



Sneak Peek PowerBuilder 2025

Chris Pollach, Appeon
November 20, 2023



DISCLAIMER

The following information is being shared in order to outline some of our current product plans. We are hopeful that the following can shed some light on our roadmap, but it is important to understand that it is being shared for **INFORMATIONAL PURPOSES ONLY**, and not as a binding commitment.

Please do not rely on this information in making purchasing decisions because ultimately, the development, release, and timing of any products, features or functionality remains at the sole discretion of Appeon, and is **SUBJECT TO CHANGE**.



Session Agenda

- New PowerBuilder Compiler!
 - Legacy Compiler vs.
 - New Compiler * - Only available in the *Professional* and *CloudPro* editions
 - New WorkSpace → Solution
 - Demo
- New PowerScript Editor
 - Better code readability, code efficiency, code navigation
 - Continual improvements planned across revisions

Tip: Product Roadmap → <https://www.appeon.com/developers/roadmap>



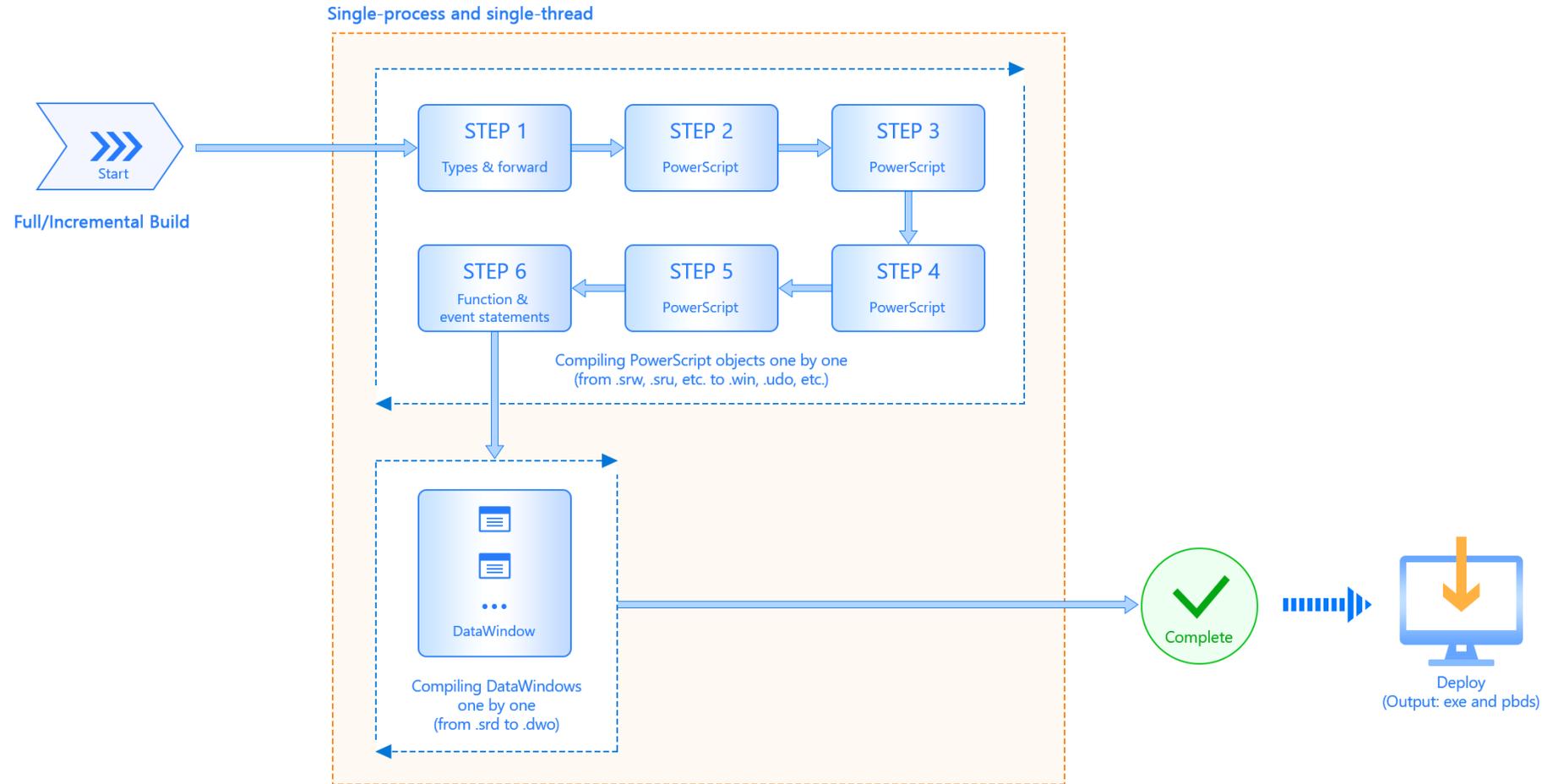
Legacy Compiler

Operational Characteristics

- The legacy compiler can only work in *single-process* and *single-threaded* mode
=> because both the source code and compiled code (P-code) are saved in the *same* PBL.
- The legacy compiler scans **every** piece of PowerScript source code for **5** times and it takes **7** steps to complete the entire compilation process!
- The legacy compiler compiles the DataWindows **one by one** after completing PowerScript compilation.
- The legacy compiler runs in the *same process* as the IDE! If anything goes **wrong** during the compilation, the running of the IDE could be affected (crashes, abnormal termination* of the IDE, stop the compilation process, etc.)
 - * When a sub-process aborts, the main task is killed by the OS!
 - * Also applies when running your App from the IDE!

Legacy Compiler – Process Flow

Legacy Compiler



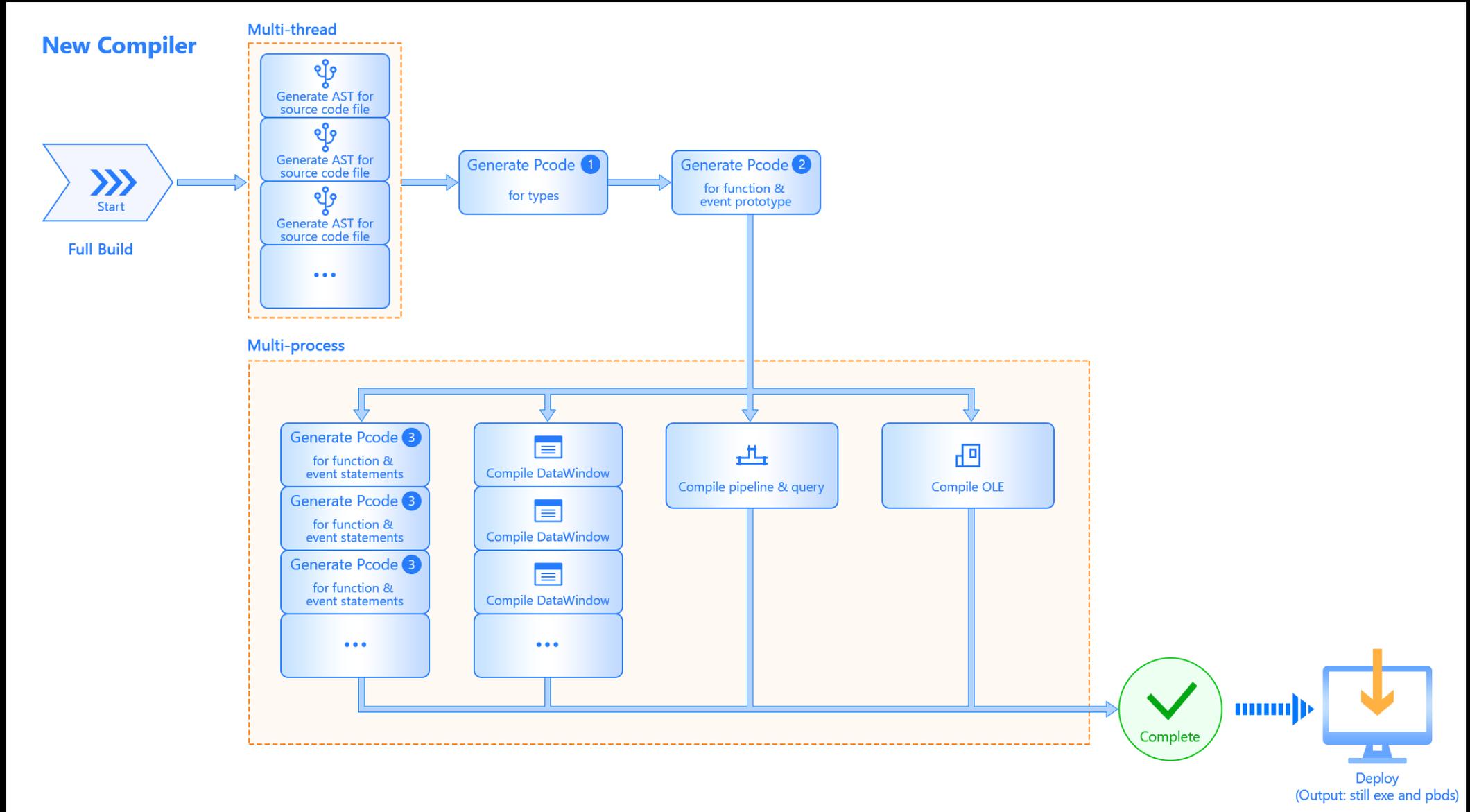
New Compiler

Operational Characteristics

- The new compiler can work in **multiple-process** and **multiple-thread** mode, because the source code and Pcode of every object are now stored in **separate** files.
- The new compiler scans every piece of source code only **once** (instead of 5 times) and takes **4** steps (instead of 7 steps) to complete the entire compilation process.
- The new compiler creates an AST (*abstract syntax tree*) for every source code file. The AST then **enables multiple-thread concurrent** compilation and thus, is *much faster*!
- The performance boost of the new compiler mainly comes from the concurrent **multiple-process** handling of the P-code generation and DataWindow compilation:
 - The P-code generation step now works in a multiple-processing mode which is much faster!
 - DataWindow compilation also now works in multiple-processing mode concurrently with the P-code generation.
- The compilation process runs in **parallel** with the IDE process. The two processes do **not** affect each other.

Note: No changes required for any Client/Server, PC, or PS App deployment!

New Compiler – “Full Build” Flow



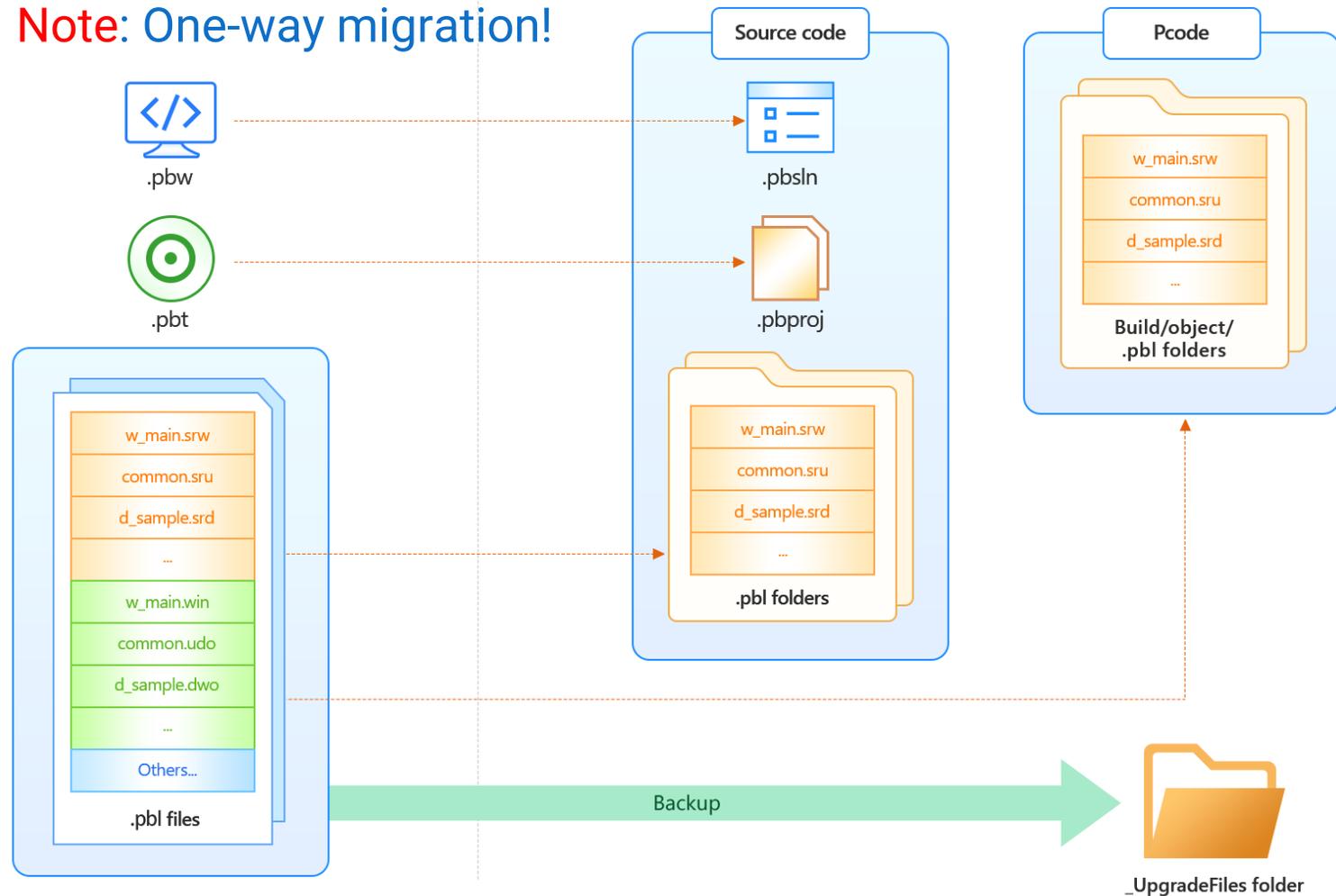
New Compiler – Convert to “Solution”!

New Compiler

Before Conversion

After Conversion

Note: One-way migration!



Workspace to Solution Migration Demo



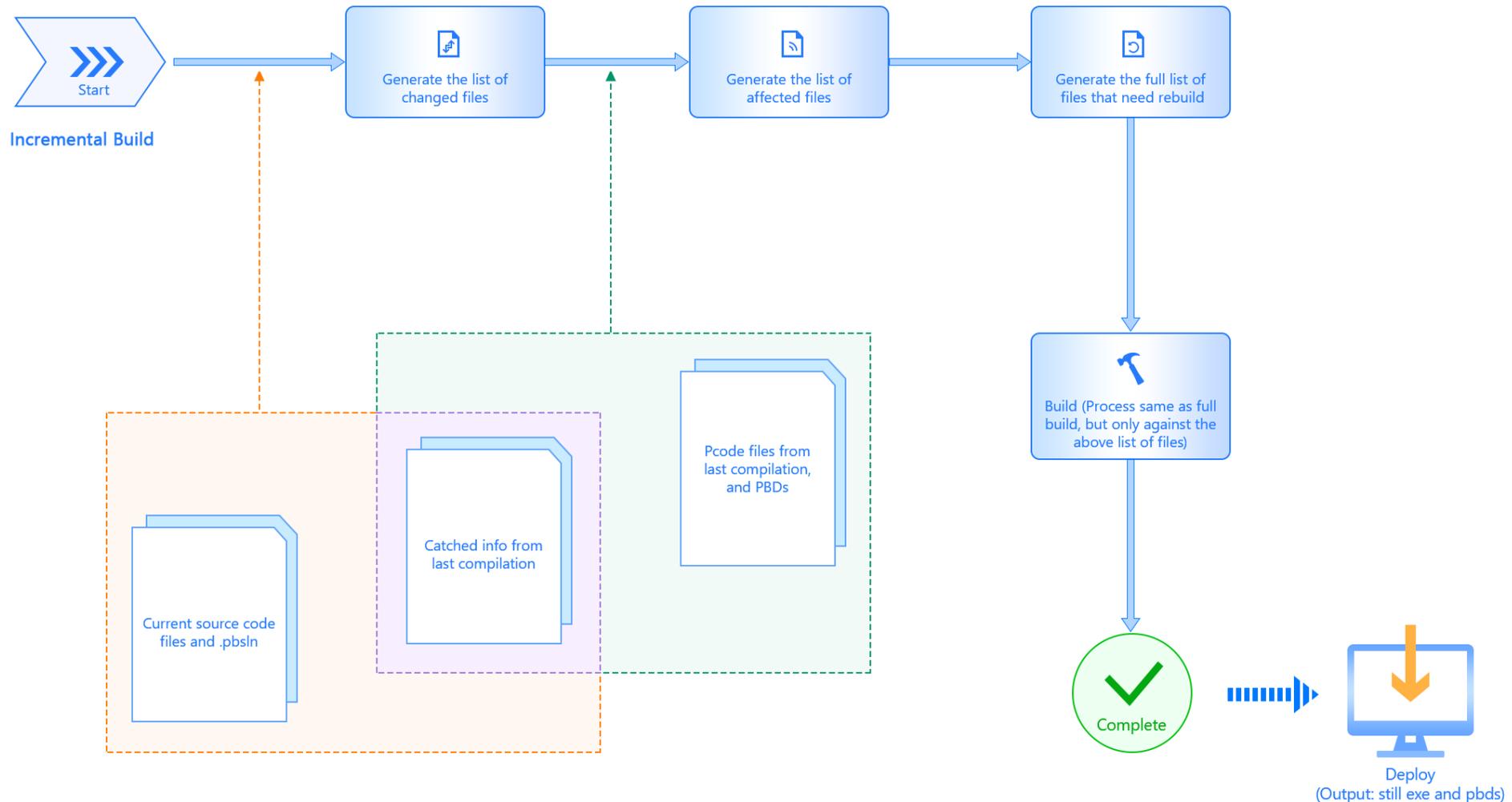
New Compiler

Incremental build

- Information **cached** from the last compilation:
 - The **forward dependencies** (inherited from or referring to) of objects
 - The information of the **global function(s)** that a DataWindow is referring to.
 - The **reverse dependencies** that are inferred from the forward dependencies.
 - The last **modified date** of all source code and P-Code files.
- The incremental build accurately identifies *which objects* **need** to be rebuilt:
 - Identifies the **modified/deleted** files based on the file list of the current projects compared against the file information cached from the last build;
 - Identifies the file list affected objects only based on the **cached** information;
- The incremental build then formulates the full list of files that need to be rebuilt and then performs the build process **only** against that list.

New Compiler – “Incremental” Flow

New Compiler



New Compiler Performance

Full Build	Legacy Compiler	New Compiler
Code Example App <ul style="list-style-type: none">• 388 PowerScript objects• 194 DataWindow objects	37s	10s
Customer App 1 <ul style="list-style-type: none">• 3,154 PowerScript objects• 5,348 DataWindow objects	18m 56s	2m 50s
Customer App 2 <ul style="list-style-type: none">• 7,495 PowerScript objects• 10,853 DataWindow objects	1hr 31m 21s	9m 33s
Incremental Build		
Code Example App	36s	6s
Customer App 1	18m	23s
Customer App 2	3m	5s

New Compiler

More Benefits

- Plus ...
 - Much less chance of PBL *corruption!*
 - Better *integration* with Git/SVN
 - *Eliminate* logging the entire PBL to SCC!
 - Allows *external tools* to be used on object source (even using VS Code)!
 - Still works with Legacy SCC (including Native)



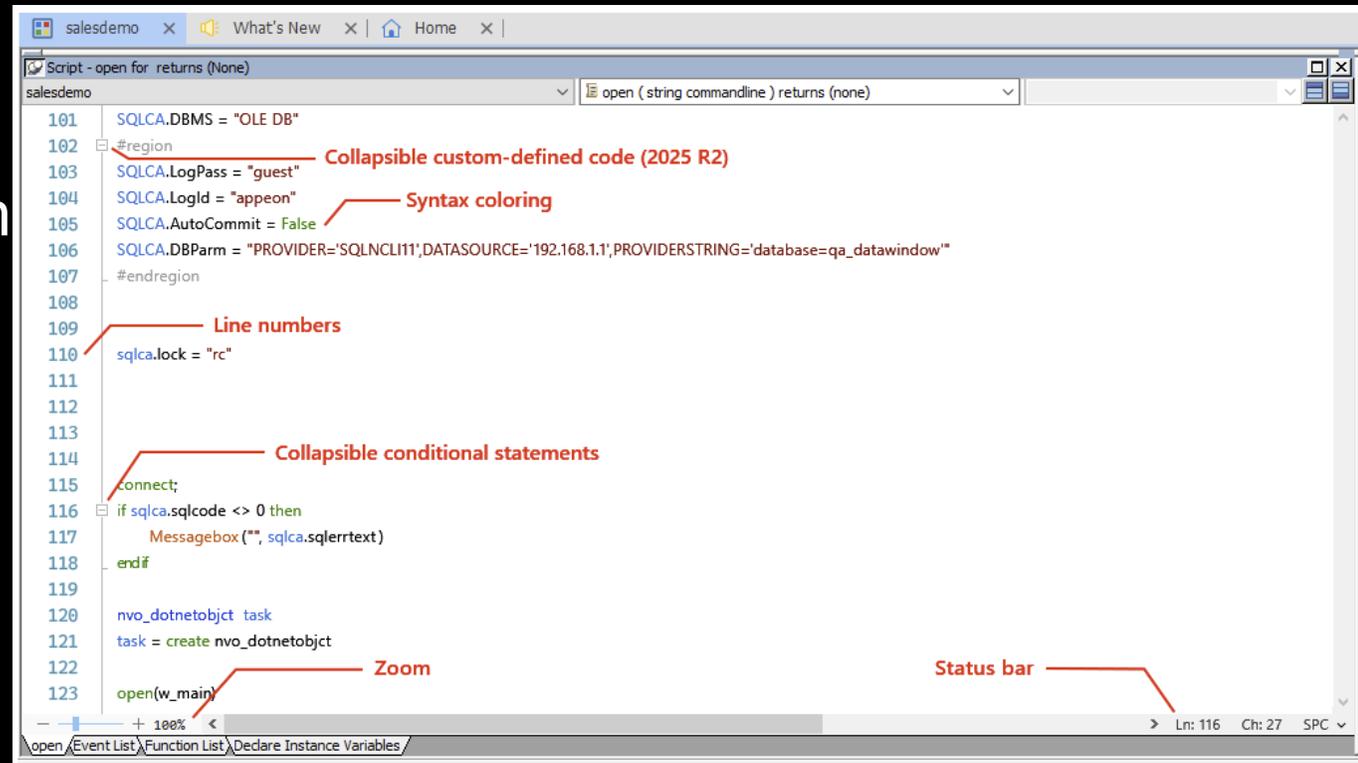
Incremental Build Demo



Modern Code Editor

Code Readability

- How to Achieve...
 - Syntax coloring
 - Line numbers/status bar/zoom
 - Collapse/expand code
 - ✓ Collapsible conditional statements
 - ✓ Collapsible custom-defined code blocks (R2)



The screenshot shows a code editor window titled 'Script - open for returns (None)'. The code is as follows:

```
101 SQLCA.DBMS = "OLE DB"
102 #region
103   SQLCA.LogPass = "guest"
104   SQLCA.LogId = "apeon"
105   SQLCA.AutoCommit = False
106   SQLCA.DBParm = "PROVIDER=SQLNCLI11;DATASOURCE='192.168.1.1';PROVIDERSTRING='database=qa_datawindow'"
107 #endregion
108
109
110 sqlca.lock = "rc"
111
112
113
114
115 .connect;
116 if sqlca.sqlcode <> 0 then
117     MessageBox("", sqlca.sqlerrtext)
118 endif
119
120 nvo_dotnetobjct task
121 task = create nvo_dotnetobjct
122
123 open(w_main)
```

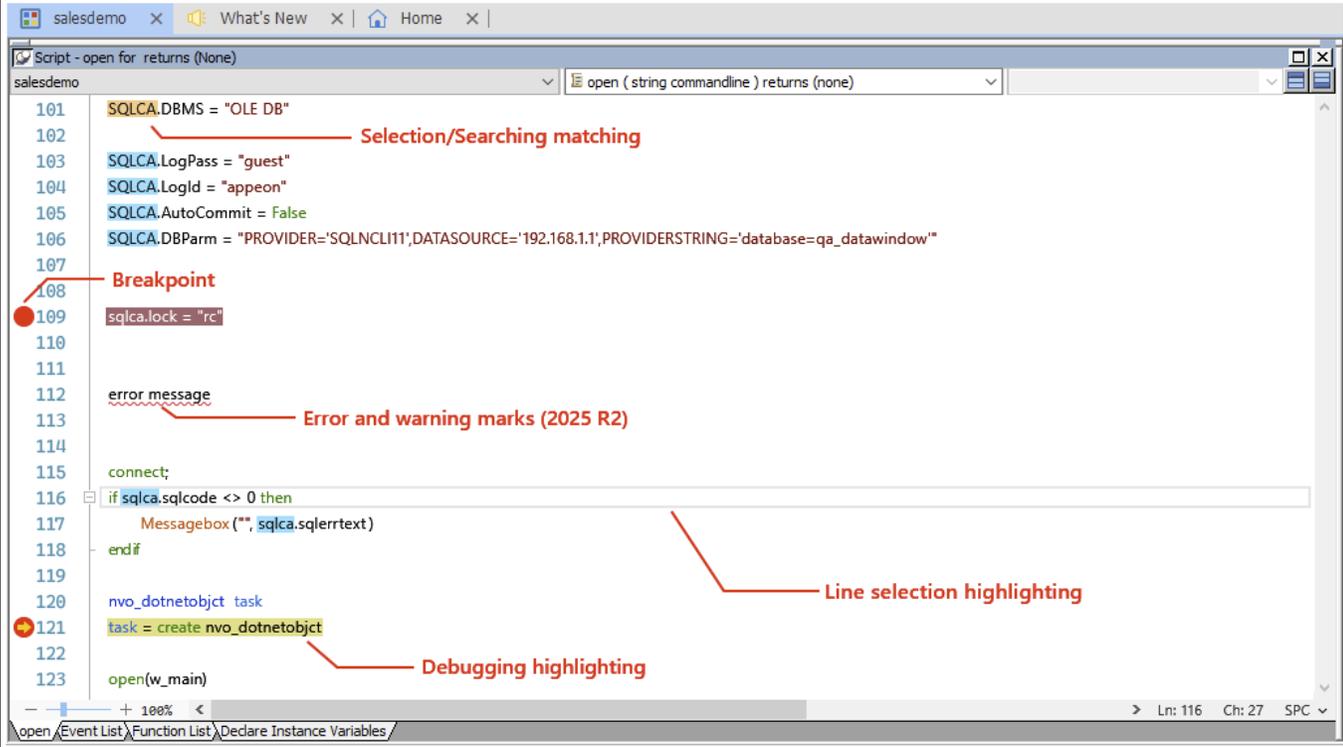
Red arrows point to various features in the editor:

- Collapsible custom-defined code (2025 R2)**: Points to the collapsed region block (lines 102-107).
- Syntax coloring**: Points to the colored text in the SQLCA.DBParm line (line 106).
- Line numbers**: Points to the line numbers on the left side of the editor.
- Collapsible conditional statements**: Points to the collapsed if statement (lines 116-118).
- Zoom**: Points to the zoom level indicator (100%) at the bottom left of the editor.
- Status bar**: Points to the status bar at the bottom right, showing 'Ln: 116 Ch: 27 SPC'.

Modern Code Editor

Code Readability (Continued)

- Improved *Highlighting*
 - Selection/search matching
 - Line selection highlighting
 - Debugging highlighting
 - Error & warning marks (R2)



The screenshot displays a code editor window with the following code and annotations:

```
101 SQLCA.DBMS = "OLE DB"
102
103 SQLCA.LogPass = "guest"
104 SQLCA.LogId = "apeon"
105 SQLCA.AutoCommit = False
106 SQLCA.DBParm = "PROVIDER='SQLNCLI11',DATASOURCE='192.168.1.1',PROVIDERSTRING='database=qa_datawindow'"
107
108
109 sqlca.lock = "rc"
110
111
112 error message
113
114
115 connect;
116 if sqlca.sqlcode <> 0 then
117     MessageBox("", sqlca.sqlerrtext)
118 endif
119
120 nvo_dotnetobject task
121 task = create nvo_dotnetobject
122
123 open(w_main)
```

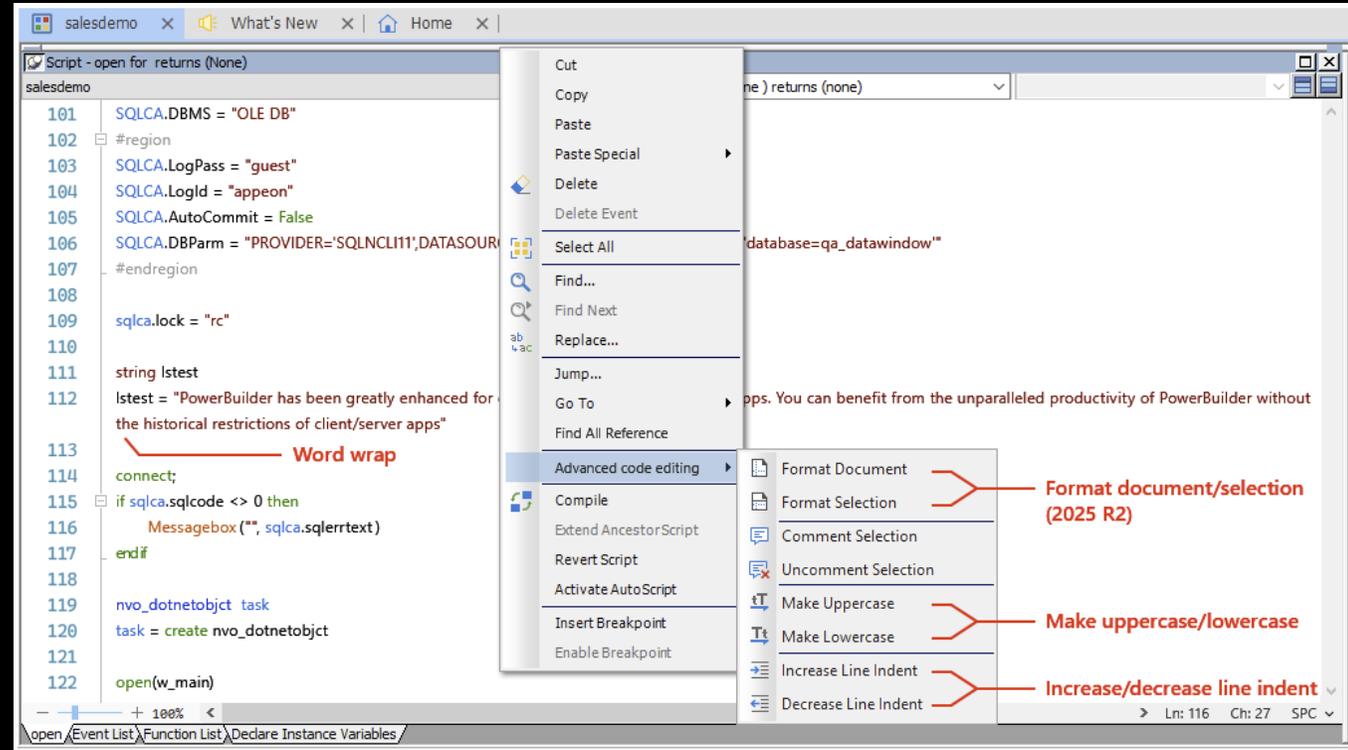
Annotations in the image:

- Selection/Searching matching:** Points to the text "OLE DB" on line 101.
- Breakpoint:** Points to a red circle on line 109.
- Error and warning marks (2025 R2):** Points to the text "error message" on line 112.
- Line selection highlighting:** Points to the entire line 116.
- Debugging highlighting:** Points to the text "task = create nvo_dotnetobject" on line 121.

Modern Code Editor

Code Readability (Continued)

- *Advanced* code editing
 - Word wrap
 - Improved Undo
 - Make uppercase/lowercase
 - Increase/decrease line indent
- Format document/selection (R2)



Modern Code Editor

Coding Efficiency

- Improved completion list
 - ✓ Include code snippet
 - ✓ Type matching completion
- Provide context-aware info
 - ✓ Quick info
 - ✓ Parameter info
- Refactoring code (R2)

The screenshot displays a code editor window with the following code and annotations:

```
101 SQLCA.DBMS = "OLE DB"
102 SQLCA.LogPass = "guest"
103 SQLCA.LogId = "apeon"
104 SQLCA.AutoCommit = False
105 SQLCA.DBParm = "PROVIDER='SQLNCLI11';DATASOURCE='192.168.1.1';PROVIDERSTRING='database=qa_datawindow'"
106
107 if
108     If Then
109     If Then Else
110     If Then Elseif
111     If Then Elseif Else
112     then
113     MessageBox("", sqlca.sqlerrtext)
114 end if
115 MessageBox("", sqlca.sqlerrtext)
116 string c, string t
117
118 nvo_dotnetobjct task
119 task = create nvo
120
121
122 open(w_main)
123
```

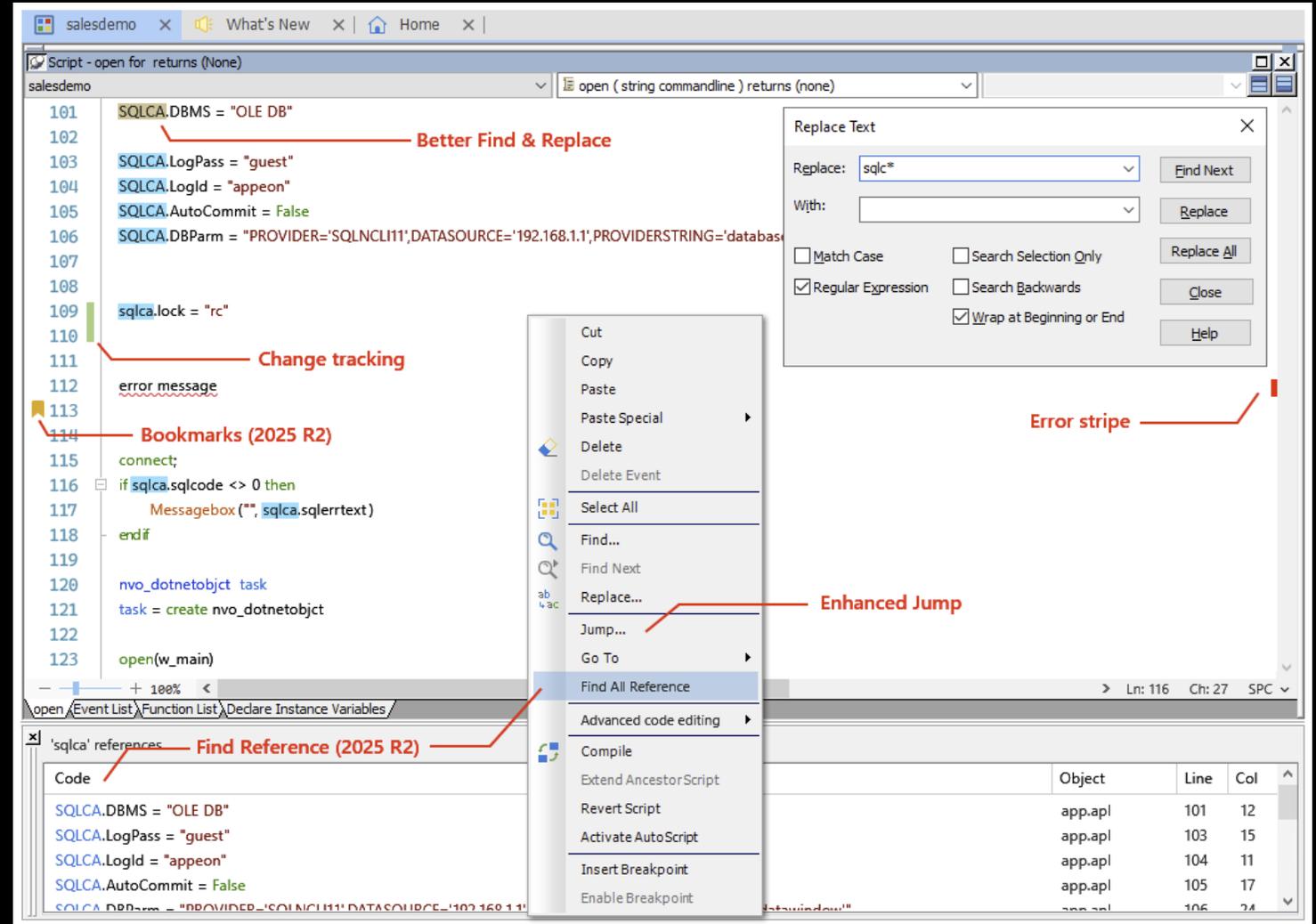
Annotations and features shown:

- Improved complete list and code snippet:** A dropdown menu is open at line 107, showing options like "If Then", "If Then Else", etc., with a code snippet for the "if" statement.
- Brace matching:** A red line connects the opening curly brace of the "if" statement to its closing brace.
- Quick info:** A tooltip shows the signature "function integer MessageBox (string c, string t)".
- Parameter info:** A tooltip shows the parameters "string c, string t".
- Refactoring code (2025 R2):** A dialog box is open for refactoring the variable "task", with options for "Include comments", "Include strings", and "Rename symbol's file".

Modern Code Editor

Code Navigation

- Better Find & Replace
- Enhanced Jump
- Change Tracking
- Error Stripe
- Find All Reference (*R2*)
- Bookmarks (*R2*)



The logo features two overlapping speech bubbles, one blue and one green, with a white ampersand between them. To the right of the bubbles, the word "Time" is written in a light blue, sans-serif font.

Q & A Time

- Q: Will the Solution be a *mandatory* migration step? A: No
- Q: Can a Solution be migrated *back* to a Workspace? A: No
- Q: Will the Solution support *Client/Server* compiles? A: Yes
- Q: Will the Solution support *PowerClient/PowerServer* compiles? A: Yes
- Q: Will the Solution support *M-code* compiles? A: No, only P-code
- Q: Will Appeon continue to support Workspace development / compilation? A: Yes
- Q: Will the Solution paradigm be added to InfoMaker? A: Possibly in the future
- Q: Will the Solution paradigm be supported by PBAutoBuild / ORCAScript? A: Yes
- Q: Will the Solution paradigm support PBR resources in an EXE/PBD? A: Yes
- Q: Will the Solution support PBD *only* compiles? A: No

=> Please join us in the **Chat** session if you have more questions!

Connect with Us



Discussions, tech articles and videos, free online training, and more.



facebook.com/AppeonPB

Encourage us with a “like”, see cool pics, and get notified of upcoming events.



twitter.com/AppeonPB

Follow Appeon and community members to get the latest tech news.



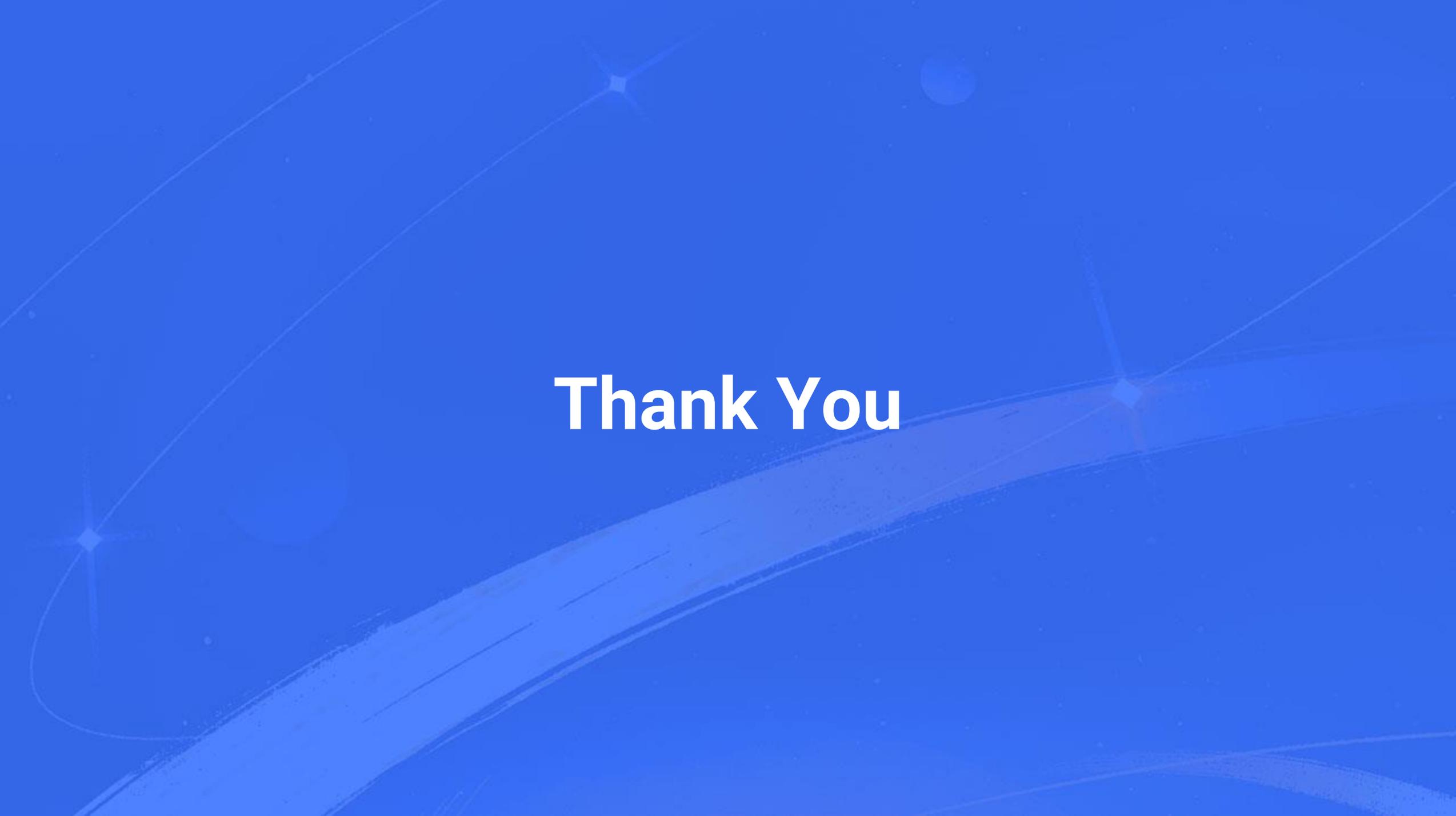
linkedin.com

Build up your career profile, and stay in contact with other professionals.



youtube.com/Appeon

Share important Appeon videos with others; no account registration required.

The background is a solid blue color with several abstract, light blue curved lines and starburst patterns scattered across it. The lines are thin and elegant, while the starbursts are small, multi-pointed shapes that resemble distant stars or light flares.

Thank You