

Z**DC**®

VS Code Extension Quick Start

Z/XDC® Release v3.1 for z/OS

Contents

CONTENTS	i
XDC VS Code Extension Requirements	1
XDC VS Code Extension Installation	1
XDC VS Code Extension: Adding a Server	4
Connecting to an XDC Debugging Session	6
Extension Features	9

List of Figures

1	Steps to install VS Code extension	2
2	XDC Extension Installed	3
3	Add XDC GUI Server	4
4	Add XDC GUI Server Panel	5
5	XDC GUI Server added	6
6	A waiting XDC CDF session	6
7	Connecting to an XDC debugging session	7
8	XDC VS Code Extension	8
9	Register Sets	9
10	Debug Console	10
11	Debug Console Response	10
12	Watch Panel	11
13	Call Stack Panel	12
14	Frame Details Panel	12
15	Loaded Scripts Panel	13
16	Command Palette	13

XDC VS Code Extension Requirements

!!!IMPORTANT!!! The XDC extension requires HTTPS to be enabled on your z/OS system. If your z/OS HTTP Server presents a certificate that is not issued by a publicly-trusted CA, you must install its trust anchor into the machine/system store of the operating system running VS Code. If the certificate is signed by your organization's internal (private) CA, install the *CA signing* certificate — not the server certificate used by z/OS HTTP Server. If it is a genuinely self-signed certificate, that certificate is its own signer, so install it directly.

Installing the CA The signing certificate must be installed into the machine/system store, not a per-user store.

Windows

- Open the certificate (.cer/.crt) → Install Certificate.
- Choose Local Machine (not Current User).
- Select Place all certificates in the following store → Trusted Root Certification Authorities.
- Confirm it appears under certlm.msc → Trusted Root Certification Authorities → Certificates.

macOS

Import into the System keychain and mark it trusted:

```
sudo security add-trusted-cert -d -r trustRoot \  
-k /Library/Keychains/System.keychain your-ca.pem
```

Or via Keychain Access: drag the certificate into the System keychain, open it, expand Trust, and set When using this certificate → Always Trust.

Linux

Add the CA to the system bundle, then rebuild it:

- Debian / Ubuntu / SUSE: copy the PEM (named *.crt) to /usr/local/share/ca-certificates/ then run `sudo update-ca-certificates`
- RHEL / Fedora / CentOS: copy to /etc/pki/ca-trust/source/anchors/ then run `sudo update-ca-trust`
- Arch: `sudo trust anchor your-ca.pem`

Note: The extension reads the system trust store when it starts, so install the certificate *before* launching VS Code — or, if VS Code is already running, reload the window (Command Palette → Developer: Reload Window). This trust step applies to desktop VS Code only; a browser-based build cannot read the local trust store.

XDC VS Code Extension Installation

To install the XDC VS Code Extension, you will need access to the extension package named **code-xdc-0.x.y.vsix**.

The quickest and simplest way to install our extension is to search for **XDC** in the Visual Studio Marketplace directly in VS Code. The extension package can also be downloaded directly from [Visual Studio Marketplace](#) or from our [website](#).

To install the extension manually, download extension package to your desktop, open up VS Code and then...

- Step 1: Open the VS Code Extensions panel.
- Step 2: Click on the hamburger menu...

- Step 3: Select "*Install from VSIX...*"
- Step 4: Navigate to where **code-xdc-0.x.y.vsix** is saved on your desktop and select it.

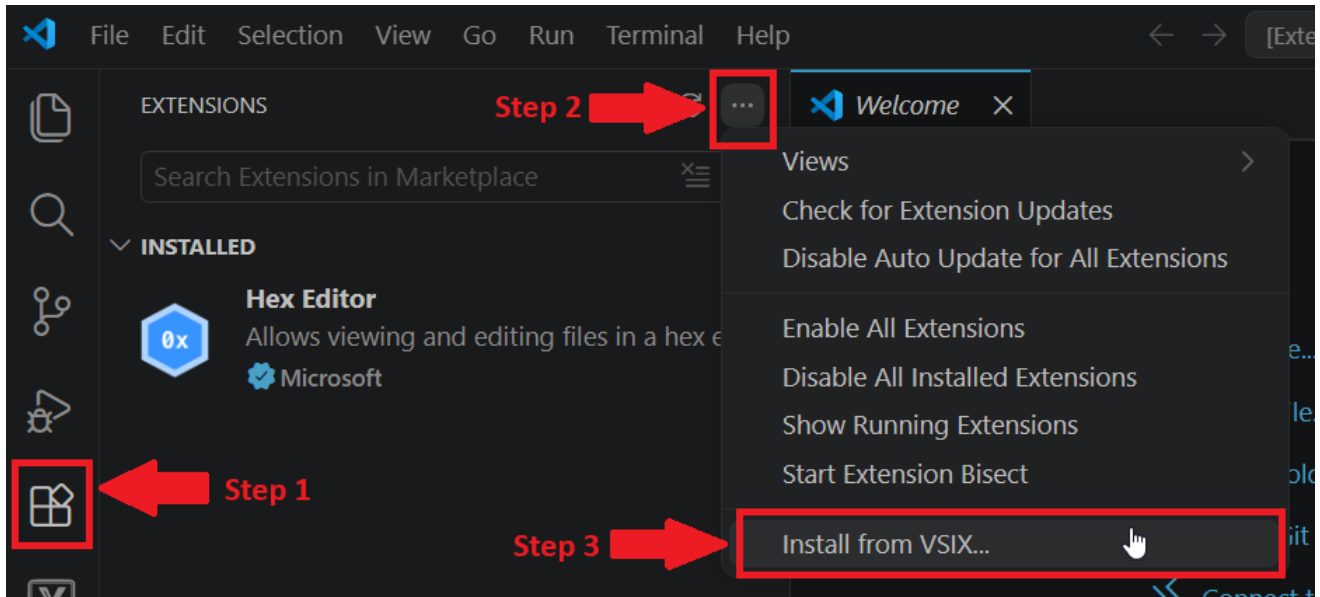


Figure 1: Steps to install VS Code extension

The XDC VS Code Extension can also be installed from the command line. Open a command prompt/shell/Powershell/etc and change to the directory containing the XDC VS Code Extension package and issue:

```
code --install-extension code-xdc-0.x.y.vsix
```

And, that's it. Once installed, the XDC Extension will appear in the side panel. When a new code drop occurs, just install the new package. VS Code will automatically replace the older version with the new one.

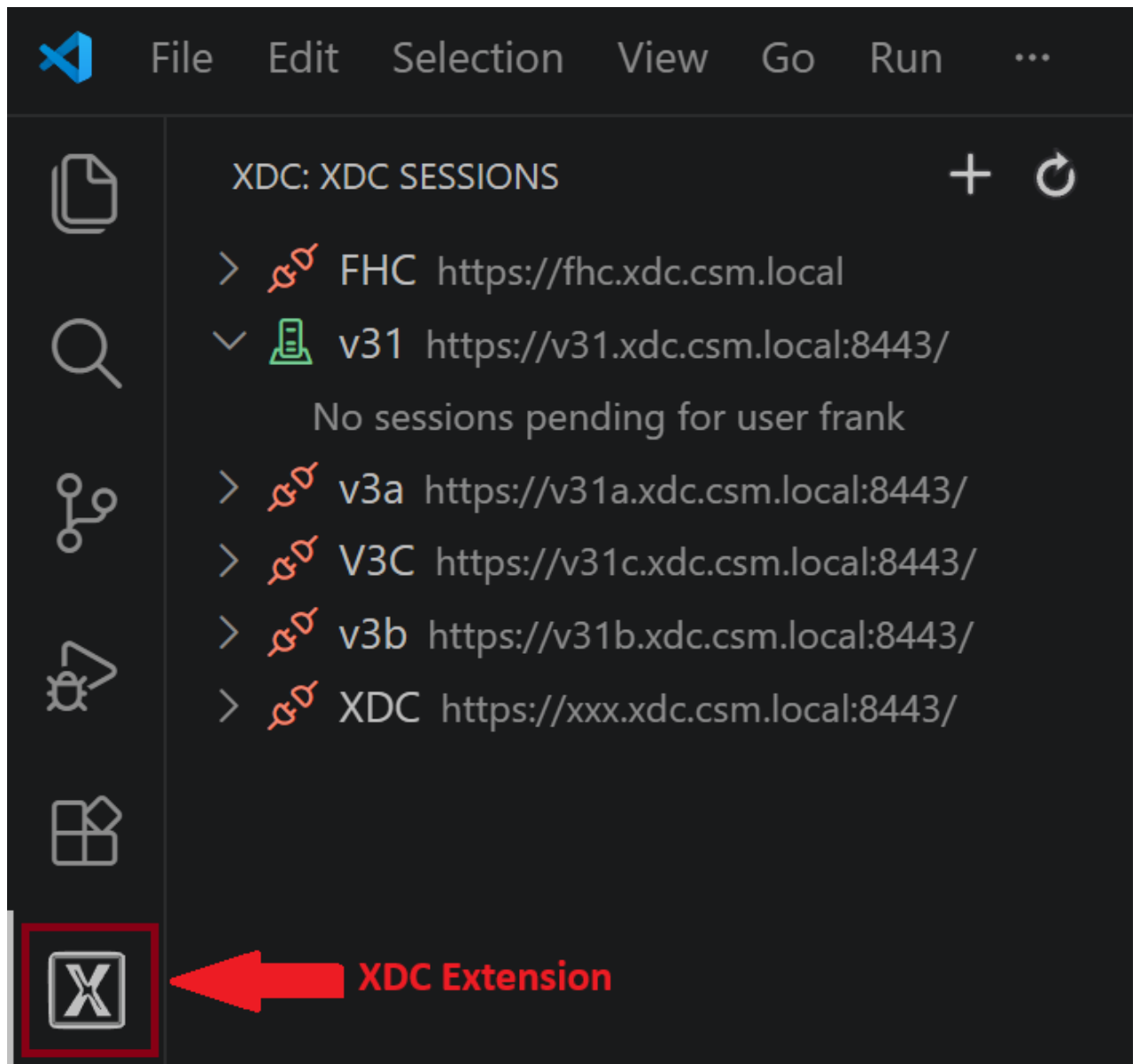


Figure 2: XDC Extension Installed

XDC VS Code Extension: Adding a Server

In order to use the XDC VS Code Extension, a profile needs to be created for the XDC GUI Server running on the z/OS system you will be connecting to. You can create multiple profiles to connect to multiple systems.

- Open the XDC VS Code Extension by clicking on the XDC icon in the side panel (see figure 2)
- Click on the + (*plus*) symbol in the top right of the panel to bring up the **Add XDC Server** panel.

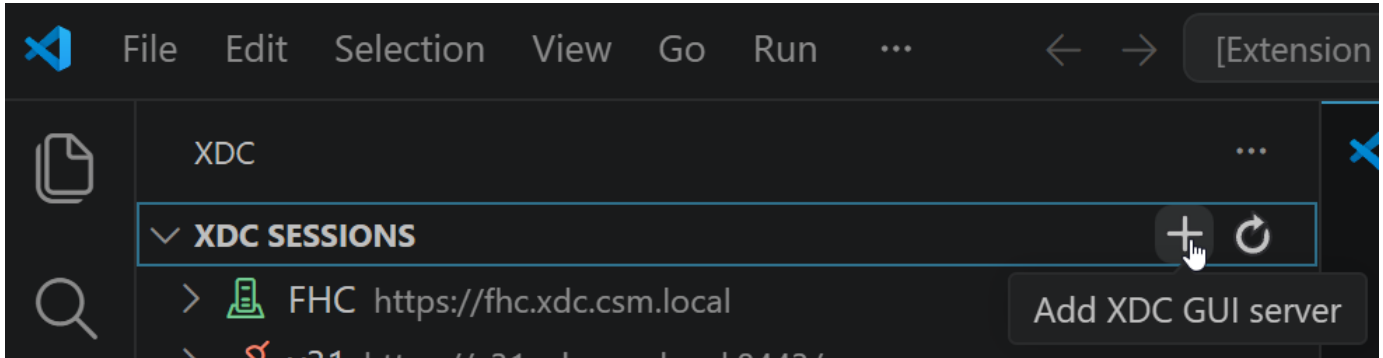


Figure 3: Add XDC GUI Server

- Fill out the panel with:
 - A descriptive profile name.
 - Your TSO user id and password.
 - Security group information, if required.
 - The HTTPS URL to the XDC GUI Server. Check with your XDC sysprog for this information.
 - Click **Connect**

ADD XDC SERVER

Add XDC Server

Configure a new connection profile to connect to an XDC server.

Profile Name *

e.g., XDC Server

Username *

Your mainframe username

Password / Passphrase *

Enter your password

Security Group

Optional security group

XDC Server Address *

https://xdc.example.com:8080

Close

Save & connect

Figure 4: Add XDC GUI Server Panel

- Once connected, the XDC VS Code Extension will be populated with the newly created XDC GUI Server profile. If there are any pending CDF sessions, they will be displayed when the XDC GUI Server profile is expanded.

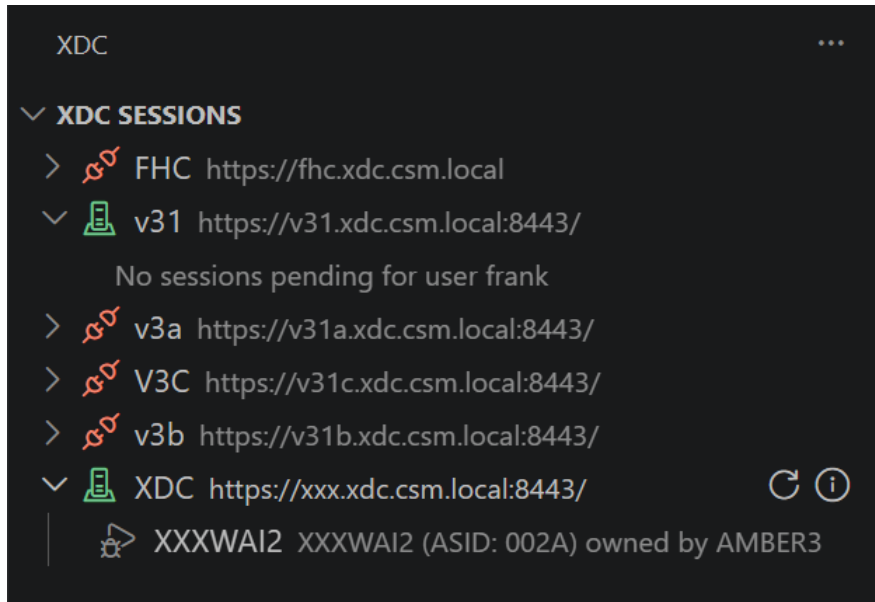


Figure 5: XDC GUI Server added

Connecting to an XDC Debugging Session

To connect to an XDC debugging session, you must first start a CDF debugging session on z/OS. A CDF session is ready to be connected to when you see a series of **DBC640Q** messages in the *SYSLOG*.

```

ICHT00011 FRANK2 - LAST ACCESS AT 14.03.43 ON FRIDAY, MARCH 21, 2020
$HASP373 XDCCAAA - STARTED - INIT 01 - CLASS A - SYS S0W1
IEF403I XDCCAAA - STARTED - TIME=16.07.06
+ DBC640Q XDC z2.2: (XDCCAAA) AWAITING USER SIGNON. 741
  DBC640Q - Reply C to cancel the wait and percolate the abend.
  DBC640Q - Reply GO to cause the user's program to reattempt
  DBC640Q execution.
  DBC640Q - Reply WTOR to issue z/XDC commands via WTO/WTOR.
  DBC640Q - Reply anything else to redisplay these messages.
@11 DBC640Q XDC z2.2: (XDCCAAA) Reply C, GO, WTOR, or RESHOW.
@11 DBC640Q XDC z2.2: (XDCCAAA) Reply C, GO, WTOR, or RESHOW.

```

Figure 6: A waiting XDC CDF session

With a pending XDC CDF debugging session, expand the XDC GUI Server profile to display a list of pending debugging sessions. What you see in the list will depend on your SAF permissions, so if you don't see anything listed, you will need to get in touch with your sysprog.

Right click on the XDC CDF debugging session, you want to connect to and select "Connect to debugging session."

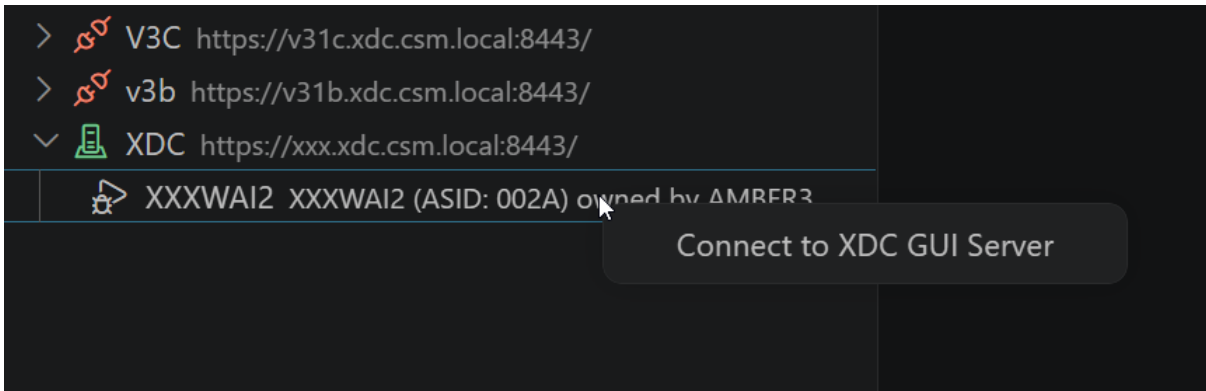


Figure 7: Connecting to an XDC debugging session

Once connected to an XDC CDF debugging session, the VS Code debug view will pop out and be populated.

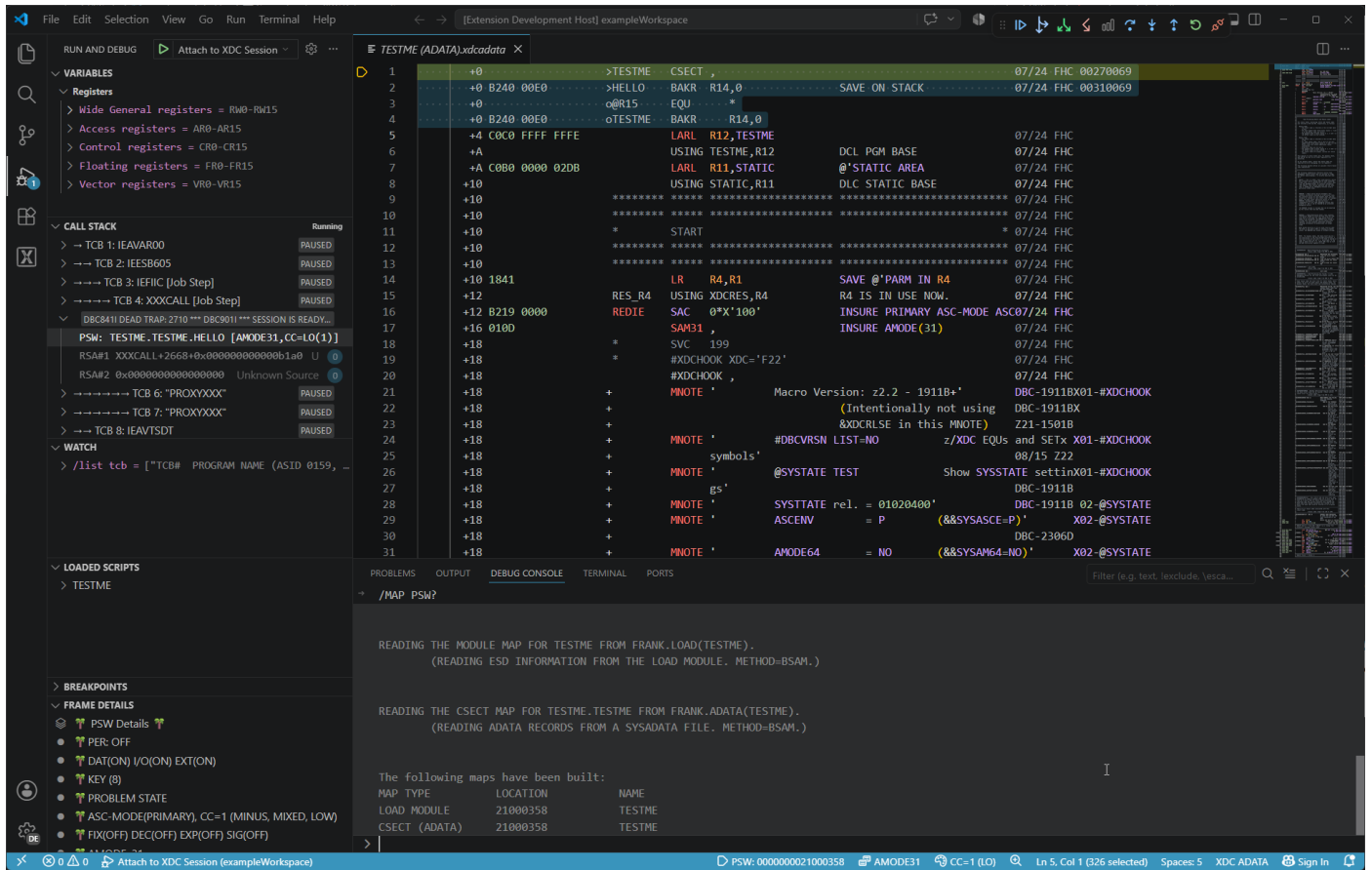


Figure 8: XDC VS Code Extension

Extension Features

- Registers are displayed in the Variables panel

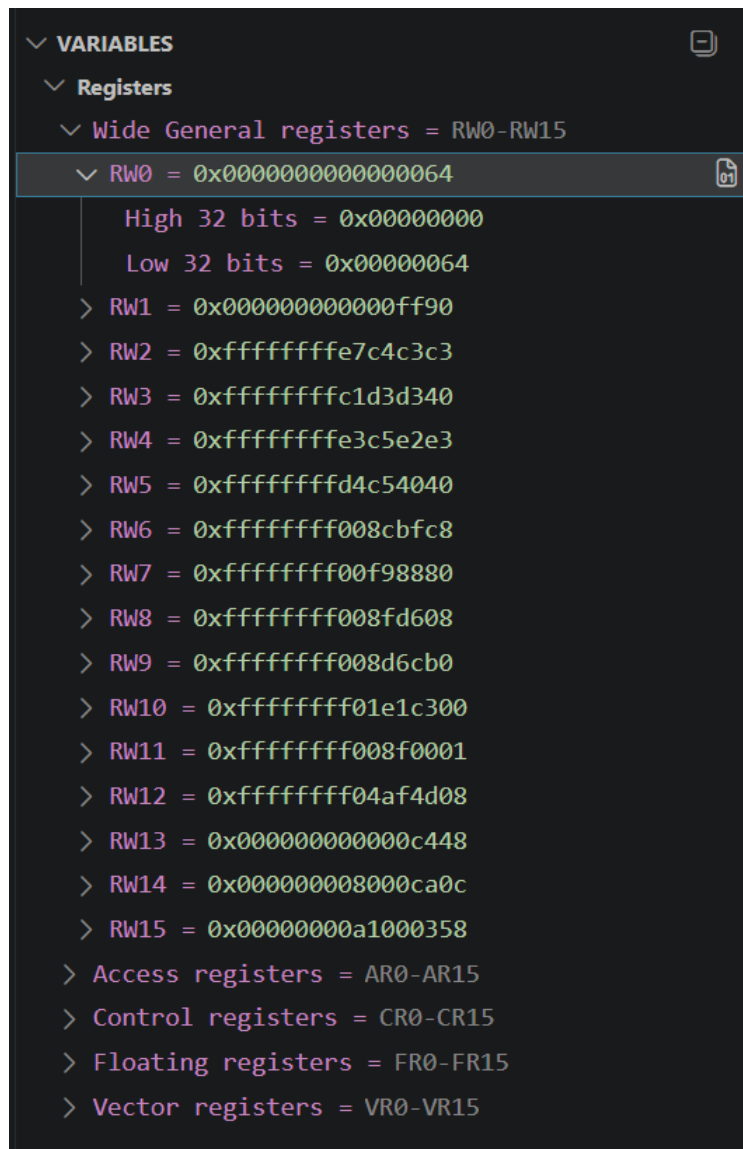


Figure 9: Register Sets

- All XDC commands can be issued via the Debug Console. Prefix any XDC command with a forward slash ("/").

```

XDC Debug Adapter (Desktop Mode)
=====
Log File: c:\Users\Frank\AppData\Roaming\Code\logs\20260326T231944\window9\exthost\colesoft-marketing.code-xdc\xdc-debug-2026-03-27T20-20-31-889Z.log
Timestamp: 2026-03-27T20:20:31.899Z

To send logs for troubleshooting:
1. Copy the log file path above
2. Open the file in a text editor
3. Send the file contents
=====
> /list tcb

```

Figure 10: Debug Console

Responses will be displayed in the Debug Console in unformatted, plain text (This is basically the same text displayed in a 3270 terminal.)

```

=====
> /list tcb
["TCB# PROGRAM NAME (ASID 009C, FRANK8)","1 IEAVAR00","2 . IEESB605","3 . . IKJEFT01, JOBSTEP","4 . . . . ISPF","7 . . . . . ISPTASK (ISPLLIB)","8 . . . . . XDCCALL","9 . . . . . "TESTME",
. . . . . \ "PROXYXDC\","11 . . . . . \ "PROXYXDC\","12 . IEAVTSDT"]
[0] = TCB# PROGRAM NAME (ASID 009C, FRANK8)
[1] = 1 IEAVAR00
[2] = 2 . IEESB605
[3] = 3 . . IKJEFT01, JOBSTEP
[4] = 4 . . . IKJEFT02
[5] = 5 . . . . IKJEFT09
[6] = 6 . . . . . ISPF
[7] = 7 . . . . . ISPTASK (ISPLLIB)
[8] = 8 . . . . . XDCCALL
[9] = 9 . . . . . "TESTME", CURRENT (XDCT0006) (ABEND: s0C1)
[10] = 10 . . . . . "PROXYXDC"
[11] = 11 . . . . . "PROXYXDC"
[12] = 12 . IEAVTSDT

```

Figure 11: Debug Console Response

- XDC commands can also be used in the Watch panel:

```

✓ WATCH
✓ /list tcb = ["TCB#  PROGRAM NAME (ASID 009C, FRANK8)", "1 ... ✕
  [0] = TCB#  PROGRAM NAME (ASID 009C, FRANK8)
  [1] = 1      IEAVAR00
  [2] = 2      . IEESB605
  [3] = 3      . . IKJEFT01, JOBSTEP
  [4] = 4      . . . IKJEFT02
  [5] = 5      . . . . IKJEFT09
  [6] = 6      . . . . . ISPF
  [7] = 7      . . . . . ISPTASK (ISPLLIB)
  [8] = 8      . . . . . XDCCALL
  [9] = 9      . . . . . "TESTME", CURRENT (XDCT0006) (AB...
  [10] = 10    . . . . . "PROXYXDC"
  [11] = 11    . . . . . "PROXYXDC"
  [12] = 12    . IEAVTSDT

```

Figure 12: Watch Panel

- The Call Stack panel displays the tasks in the current address space, along with the PSW and the RSA chain of the active task.

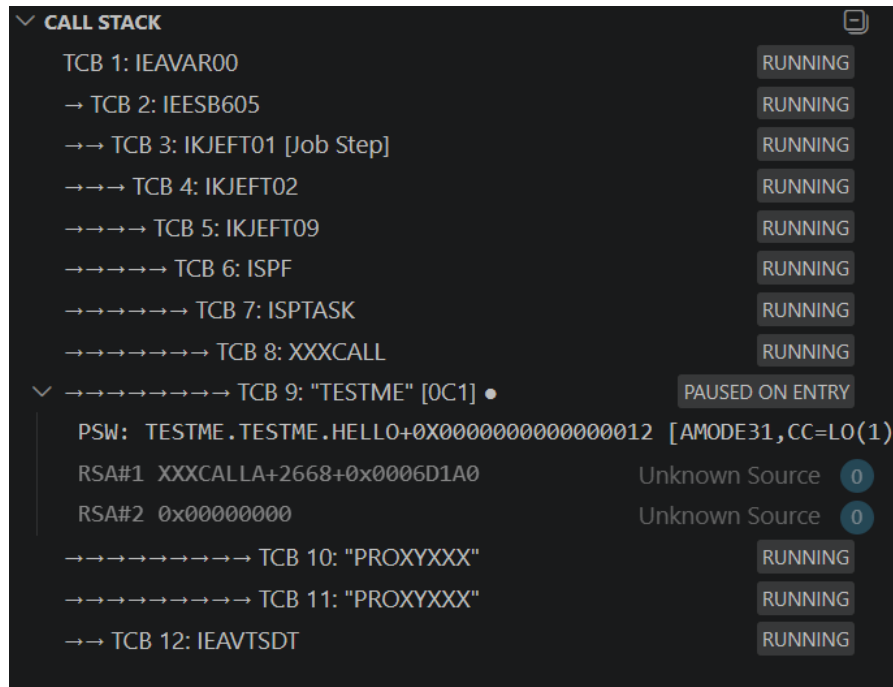


Figure 13: Call Stack Panel

Clicking on the PSW or any of the RSA entries will populate the Frame Details Panel with the relevant information.



Figure 14: Frame Details Panel

- The Loaded Scripts panel displays a formatted version of the LIST LKED command.

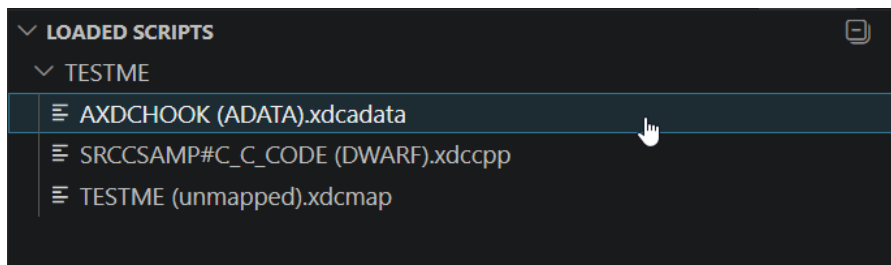


Figure 15: Loaded Scripts Panel

Clicking on any of the entries will display the ADATA source, SYMDATA or disassembly of the CSECT.

- The Command Palette contains several extension functions that are prefixed with "XDC".

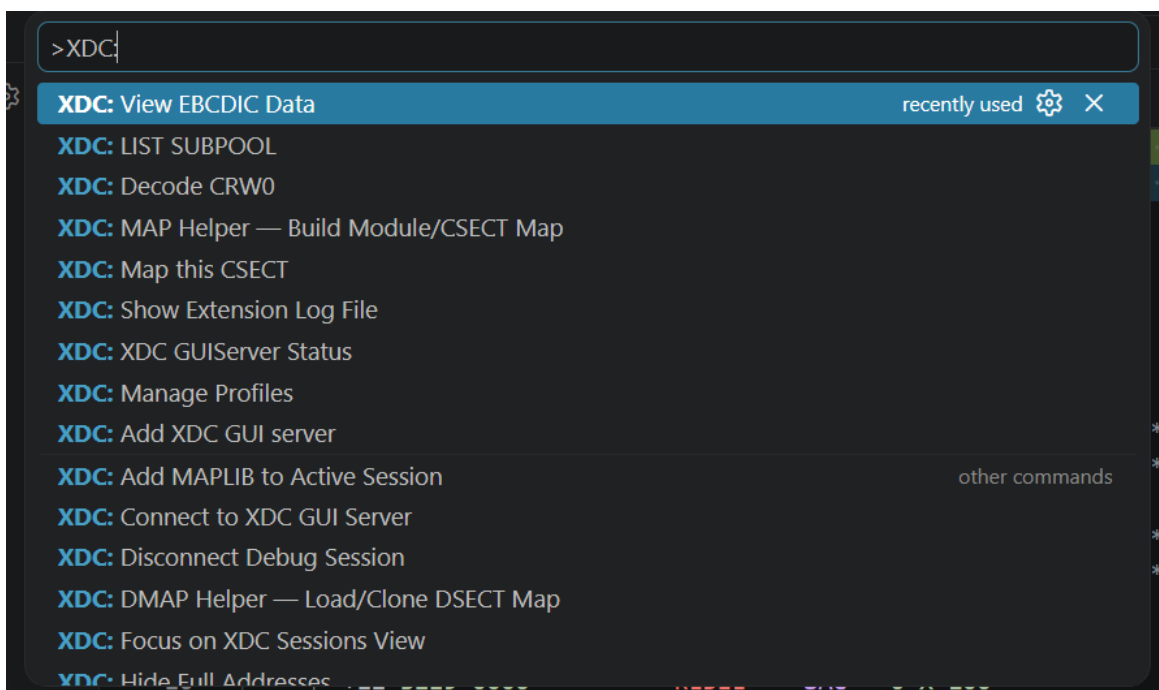


Figure 16: Command Palette

