SEE/Change

www.thenon.com

Many of the world leading companies use Thenon's products to change manage and test their software.

Thenon – designers of SEE/Change, the leading iSeries change management product.

SEE/Change

PE Notes 4.5503

(Version 4.5503)

1 Contents

2 Primary PE Enhancements

2.1 Test systems

SEE/Change now supports the concept of a test system. This allows you to deploy developments from CRs at development status directly into isolated test environments on the development machine.

A test system is an adjunct to a development system. It is a library environment into which CRs can be deployed before formal promotion through the SEE/Change environments begins, A development system can have several associated test systems. For example, if you have separate CRs containing alternative solutions to an application problem, and wish to evaluate them in parallel, you might deploy each to a test system for that purpose, without infringing any concurrent development restriction that might otherwise exist.

Test system deployments do not support reversion or archiving. You use test systems on a 'forward fix' basis, whereby errors are corrected either by redeployment or the deployment of a secondary CR. Existing test system objects are overwritten by incoming objects having the same names.

Developments in several CRs can utilise the same test system concurrently, but CRs should be deployed sequentially, and you should be aware that objects under concurrent development may be superceded during deployment. Test system objects can be cleared at any stage on a per-CR basis. If this is done for all objects in a test system CRs, it will revert to the objects contained in the libraries when it was configured.

2.1.1 Defining a test system

A test system is defined by taking the new option 17=Crt tst sys in Work with Systems Configuration against the parent development system, as shown below. A pop-up window allows the entry of a unique 3-character code for the test system, along with descriptive text.

At any stage, the description can be changed by taking option 2=Change against the test system, which will appear immediately beneath the parent development system in this display. For example, you may create a test system for 'Invoice layout changes' and when that project is finished reuse tor another e.g. 'Picking snake revision'.

```
THENON              THENON Change Management System 4.5503D
                        Work with Systems Configuration

 2=Change        3=Copy          4=Delete        5=Display      13=Sites
14=Passthru     17=Crt Tst Sys  20=Local system

Opt Sys  Description                              OS/400
         <-- Locate                              Version
    D43  Dev Sys 4.5503                          V7R3M0         Local system
           Invoice print changes                 V7R3M0         Test System
           Acc Rcv invoice match fix             V7R3M0         Test System
           Claims upload API                     V7R3M0         Test System
    PRD  PRD LON                                 V7R2M0
    PR3  PRD3 AUS                                V7R2M0
    PR4  PRD4 Aus                                V7R3M0
    P43  Prd Sys 4.5503                          V7R3M0

                                                                Bottom

F3=Exit  F4=Prompt  F5=Refresh  F6=Create  F9=Cmd  F12=Cancel
F21=Include Thenon sys  F24=Messages
```

Options 2=Change, 4=Delete and 5=Display can be used against a test system.

2.1.2 Configuring a test system

When a test system has been defined, you can use Work with Application Configuration to specify its usage by each application in the scope of its parent development system.

Test systems appear beneath the parent development system in the Work with Application Configuration – Where Used as shown below. Here a test system can be enabled for the application, and you can proceed to define its libraries.

```
THENON              SEE/Change Development Environment.
                Work with Application Configuration - Where used

Application  . . . . : AP1  Demo Appl                    Local Sys. :  DEV

 3=Add site
                                                    Applic   Site      Grou
Opt System      Type  Site                          Used ?   Specific? Code

    DEV System  Dev   Dev System                     Y         N
                      Dev Sys 2                     N         N
                      Muswell Hill site.            N         N
                      Prd Site                      N         N
                      Twickenham site.              N         N
                      Twickenham site.              N         N
    Acc Rcv in  TST   Invoice print changes          Y         N
    Claims upl  TST   Claims upload API             Y         N
    Invoice pr  TST   Invoice print changes         N         N

    PRD LON     Prod  PRD LON                       Y         Y       TWK
                                                                   More...

F1=Help  F3=Exit  F5=Refresh  F9=Cmd  F12=Cancel
F16=Update  F24=Messages
```

In the Work with Application Configuration – Libraries display (below), enabled test systems are again shown beneath the parent development system. A single set of libraries is all that is required; you enter their names in the Live/Prod column.

```
THENON              THENON Change Management System 4.5503D
                Work with Application Configuration - Libraries

Application  . . . . : V43  V43 Acme Order Entry System      Local Sys. :  D43

12=Obj typ overrides  13=CR type overrides  15=Environment settings
16=Additional libraries
                                                                        Ovr/
Opt System     Type Type:Description      -------- Target Library -------- Addn
                                          Live/Prod  Accept/QA  Integr/Tst
    Dev Sys 4. Dev  Obj: Base application V43DBASOL  V43DBASOA  V43DBASOM
                    Obj: Site Gatwick     V43DGATOL  V43DGATOA  V43DGATOM
                    Obj: Site Glasgow     V43DGLAOL  V43DGLAOA  V43DGLAOM
                    Obj: Site Liverpool   V43DLIVOL  V43DLIVOA  V43DLIVOM
                    Obj: Site London      V43DLONOL  V43DLONOA  V43DLONOM
                    Obj: Group NTH        V43DNTHOL  V43DNTHOA  V43DNTHOM
                    Obj: Group STH        V43DSTHOL  V43DSTHOA  V43DSTHOM
                    DB : Base Dev Site    V43DBASDL  V43DBASDA  V43DBASDM
    Acc Rcv in TST  Obj: Base application V43OBAST02
                    Obj: Site Gatwick     V43OBASGT2
                                                                  More...


F1=Help  F3=Exit  F5=Refresh  F9=Cmd  F12=Cancel  F16=Update  F24=Messages
```

Note that object libraries are shown before the database library, which is truncated in the display above. You would need to page down on the display to see the data library for Acc Rcv invoice test system, in this example.

Note also that it is your responsibility to populate the test system libraries. In this example, the library names suggest are copied (or derived from) a production environment.

2.1.3 Deploying to a test system

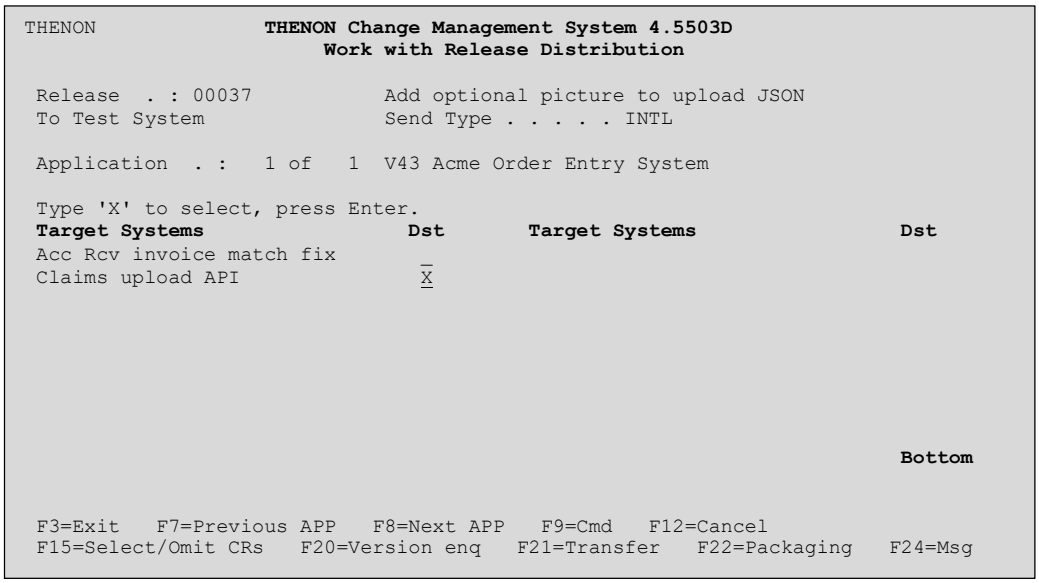
You can deploy a CR at status *DEV to one or more configured test systems Work with Change Requests, option 11=Promote. New dialog has been introduced here giving the opportunity of deployment to a test system. If you decline, the dialog will proceed as for a conventional promotion into a SEE/Change environment.

If you accept (Y=Yes) you will receive the Work with Release Distribution display (below) showing the available test systems. Choose your desired test system and press F21 to make an internal delivery of the CR contents.

When deploying a CR in GitHub-enabled application, CHKCR processing is skipped. CR objects are deployed to the test system on an ’as is’ basis. This means a CR source may have changed since the last compile, but it is always the last compiled object that is deployed to the test system.

Any database changes made to a test system will be applied in the configured database library and persist even if the test system is cleared down (see section 2.1.4) . Corrections to DB changes must be effected by changing the CR definitions and redeploying.

*BEFORE or *AFTER programs in the CR are executed when the CR is received into the test system.

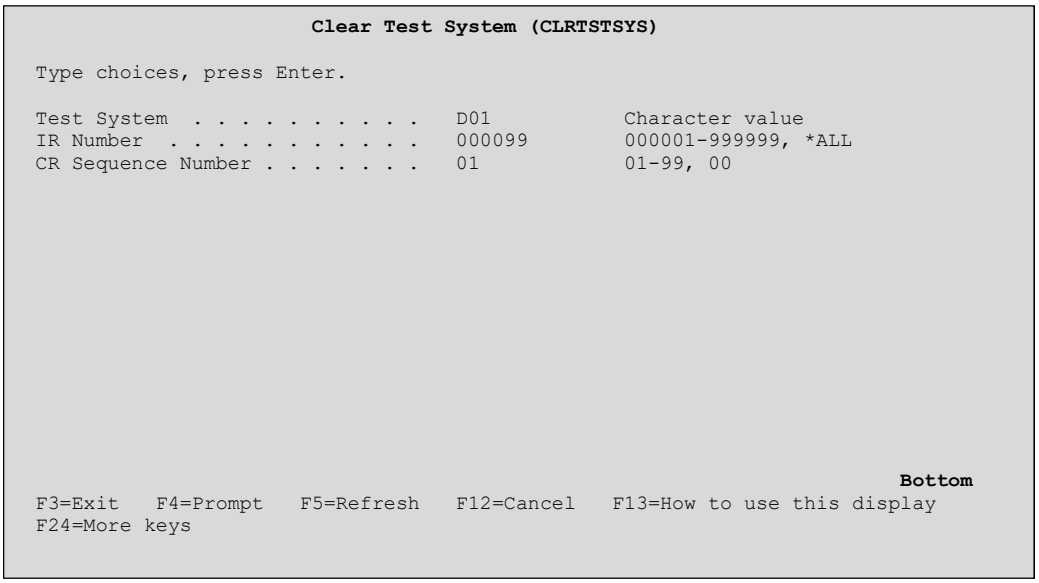


The Work with Release Distribution display shows the test systems as targets and the send type as 'INTL', indicating an internal transfer.

2.1.4 Clearing a test system

Objects placed into a test system remain there until removed.

You can use the new command CLRTSTSYS to clear a test system. It runs from the SEE/Change command line and can remove the contents of a previously deployed CR from a test system, Alternatively, you can remove all deployed objects from a test system.



This command is used on an ad-hoc basis.

2.2 DB2 Multisystem Support

IBM allows DB2 Multisystem allowing SQL partitioned tables across multiple systems. See https://www.ibm.com/docs/en/i/7.5?topic=multisystem-db2-overview

This PE now includes support for the DDL syntax that enables SQL tables to be spanned across multiple partitions. While this is not recommended by IBM, it has been added to this PE for compatibility reasons. Further information on DB2 multisystem, overview and partitioned tables can be found through the above link.

2.3 GitHub Interface

2.3.1 Introduction

GitHub is a cloud-based service for software development and version control, allowing developers to store and manage their code. It provides bug tracking, software feature requests, task management, and continuous integration for almost any project. The main benefit of GitHub for iSeries is that it enables new developers, who may not have encountered SEU and PDM, to work on iSeries projects using familiar tools.

GitHub is essentially a source repository with change branches, many editors, and third-party IDEs. However, it is not an appropriate CM solution for iSeries objects. For this reason, SEE/Change now contains a GitHub interface that exposes new source editing capabilities, while retaining its proven object management functions. In the following text, it is referred to as ‘the interface’.

2.3.2 Overview

To accommodate GitHub the interface extends workflow of the SEE/Change Development Manager (SCDM) as shown in Figure 1 below. In the Figure, the shaded area depicts the new functionality, and in the text references to the numbered workflow elements are parenthesized.

- Each SEE/Change user of the interface must possess a valid GitHub account (preferably at the Team level or higher). The interface maintains a register of known GitHub account holders.
- The interface creates a GitHub repository (repo) for the source of a single application (1) and links it to the application. The repo name can be freely chosen, and such applications are referred to as ’GitHub-enabled’. The interface establishes ownership of the repo and all account holders as collaborators. Existing source can be loaded with an interface command.
- A CR is represented by a branch in the application repo. The interface can create the branch based on the last commit of the initial load (2), with a name derived from the CR description. The repo main branch is updated from the CR branch immediately before the CR attains *LIV status. This scheme resembles GitHub Flow.

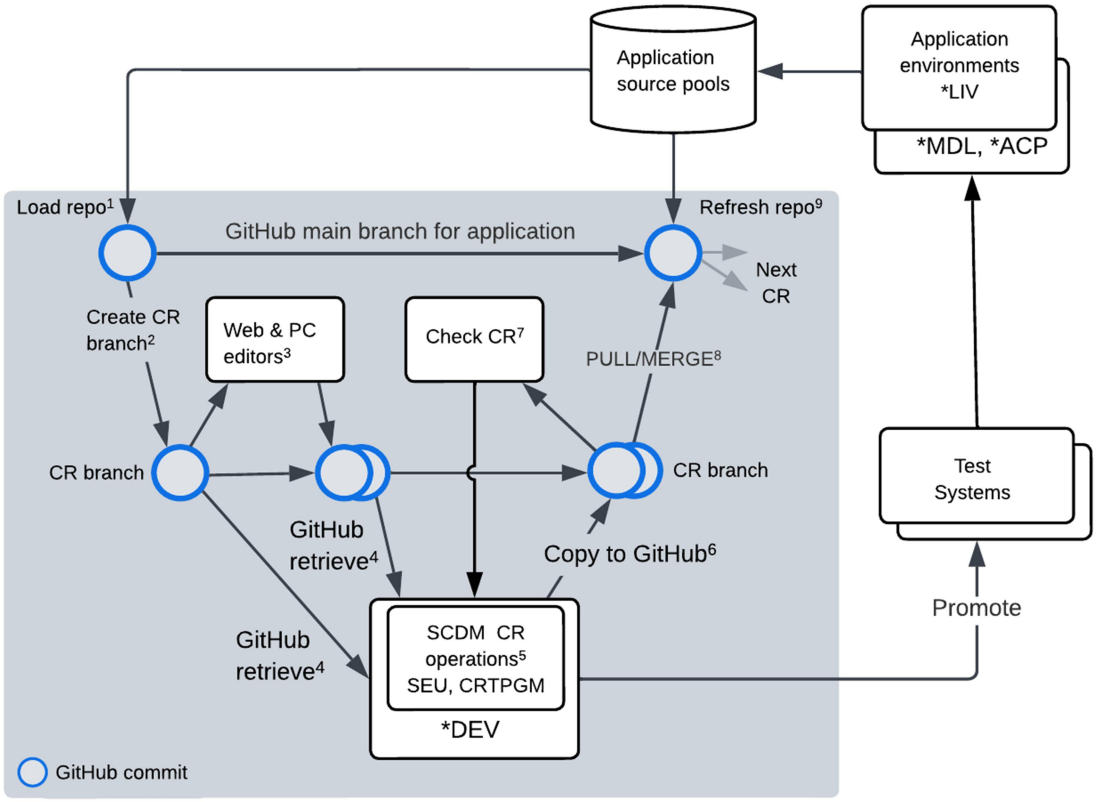


Fig 1: The GitHub Interface - workflow for a single CR

(for refs 1-9, please see text)

- You can edit CR branch files using a GitHub-compatible editor (e.g. github.dev) or a local Git client established with a remote tracking branch (3). In either case, multiple commits will arise in the CR branch. When convenient, you can retrieve edited source from the branch into the CR library (4) for compilation and further editing with SCDM (5). If you edit members in the CR, you must copy them back to the branch using another interface function (6). Note that in a GitHub-enabled application, conventional source pool retrieval is disabled.
- When you set the CR status to *TST, the interface ensures the branch and library sources are synchronized. You can remove synchronization errors by repeating the steps (3,4,6) , as required. You can then deploy the CR to a test system or promote it through the usual SEE/Change environments.
- Before the CR reaches the *LIV development environment, you should issue a GitHub PULL (8) request on the CR branch. As a collaborator in the repo, you can then MERGE the branch into main, completing the GitHub Flow cycle mentioned above. You then promote the CR to *LIV, synchronizing the application's source pools with its repo, and delete the branch. The interface warns if you attempt the promotion before updating the repo's main branch.

- A difficulty could arise if you want to revert the CR, because the equivalent GitHub operations can be problematic. If reversion is anticipated, it is better to delete the CR branch, promote the CR to *LIV and refresh the repo from the source pools (9). Alternatively, you can avoid reversion by adopting a 'forward fix' strategy, i.e. create a new CR branch correcting any residual errors that you have promoted to *LIV.

2.3.3 Using SEE/Change with GitHub

When version 4.5503 is installed, a new general parameter @GIT holds the base GitHub URL. This should not need changing.

Your iSeries must have internet access. You can verify this by issuing the command:

PING RMTSYS(API.GITHUB.COM)

You will need to inspect the job log to view the IP replies. Further diagnostics can be obtained by issuing thise command:

TRACEROUTE RMTSYS(API.GITHUB.COM)

Each interface user must possess a GitHub account, a current personal access token (classic PAT). You will require a two-factor authentication mechanism to access the GitHub web client.

2.3.4 Granting SEE/Change users access to GitHub

To use the interface, an enrolled SEE/Change user must also be registered as a GitHub user. To register a GitHub user, take the Configuration Manager's Work with User Enrolment menu option and use F18 to obtain the new Work with GitHub Users display:

```
                      SEE/Change Development Environment.
                           Work with GitHub Users
  2=Change  4=Delete  5=Display

Opt User Profile  User description
    __________ <--- Locate user
    JAMESA        James Anderson
_   GILLB         Gillian Brown
_   NATHANW       Nathan Wilkinson

                                                                  Bottom
F1=Help  F3=Exit  F5=Refresh  F6=Create  F9=Cmd  F12=Cancel
```

Press F6 to obtain the Create GitHub User window below:

```
 .....................................................
:                Create GitHub User                   :
: iSeries User USRPRF                                 :
:                                                     :
: GitHub User. GitHub-User-ID                         :
: Token. . . . GitHub_token_this_will_be_40_Characters. :
:                                                     :
: Enter=Create  F12=Previous                          :
:.....................................................:
```

You should enter a valid SEE/Change user profile, the corresponding GitHub user account name and PAT (Personal Access Token).

2.3.5 Configuring an application to use a GitHub repo

Each GitHub-enabled application has its own repo, set up via the interface. You will need to provide the repo name and the identity of the GitHub user / organisation that will own the repo. It is the responsibility of the repo owner to implement any GitHub branch protection rules mandated by your organization.

In Work with Application Configuration, a new option 60 (60=GitHub link) signals that the application is GitHub-enabled:

```
.....................................................
:                GitHub Application Link              :
:                                                     :
: Application . .  V43 Acme Order Entry System        :
: Repository Name  Acme-Order-Entry-System            :
: Repository Owner Thenon-Holdings-Ltd                :
:                                                     :
:                                                     :
: Enter=Continue  F12=Previous  F23=Unlink repository :
:.....................................................:
```

The Repository Name is the GitHub repo that this application will use. It is in free format and up to 40 characters in length. Allowable characters are letters (a-z and A-Z), numbers (0-9), hyphen (-) and period (.). GitHub will replace other characters (including space) with a hyphen.

The Repository Owner must be a GitHub user as described at 2.3.4 above or an organisation. This field is required as the repo is uniquely specified by name and owner/organisation.

Sources are placed in subdirectories of the source file name within GitHub If a source file is empty no corresponding subdirectory is created in the repository. If an empty source file will be targeted by a subsequent CR movement to *LIV, you should manually create a subdirectory in the repository .main branch.

The repository name and owner are checked on pressing Enter. Unsuccessful completion is indicated by an HTTP response code (typically 404) on the display base line.

Note that in a GitHub-enabled application:

- Source retrieval is done from a CR branch in the repo, not from the local SEE/Change source pools. However, the new workflow maintains the source pools and these can be regarded as an internal backup of the application repo.

- The automatic documentation parameter @DOC is ignored because compiling a retrieved source would update it with documentation information. This would cause the CR source to lose synchronization with the branch source.
- Any CR you create will allow planned concurrent development and retrieval of site and group-specific source.

2.3.6 Initial source load to GitHub

With a repo created and your user registration complete, you can load application sources into GitHub. This begins the workflow of Figure 1 above and the interface provides the single command GITLOAD (1) for this purpose.

You can invoke GITLOAD from the application take on jobs panel or command line e.g.

GITLOAD APPL(ACM) GROUP(*ALL) SITE(*ALL)

loads the base, group, and site versions of all source files in GitHub-enabled application ACM into a linked repo. For example, assume the linked repo is named SEE-ACM and the application contains source pools BAS01, GRPGR1 and SITSI1. GITLOAD would then create a subdirectory structure like that shown in Figure 2 below:

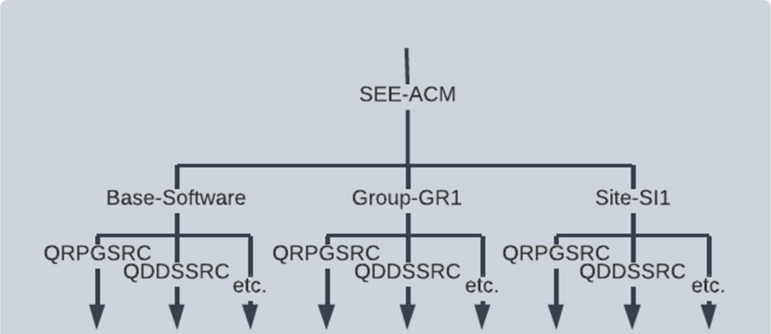


Fig 2: Repo directory from GITLOAD, showing source type and site/group subdirectory structure

As it executes, the command directs HTTP status messages to the display base line. It also generates the spooled print report O#9550, detailing the source files and members that were loaded. Please note that source sequence numbers and amendment dates are removed during loading.

2.3.7 Working with CRs and GitHub

As discussed above, a CR is represented by a branch in the application repo. The linkage is established through the branch name and the CR's descriptive text.

A branch is linked to a CR if the branch name contains the CR description. For example, you might use the GitHub web client to create a branch named 'ACM-00004602-File-changes' in the repo of application ACM. The interface would dynamically link that branch to the ACM CR having descriptive text 'File changes'.

Alternatively, you can create a branch for an existing CR with the new option 21 (21=Git crt brn) in Work with Change Requests.

You should create a separate CR for each individual site or group change. These CRs should have the branch-related descriptive text described above. This allows the interface to identify the branch subdirectory where the site and group variations reside,

The GitHub workflow allows you to freely edit application source on a CR branch with any compatible editor (3). Note that you should not edit source in the main branch. This can occur unintentionally, and you should consider establishing a branch protection rule to prevent it.

For compilation, you can load the linked CR with all CR branch changes using Work with Change Requests option 7=Git retrieve (4).

Alternatively, use option 1=Retrieve from within the CR. This also retrieves a changed source file from the CR branch. If there is no such file in the branch, the source is retrieved from the main repo branch. If the source file cannot be found there an HTTP 404 error will be presented. In this case, you should manually create the file in the CR branch and retrieve it using Work with Change Requests options 1 or 7.

After editing in the CR with SCDM, be sure to use Work with Change Requests option 3=Copy to Git (6) to upload the CR source to the linked branch.

When you take Work with Change Requests option 11=Promote, CHKCR will verify the sources in the CR library and the CR branch are synchronized (7). It does this by comparing commit and member amendment timestamps. Therefore, CHKCR will fail if the branch is found to contain an edited file that has not been retrieved / changed since the last retrieve, or a CR source member that has been edited but not copied back to the branch. If CHKCR succeeds, you can deploy the CR to a test system or promote it to a CR environment in the normal way. The synchronization check ensures the interface workflow delivers the same level of object integrity as conventional SCDM operations.

Before promoting a CR to *LIV on the development machine, you should consider whether reversion will be required. If so, it may be preferable to close the branch and promote the CR to *LIV. After reversion you can continue working in SCDM. When the CR eventually attains *LIV status, you can synchronize the application repo using the GITLOAD command (9).

If you do not anticipate reversion, you can issue PULL and MERGE operations on the CR branch, using the GitHub web client (8). When this has been done, you can promote the CR to *LIV. A warning will be issued if the CR's branch is still open. This ends the enhanced workflow.

2.3.8 Notes

- Repo source files are named after the originating source pool library member, with an extension indicating member type.
- Directory and source file names are in uppercase. These conventions should be followed when creating new repo files, e.g. NEWPROG.RPGLE.
- At this time (Feb 2024) the SEE/Change RDI feature does not support the GitHub interface or test systems described above.
- In version 4.5503, the interface does not support IFS files.

3 Software Performance Reports

The following table lists software performance reports that have been resolved in this PE.

SPR Log Number	Description
5927	SQL Multisystem tables. Within the CR, the create completes correctly, as does testing of the CR. However, promotion of Multisystem / partition tables fails.
5928	Version 4.5502 close release failure corrected
5929	RCVRLS would not tidy the release library if SNDTCPSPLF failed to process (e.g. if the system is not configured to use SNDTCPSPLF or there is a network issue). RCVRLS now continues processing regardless of the SNDTCPSPLF result.

4 Installation

4.1 Warnings

To ensure a successful upgrade please use the SEE/Change upgrade guide. This comprehensive guide is available from the thenon.com website Download Area or from the support line.

Steps must be taken to preserve any customised code especially if upgrading by reinstalling the product libraries and running a conversion.

4.2 Dependencies

- Must be applied after 4.5502.
- SEE/Change 4.5503 will run on IBM i 7.4 or higher.
- Must be applied to all remote sites? **See SEE/Change compatibility chart.**

5 RDi Feature Compatibility Chart

SEE/Change Rational Developer Feature Compatibility

SEE/Change Server Version	IDE Version	
	RDP 8.x.x.x	RDP 9.x.x.x
4.5401	1.4.2	1.4.2
4.5402	1.4.2	1.4.2
4.5403	N/A	1.4.3
4.5500	N/A	1.4.3
4.5501	N/A	1.4.3
4.5502	N/A	1.4.3
4.5503	N/A	1.4.3

6 SEE/Change Compatibility Chart

SEE/Change Release Compatibility Chart

		Development										
		4.5200	4.5201	4.5300	4.5400	4.5401	4.5402	4.5403	4.5500	4.5501	4.5502	4.5503
Production	4.5200	Yes	1	1,2	1,2	1,2	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3
	4.5201	No	Yes	2	2	2	2,3	2,3	2,3	2,3	2,3	2,3
	4.5300	No	No	Yes	Yes	Yes	3	3	3	3	3	3
	4.5400	No	No	No	Yes	Yes	3	3	3	3	3	3
	4.5401	No	No	No	No	Yes	3	3	3	3	3	3
	4.5402	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
	4.5403	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes
	4.5500	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes
	4.5501	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes
	4.5502	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes
	4.5503	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes

Notes

Note 1: Yes, however the Alternative ALTER TABLE support will be ignored on production machine

Note 2: Yes, however changes to the default object delivery sequence will be ignored on production machin

Note 3: Yes, however SSH distribution is not supported.

Minimum i5/OS Levels

4.5200	V5R3
4.5201	V5R4
4.5300	V5R4
4.5400	V6R1
4.5401	V6R1
4.5402	V6R1
4.5403	V6R1
4.5500	V7R1
4.5501	V7R3
4.5502	V7R3
4.5503	V7R4

Disclaimer: Every effort has been made to ensure accuracy however we cannot take responsibility for any errors caused by using this information