

The Dynamics of z/OS

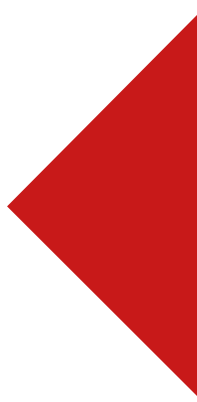
Peter Relson

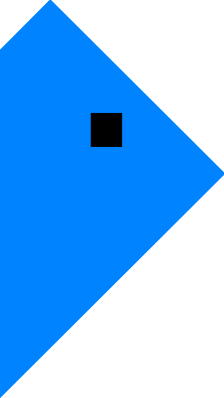
z/OS Core Technology Design

Corporation, Poughkeepsie, NY

(845)-435-8390

relson@us.ibm.com

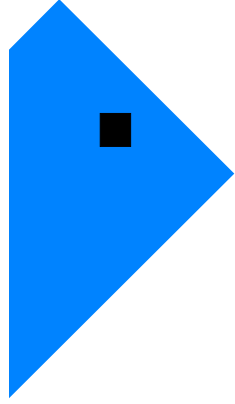




Abstract

This talk explores dynamic APF, dynamic Exits, dynamic LNKLST, dynamic LPA, and dynamic SSI -- what they can do, and what they can't do, and how you can use them.





Trademarks

■ **IBM**

■ **z/OS**

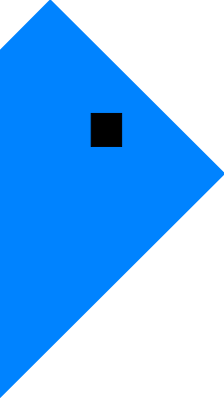




Introduction

- PROGxx
 - Dynamic APF
 - Dynamic Exits
 - Dynamic LNKLST
 - Dynamic LPA
- Dynamic SSI

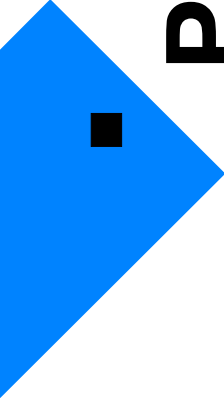




PROG

- PROG=xx system parameter (PROGxx)
- SET PROG=xx command (PROGxx)
- SETPROG command
- DISPLAY PROG
- Program Interfaces

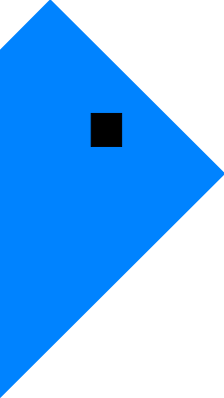




PROG (cont)

- Parmlib supports keyword(value) notation, blank separators
- Command supports keyword=value notation, comma separators
- Parmlib supports command notation
- Command supports parmlib notation





Dynamic APF

- Dynamic APF is often used when installing a new product which requires additional data sets to be APF-authorized.





Dynamic APF, command

- **SETPROG APF,FORMAT=f**
 - Sets APF list to the specified format (dynamic or static)
- **SETPROG APF,ADD,DSNAME=d,VOL=v**
 - Adds the named data set on the named volume
- **SETPROG APF,ADD,DSNAME=d,SMS**
 - Adds the SMS-managed named data set
- **SETPROG APF,DELETE,DSNAME=d,VOL=v**
 - Deletes the named data set on the named volume
- **SETPROG APF,DELETE,DSNAME=d,SMS**
 - Deletes the SMS-managed named data set





Dynamic APF, display

- `DISPLAY PROG,APF`
 - Display the entire list
- `DISPLAY PROG,APF,DSNAME=d`
 - Display information about a particular data set





Dynamic APF, CSVAPF

- Request=QUERY: by dsname/volume
- Request=ADD: by dsname/volume
- Request=DELETE: by dsname/volume
- Request=LIST: extract entire APF list
- Request=QUERYFORMAT
- Request=DYNFORMAT

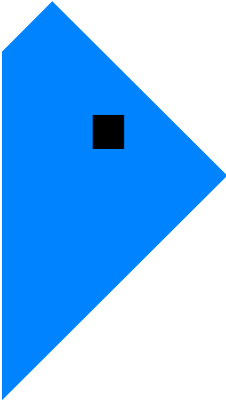




Dynamic APF, comments

- By now, everyone should be using dynamic APF
- Future release: remove the option and enforce dynamic-only

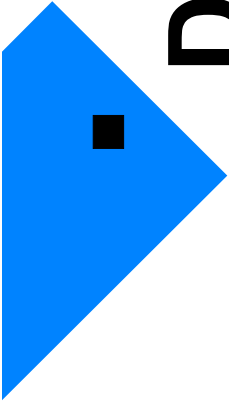




Dynamic Exits

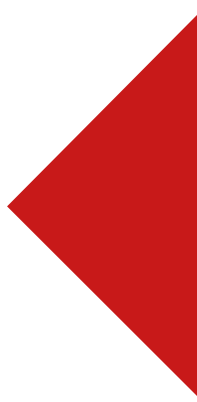
- The Dynamic Exits facility has two main users -- installation exits and program exits
- Since it accommodates multiple exit routines at an exit point it facilitates multi-application use of an exit.

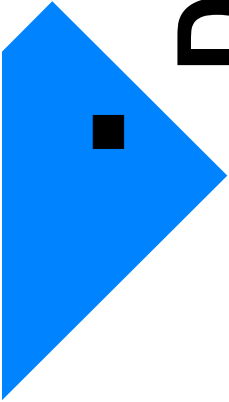




Dynamic Exits, command

- SETPROG
EXIT,ADD,EXITNAME=e,MODNAME=m
 - Associate exit routine with exit.
 - Can also define whether
 - ▶ state is active/inactive
 - ▶ limit to a jobname
 - ▶ number of abends
 - ▶ first or last to be called
 - ▶ data set to fetch from





Dynamic Exits, command (*continued*)

- SETPROG EXIT,ATTRIB,EXITNAME=e
 - keeprc: comparison, value
- SETPROG EXIT,DELETE,EXITNAME=e,
MODNAME=m,FORCE=YES
- SETPROG EXIT,MODIFY,EXITNAME=e,
MODNAME=m
 - Change whether
 - state is active/inactive
 - limit to a jobname
- SETPROG EXIT,UNDEFINE,EXITNAME=e





Dynamic Exits, display

- `DISPLAY PROG,EXIT`
 - all exit names
- `DISPLAY PROG,EXIT,EXITNAME=e`
 - all exit routines for a given exit name
- `DISPLAY PROG,EXIT,EXITNAME=e,DIAG`
 - all exit routines for a given exit name with diagnostic information
- `DISPLAY PROG,EXIT,MODNAME=m`
 - all exit routines that have a particular exit routine (module)

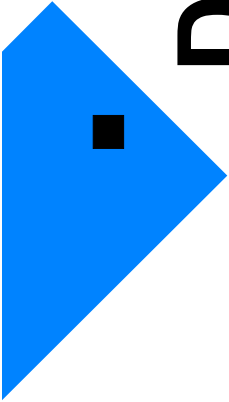




Dynamic Exits, CSVDSYNEX

- Request=DEFINE by exitname
 - AMODE requirement
 - Reentrancy requirement
 - Persistence attribute
 - Abend number characteristic
 - Use of fastpath
 - Return code manipulation
 - Return code to stop calling exit routines

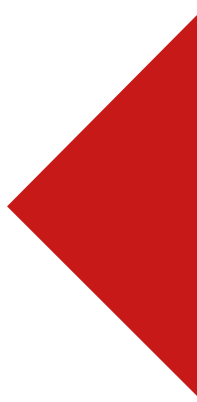


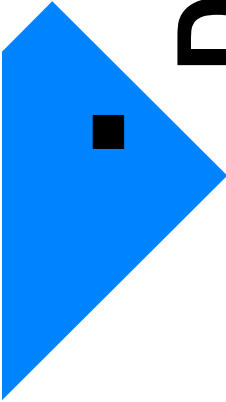


Dynamic Exits, CSVSYNEX

(continued)

- **Request=ADD:** by exitname, modname
 - State (activate or inactive)
 - Position in list of exit routines
 - Where to locate exit routine
 - Jobname or STOKEN characteristic
 - Abend number characteristic
- **Request=DELETE:** by exitname, modname
- **Request=UNDEFINE:** by exitname



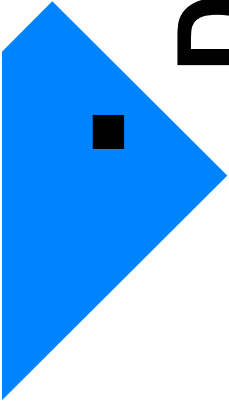


Dynamic Exits, CSV DYNEX

(continued)

- Request=ATTRIB: by exitname
 - Change KEEPRC attribute
- Request=LIST: extract information
 - By Exitname or all exits
 - Indicate version of output



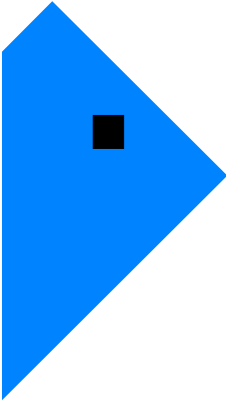


Dynamic Exits, CSVDSYNEX

(continued)

- Request=CALL: by exitname
 - Register information for the exit routine
 - Return information characteristics (last, lowest, highest, all)
 - Fastpath or not
- Request=RECOVER: by exitname
- Request=QUERY: by exitname
 - Any exit routines if I issued "CALL"?
 - Any exit routines added at all?

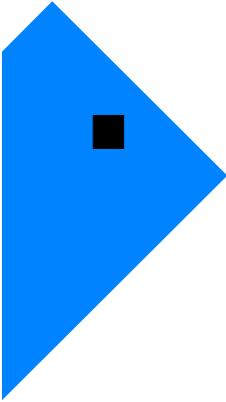




Dynamic Exits, comments

- For SMF, exit names are **not** "IEFUSI" and "IEFACTRT". They are
 - **SYS.IEFUSI** etc.
 - **SYSSTC.IEFUSI** etc.

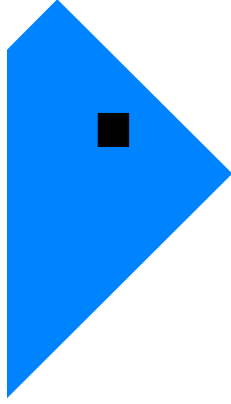




Dynamic LNKLST

- Dynamic LNKLST is typically used when adding a new product that requires additional data set(s) in the LNKLST.
- It is hoped that such new data sets are (for the most part) to be used by jobs that have not yet started

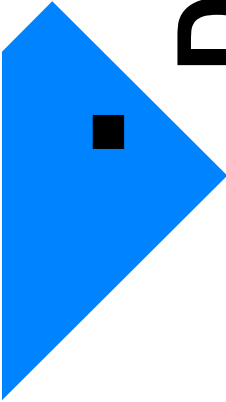




Dynamic LNKLST, command

- `SETPROG LNKLST,DEFINE,NAME=n,COPYFROM=c`
 - define a new LNKLST set copied from the named LNKLST set

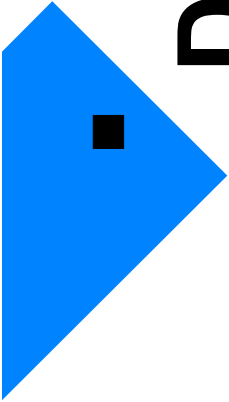




Dynamic LNKLST, command (*continued*)

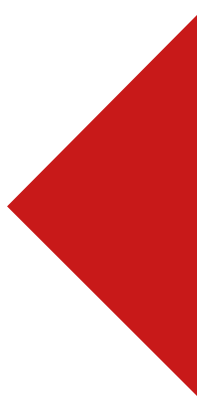
- `SETPROG LNKLST,ADD,NAME=n,DSNAME=d`
 - add the data set to the named LNKLST set
 - can also specify
 - ▶ volser if not cataloged
 - where to place it:
 - ▶ at top (after system-defined data sets)
 - ▶ at bottom
 - ▶ after some specific data set

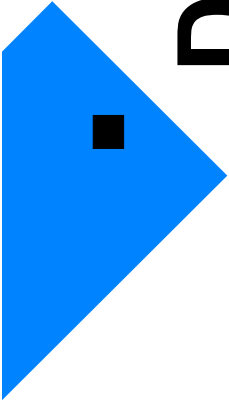




Dynamic LNKLST, command (*continued*)

- SETPROG LNKLST,DELETE,NAME=n,DSNAME=d
 - delete the data set from the named LNKLST set
- SETPROG LNKLST,UNDEFINE,NAME=n
 - remove definition of the LNKLST set
- SETPROG LNKLST,TEST,NAME=n,MODNAME=m
 - test if module is in that LNKLST set
- SETPROG LNKLST,ACTIVATE,NAME=n
 - activate the LNKLST set

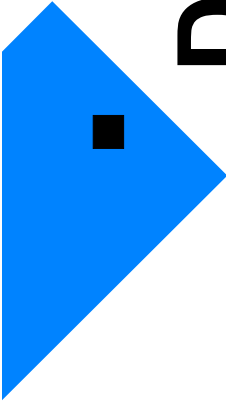




Dynamic LNKLST, command (*continued*)

- **SETPROG LNKLST,UPDATE,{JOB=j | ASID=a}**
 - change the specified job(s) to use the current LNKLST
 - **DANGER DANGER DANGER**
 - This results in closing the previously-opened DCB
 - That in turn results in freeing the DCB and DEB
 - Operations in progress can easily blow up for many reasons
 - Operations not yet begun might even blow up





Dynamic LNKLSST, command (*continued*)

- SETPROG LNKLSST,UNALLOCATE
 - Get rid of data set allocations
 - **ONLY REASON:** permit manipulation of uncataloged data set of same name
- SETPROG LNKLSST,ALLOCATE
 - Restore data set allocations

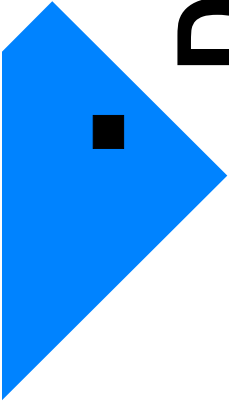




Dynamic LNKLST, display

- `DISPLAY PROG, LNKLST`
 - display the current LNKLST set
- `DISPLAY PROG, LNKLST, NAME=n`
 - display the named LNKLST set
- `DISPLAY PROG, LNKLST, NAMES`
 - display the names of all LNKLST sets
- `DISPLAY PROG, LNKLST, USERS, NAME=n`
 - display the jobnames using the named LNKLST set



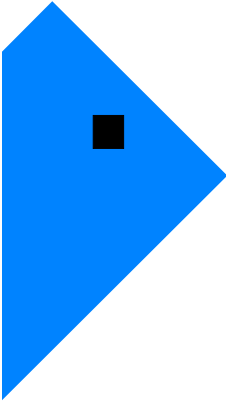


Dynamic LNKLSST, display

(continued)

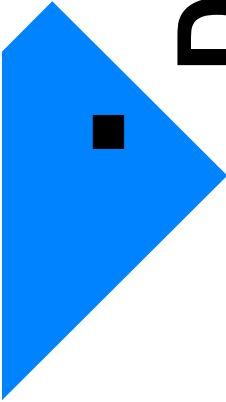
- `DISPLAY PROG, LNKLSST, ASID=a`
 - display the LNKLSST set associated with the ASID
- `DISPLAY PROG, LNKLSST, JOBNAME=j`
 - display the LNKLSST set associated with the job





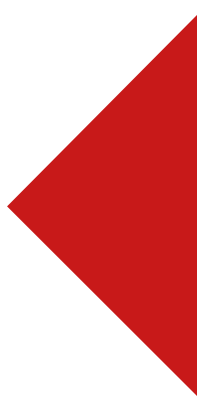
Dynamic LNKLST, CSVSYNL

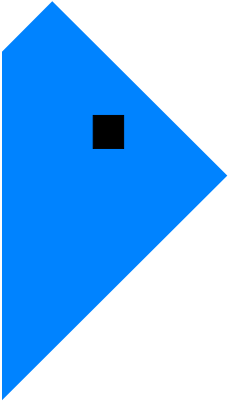
- Request=DEFINE a new LNKLST set
 - Where to copy from
- Request=ADD a data set to LNKLST set
 - By volume or catalog
 - Position in LNKLST
- Request=DELETE a data set from LNKLST set
- Request=UNDEFINE a LNKLST set
- Request=ACTIVATE a LNKLST set
- Request=TEST a LNKLST set for a module



Dynamic LNKLIST, CSVDYNL (*continued*)

- Request=LIST
 - All LNKLIST sets or specific LNKLIST set
 - Users of LNKLIST set(s)
 - LNKLIST set associated with a job
- Request=UPDATE job(s)
 - By (wildcarded) jobname or ASID
 - **Same DANGERS as command apply**





Dynamic LNKLST, comments

- Things not available
 - Defaults such as COPYFROM=CURRENT
 - Safe UPDATE (sorry, never will be)





Dynamic LPA

- Dynamic LPA is typically used when
 - Installing a new product that needs things in LPA
 - When a product has items that it needs in common storage that must reside in a PDSE

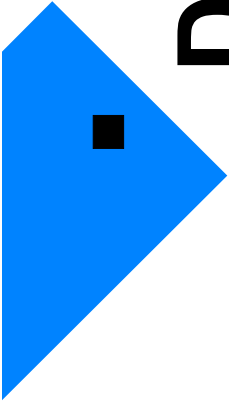




Dynamic LPA, command

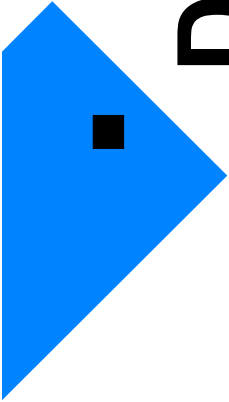
- SETPROG LPA,ADD,MODNAME=(m1,...,mN),
DSNAME=d
 - Add to LPA the named module(s) from the data set
 - Important to add, in same operation, module and all is aliases
 - Can also indicate
 - Put into fixed storage
 - Page protect only the full pages (save space, but more dangerous)





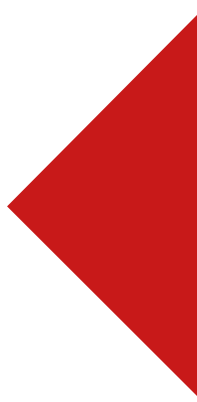
Dynamic LPA, command (*continued*)

- `SETPROG LPA,ADD,MASK=m,DSNAME=d`
 - Add to LPA all module(s) from the data set whose names match the mask
 - Can also indicate
 - Put into fixed storage
 - Page protect only the full pages (**save space, but more dangerous**)



Dynamic LPA, command (*continued*)

- SETPROG LPA,DELETE,MODNAME=(m1,...,mN)
 - Delete from LPA the named module(s)
 - Must specify FORCE=YES
 - Can also indicate
 - ▶ Delete the CURRENT dynamic copy
 - ▶ Delete the OLDEST dynamic copy
- SETPROG LPA,CSAMIN=(csa,ecsa)
 - Set remaining-storage thresholds for CSA and ECSA





Dynamic LPA, display

- `DISPLAY PROG,LPA,MODNAME=m`
 - display information about the named module
- `DISPLAY PROG,LPA,CSAMIN`
 - display information about the (E)CSA minimums for LPA





Dynamic LPA, CSV DY LPA

- Request=ADD module(s) to LPA
 - By data set or by address
 - By module name or by member mask
- Request=DELETE module(s) from dynamic LPA
 - Specific one by token
 - Current dynamic copy
 - Oldest dynamic copy

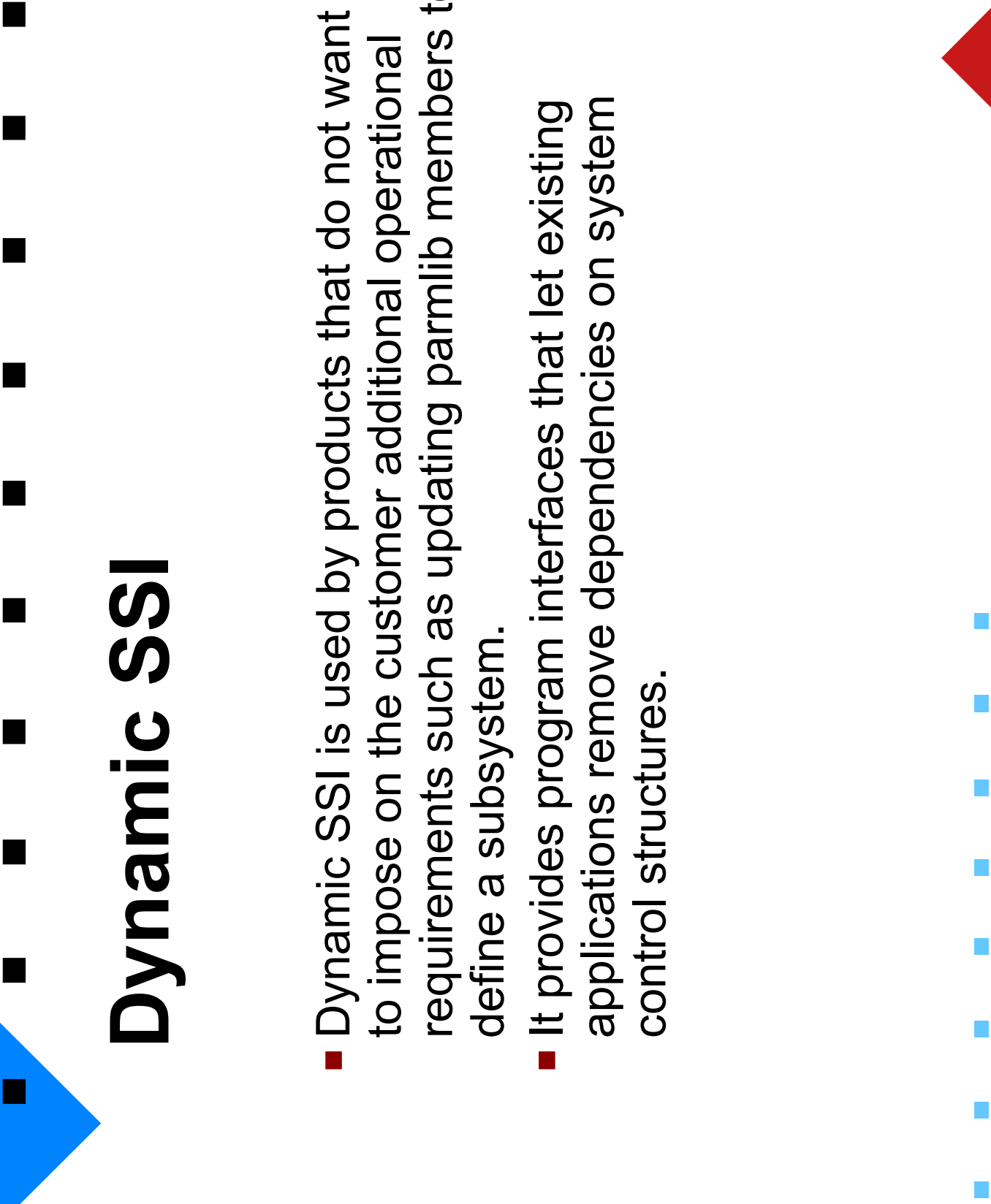
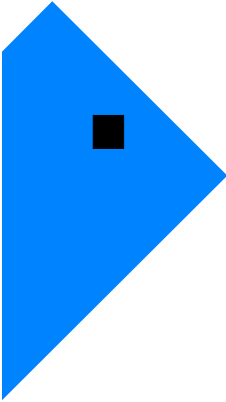




Dynamic LPA, comments

- Things not available
 - Safe Delete from LPA (will never be truly safe, but might reject a delete when the use count is not 0)
 - Automatic ALIASes for LPA ADD (and providing a way to default to that)
 - IPL-time specification of PDSEs for dynamic LPA





Dynamic SSI

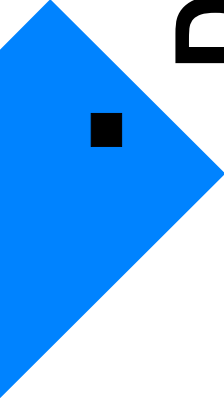
- Dynamic SSI is used by products that do not want to impose on the customer additional operational requirements such as updating parmlib members to define a subsystem.
- It provides program interfaces that let existing applications remove dependencies on system control structures.



Dynamic SSL, command

- **SETSSI ADD,SUBNAME=subname**
 - adds a subsystem (table update)
- **SETSSI ACTIVATE,SUBNAME=subname**
 - starts routing commands
 - must have SSL-managed vector table (via IEFSSVT)
 - only if dynamic and "permits" SETSSI

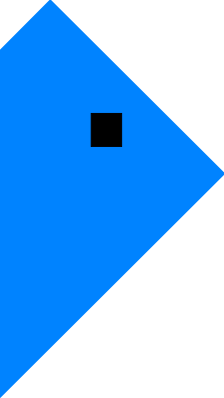




Dynamic SSL, command (*continued*)

- SETSSI DEACTIVATE,SUBNAME=subname
 - stops routing commands, ones in progress continue to completion
 - still defined!
 - only if dynamic and "permits" SETSSI

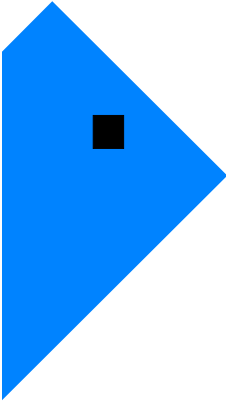




Dynamic SSL, display

- Display SSL
 - by dynamic yes/no
 - by name

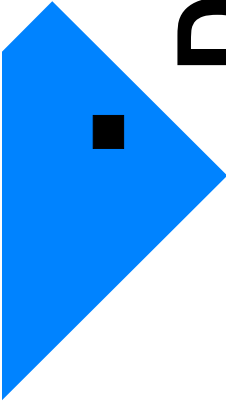




Dynamic SSI, IEFSSI

- REQUEST=ADD
 - adds subsystem
- REQUEST=ACTIVATE
 - activates subsystem
- REQUEST=OPTIONS
 - indicates if/that the subsystem responds to SETSSI
- REQUEST=DEACTIVATE
- REQUEST=SWAP
 - replaces subsystem vector





Dynamic SSL, IEFSSI (*continued*)

- REQUEST=PUT
 - stores subsystem-defined data, 2 fullwords
- REQUEST=GET
 - retrieves subsystem-defined data, 2 fullwords
- REQUEST=QUERY
 - whether active/inactive, whether dynamic, whether responds to SETSSI commands
 - list of function codes. Mapped by IEFJSQRY.

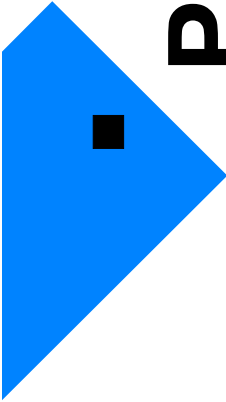




Dynamic SSL, comments

- Things not available
 - Delete
 - Tracing of calls
 - Reinitialization
 - Document of the internal SSL's used for managing data sets
 - Automatic disabling
 - Timing Support
 - SET SSN=xx
 - Support for Non-ENQ environments





Publications

- z/OS V1R5.0 MVS Auth Assm Services Reference
ALE-DYN SA22-7609
- z/OS V1R5.0 MVS Auth Assm Services Reference
ENF-IXG SA22-7610
- z/OS V1R5.0 MVS Initialization and Tuning
Reference SA22-7592
- z/OS V1R5.0 MVS Installation Exits SA22-7593
- z/OS V1R5.0 MVS System Commands SA22-7627
- z/OS V1R5.0 MVS Using the Subsystem Interface
SA22-7642