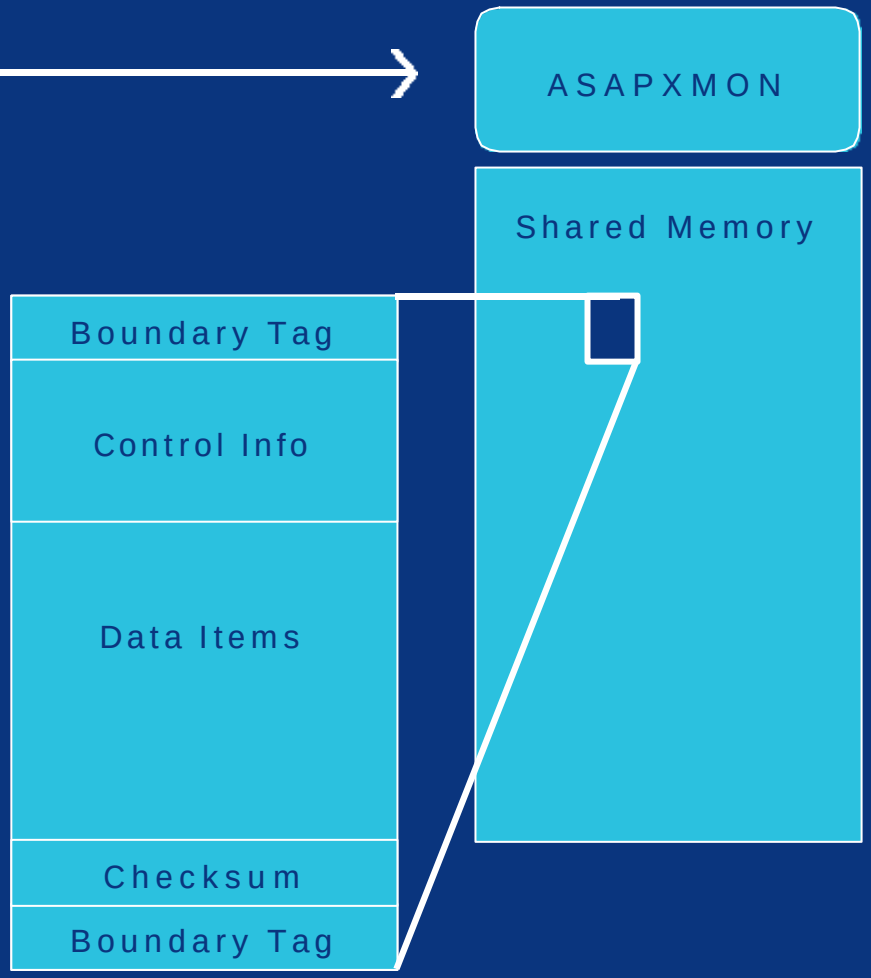
Metrics (EDL Attributes)

- Statistical properties computed at each interval
- Computed using DataItems and user-defined formula

Display Formats

Registering a Domain with ASAP

```
error := ASAP_REGISTER_ (
          domain^name:name^len
        , seg^offset
        , [ error^detail ]
        , [ segment^id ]
        , [ segment^base ]
        , [ version ]
        , [ asap^id:id^len ]
        , [ flags ]
        , [ timeout ] ) extensible;
```

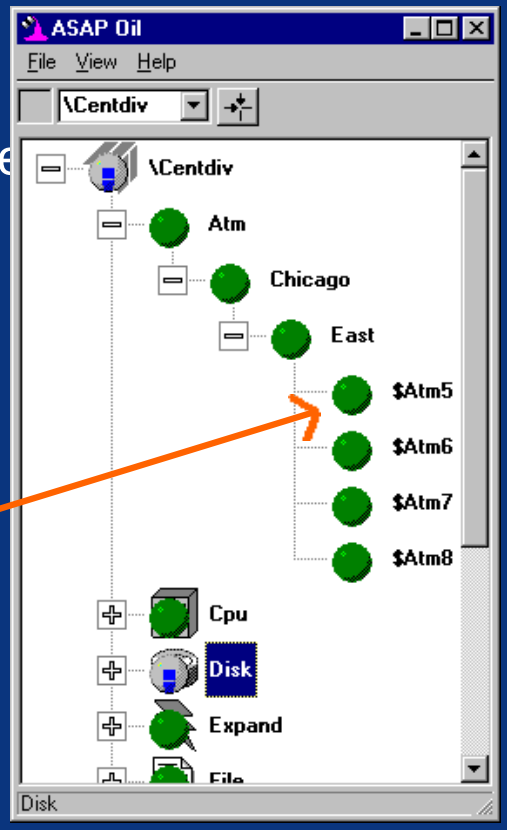


flags.<0:12> - reserved
flags.<13> - allow replace of non-constants
flags.<14> - don't concatenate process name
flags.<15> - start with ranking deactivated

Domain Names

<entity>\<domain>[\<subdom>...]

- 1st level must be EDL defined Entity name
- 5 levels
- 64 bytes max
- Must be unique
- Examples
  - Atm\Chicago\East\$Atm5
  - Assembly\Belt12\Station6\P34345
  - Basketball\Scores\Clippers

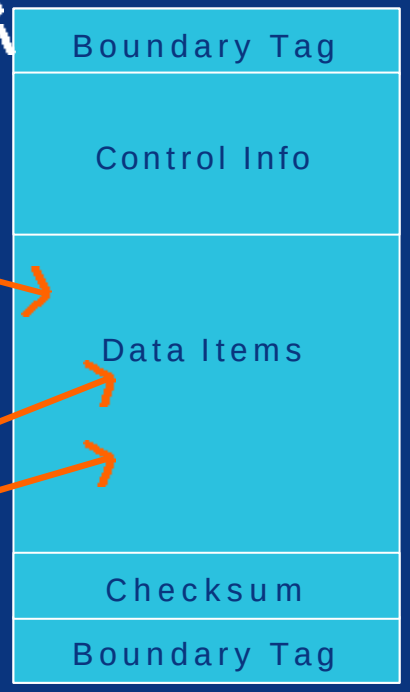


Domains are logical representations of business functions

Updating DataItems

```
error := ASAP_UPDATE_( seg^offset
                     , [ error^detail ]
                     , data^item
                     , value
                     , [ math ] ) extensible;

error := ASAP_UPDATELIST_( seg^offset
                         , [ error^detail ]
                         , num
                         , list)   extensible;
```



math ➔ 0 = add (default)
1 = replace
2 = replace text

Creating Metrics (Attributes)

Defined as EDL Attributes

TransRate is the Attribute name

Format controls display

MetricRule is formula

- #<n> refers to a DataItem
  - Difference between samples
  - Or constant value at sample
- S is seconds in interval
- C<n> means <n> is a constant

```
AT TransRate Grid YES
   Graph NO
   GraphMax 1000
   Format “F7.2”
   Help
   ”Successful transaction rate"
   StatePair YES
   StateRule UseStateGraphState
   MetricRule “#0/S"
   TypeData REAL64;

AT S0 Grid NO Graph NO
   GraphMax 9
   Help "State of TransRate ";
```

ASAPX Built-in Metrics (Attributes)

11 built-in Metrics

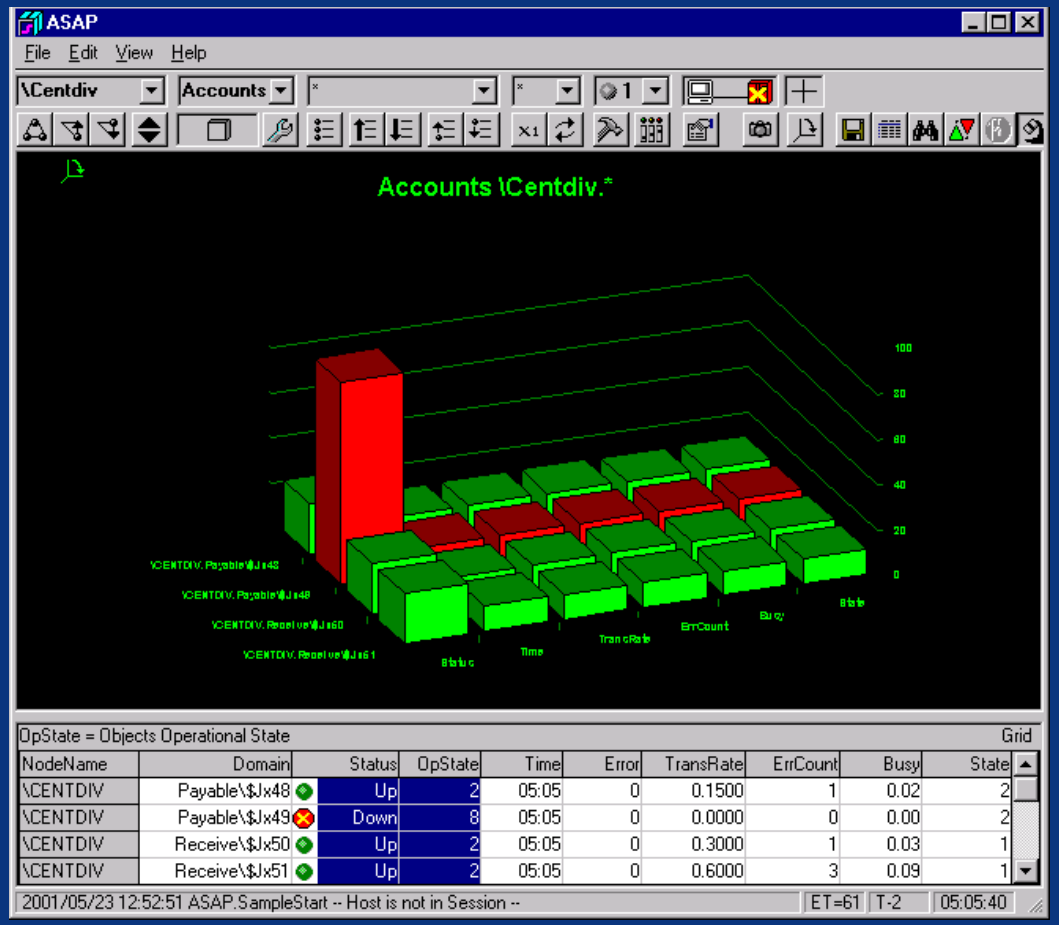
- Avail - Availability since registration
- Busy - Percent busy for interval
- Cpu - Process Cpu
- DownTime - Down time seconds since registration
- Nak - Number of intervals without update
- Pri - Priority of registering process
- PState - Process state of registering process
- RegTime - Domain registration date/time
- UnAvail - Unavailabilty for interval
- Version - Application supplied version
- WState - Wait state of registering process

Include EDL definitions as described in the ASAPX Manual

Operational Status and State

Set by ASAPX to:

- “Up” - OpState 2
- “Inactive” - 6*
- “Down” - 8
- “Removed” - 0
- “Unknown” - 6

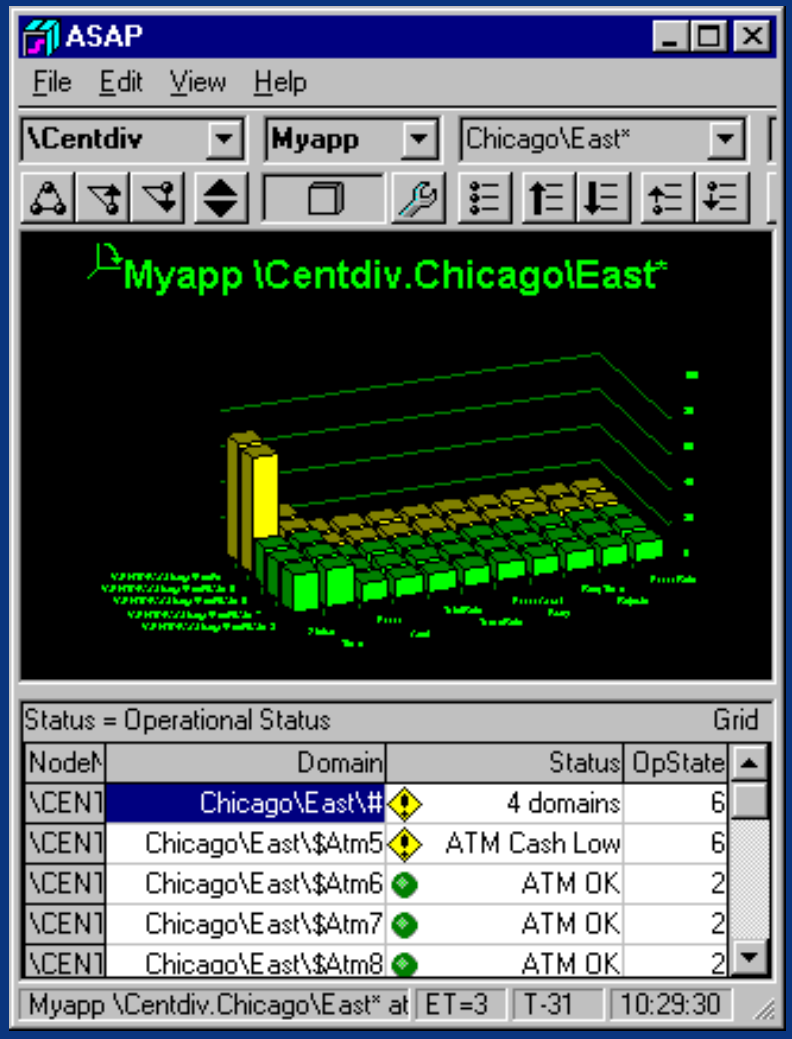


* If the Nak built-in is used, the state reported for “Inactive” is the OEM state for the Nak attribute instead of state 6

Operational Status and State

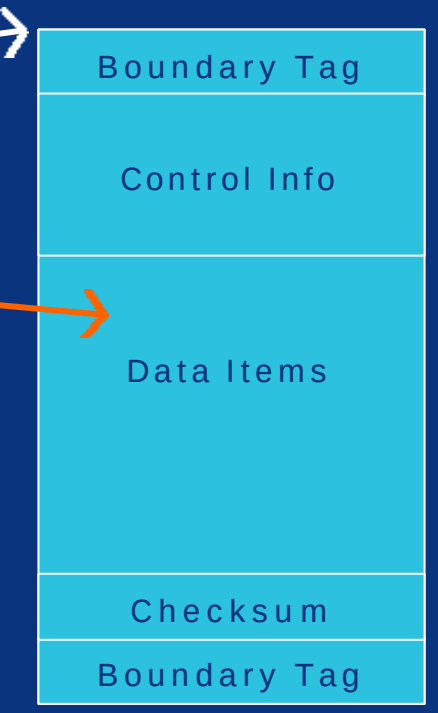
Applications can override ASAPX and set Status Text and OpState directly

- Status text is 1 to 15 ASCII bytes
- OpState is a valid OEM State, 1-8



ASAP_OPSTATE_

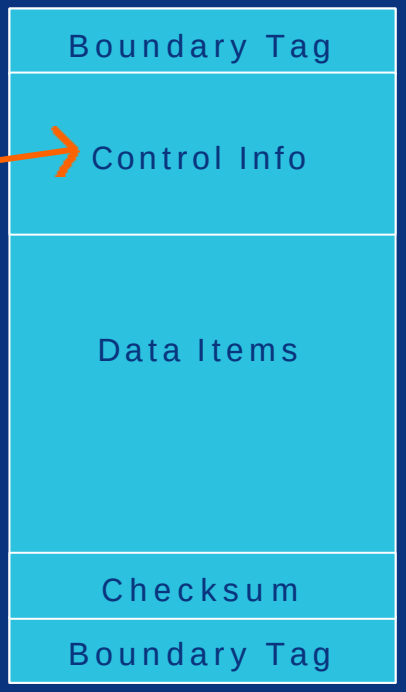
```
Error := ASAP_OPSTATE_(
            seg^offset
            , [ error^detail ]
            , OpText:OpText^Len
            , OpState
            ) EXTENSIBLE;
```



If an application sets Status and OpState and then fails, ASAPX overrides OpState to DOWN (8). Status text is overridden if and only if it is “Up”.

Domain Control

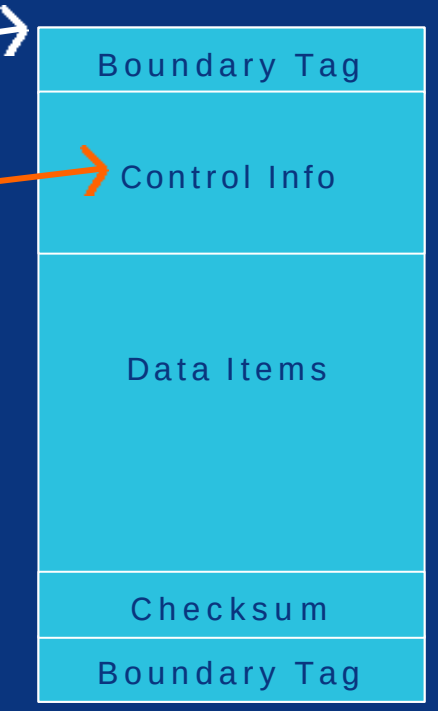
```
error := ASAP_CONTROL_(
                    seg^offset
                    , [ error^detail ]
                    , flags ) extensible;
```



flags.<0:13>	-	reserved
flags.<14>	-	enable ranking
flags.<15>	-	disable ranking

Domain Removal

```
error := ASAP_REMOVE_(
                seg^offset
                , [ error^detail ]
                , [ segment^id ]
                , [ flags ] ) extensible;
```



flags.<0:14> - reserved
flags.<15> - de-allocate segment

Discrete Object Thresholds

Standard

Icon	State
	Exists
	Up
	Critical
	Down

Percent/Historical

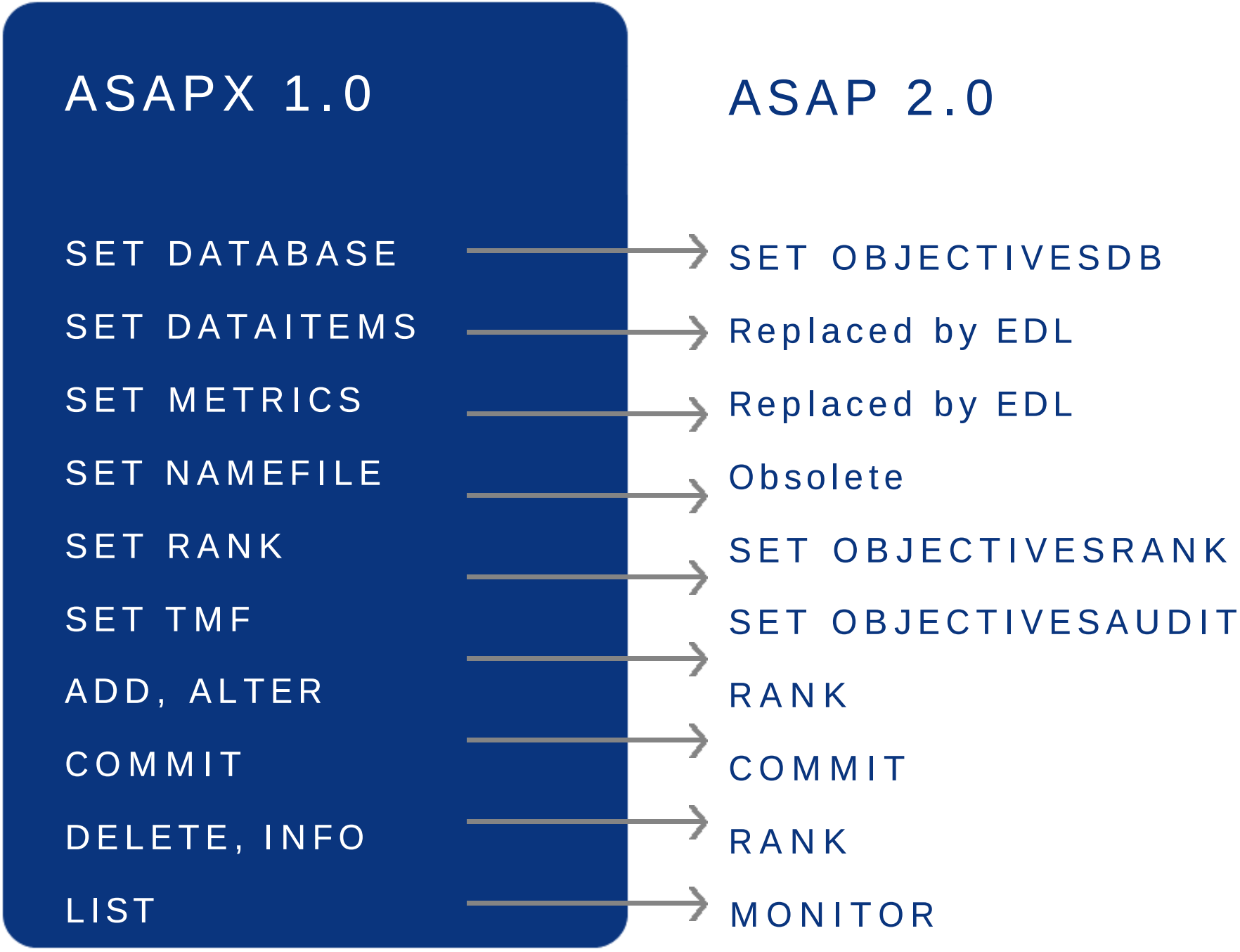
Icon	State
	Exists
	Up
	Low
	Medium
	High
	Warning
	Critical
	Down

ASAPX fully supports ASAP DOTs

- Except monitored via API

Metrics ranked to set alert level

All version 1 Objectives functions have been removed from ASAPX.


ASAPX 1.0
ASAP 2.0
SET DATABASE
SET OBJECTIVESDB
SET DATAITEMS
Replaced by EDL
SET METRICS
Replaced by EDL
SET NAMEFILE
Obsolete
SET RANK
SET OBJECTIVESRANK
SET TMF
SET OBJECTIVESAUDIT
ADD, ALTER
RANK
COMMIT
COMMIT
DELETE, INFO
RANK
LIST
MONITOR

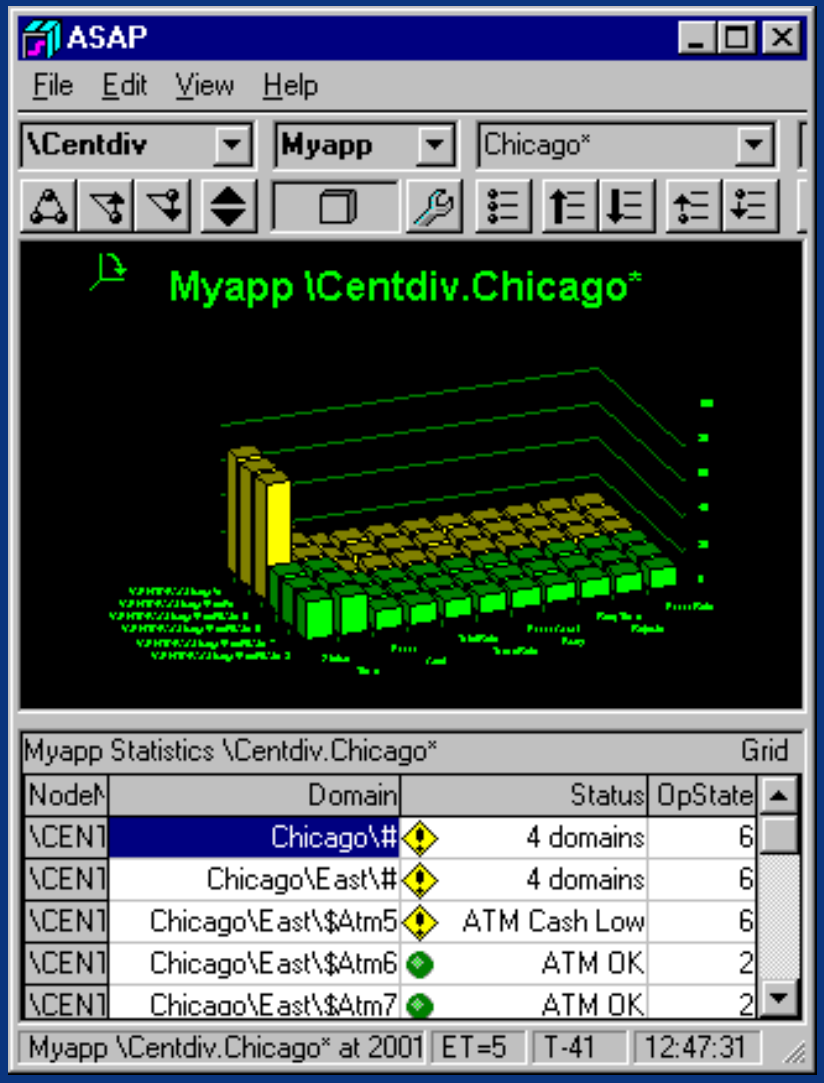
Domain Aggregation

Atm\Chicago\East\$Atm37

Atm\Newyork\East\$Atm39

- 3rd Level Aggregates
  - Atm\Chicago\East\#
  - Atm\Newyork\East\#
- 2nd Level Aggregates
  - Atm\Chicago\#
  - Atm\Newyork\#
- 1st Level Aggregate
  - Atm\#

State propagation

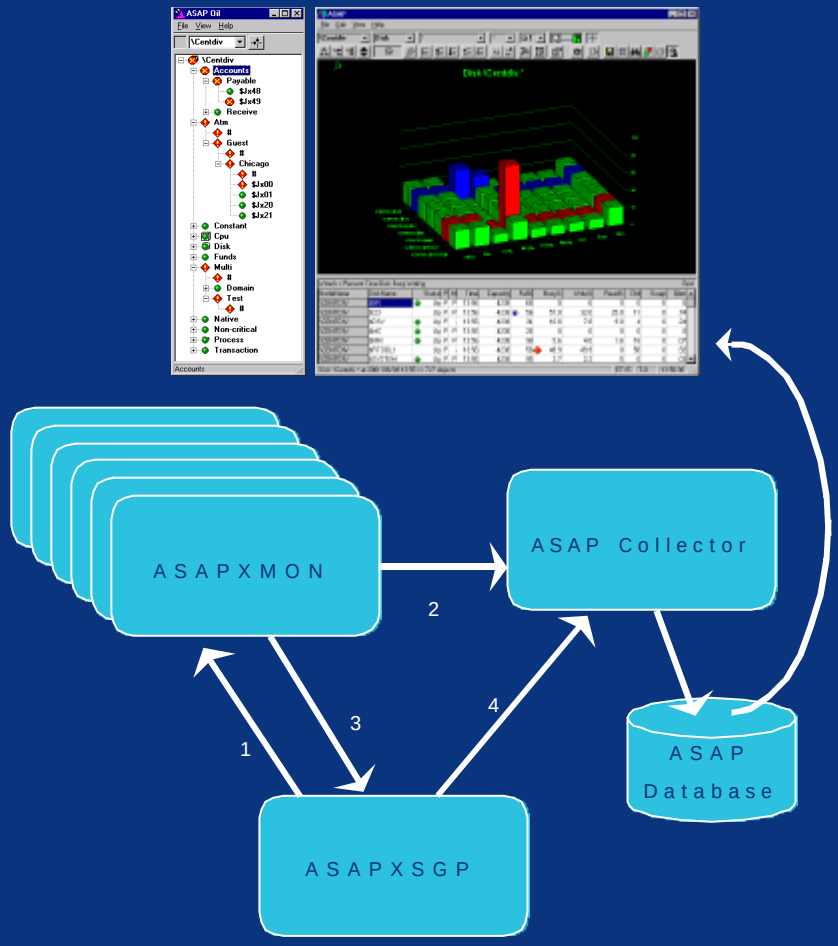


Write-to-Collector Mode

ASAPX SGP collects statistics serially from all ASAPXMON processes then forwards them to the ASAP Collector

In Write-to-Collector mode all ASAPXMON processes work in parallel

- Detail records sent directly to ASAP Collector
- Aggregate records (if any) are still collected by the SGP



Limits Increased

128 domains per CPU to 1024 domains per CPU!

- Done for K to S migration customers who end up with fewer processors
- Use this increase wisely, be careful of too much APP data
- ASAP 1.0 memory segments cannot be used with ASAP 2.0 due to the segment size increase for this enhancement!

20 metrics to 30 metrics (EDL Attributes)

- DataItems remains 12
- Metrics increased due to new built-in metrics

If you are approaching limits, we want to know!

Instrumenting an Application with ASAPX 2.0

Joe Davis
Software Designer
NonStop™ Enterprise Division

month ##, 2002

Follow These Simple Steps

Define Metrics (Attributes)

Reverse Engineer DataItems

Define EDL

Load/Distribute/Verify EDL

Test EDL

Identify code points

Instrument the Application

Final Test

Define Metrics - What to measure?

What criteria is used to judge the application at the end of the year, month, day?

What don't you know about the application?

General suggestions:

- Success rate
- Error rate
- Server response time
- Total transaction rate
- Success percentage
- Processing states

Application specific:

- Cash remaining
- Shares traded
- Web site visits
- Quantity on hand

Define Metrics - What to measure?

What criterion is most important?

Order them in descending order from most to least important.

Each item will be defined as an attribute in ASAP EDL.

Application Metrics (in priority order)

- Cash remaining
- Total transaction rate
- Successful transaction rate
- Failed transaction count
- CPU utilization
- Server response time
- Card reject rate
- Failed transaction rate

Reverse Engineer DataItems - Formulas

Cash	-	computed and stored as a constant by the application.
Total rate	-	(Trans count + Error count)/seconds in interval
Trans rate	-	Trans count/seconds in interval
Error count	-	Error count
CPU busy	-	an ASAPX built-in metric.
Resp time	-	Time/(Trans count + Error count)
Reject rate	-	Reject count/seconds in interval
Error rate	-	Error count/seconds in interval

Reverse Engineer DataItems - Variables

Application DataItems

- Constant representing cash remaining
- Successful transaction count
- Failed transaction count
- Accumulated time spent processing in the server
- Card reject count

ASAPX Computed Data

- Number of seconds in the interval
- Busy attribute

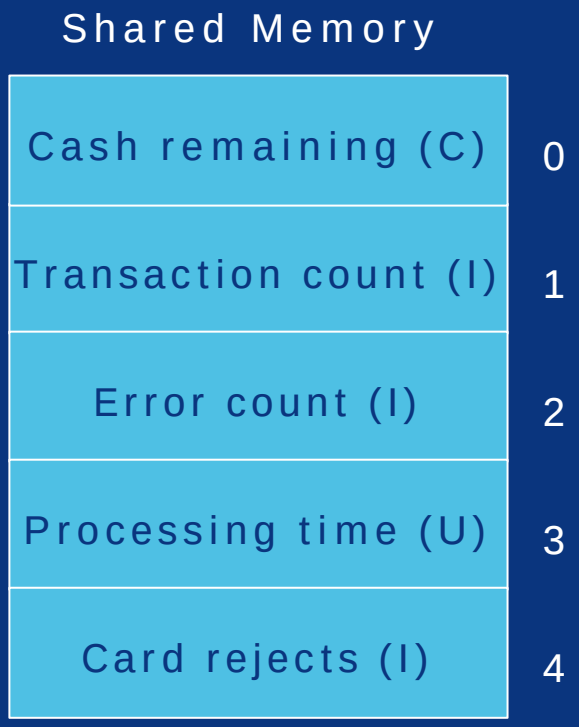
Reverse Engineer DataItems - Types

DataItems are defined in EDL using the DataItems EDL entity property.

They are assigned numbers 0-11, beginning with 0.

DataItems are updated with 64-bit binary values.

DataItems are stored as 64-bit floating point.



I - integer
S,M,U - time units
C - constant

Define EDL - What is an entity?

Entity A = Application A

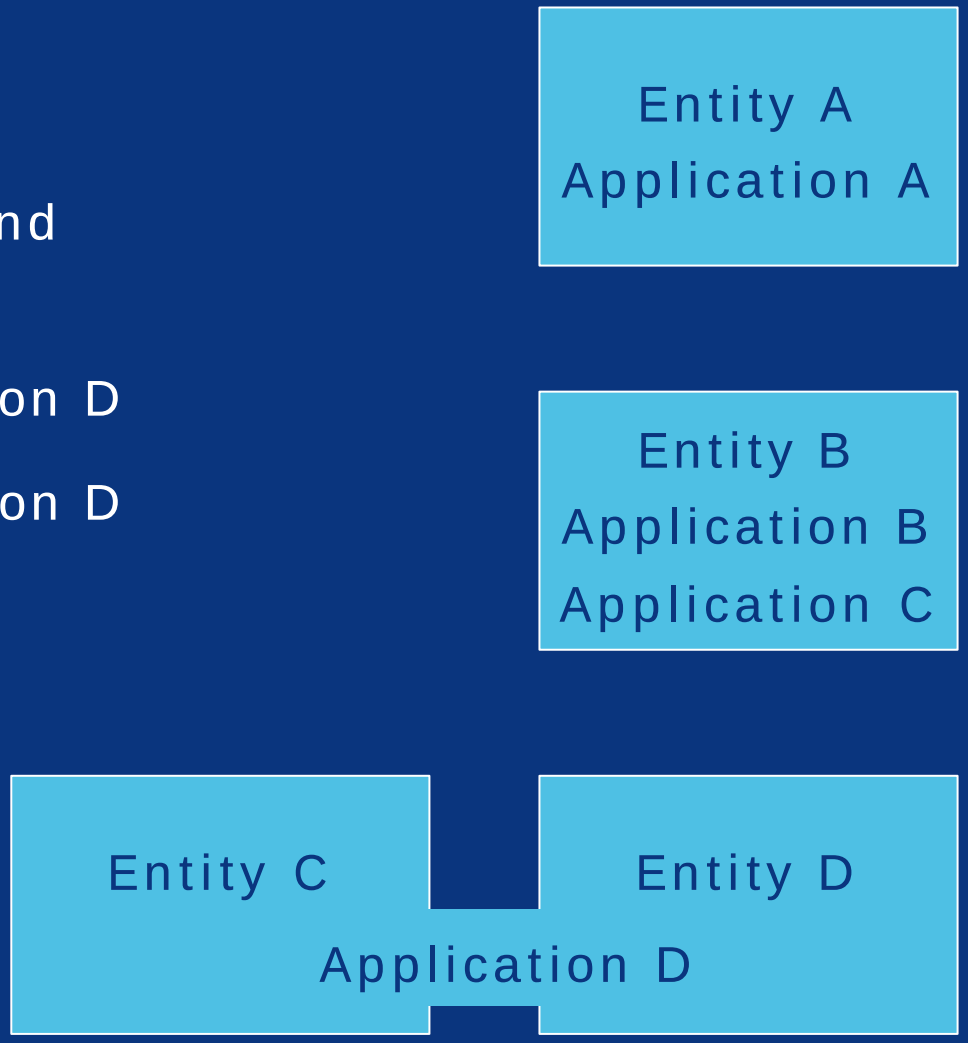
Entity B = Application B and Application C

Entity C = 1/2 of Application D

Entity D = 1/2 of Application D

Common Properties

- DataItems
- Metrics



Define EDL - Application Name Space

Abstract application domain names

Entity name is 1st (leftmost) level

Rules

- 64 bytes or less
- up to 5 levels
- lowest level cannot be “#”
- no commas, quotes, etc.
- no unbalanced trees
- auto name append

Examples

- Atm\Chicago\Loop\Branch12\$Atm43
- Deposit\Chicago\Atm\$Atm43

Define EDL - Entity Properties

```
ENTITY ATM
    CI      ASAP
    Command
     "APP\*ATM,DETAIL,RAW,TAB,STATE,AGGREGATE"
    Detail      "APP^,TAB,STATE,DETAIL,MINSTATE"
    DataItems "0 C, 1 I, 2 I, 3 U, 4 I"
    Enabled   YES
    ErrorState ErrorState
    Help   "ATM Application"
    KeyForNode NodeName  KeyForObj  Domain
    KeyForRow  "Dateymd Time"
    MaxObjectives 200
    SGPFile   ASAPXSGP  SGPManaged YES  SGPSuffix H
    Reserved   NO
    Version   1.00000;
```

Define EDL - Attributes

Copy the Error and ErrorState attributes and place them at the end of the file, following the original ErrorState attribute.

```
AT ErrorState Grid NO Graph NO GraphMax 9
Help "State of Error";

AT Error Grid YES Graph NO GraphMax 0
    Format I4
    Help "Collection Error"
    StatePair YES
    StateRule UseStateGraphState
    TypeData INT64;

AT ErrorState Grid NO Graph NO GraphMax 9
    Help "State of Error";
```

Define EDL - Attributes

```
AT Cash Grid YES Graph NO GraphMax 3000
    Format “I6”
    Help “Cash remaining in ATM“
    StatePair YES
    StateRule UseStateGraphState
    MetricRule “#0“
    TypeData REAL64;

  AT S0 Grid NO Graph NO GraphMax 9
    Help "State of Cash";
```

Define EDL - Attributes

```
AT TotalRate Grid YES Graph NO GraphMax 10
    Format “F7.2”
    Help “Total transaction rate“
    StatePair YES
    StateRule UseStateGraphState
    MetricRule “(#1 + #2)/S“
    TypeData REAL64;

  AT S1 Grid NO Graph NO GraphMax 9
    Help "State of TotalRate";
```

Define EDL - Attributes

```
AT TransRate Grid YES Graph NO GraphMax 1000
    Format “F7.2”
    Help “Successful transaction rate“
    StatePair YES
    StateRule UseStateGraphState
    MetricRule “#1/S“
    TypeData REAL64;

  AT S2 Grid NO Graph NO GraphMax 9
    Help "State of TransRate ";
```

Define EDL - Attributes

```
AT ErrorCount Grid YES Graph NO GraphMax 10
    Format “I3“
    Help “Failed transaction count“
    StatePair YES
    StateRule UseStateGraphState
    MetricRule “#2“
    TypeData REAL64;

  AT S3 Grid NO Graph NO GraphMax 9
    Help "State of ErrorCount";
```

Define EDL - Attributes

```
AT Busy Grid YES Graph NO GraphMax 100
    Format “F6.2”
    Help “Process percent busy"
    StatePair YES
    StateRule UseStateGraphState
    TypeData REAL64;

AT BusyState Grid NO Graph NO GraphMax 9
    Help "State of Busy";
```

Busy is 1 of the 11 ASAPX built-in attributes. No DataItems or MetricRule formulas are required. Just include the definition of a built-in in your EDL file exactly as indicated.

Define EDL - Attributes

```
AT RespTime Grid YES Graph NO GraphMax 10
    Format “F6.3“
    Help “Server response time seconds“
    StatePair YES
    StateRule UseStateGraphState
    MetricRule “#3/(#1 + #2)“
    TypeData REAL64;

  AT S5 Grid NO Graph NO GraphMax 9
    Help "State of RespTime ";
```

Define EDL - Attributes

```
AT RejectRate Grid YES Graph NO GraphMax 0
    Format “F5.3“
    Help “Card reject rate“
    StatePair YES
    StateRule UseStateGraphState
    MetricRule “#4/S“
    TypeData REAL64;

  AT S6 Grid NO Graph NO GraphMax 9
    Help "State of RejectRate";
```

Define EDL - Attributes

```
AT ErrorRate Grid YES Graph NO GraphMax 0
    Format “F6.3“
    Help “Failed transaction rate“
    StatePair YES
    StateRule UseStateGraphState
    MetricRule “#2/S“
    TypeData REAL64;

  AT S7 Grid NO Graph NO GraphMax 9
    Help "State of ErrorRate ";
```

Load/Distribute/Verify EDL

Place new EDL file

- On all NSK servers
- On all ASAP Client workstations
  - C:\Program Files\Tandem\Asap\Edl\MYEDL.edl

Add EDL file to ASAPUSER file

Compile and verify EDL

- On ASAP Client using IDE window
- On NSK Server using ASAP CI
  - EDL automatically compiled at startup
  - Use SHOW command

Load/Distribute/Verify EDL

Restart ASAP on all NSK servers to pick up new EDL

- Watch EMS events for possible EDL error messages

Restart all ASAP Client sessions

- Restarts the Client's private ASAP CI

Test EDL - Use ASAPXTST program

Register a domain for your new Entity.

Update DataItems with expected values.

- SET TEST ON

Validate results using APP commands and ASAP Client.

```
> ASAPXTST
1+ register

error := ASAP_REGISTER_(
      domain^name:domain^name^len -- in/required
      ,seg^offset          -- out/required
      ,error^detail        -- out/optional
      ,segment^id          -- in/optional, default 0
      ,segment^base        -- in/optional
```

Identify Code Points

Where is work committed or failed?

Where are important values computed?

Where are the I/O operations?

Application Source

```
ComputeCash;
<code point 0>

EndTransaction;
<code point 1>

AbortTransaction;
<code point 2>

ReadUpdate $Receive
Reply $Receive
<code points 3>

ComputeShares;
<code point 4>
```

Instrument the Application

Develop an ASAP interface procedure to call from your program whenever a DataItem needs to be updated

- Handle registration as an automatic function of the ASAP interface procedure
- Handle errors
  - For some errors call ASAP_REMOVE_ to remove the domain before re-registering with ASAP using ASAP_REGISTER_.

Add ASAP Globals

- Segment offset variable(s)
- Domain name(s)
- ASAPXDEC or ASAPXH
- Optional Library Reference
- ZASPXC, ZASPXCOB or ZASPXTAL

pTAL Example - Globals

```
int(32)      offset     := -1d;
string  .domain[0:9]  := ["Atm\Server"];

?nolist, source zaspxtal

?nolist, source $system.system.asapxdec

proc asap(ditem,val,math) extensible;
 int ditem; int(64) val; int math;
 forward;
```

pTAL Example - Interface Procedure

```
proc asap(ditem,val,math) extensible;
 int ditem; int(64) val; int math;
begin
int     err  :=  0, err^dtl  :=  0;
int(32) .ext offset^ptr;
if offset <= 0d then begin          ---- if offset <= 0 then register
   @offset^ptr  :=  $xadr(offset);
   if (err := asap_register_(domain:10, offset^ptr,
     err^dtl)) <> 0 then error^msg(4021,err,err^dtl);
   end;
if offset <= 0d then return;
```

pTAL Example - Interface Procedure

```
-- we are registered, update the dataitem
if (err := asap_update_(offset,err^dtl,
  ditem,val,$optional($param(math),math))) <> 0
  then begin
  error^msg(4022,err,err^dtl);
  err   := asap_remove_(offset,err^dtl,,1);
  offset := -1d;
  end;
end; -- of proc asap
```

Calling the ASAP Interface Procedure

```
ComputeCash(cash);
asap(0,cash,1);
.
endtransaction;
asap(1,1f);
aborttransaction;
asap(2,1f);
.
readupdate(…;
rtime = juliantimestamp;
Reply(…;
asap(3,juliantimestamp-rtime);
```

Options?

What about that old code you don't want to touch?

What about that Third Party application you need to know more about?

Check out AutoASAP from TandSoft !


hp
invent