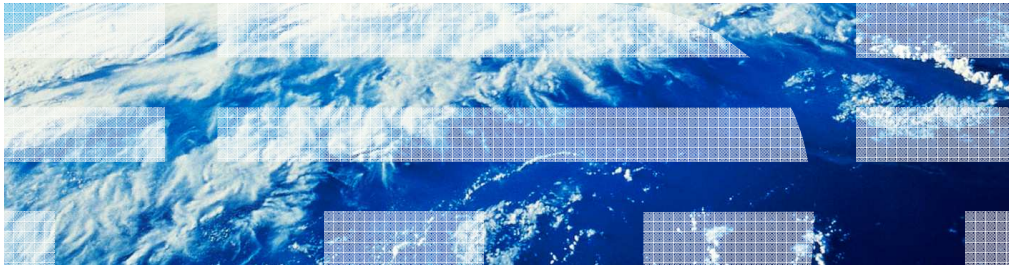


# z/OS V1R13

Language environment:  
LE CEEPIPI multi main and user word



---

## Session objectives

- Describe new CEEPIPI interfaces that support conversions from PICI to CEEPIPI

## Overview

- Problem statement / need addressed
  - Conversion from the preinitialization compatibility Interface (PICl) to preinit would be easier if additional interfaces were provided in preinit. Specifically:
    - Support for multiple main environments on one TCB;
    - Support for a user word that can be accessed from both outside and within a preinit environment
- Solution
  - Provide additional preinit interfaces
- Benefit / value
  - Facilitates conversion from PICl to preinit, so that users can move to the more strategic interface

Preinitialization Compatibility Interface (PICl) is an older form of Preinit that is supported but no longer being enhanced. Users that would like to take advantage of newer functionality need to change their assembler programs to use the Preinit interfaces.

## Usage and invocation (1 of 5)

- Support for multiple main environments on one TCB
- New CEEPIPI function `init_main_dp`
  - Allows the preinit assembler driver to create multiple main CEEPIPI environments on the same TCB
  - Main programs can be called on these environments, but only one call can be active at a time on a given TCB

## Usage and invocation (2 of 5)

Support for multiple main environments on one TCB...

```
CALL CEEPIPI(init_main_dp,ceexptbl_addr,service_rtns,token)
```

- `init_main_dp` (input) - A fullword containing the `init_main_dp` function code (integer value = 19).
- `ceexptbl_addr` (input) - A fullword containing the address of the PreInit table to be used during initialization of the new environment.
- `service_rtns` (input) - A fullword containing the address of the service routine vector or 0, if there is no service routine vector.
- `token` (output) - A fullword containing a unique value used to represent the environment.

## Usage and invocation (3 of 5)

### Support for preinit user word

- Facilitates communication between the preinit assembler driver and the user code running within a preinit environment
- Preinit assembler driver uses CEEPIPI interfaces to access the user word
  - CEEPIPI(set\_user\_word,...) sets the user word value
  - CEEPIPI(get\_user\_word,...) retrieves the user word value from the last set\_user\_word call
- Code running within preinit environment accesses the user word from within the CAA control block
  - Field CEECAA\_USER\_WORD in the assembler CEECAA mapping
    - 4 byte field located at offset +3F0x
  - Modifications to this field by the user code running in the preinit environment are not saved between CEEPIPI calls
    - Next CEEPIPI call will use value from last set\_user\_word call

## Usage and invocation (4 of 5)

Support for preinit user word...

CALL CEEPIPI(set\_user\_word,token,value)

- set\_user\_word (input) - A fullword containing the set\_user\_word function code (integer value = 17)
- token (input) - A fullword with the value of the token of the environment
- value (input) - A fullword value that will be used to initialize the user word in the initial thread CAA when the application is invoked

### Usage Notes:

.This value will be saved away in an area associated with the passed-in environment token. It will be copied into the CAA for the initial thread when the next call\_main/\_sub/\_sub\_addr/\_sub\_addr\_nochk/\_sub\_addr\_nochk2 function is done to start an application. The application can then examine or update this user word in the CAA (CEECAA\_USER\_WORD). When the application ends, the final value in CEECAA\_USER\_WORD is not copied back into the area associated with the environment token. When the next application is started using call\_main/\_sub etc. function, the user word value last established by (set\_user\_word) will be used again.

.The user word associated with the environment token is initialized to 0 when (init\_main), (init\_sub), or (init\_sub\_dp) is done. The CAA for the initial process thread will be initialized with 0 if no (set\_user\_word) function call has been done before the application is started.

.The user word in all CAAs other than the initial thread CAA is set to 0.

.When fork() is done, the user word in the CAA for the new process will inherit whatever value is in the CAA at the time fork() is done.

.The use of the CAA user word is not supported in the Assembler User Exit Routine (CEEBXITA and related modules), or in the CEEPIPI service routines specified in the service routine vector.

.Any user code that runs on a CEEPIPI environment before the first call\_main/\_sub etc. request will see zero in the CAA\_USER\_WORD. Examples of this code include static constructors run for programs that get loaded when a CEEPIPI environment is initialized. Any changes to the CAA\_USER\_WORD made by this code will be overlaid when the next call\_main/\_sub etc. is done for that environment.

zOS\_V1R13\_Language\_Environment\_LE-CEEPIPI-MultiMain-UserWord.ppt

## Usage and invocation (5 of 5)

Support for preinit user word...

CALL CEEPIPI(get\_user\_word,token,value)

- get\_user\_word (input) - A fullword containing the get\_user\_word function code (integer value = 18)
- token (input) - A fullword with the value of the token of the environment
- value (output) - A fullword that will be returned containing the current value that will be used to initialize the CAA user word when the next application is invoked

### Usage Notes:

.This value will be saved away in an area associated with the passed-in environment token. It will be copied into the CAA for the initial thread when the next call\_main/\_sub/\_sub\_addr/\_sub\_addr\_nochk/\_sub\_addr\_nochk2 function is done to start an application. The application can then examine or update this user word in the CAA (CEECAA\_USER\_WORD). When the application ends, the final value in CEECAA\_USER\_WORD is not copied back into the area associated with the environment token. When the next application is started using call\_main/\_sub etc. function, the user word value last established by (set\_user\_word) will be used again.

.The user word associated with the environment token is initialized to 0 when (init\_main), (init\_sub), or (init\_sub\_dp) is done. The CAA for the initial process thread will be initialized with 0 if no (set\_user\_word) function call has been done before the application is started.

.The user word in all CAAs other than the initial thread CAA is set to 0.

.When fork() is done, the user word in the CAA for the new process will inherit whatever value is in the CAA at the time fork() is done.

.The use of the CAA user word is not supported in the Assembler User Exit Routine (CEEBXITA and related modules), or in the CEEPIPI service routines specified in the service routine vector.

.Any user code that runs on a CEEPIPI environment before the first call\_main/\_sub etc. request will see zero in the CAA\_USER\_WORD. Examples of this code include static constructors run for programs that get loaded when a CEEPIPI environment is initialized. Any changes to the CAA\_USER\_WORD made by this code will be overlaid when the next call\_main/\_sub etc. is done for that environment.

zOS\_V1R13\_Language\_Environment\_LE-CEEPIPI-MultiMain-UserWord.ppt



## Interactions and dependencies

- Software dependencies
  - None
- Hardware dependencies
  - None
- Exploiters
  - None

## Migration and coexistence considerations

- None

## Installation

- None

---

## Session summary

- Additional preinit interfaces are available to make conversions from PIC1 easier
  - Support for multiple main environments on one TCB;
  - Support for a user word that can be accessed from both outside and within a preinit environment

---

## Appendix - References

- z/OS language environment programming guide - Sa22-7561



## Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, and z/OS are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at <http://www.ibm.com/legal/copytrade.shtml>

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

© Copyright International Business Machines Corporation 2012. All rights reserved.