Start the Debugger

This section describes basic operational requirements and provides a rough guideline on how to proceed when planning to apply the debugger.

- Debugger under Natural Security
- Operational Requirements
- Invoke the Debugger
- Default Object

Debugger under Natural Security

The use of the debugger can be controlled by Natural Security:

- You can protect the debugger against unauthorized use by disallowing the `TEST` system command, which invokes the debugger; see *Command Restrictions* in the section *Library Maintenance* in the *Natural Security* documentation.
- You can disallow or restrict the use of the debugger as described in *Components of an Environment Profile* in the *Natural Security* documentation.

Operational Requirements

The debugger is only invoked when you execute a cataloged object stored in the current library in the current Natural system file. The debugger is *not* invoked when you execute source code contained in the work area by using the `RUN` command.

Efficient and correct debugging requires that the source code in the source object corresponds to the compiled source code in the cataloged object which can be guaranteed with the system command `STOW`. If you change a source object *after* you cataloged it, it is possible that a debug entry (breakpoint or watchpoint) does not function properly because the referenced statement or variable has changed or no longer exists. When the debugger detects that a source object has an earlier time stamp than the corresponding cataloged object, the following warning appears `Time stamps of source and cataloged object do not match.`

The debugger investigates all Natural objects contained in the current library or in one of its steplibs. The debugger does not investigate Natural objects stored in the Natural system library SYSLIB or SYSLIBS.

The following restrictions apply to the use of the debugger:

- The debugger can only be applied to objects of Natural Version 2.3 and above, but not to Natural objects cataloged with any previous version. The debugger supports only debug environments which were created with Natural Version 2.3 and above; debug environments created with any previous version will be ignored. For detailed information on debug environments, see *Debug Environment Maintenance*.

- For objects cataloged in Natural versions earlier than Version 4.1, breakpoints can only be set on lines with WHEN and VALUE clauses. These restrictions do not apply to objects cataloged in Natural Version 4.1 and above.

Batch Processing

Although the debugger is mainly designed for interactive usage in online mode, the debugger commands can also be used for batch execution such as for setting breakpoints or watchpoints.

Note:
There are restrictions for batch processing which can cause a debugger command to be rejected. For example, the debugger does not support the commands ++ and +4.

Example of Generating and Printing Statistics in Batch

The following is an example of using debugger direct commands in batch mode to generate and print a report about call statistics:

```
//NATBATCH EXEC PGM=NATBAT42,
//   PARM=('INTENS=1,IM=D,CF=$,PRINT=((1-2),AM=STD)')
//STEPLIB  DD DISP=SHR,DSN=NATURAL.V2.TEST.NUCLEUS
//CMPRINT  DD SYSOUT=X
//SYSOUT   DD SYSOUT=X
//CMPRT01  DD SYSOUT=X
//CMSYNIN  DD *
LOGON DEBUGLIB
TEST PROFILE
,,,,CMPRT01
,,,,,,$K3
,,$K3
TEST ON
TEST SET XSTAT COUNT
DEBUG2P
TEST PRINT XSTAT
FIN
/*
```

Invoke the Debugger

To invoke the debugger

1. Establish a debug environment for a Natural object or application:

   - Invoke the **Debug Main Menu** by entering the Natural system command TEST.

     Or:
     From within a running application, enter the terminal command %<TEST.

   - Use the functions of the **Debug Main Menu** to specify debug entries for a Natural object or application:

     Debug environment maintenance
     Spy maintenance
     Breakpoint maintenance
     Watchpoint maintenance

Call statistics maintenance
Statement execution statistics maintenance
Variable maintenance
List object source

2. Activate the debugger:

   - At a command prompt, enter the command `TEST ON`.

     Or:
     In the **Debug Main Menu**, enter function code `T`.

3. Execute the Natural object or application.

   The debugger pauses program execution at the specified debug entries and invokes the **Debug Break** window.

To invoke the debugger for error handling

- At session start, set the profile parameter `DBGERR` to `ON`.

  See also *DBGERR - Automatic Start of Debugger at Runtime Error* in the *Parameter Reference* documentation.

  Or:
  During the session, enter the command `TEST ON` at a command prompt or enter function code `T` in a main debug maintenance menu.

  The debugger invokes the **Debug Break** window when a Natural error occurs.

See also the section *Error Handling*.

Default Object

The maintenance functions of the debugger as described in the relevant sections, refer to objects you specify either in the corresponding name fields of menus or with direct commands. If you do not specify an object name, by default, the debugger assumes the name of the current object as it is displayed in the **Object** field, in the upper right corner of the **Debug Main Menu**. With a default object specified, no object name is required in direct commands and menu options used to specify breakpoints or watchpoints. To change the default object, see the syntax of the command `SET` in the section *Command Summary and Syntax*.