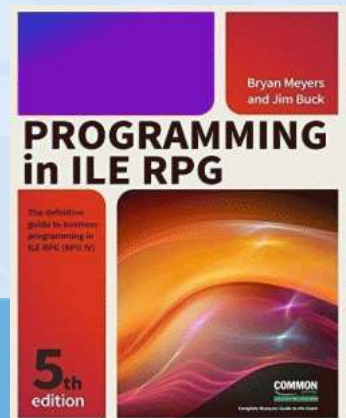


**Jim Buck**  
 Phone 262-705-2832  
[jbuck@impowertechnologies.com](mailto:jbuck@impowertechnologies.com)  
 Twitter - @jbuck\_imPower  
[www.impowertechnologies.com](http://www.impowertechnologies.com)



# How can imPower Technologies help your company?



## IBM i Education

Online IBM i Classes: Unique offering

IBM i Concepts

Programming in ILE RPG

RDi / Modular Programming – **Updated Workshop**

SQL Queries Workshop – Birgitta Hauser **New Workshop**

Onsite IBM i Classes: **Now Available**

Two-day hands-on lecture and exercises

Optional Third day - Let's design and code a new application

Modernization: Getting started

Helping a company getting started down the modernization road

The thought process of modern development

Learn to use these new tools and concepts

# Session Objectives

Relationship of OPM, EPM & ILE to IBM i OS

Converting RPG III to RPG IV (ILE RPG)

Additional Steps in the Modernization Process

ILE vs OPM – 80's Thinking

ILE ~~vs OPM~~ – A New way of thinking

The Next Generation of RPG programs

MVC Architecture!

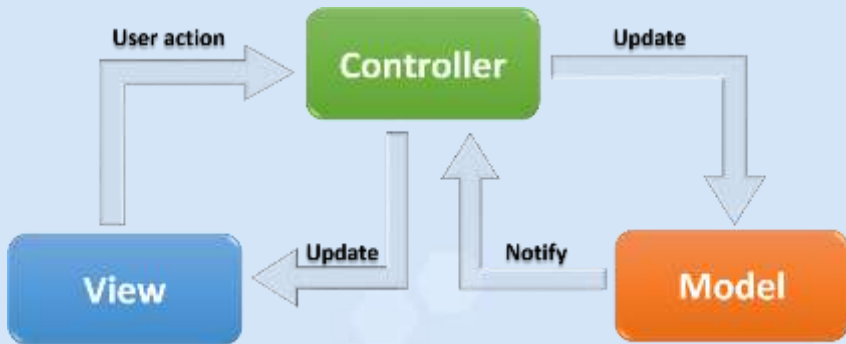
A Modern IBM i Application Overview

Activation Groups

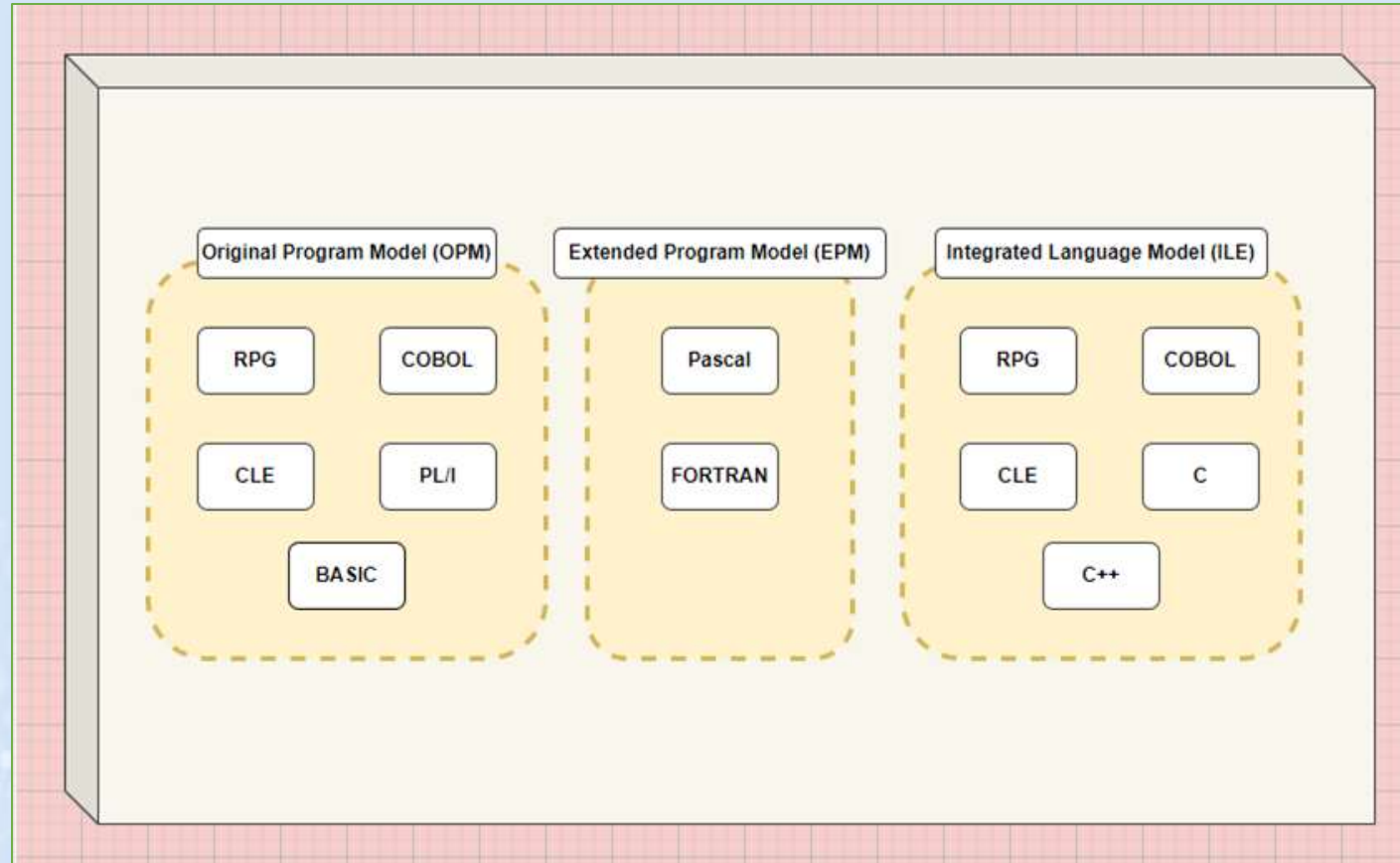
Introduction to Binding Directories

Introduction to Binder Source

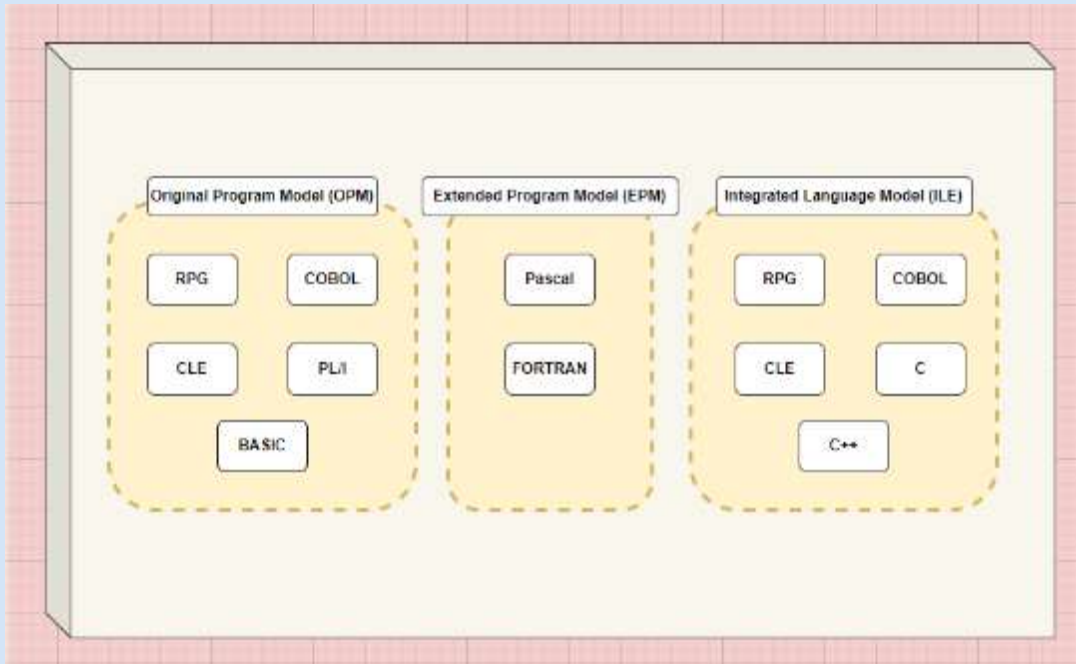
Create the Application



# Relationship of OPM, EPM & ILE to IBM i OS



# Original Program Model (OPM)

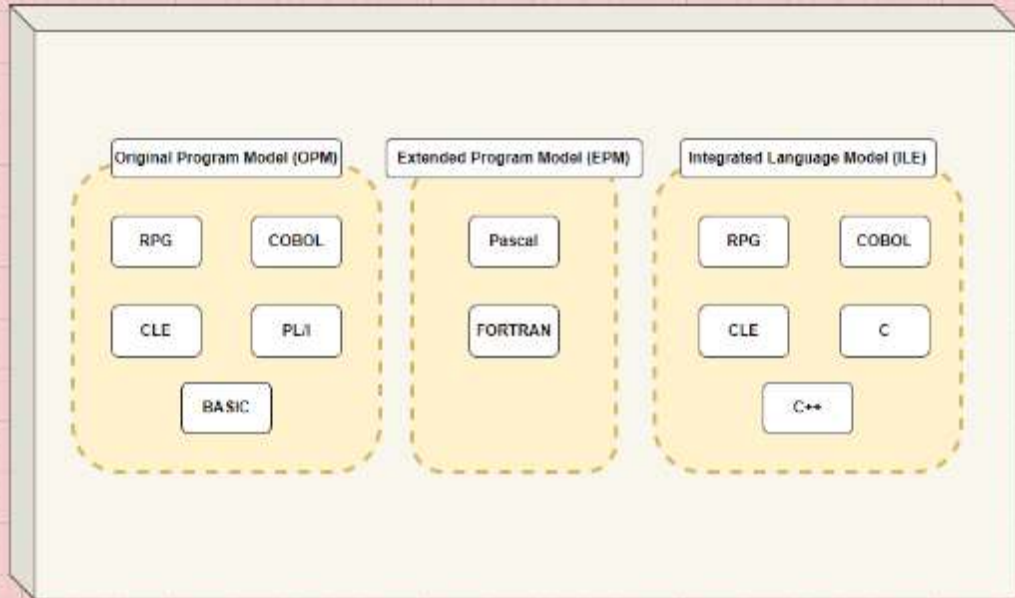


Introduced in OS/400 Version 1

- OPM compiler produced the program object and additional code to handle any special processing required. Example handling input parameters
- Special processing defined the **Entry Point** for the program.
- All Calls to a program were Dynamic (overhead)
- Dynamic calls can require significant resources. This led to large programs which reduced these calls
- Functions often provided by the compiler are included in the OS. This allows one language to call programs written in another
- As of version 6.1 BASIC compiler isn't available



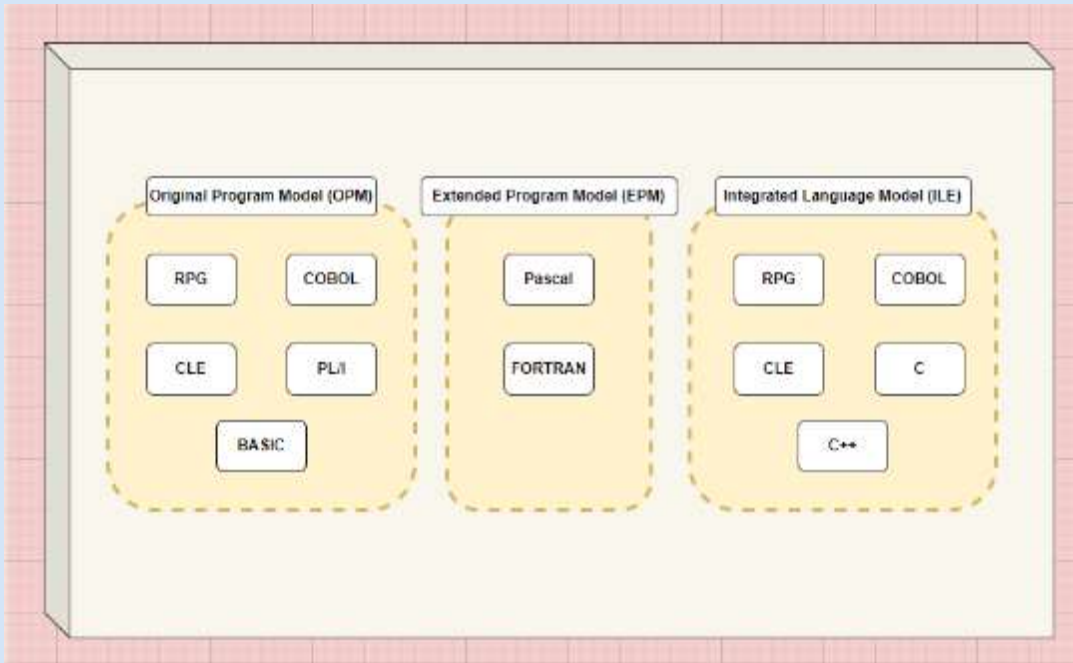
# Enhanced Program Model (EPM)



Introduced in OS/400 Version 1, release 2

- An enhancement to OPM which allowed the definition of procedure calls.
- Interim solution that allowed calls to languages like C, FORTRAN and Pascal and return data from these procedures
- The system no longer provides EPM compilers

# Integrated Language Environment



Introduced in OS/400 Version 2, release 3 (1994)

- Provided the same type procedure integration that EPM did but more robust and better performance
- The ILE compiler doesn't create an executable program object but a module object (CRTRPGMOD)
- The Executable is created in a separate step using the CRTPGM command.
- CRTBNDPGM combines the CRTRPGMOD and CRTPGM commands
- These program modules; written in COBOL, RPG, C, C++ and CL can be assembled into an executable program object

# Converting RPG III to RPG IV (ILE RPG)

## Convert RPG Source (CVTRPGSRC) command

- Can convert single member, source file or member with a common source name
- Converts line-by-line and updates a log file of the results
- Assumes that your code will compile
- Will mark RPG III code that isn't supported and this is usually a small amount





# Converting RPG III to RPG IV (ILE RPG)

## What the conversion tool won't do!



- Will not convert source back to RPG III or RPG/400
- Doesn't support conversion at compile time
- Doesn't support converting RPG II code, there is a utility for converting RPG II to RPG III
- The conversion utility **DOES NOT** re-engineer the source code, except where needed to convert to RPGILE
- Doesn't create files; the Log and output file must already exist

# Additional Utilities and Tools

## Craig Rutledge - Open Source iSeries Tools (JCRCMDS)

[Utilities](#) [API Xref](#) [Links](#) [Path to /Free](#) [Games](#) [Download](#) [Resume](#) [Help with domain/hosting fees](#)

FREE (no cost) rpg Free Format source using the [Free Software Foundation](#) license.

The JCRCMDS library has been featured many times in the [Iseries ClubTech Newsletter](#), recently JCRHFD and JCRK in IBM Redbook [Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between Rochester Initiative](#). The library continues to be a practical set of API-based utilities to really help AS/400, Iseries

SourceForge Project! [RDI/WDSCI plugins for JCRCMDS.com](#). First plug-in is JCRHFD (H F D specs free format) plug-in way to transition the jrcrmds tools into Rdi environment. If you wish to contribute plug-ins (and one day I hope to contact to add you as a project contributor.

v7.3 with current CUME tapes installed. Release June 2, 2022

### Free H F D

- [JCRHFD](#)
- [JCRPROTO](#)
- [JCRPRGEN](#)
- [JCRNUMB](#)
- [JCRSDENT](#)

### Outline View

- [JCRREFL](#)
- [JCRFELD](#)
- [JCRIND](#)
- [JCRSUBR](#)

### Source

Help text is provided with each utility. See [Download](#) link for install instructions.

Convert fixed format H, F and D specifications into free format DCL-\* statements

Create free format DCL-PI and DCL-PR statements from selected member fixed format

Generate free format DCL-PR prototype in selected member to call selected program

Automatically re-indent your free format code based on logic structures. Update statements in your source code. (not for use on v7.3 \*\*free yet. I have some

Show Source Indentation. Updated to show indentation lines for DCL-\* structures

File name / Record Format Xref for RPG source (you NEED this one!). Updated to read DCL-F source.

Field names, length and attributes used in selected RPG source code.

Indicator list used in \*RPG\* / CL\* / DSPF or PRTF source members. (This utility is now celebrating its 35th birthday)

Subroutine List has been replaced with RDI Visualize Application Diagram.

TARGET/400

Home About Us Software Downloads Support Contact

Get Started

## Protect your S/36 legacy software investment

TARGET/400 modernizes System/36 applications on the IBM i

Get Started

John Caine





# ILE vs OPM

Integrated Language Environment (ILE)

Main Benefit of ILE – Many small fast procedures

ILE procedure calls optimized to be VERY FAST!

Easier to maintain, when using smaller code components

More Exception handling options

Easier Access to APIs, including the entire C runtime library

Results in better performance! 😊



# ILE vs OPM – 80's Thinking

In the old days large programs used to do complicated applications

## Inefficient

- Programs hard to understand
- Difficult to test, all or nothing
- Programs difficult to maintain! 😞
  - Small changes can cause much effort (and heartache) to implement
  - Sometimes won't support current technologies

## Benefits?

- All your code in one place
- Torment young developers
- What else ????

# ILE vs OPM – A New way of thinking

Today's Applications – Should be comprised of small procedures

- Program flow is easier to see
- Small procedure is easier to understand
- Easier to test a procedure at a time
- Easier to share across multi-applications
- Easier to divide up the application among your developers
- Good procedure names can make the code self-documenting
- Easily step over procedures when debugging
- Easier to define and debug where problems are occurring

More Control

- Activation Groups
- Exception handling

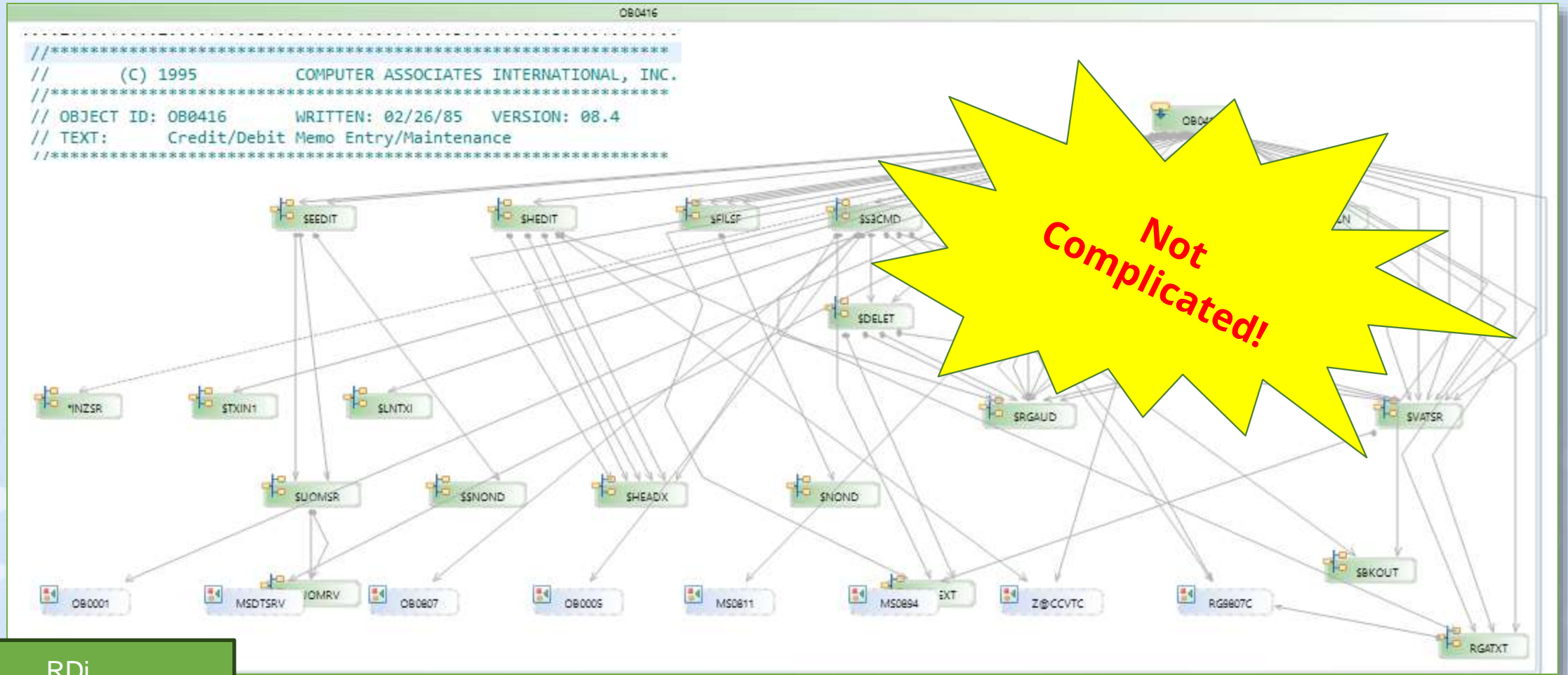
# What NOT to consider!

Is there an optimum????

- Size of a procedure
- Number of procedures per module
- Depth of code nesting level
- Number of Binding directories
- Service programs per binding directory
- Modules per Service Program

**ILE doesn't  
care and  
neither  
should you!**

# Traditional Monolith Program



RDi  
Visual Application  
Diagram



# The Next Generation of RPG programs



## The future of RPG

- RPG isn't going anywhere
- Need to expend resources to update older code. The tools are available!
- Develop new code that will be usable regardless of the changes made to the VIEW
- Write RPG programs that can be used for Web Services

# MVC Architecture!

## Break up your code into three components

- Introduced in 1979 by Trygve Mikkjel Heyerdahl Reenskaug
- His idea was to break-up complicated applications into smaller more manageable parts
- 80's and early 90's popular with desktop application
- Today it is widely used in Web Application Frameworks
- Some frameworks, ASP.NET, Angular, Ruby on Rails

# MVC Architecture!

Break up your code into three components

- **Model** – responsible for the data-logic behind the application
- **View** - How the user sees and interacts with the application
- **Controller** - Makes decisions for the application based on the actions sent from the View and data from the Model.  
The “*Decision maker*” of the application

# A Modern IBM i Application Overview

## PROG175SQL.sqlrpgle

Program drives the 5250 screens assorted Subfiles and handles the interactions needed to maintain part orders for the company

## PROG175D Screen

DDS needed to Display, Delete and maintain orders

## CUSTSRVPGM.sqlrpgle

Handles all of the Customer table database work

## CINVSrvPGM.sqlrpgle

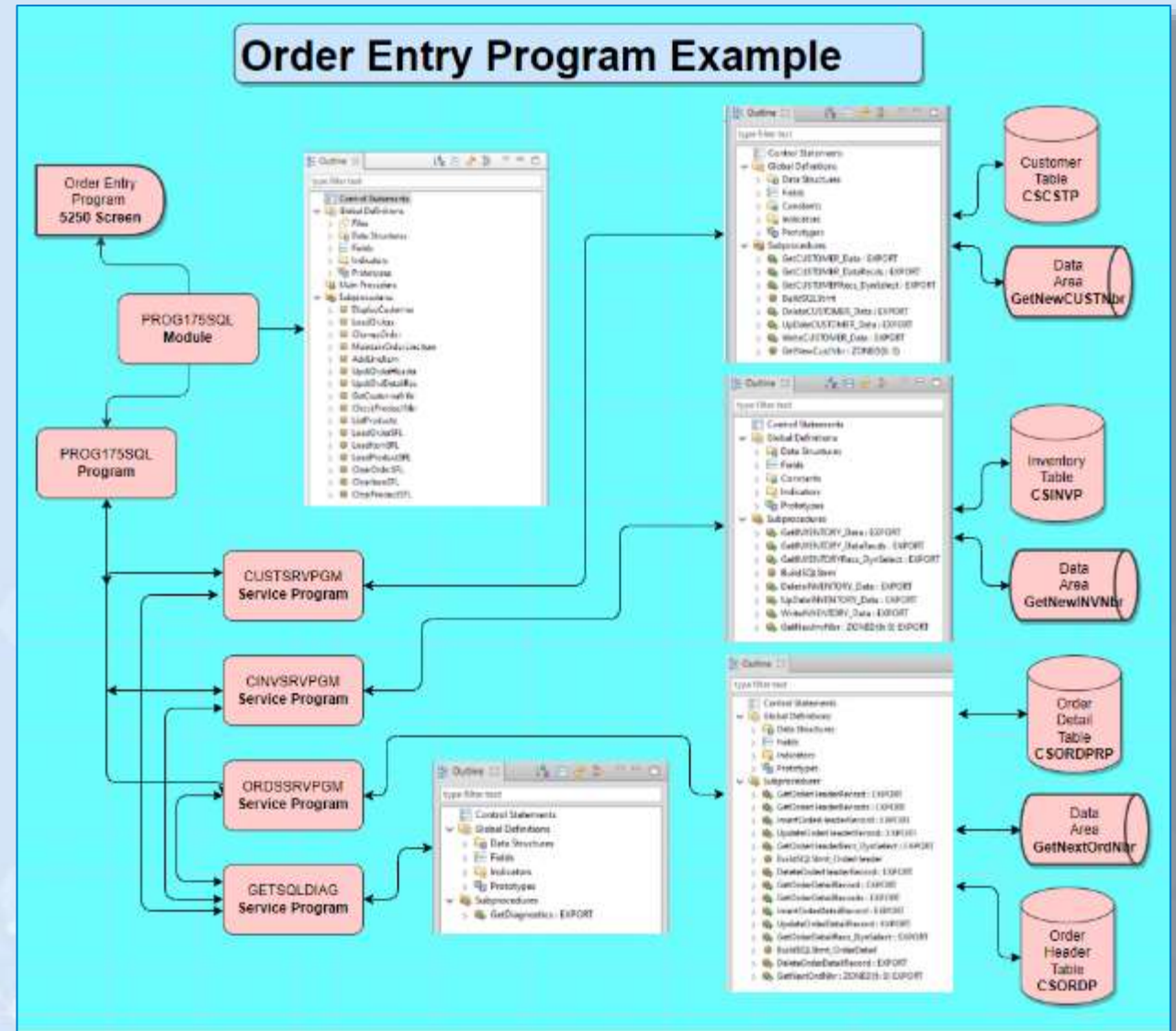
Handles all of the Inventory table database work

## ORDSSRVPGM.sqlrpgle

Handles all of the Order Header/Detail table database work

## GETSQLDIAG.sqlrpgle

Service program called to check results of any SQL Statement





# The View

Allow the user to:

Add / change shipping orders

1. Enter Customer number
2. Lists and change orders for the selected customer

```
A - IBMIMPOWERTECHNOLOGIES.COM
File Edit View Communication Actions Window Help
Program ID: PROG175SQL      CloudServices24x7, Inc.      12/20/21
Order Entry Application

Customer Number: 100008

Enter Customer Number Then ENTER: . . .

F3=Exit
```

```
A - IBMIMPOWERTECHNOLOGIES.COM
File Edit View Communication Actions Window Help
Program ID: PROG175SQL      CloudServices24x7, Inc.      12/20/21
Screen: OrdCTL      Order Entry Application

Customer Information
Customer # 100008      Order Date 4-19-2009      Balance Due $ .00
First Last      Phone:      Email Address
DAVID RYAN HUNTER      317-223-5277
Street      City      ST      Zip Code
241 CROWN AVE.      WATERLOO      IA      50701-0037
Press F9 to Add an Order      Enter C to maintain Order

OPT      Order      Line      Tracking      Amount      Order
Number      Items      STS      Date      Number      Paid      Total
-      -      -      -      -      -      -
20331      00010      2021-12-02      $ .00      $258.00
20332      00007      2021-12-02      $ .00      $17.90
20333      00001      2021-12-02      $ .00      $22,574.25
20334      00001      2021-12-06      $ .00      $380.00
20335      00001      2021-12-06      $ .00      $4,250.00

More . . .

MW
```

# The Controller Program

PROG175SQL – The Controller

Contains Controller Subprocedures

Displays the correct screen based on user interactions

Based on user interactions contains logic to do calculations needed by the application

Calls correct service program and subprocedure to handle database updates (The Model)

Evaluates the results of database requests and sends these results to the user of the application (The View)

```
PROG175SQL.SQLRPGL
Line 1      Column 1      Replace
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...0
**Free
Ctl-Opt Option(*NoDebugIO:*SrcStmt:*NoUnRef) alwnull(*usrctl);
Ctl-Opt bnmdir('ORDSSHEMA/ORDERSBIND');
//
// =====
//   Created By:.... Jim Buck
//   Program Name:.. PROG175SQL
//   Created Date:.. 03/06/2021
//   Updated Date:.. 02/16/2022
// =====
//   Description: CloudServices24x7 Order Entry Program
//   This program prompts for a customer number and then displays the
//   orders for a customer. The user can then modify or add a new order
//   or line items on an order.
//
//   This program is used to show how to put two modules together to
//   produce a functional Order Entry Application. This example has a
//   Module to run the screen (View and Controller) and a module to
//   handle the disk I/O (The Module)
//
// =====
// *****
// *** This is used for the Module Section of the class ***
//   Copy in the Module Prototype
//   This changes depending on where the member is stored
//copy ORDSSHEMA/QRPGSQLSRC,PROG175CY
// *****
// *** These copybooks are used for the Service program Section ***
// *** Copy in the Service Program Prototypes
```

- Control Statements
- Global Definitions
- Main Procedure
- Subprocedures
  - DisplayCustomer
  - LoadOrders
  - ChangeOrder
  - MaintainOrderLineItem
  - AddLineItem
  - UpdtOrderHeader
  - UpdtOrdDetailRec
  - GetCustomerInfo
  - CheckProductNbr
  - ListProducts
  - LoadOrderSFL
  - LoadItemSFL
  - LoadProductSFL
  - ClearOrderSFL
  - ClearItemSFL
  - ClearProductSFL

# The Model Programs

CINVSrvPGM – A model Program

Contains the Data base Subprocedures

Updates the Inventory Database Tables

Checks the results of the database access

Calls GETSQLDIAG to check results of any SQL Statements

Returns the database results and status to the Controller

The screenshot displays the source code of the CINVSrvPGM program in an IBM i development environment. The code includes comments and SQL-related definitions. The Outline window on the right lists the following subprocedures:

- Control Statements
- Global Definitions
  - Data Structures
  - Fields
  - Constants
  - Prototypes
- Subprocedures
  - GetINVENTORY\_Data : EXPORT
  - GetINVENTORY\_DataRecds : EXPORT
  - GetINVENTORYRecs\_DynSelect : EXPORT
  - BuildSQLStmnt
  - DeleteINVENTORY\_Data : EXPORT
  - UpdateINVENTORY\_Data : EXPORT
  - WriteINVENTORY\_Data : EXPORT
  - GetNewInvNbr : ZONED(6: 0) EXPORT

**CINVSrvPGM  
Service PGM**

# The Model Programs

The screenshot displays the IBM i development environment. The main window shows the source code for the CUSTSRVPGM.SQLRPGL program. The code includes comments and declarations for subprocedures. The Outline window on the right lists the subprocedures and their return types.

```
Line 1      Column 1      Replace
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+
000101 **Free
; //
creates internal PCML Info
*****
ies
*
*
here are *
the code! *
*****
001009 DynUsrPrf = *User,
001010 Datfmt = *iso,
001011 CloSqlCsr = *EndMOD;
001012
001100 // =====
001101 // Data Area To Assign Next Customer Number
001102 // =====
001103 Dcl-ds NewCustNbr Dtaara(*Usrctl : '*LIBL/NEWCUSTNBR');
001104 CharCustNextNbr char(6) pos(01);
001105 End-ds;
```

**Outline:**

- Control Statements
- Global Definitions
  - Data Structures
  - Fields
  - Constants
  - Prototypes
- Subprocedures
  - GetCUSTOMER\_Data : EXPORT
  - GetCUSTOMER\_DataRecds : EXPORT
  - GetCUSTOMERRecs\_DynSelect : EXPORT
  - BuildSQLStmnt\_Customer
  - DeleteCUSTOMER\_Data : EXPORT
  - UpDateCUSTOMER\_Data : EXPORT
  - WriteCUSTOMER\_Data : EXPORT
  - GetNewCustNbr : INT(10)

CUSTSRVPGM – A model Program

Contains the Data base Subprocedures

Updates the Customer Database Tables

Checks the results of the database access

Calls GETSQLDIAG to check results of any SQL Statements

Returns the database results and status to the Controller



# The Model Programs

ORDSSRVPGM – A Model Program

Contains the Database Subprocedures

Updates the Order Header/Detail Database Tables

Checks the results of the database access

Calls GETSQLDIAG to check results of any SQL Statements

Returns the database results and status to the Controller

The screenshot displays the source code of the ORDSSRVPGM program in a development environment. The code includes comments and SQL-related logic. A table at the bottom of the code block shows global definitions for SQL execution options.

Line	Column	Replace
000120		
000121		EXEC SQL
000122		Set Option
000123		Naming = *Sys,
000124		Commit = *None,
000125		UsrPrf = *User,
000126		DynUsrPrf = *User,
000127		Datfmt = *iso,

The Outline window on the right shows the program's structure, including Global Definitions, Fields, Indicators, Prototypes, and Subprocedures. The subprocedures listed are:

- GetOrderHeaderRecord : EXPORT
- GetOrderHeaderRecords : EXPORT
- InsertOrderHeaderRecord : EXPORT
- UpdateOrderHeaderRecord : EXPORT
- GetOrderHeaderRecs\_DynSelect : EXPORT
- BuildSQLStmnt\_OrderHeader
- DeleteOrderHeaderRecord : EXPORT
- GetOrderDetailRecord : EXPORT
- GetOrderDetailRecords : EXPORT
- InsertOrderDetailRecord : EXPORT
- UpdateOrderDetailRecord : EXPORT
- GetOrderDetailRecs\_DynSelect : EXPORT
- BuildSQLStmnt\_OrderDetail
- DeleteOrderDetailRecord : EXPORT
- GetNextOrdNbr : ZONED(5: 0) EXPORT



# Activation Groups

## Activation Groups

- Nothing more than an isolated area with its own resources
- Contains shared-open, overrides and commitment control to a portion of your job
- Applications can't interfere with each other
- When an activation group ends all the files used by the group are closed

# Activation Groups

## Basic Facts

- **OPM Programs** – run in their own environment. If a program is run multiple times the environment needs to be recreated
- **ILE Programs** – still need an environment to run in but they can share these environments and even keep them open between programs. Used properly results in better management
- What's the goal of using ILE Activation groups?
  - Minimize the number of resources consumed creating environments and reduce the overhead of multiple environments
  - Run related programs in the same activation group

# Activation Groups

## Types of Groups

- Using the CRTRPGPGM command the DFTACTGRP parameter is available
  - Two options
    - **\*YES**, the ILE program will act like a OPM program, Why bother?
    - **\*NO**, the ILE program will need a named Activation group
      - When set to \*NO, the ACTGRP parameter appears



# Activation Groups

## ACTGRP Parameter - \*NEW

- **\*NEW** – Causes a NEW activation group to be created when the program is run
  - System will create and name the group every time the program is run.
  - This will create system overhead
  - The system will act like an OPM system

# Activation Groups

## ACTGRP Parameter – “NAME”

- **“NAME”** – If Specified on every compile, WORKS LIKE \*NEW
  - The system will work as an OPM system
  - Except when the program ends the Activation group continues to run and can only be cleaned up is to run the **RCLACTGRP**.

# Activation Groups

## ACTGRP Parameter – \*CALLER

- **\*CALLER** – causes the program to run in the same activation group as the program that called it.
- Should only be used when you want a program to run in a known activation group
- Most often used with service programs.

# Activation Groups

## When the Activation group closes

- Files opened by the activation group close
- Activation-scoped file overrides are freed up
- Static storage used by the programs and service programs gets released
- Allocated storage (%ALLOC) gets released
- Activation-scoped Commitment Control will end and if commitment control is being used database changes will be committed

# Error Handling

OPM Program or program created with DFTACTGRP(\*YES)

- Handle Exceptions using error indicator, (e), \*PSSR, INFSR or an Inquiry message

When using ILE there are additional options

- Percolate the exception to the caller (up the stack)
- Register a procedure to run if the procedure crashes (CEEUTX)
- Register a procedure to run when there is an exception (CEEHDLR)



# Introduction to Binding Directories

## Some Characteristics of Binding Directories

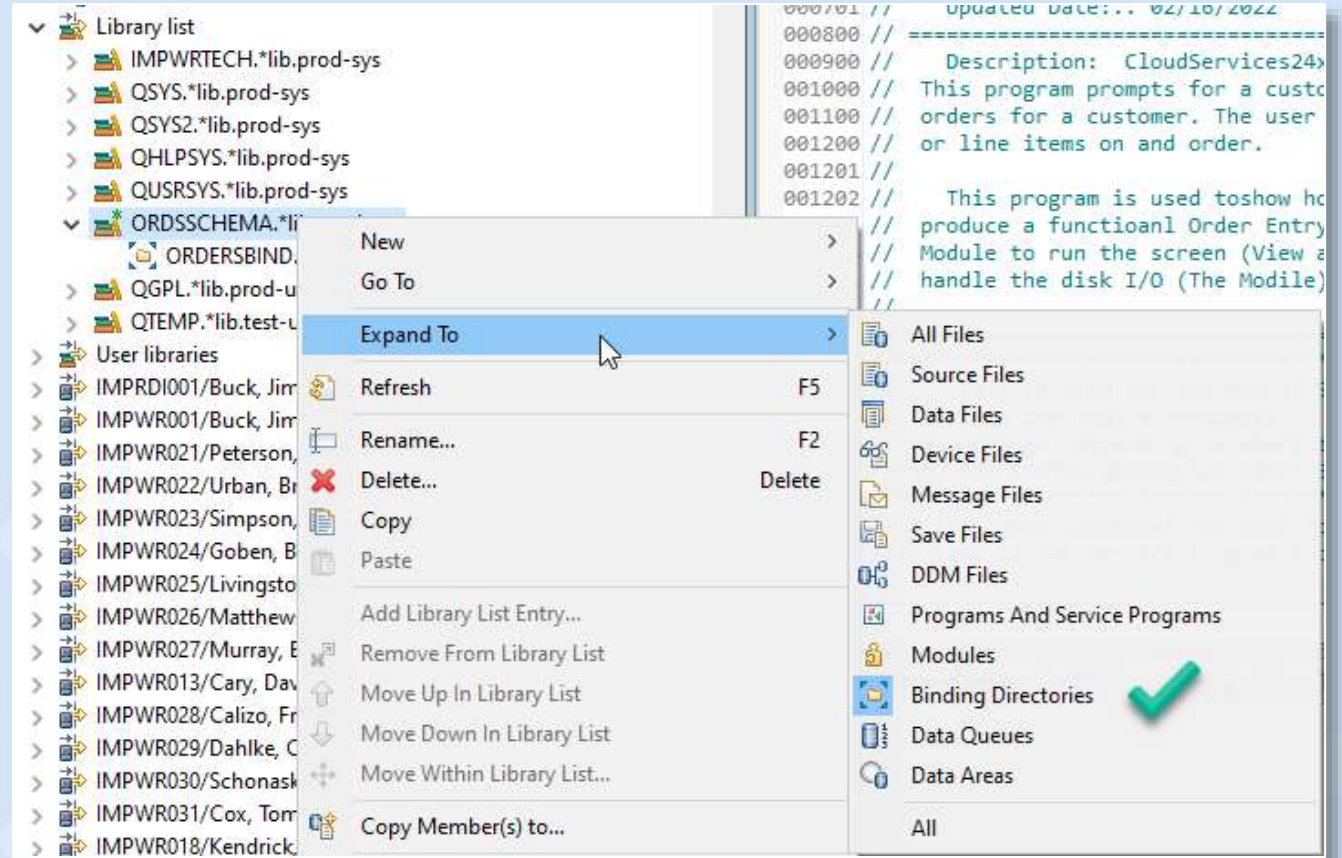
- Convenient method of grouping the names of modules and service programs ***that may be needed*** to create an ILE program or service program
- The object names listed do not have to exist at the time the binding directory is created and populated
- \*LIBL or a specific library name are the only valid entries
- The object names listed are optional:
  - The named objects are used only if unresolved imports exist and if named object provides a needed export for an unresolved import request

# Binding Directories

Used to list the names of modules and service programs that may be used in an application

They are optional and are used as a convenience and to reduce program size

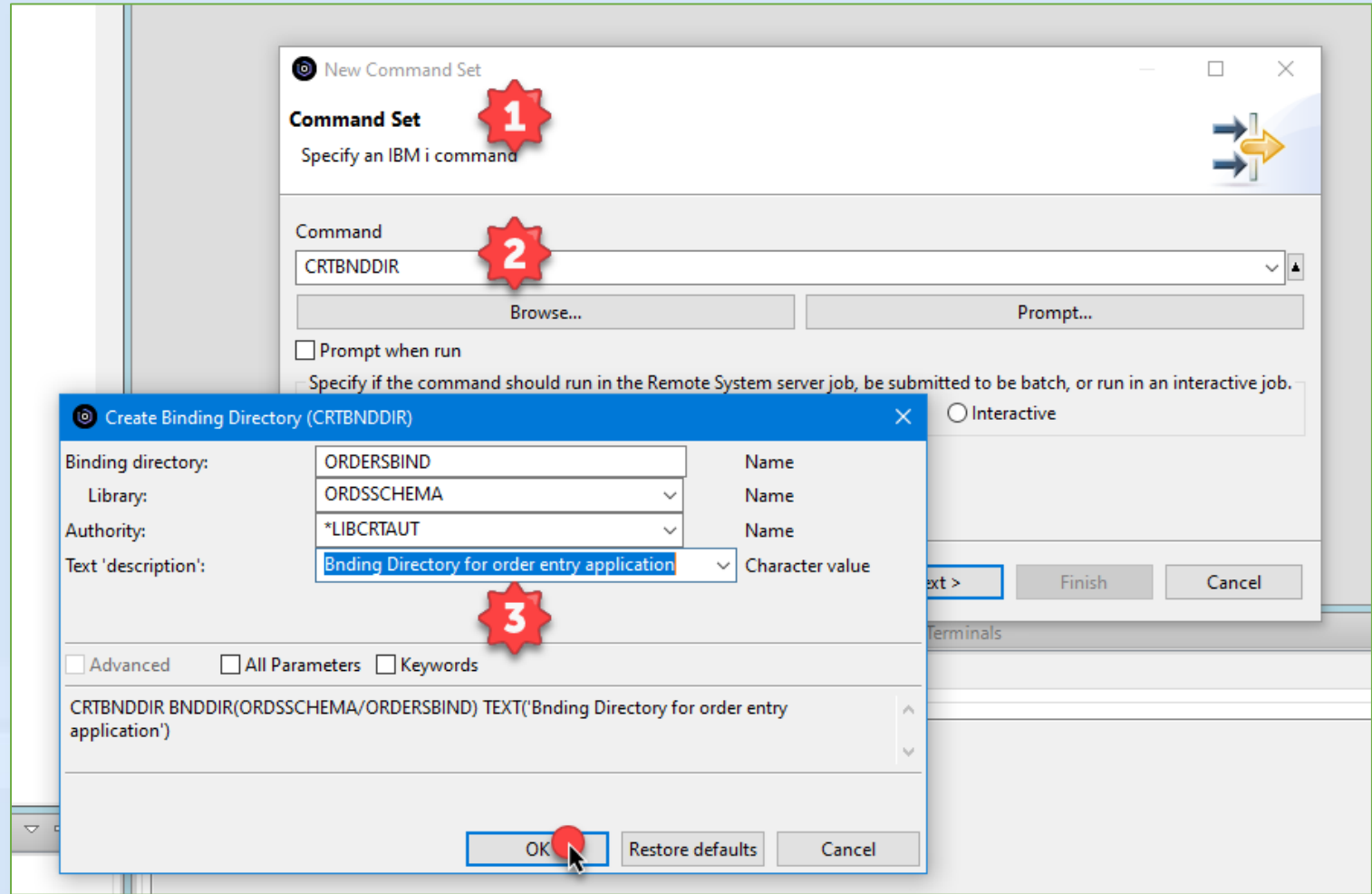
Important to list only modules and service programs that are potentially used



# Binding Directories

## Create a Binding Directory

1. Create a new Command
  2. Type the command name
  3. Fill in the blank
- Then Ok



# Binding Directories



The screenshot shows the iSphere Binding Directory Editor interface. On the left, a tree view displays a 'Library list' with various system libraries. A context menu is open over the 'ORDSSCHEMA.\*lib.prod-cur' library, with the 'iSphere Binding Directory Editor' option highlighted and marked with a red '1'. A red arrow points from this option to a table of objects. The table has columns for 'Library', 'Object', 'Object type', and 'Activation'. The row for 'ORDSSCHEMA' with object 'GETSQLDIAG' is selected. A context menu is open over this row, with the 'New' option highlighted and marked with a red '2'.

Library	Object	Object type	Activation
ORDSSCHEMA	ORDSSRVPGM	*SRVPGM	*IMMED
ORDSSCHEMA	CUSTSRVPGM	*SRVPGM	*IMMED
ORDSSCHEMA	CINVSrvPGM	*SRVPGM	*IMMED
ORDSSCHEMA	GETSQLDIAG	*SRVPGM	*IMMED

## Populate Binding Directory

- Expand to: set to Binding Directory
1. Right-Click and select ***iSphere Editor***
  2. Right-click and select New



# Using Binding Directory

```
CINVRVPGM.BND  PROG175D.DSPF  PROG175SQL.SQLRPGLE  ✕
Line 28      Column 1      Replace
  |...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
000001 **Free
000002 Ctl-Opt Option(*NoDebugIO:*SrcStmt:*NoUnRef) al null(*usrctl);
000003 Ctl-Opt bnmdir('ORDSSCHEMA/ORDERSBIND');
000300 // =====
000500 //      Created By:.... Jim Buck          *
000600 //      Program Name:.. PROG175SQL        *
000700 //      Created Date:.. 03/06/2021       *
000701 //      Updated Date:.. 02/16/2022     *
000800 // =====
000900 //      Description: CloudServices24x7 Order Entry Program *
001000 // This program prompts for a customer number and then displays the *
001100 // orders for a customer. The user can then modify or add a new order*
001200 // or line items on and order.          *
001201 //                                     *
001202 //      This program is used to show how to put two modules together to *
001203 // produce a functional Order Entry Application. This example has a *
001204 // Module to run the screen (View and Controller) and a module to *
001205 // handle the disk I/O (The Module)    *
001206 //                                     *
001501 // =====
001516 //*****
001517 // *** This is used for the Module Section of the class ***
001700 // Copy in the Module Prototype
001800 // This changes depending on where the member is stored
001900 //copy ORDSSCHEMA/QRPGSQLSRC,PROG175CY
001901 //*****
001902 //// *** These copybooks are used for the Service program Section ***
001903 //// Copy in the Service Program Prototypes
(Main Procedure)
```

## Specifying the Binding Directory

- Use the BNDDIR keyword on the Ctl-Opt spec of your RPG modules
- Use the BNDDIR keyword on the CRTPGM or CRTBNDRPG command
- Use the BNDSRVPGM keyword on the CRTPGM (or CRTSRVPGM) command

# Intro to Binder source

Binder source is used when creating a service program. A signature is created from the order the modules are listed and it shows the order modules are exported

- The file must contain:
  - **Start Program Export (STRPGMEXP)** command identifies the beginning of the list of exports from the service program.
  - **Export Symbol (EXPORT)** commands identify each symbol name (Procedure) available to be exported from the service program.
  - **End Program Export (ENDPGMEXP)** command identifies the end of the list of exports from the service program.

## Important

Do not to change the order of exports in the binder source!



# Retrieve Binder source

The screenshot shows the IBM i development environment. On the left, a 'Library list' pane displays a tree view of libraries. A context menu is open over the 'CINVSrvPGM' object, with 'Retrieve Binder Source' highlighted. A green arrow points from this menu item to the right-hand pane. The right-hand pane displays the binder source code for 'CINVSrvPGM.BND'. The code includes a signature line (000001) and a list of sub-procedures (000005-000011). Two red starburst callouts are present: '1' points to the signature line, and '2' points to the list of sub-procedures.

```
000001 STRPGMEXP  PGMLVL(*CURRENT) SIGNATURE(X'AF573EB968754D5A6D4F26FAF8C133F1')
000002 /*****
000003 /*  *SRVPGM      CINVSrvPGM  ORDSSSCHEMA  03/27/22  16:34:29
000004 /*****
000005 EXPORT SYMBOL("DELETEINVENTORY_DATA")
000006 EXPORT SYMBOL("GETINVENTORY_DATA")
000007 EXPORT SYMBOL("GETINVENTORY_DATARECS")
000008 EXPORT SYMBOL("GETINVENTORYRECS_DYNSELECT")
000009 EXPORT SYMBOL("GETNEWINVNBR")
000010 EXPORT SYMBOL("UPDATEINVENTORY_DATA")
000011 EXPORT SYMBOL("WRITEINVENTORY_DATA")
000012 ENDPGMEXP
```

Retrieve the Binder source

- Right-click the program object
- 1.The program current signature
- 2.The list of current sub-procedures

# Original Service Program

```
Line 11      Column 43      Insert  6 changes
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----
000001
000014 STRPGMEXP PGMLVL(*CURRENT) SIGNATURE(X'6DA057FC9680A07C5244F5EFA13FD6')
000015 /*****
000016 /*  *SRVPGM      CUSTSRVPGM  ORDSSHEMA  03/22  16:03:39  */
000017 /*****
000018 EXPORT SYMBOL("GETCUSTOMER_DATA")
000019 EXPORT SYMBOL("GETCUSTOMER_DATARECDS")
000020 EXPORT SYMBOL("GETCUSTOMERRECS_DYNSELECT")
000021 EXPORT SYMBOL("DELETECUSTOMER_DATA")
000022 EXPORT SYMBOL("UPDATECUSTOMER_DATA")
000023 EXPORT SYMBOL("WRITECUSTOMER_DATA")
000024 ENDPGMEXP
```

Outline

Type filter text

- Control Statements
- Group Definitions
  - Data Structures
  - Constants
  - Prototypes
- Subprocedures
  - GetCUSTOMER\_Data : EXPORT
  - GetCUSTOMER\_DataRecds : EXPORT
  - GetCUSTOMERRecs\_DynSelect : EXPORT
  - BuildSQLStmt\_Customer
  - DeleteCUSTOMER\_Data : EXPORT
  - UpDateCUSTOMER\_Data : EXPORT
  - WriteCUSTOMER\_Data : EXPORT
  - GetNewCustNbr : INT(10)



# Changed Service Program

Line 1	Column 1	Replace
000001	STRPGMEXP	PGMLVL(*CURRENT) SIGNATURE(X'7961D5E649483B4651E76B5C706CBBAE')
000002	/*****	
000003	/*	*SRVPGM CUSTSRVPGM ORDSSHEMA 5/24/22 11:41:10 */
000004	/*****	
000005	EXPORT	SYMBOL("DELETECUSTOMER_DATA")
000006	EXPORT	SYMBOL("GETCUSTOMER_DATA")
000007	EXPORT	SYMBOL("GETCUSTOMER_DATAARECDS")
000008	EXPORT	SYMBOL("GETCUSTOMERRECS_DYNSELECT")
000009	EXPORT	SYMBOL("SORTCUSTOMER_DATAARECDS")
000010	EXPORT	SYMBOL("UPDATECUSTOMER_DATA")
000011	EXPORT	SYMBOL("WRITECUSTOMER_DATA")
000012	ENDPGMEXP	

Outline window showing subprocedures:

- Control Statements
- Global Definitions
- Data Structures
- Fields
- Constants
- Prototypes
- Subprocedures
  - GetCUSTOMER\_Data : EXPORT
  - SortCUSTOMER\_DataRecds : EXPORT**
  - GetCUSTOMER\_DataRecds : EXPORT
  - GetCUSTOMERRecs\_DynSelect : EXPORT
  - BuildSQLStmt\_Customer
  - DeleteCUSTOMER\_Data : EXPORT
  - UpDateCUSTOMER\_Data : EXPORT
  - WriteCUSTOMER\_Data : EXPORT
  - GetNewCustNbr : INT(10)

New Binder Source shows the new procedure

# Updated Service Program

The screenshot displays the IBM i development environment. The main window shows the binder source file `CUSTSRVPGM.BND` with the following code:

```
Line 24      Column 8      Insert
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
000027 STRPGMEXP  PGMLVL(*CURRENT) SIGNATURE(X'7961D5E649483B4671E76B5C706CBBAE')
000028 /*****
000029 /* *SRVPGM      CUSTSRVPGM  ORDSSHEMA  04/14/22  11:41:10  */
000030 /*****
000031 EXPORT SYMBOL("DELETECUSTOMER_DATA")
000032 EXPORT SYMBOL("GETCUSTOMER_DATA")
000033 EXPORT SYMBOL("GETCUSTOMER_DATAARECDS")
000034 EXPORT SYMBOL("GETCUSTOMERRECS_DYNSELECT")
000035 EXPORT SYMBOL("SORTCUSTOMER_DATAARECDS")
000036 EXPORT SYMBOL("UPDATECUSTOMER_DATA")
000037 EXPORT SYMBOL("WRITECUSTOMER_DATA")
000038 ENDPGMEXP
000039
000040 STRPGMEXP  PGMLVL(*PRV) SIGNATURE(X'6DA057...91A3A07C5244F58...')
000041 /*****
000042 /* *SRVPGM      CUSTSRVPGM  ORDSSCHEM  04/13/22  16:03:
000043 /*****
000044 EXPORT SYMBOL("GETCUSTOMER_DATA")
000045 EXPORT SYMBOL("GETCUSTOMER_DATAARECDS")
000046 EXPORT SYMBOL("GETCUSTOMERRECS_DYNSELECT")
000047 EXPORT SYMBOL("DELETECUSTOMER_DATA")
000048 EXPORT SYMBOL("UPDATECUSTOMER_DATA")
000049 EXPORT SYMBOL("WRITECUSTOMER_DATA")
000050 ENDPGMEXP
```

An "Update Service Program (UPDSRVPGM)" dialog box is open, showing the configuration for updating the service program. The "Service program" is `CUSTSRVPGM` in the `ORDSSHEMA` library. The "Export" is `*SRCFILE` in the `*LIBL` library, with the source file `QSRVSRC` and source member `*SRVPGM`. A green arrow points from the `EXPORT SYMBOL("DELETECUSTOMER_DATA")` line in the binder source to the `*SRCFILE` export selection in the dialog.

At the bottom, the "Commands Log" shows the execution of the `UPDSRVPGM` command:

```
UPDSRVPGM SRVPGM(ORDSSHEMA/CUSTSRVPGM) MODULE(ORDSSHEMA/CUSTSRVPGM)
AUT and USRPRF parameter values were ignored.
Cause . . . . . : The AUT and USRPRF parameter values were ignored because REPLACE(*
attribute and the USEADPAUT attribute were copied from the existing object to the new
determine if the authority for object CUSTSRVPGM type *SRVPGM in library ORDSSCHEM
authority for object CUSTSRVPGM type *SRVPGM in library ORDSSHEMA if needed.
Replaced object CUSTSRVPGM type *SRVPGM was moved to QRPLOBJ.
Cause . . . . . : Replaced object CUSTSRVPGM type *SRVPGM from the ORDSSHEMA lib
```

Using the combined Binder Source

# Updated Service Program

```
A - IBM.IMPOWERTECHNOLOGIES.COM
File Edit View Communication Actions Window Help
[Icons]
Display Service Program Information                                     Display 1 of 1
Service program . . . . . : CUSTSRVPGM
Library . . . . . : ORDSSCHEM
Owner . . . . . : JBUCK
Service program attribute . . . . . : RPGLE
Detail . . . . . : *SIGNATURE

Signatures:

7961D5E649483B4671E76B5C706CBBAE
6DA057FC9691A3A07C5244F5EFA13FD6

F3=Exit F11=Display character
(C) COPYRIGHT IBM CORP. 1980
M^+ ^ MW
IBM.IMPOWERTECHNOLOGIES.COM:992 128
```

Updated Service program with multiple signatures

# Create the Application

## Step 01

- Create the 5250 Display file PROG175D

## Step 02

- Create The Module PROG175SQL

## Step 03

- Create the Module CUSTSVRPGM
- Create the Service Program CUSTSVRPGM

## Step 04

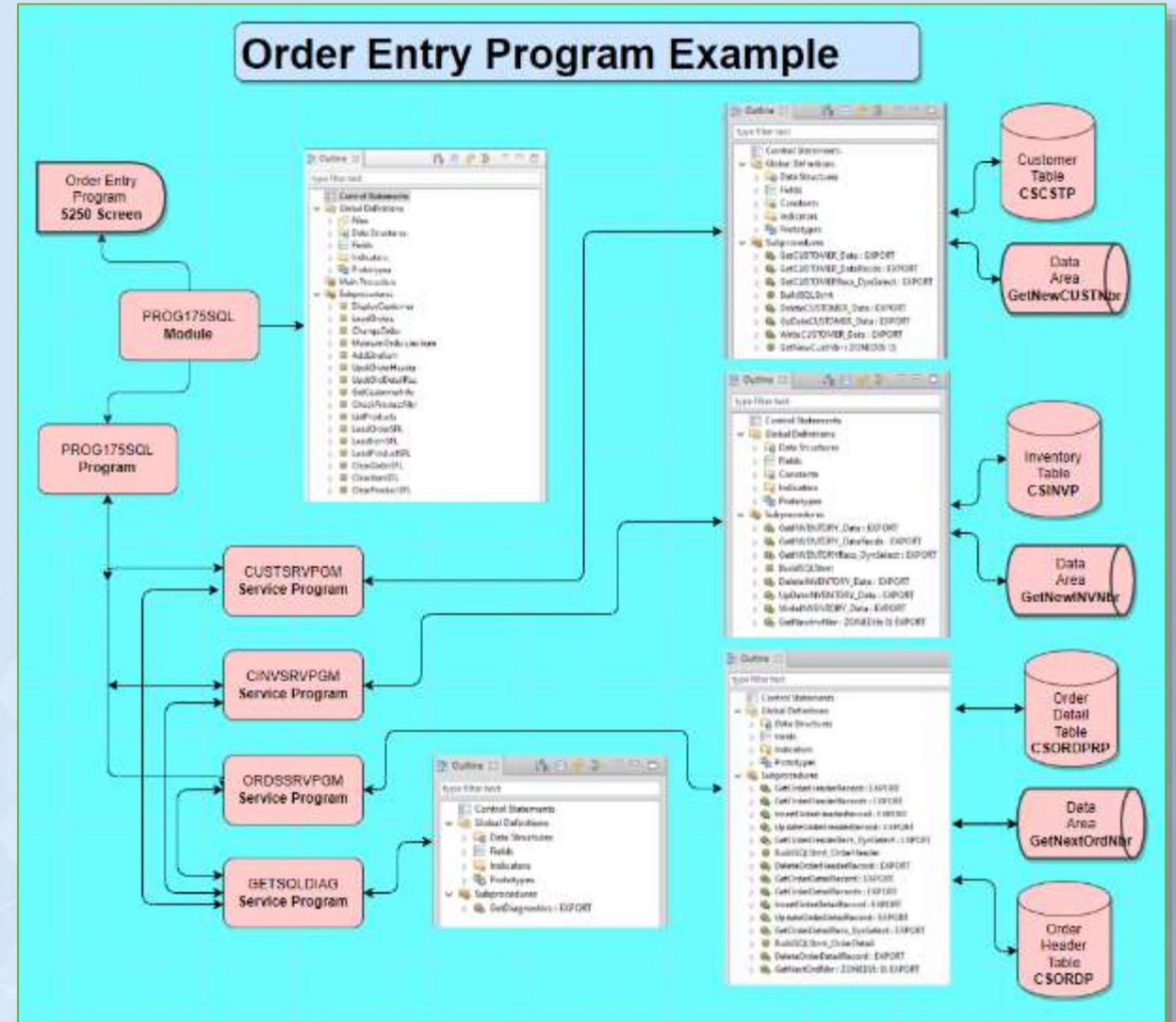
- Create The Module CINSVRPGM
- Create The Service Program CINSVRPGM

## Step 05

- Create The Module ORDSSVRPGM
- Create The Service Program ORDSSVRPGM

## Step 06

- Create the Executable PROG175SQL





# Create PROG175D Display File

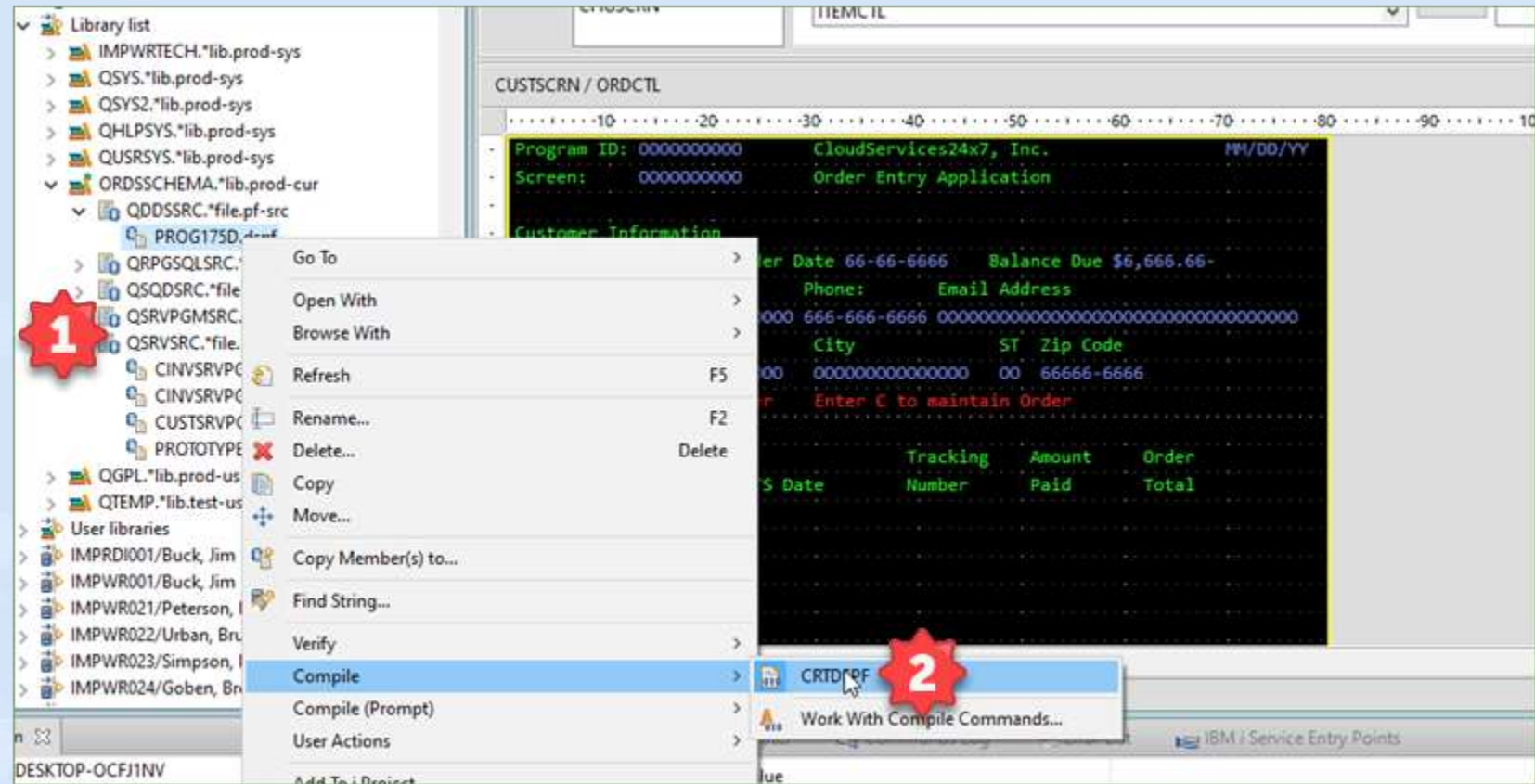
## Create the Display file

Right-click the program source member

1. Expand the Library>source file until you find the source member

2. Select the CRTDSPF

Always check that the compile worked



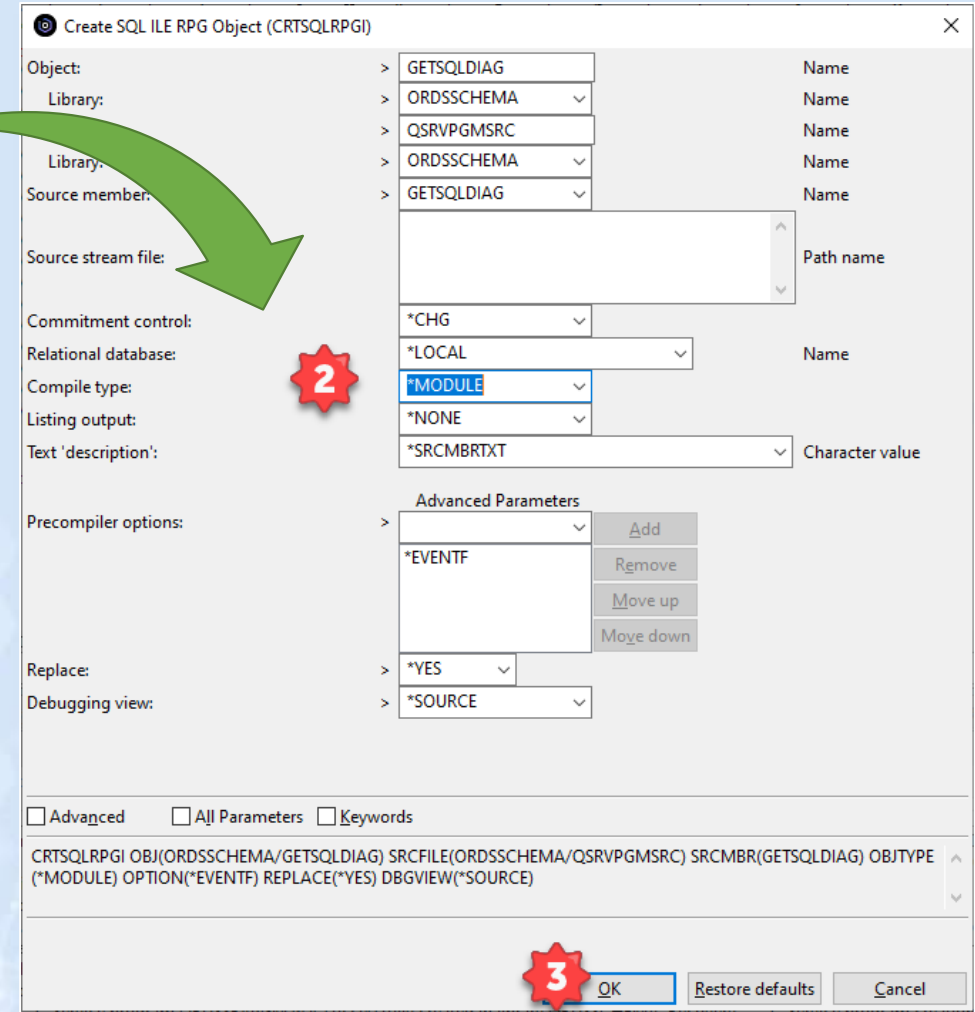
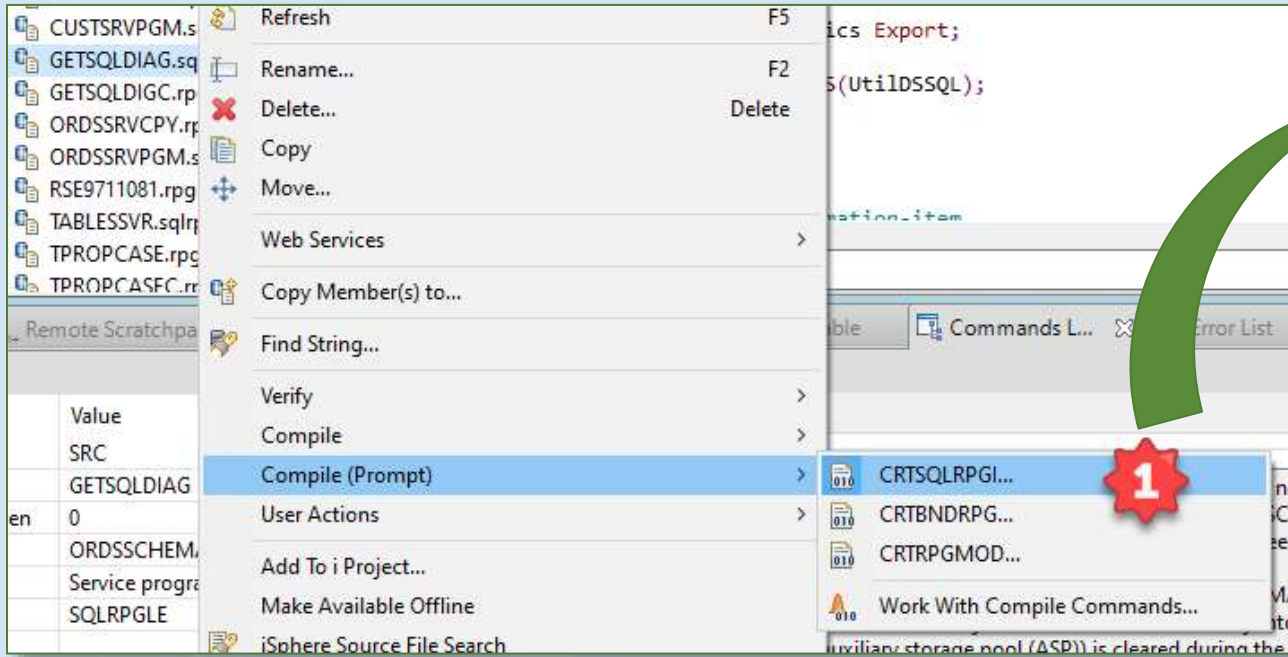


# Activation Group information

## Application ILE Information

- PROG175SQL – Will run in its own named **ORDERENTRY** activation group
- All Service programs will use \*CALLER for the ACTGRP parameter
- Binder Directory name is **ORDERSBIND**

# Create GETSQLDIAG Module

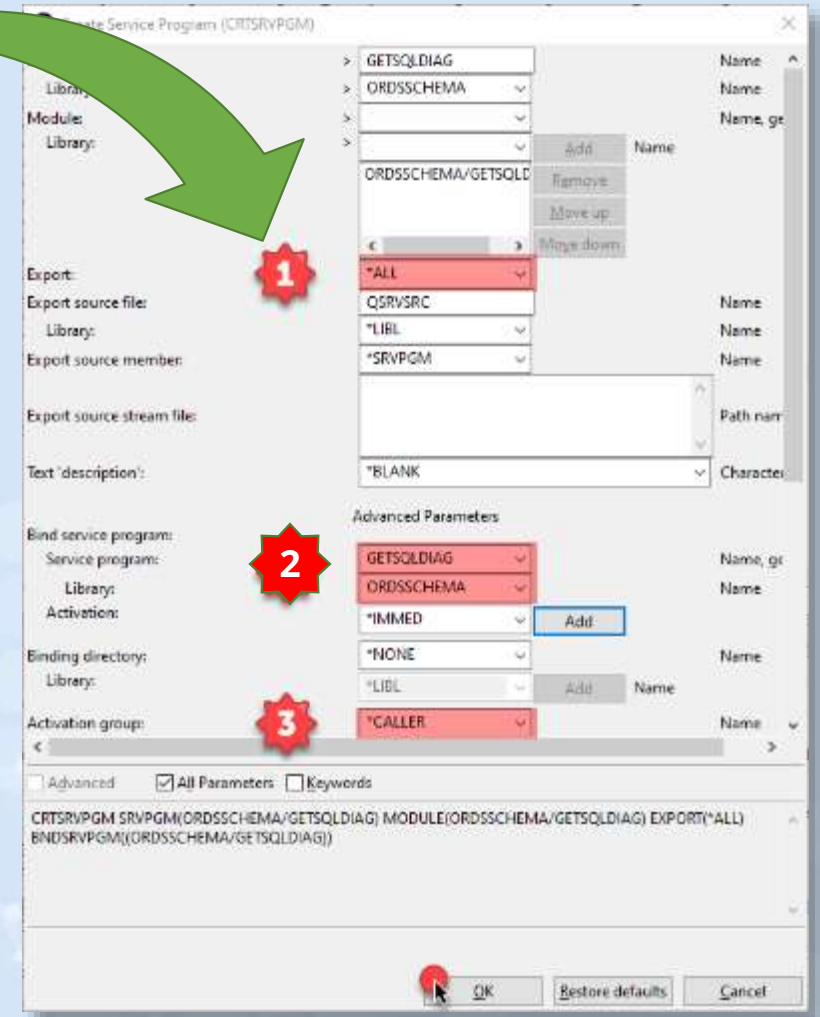
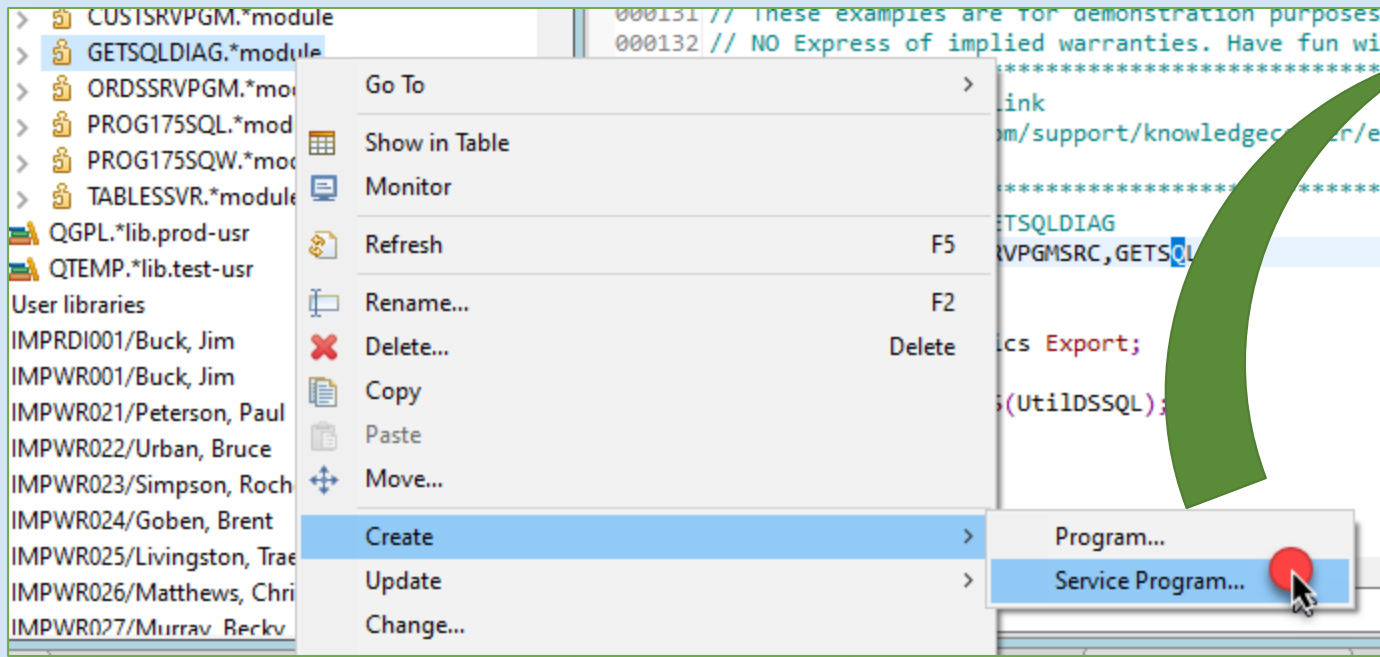


## Create GETSQLDIAG Module

- Right-click the program source member
1. Select the Compile (Prompt) > CRTSQLRPGI
2. Change Compile type to \*Module
3. Click on Ok

Always check that the compile worked under commands log view

# Create GETSQLDIAG Service Program

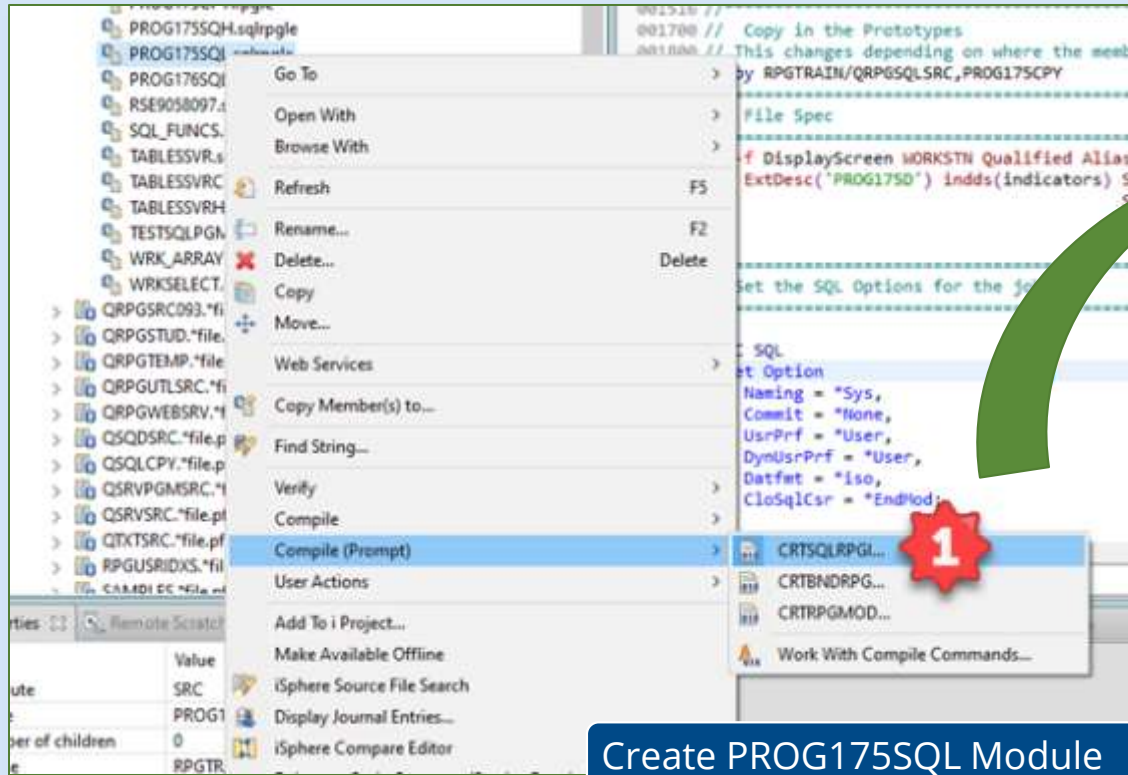


## Create GETSQLDIAG Service Program

Right-click the program module THEN Select Create -> Service program

1. Change export to \*ALL
  2. Add the **YourLib**/GETSQLDIAG Service program
  3. Use the \*CALLER to the ACTGRP parameter
- Click "Ok"

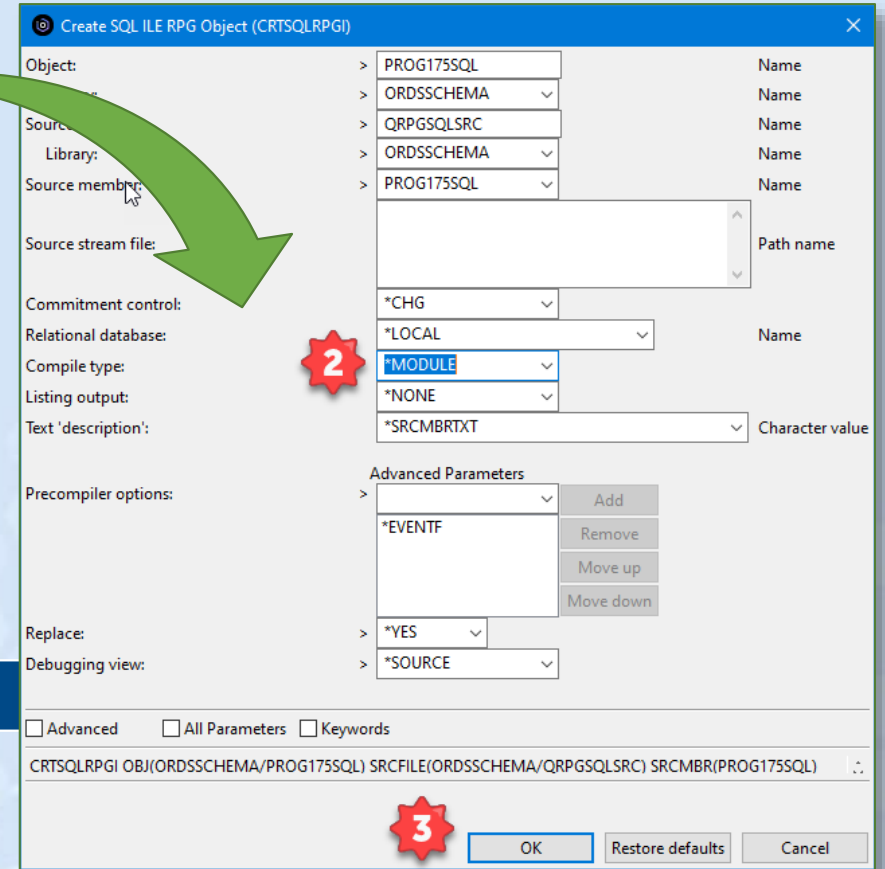
# Create PROG175SQL Module



## Create PROG175SQL Module

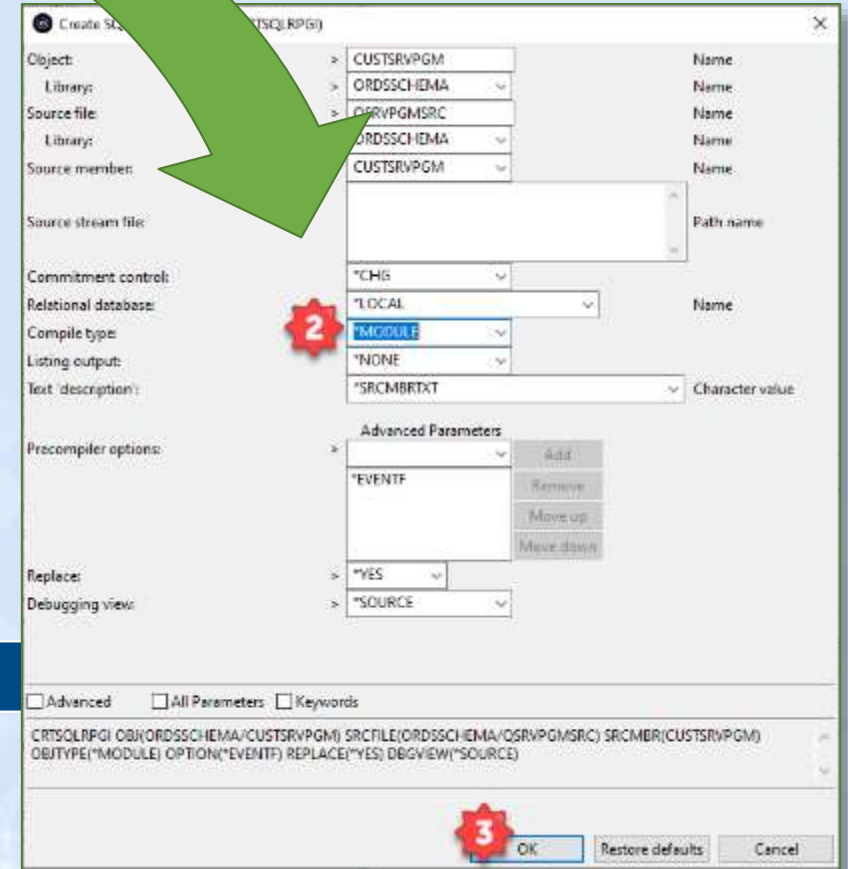
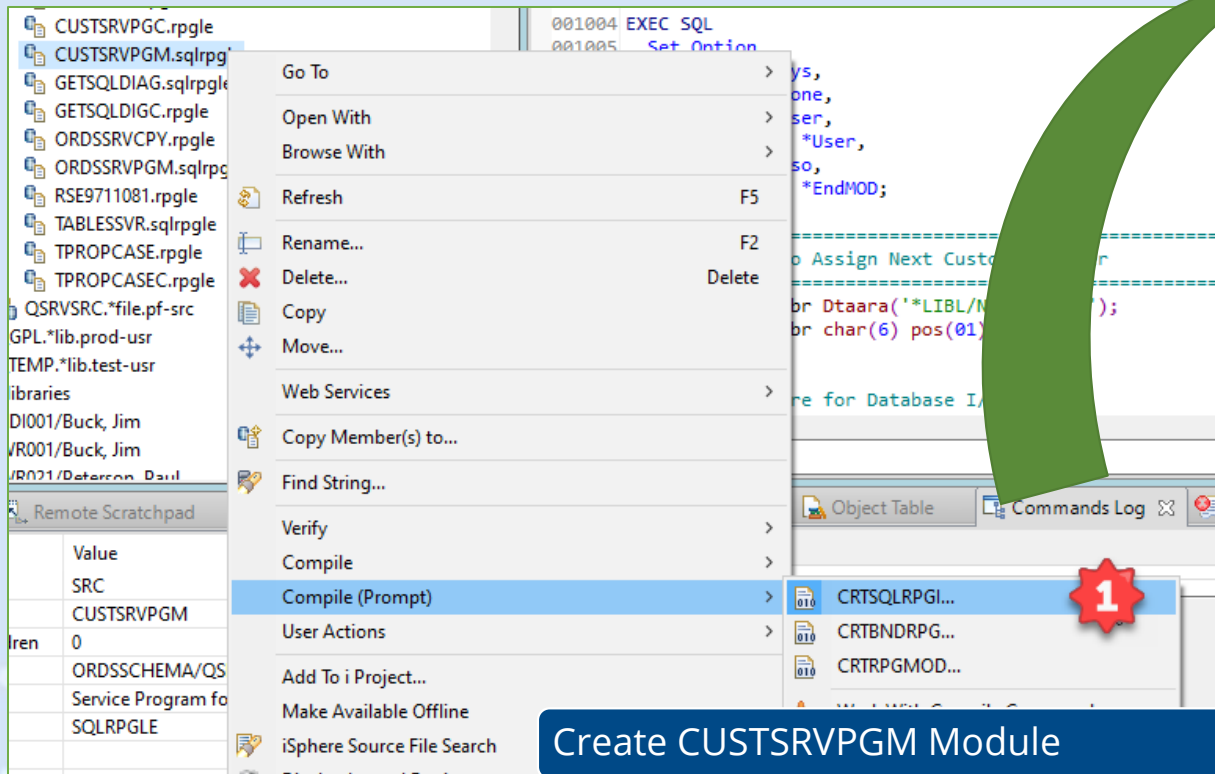
1. Right-click the program source member
1. Select the Compile (Prompt) > CRTSQLRPGI
2. Change Compile type to \*Module
3. Click on Ok

Always check that the compile worked under commands log view





# Create CUSTSRVPGM Module



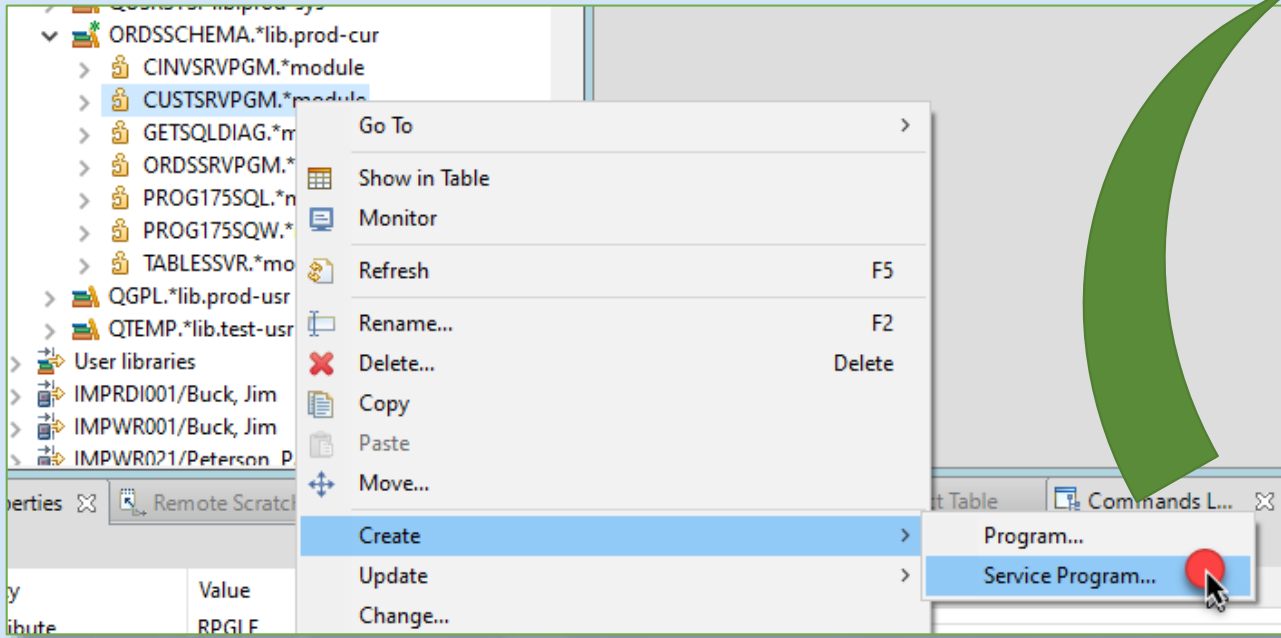
## Create CUSTSRVPGM Module

- Right-click the program source member
1. Select the Compile (Prompt) > CRTSQLRPGI
2. Change Compile type to \*Module
3. Click on Ok

Always check that the compile worked under commands log view



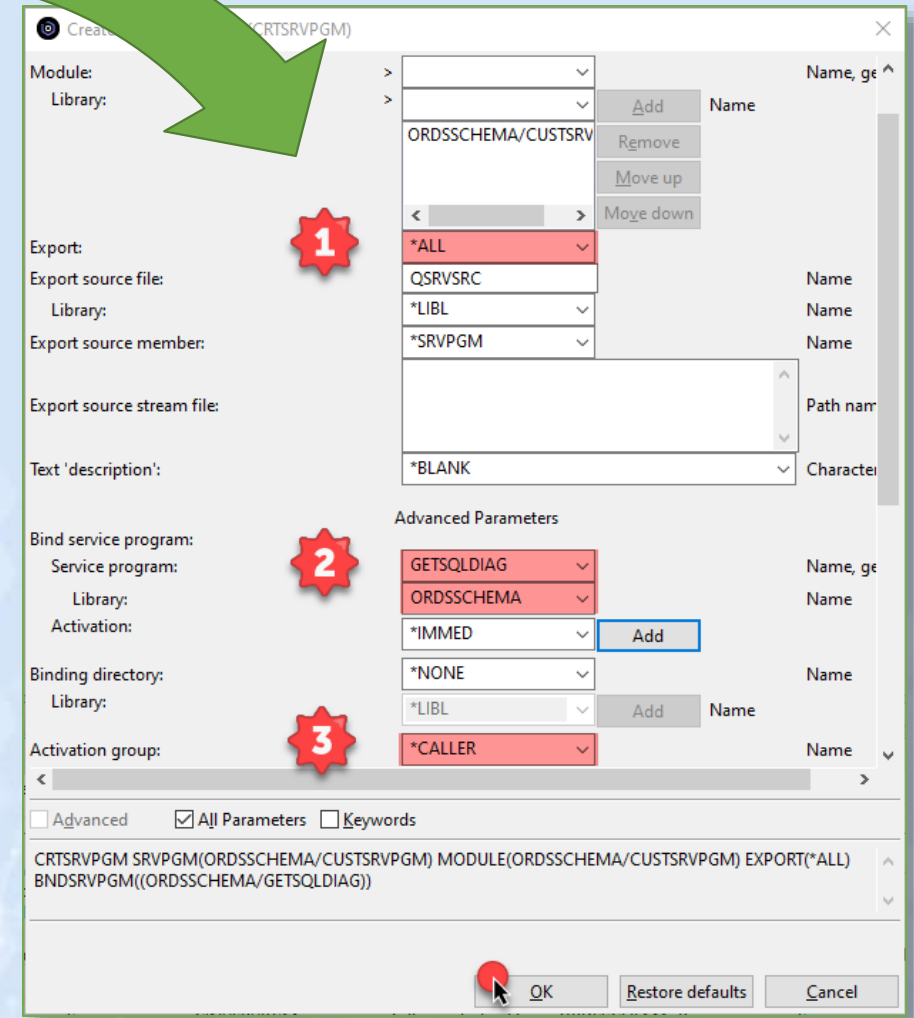
# Create CUSTSRVPGM Service Program



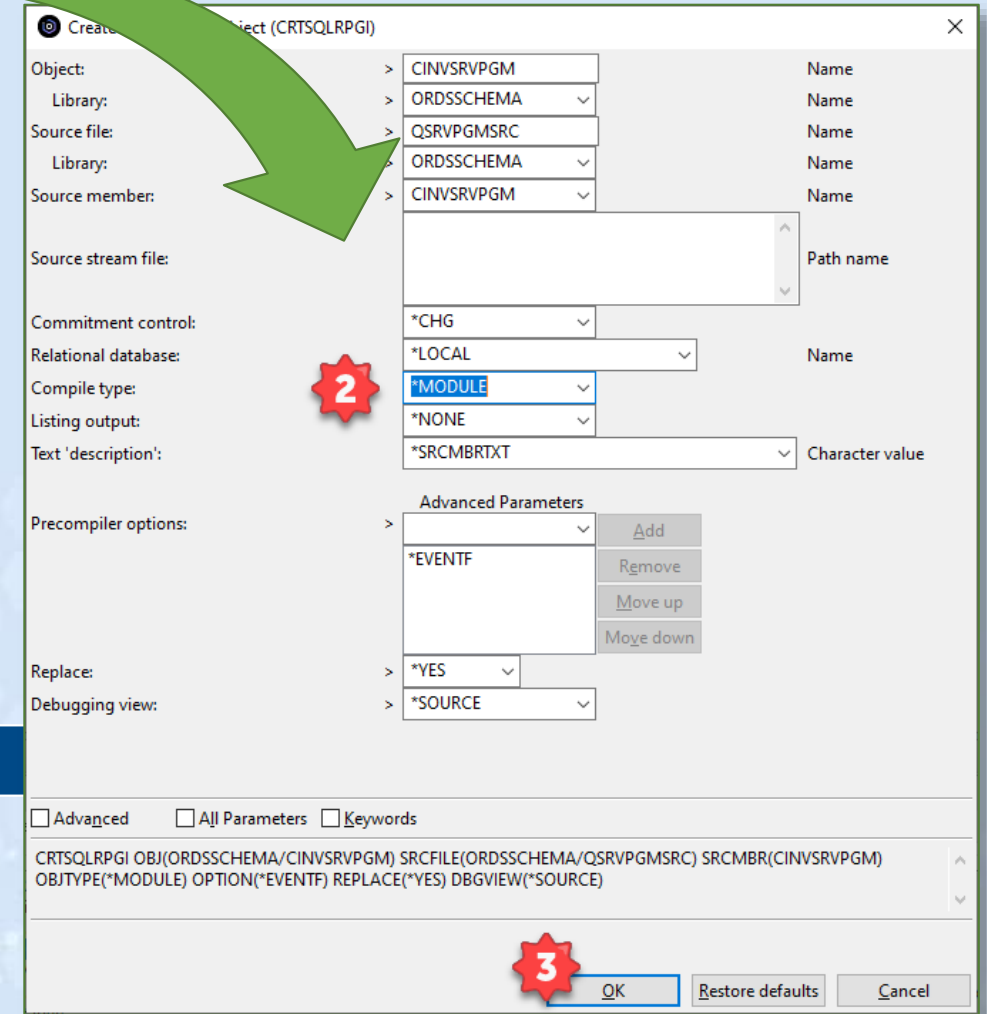
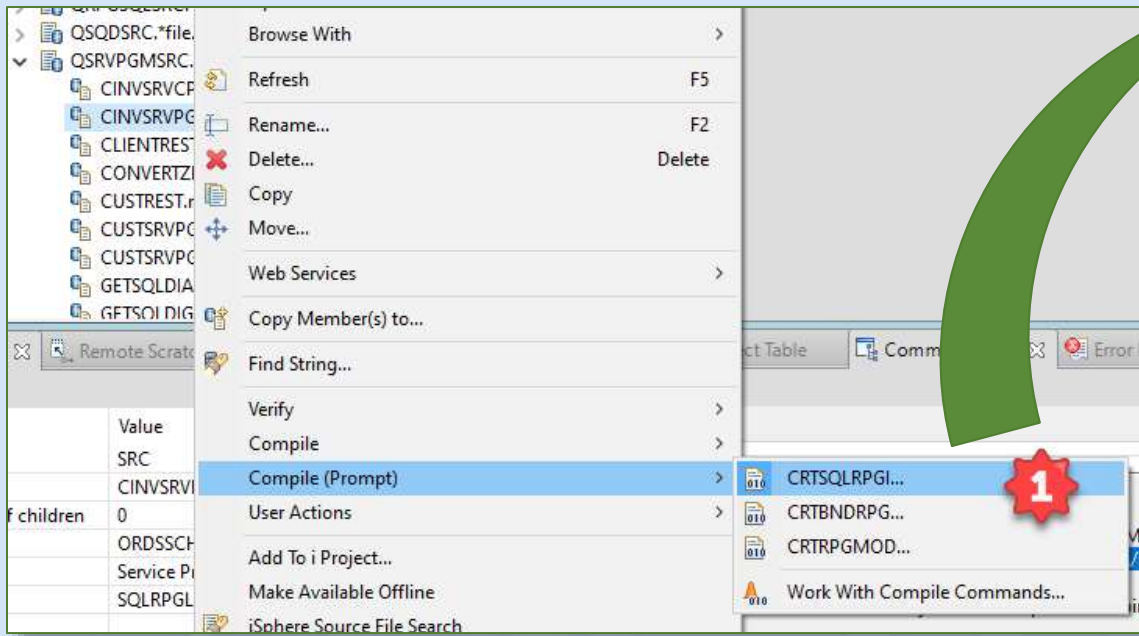
## Create CUSTSRVPGM Service Program

Right-click the program module THEN Select Create -> Service program

1. Change export to \*ALL
  2. Add the **YourLib**/GETSQLDIAG Service program
  3. Use the \*CALLER to the ACTGRP parameter
- Click "Ok"



# Create CINVSRVPGM Module



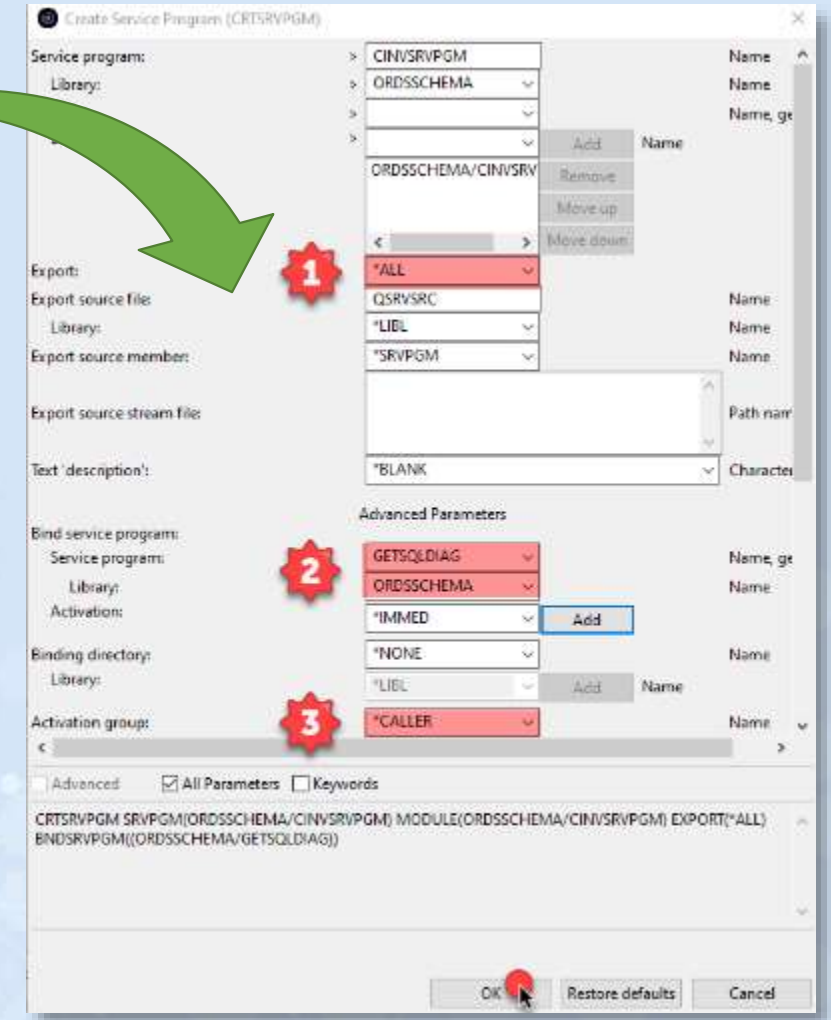
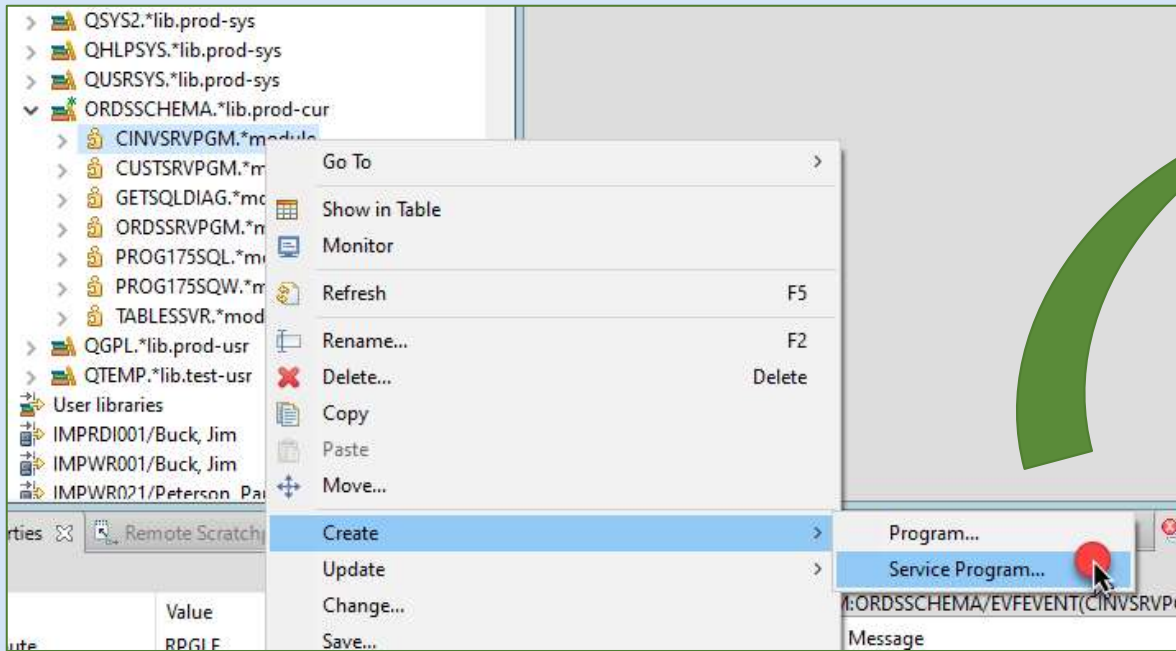
## Create CINVSRVPGM Module

Right-click the program source member

1. Select the Compile (Prompt) > CRTSQLRPGI
2. Change Compile type to \*MODULE
3. Click on Ok

Always check that the compile worked under commands log view

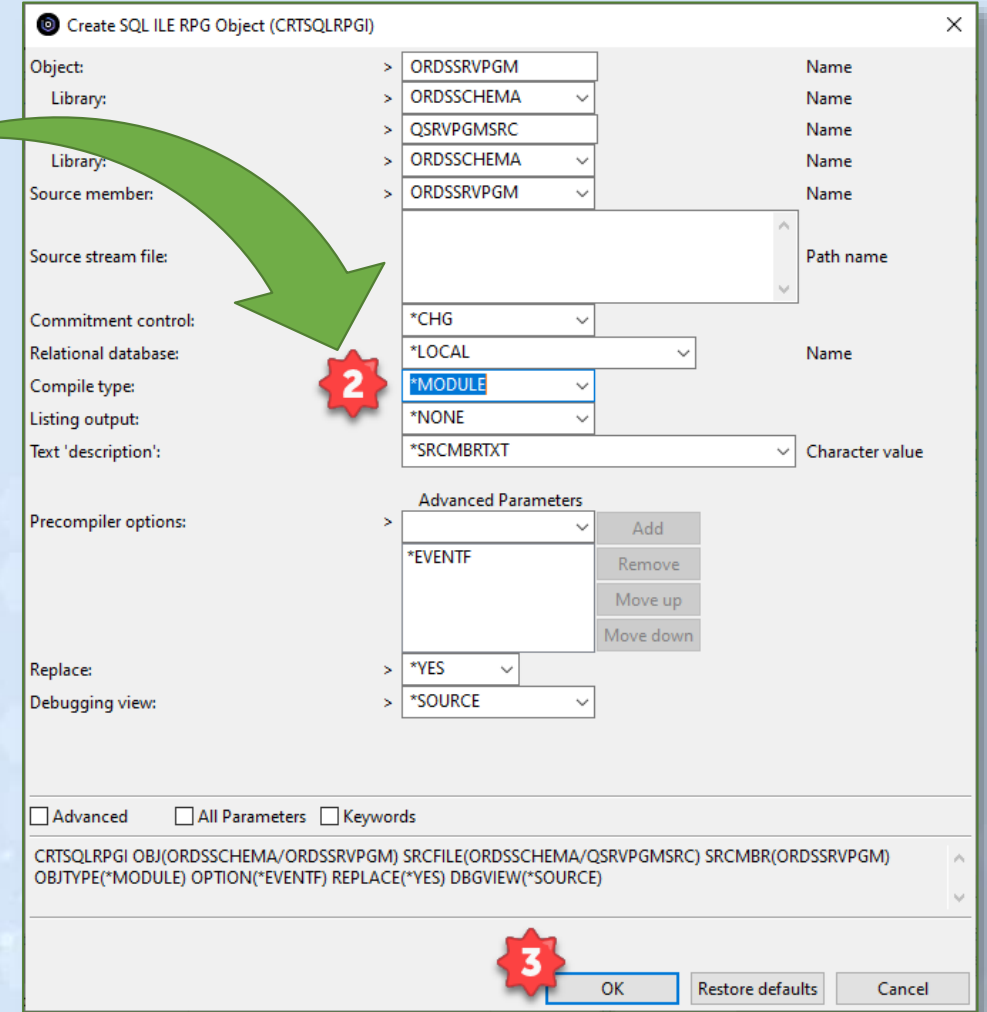
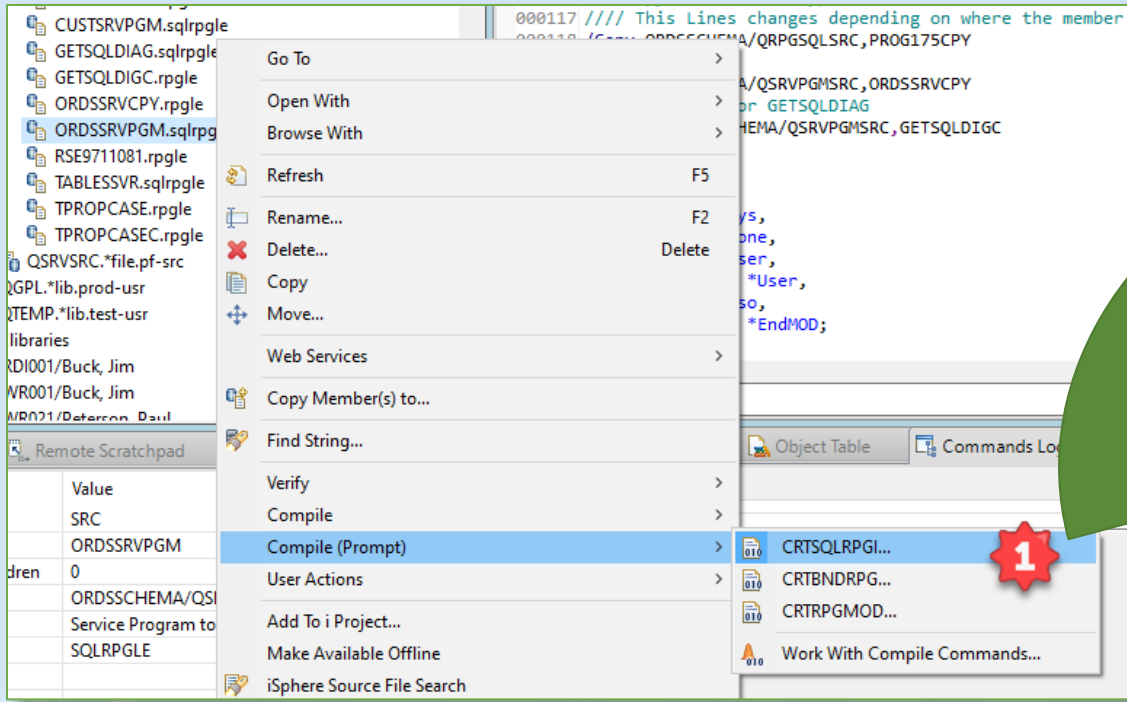
# Create CINVSRVPGM Service Program



## Create CINVSRVPGM Service Program

- Right-click the program module THEN Select Create -> Service program
- 1.Change export to \*ALL
- 2.Add the **YourLib**/GETSQLDIAG Service program
- 3.Use the \*CALLER IN the ACTGRP parameter
- Click "Ok"

# Create ORDSSRVPGM Module



## Create ORDSSRVPGM Module

Right-click the program source member

1. Select the Compile (Prompt) > CRTSQLRPGI

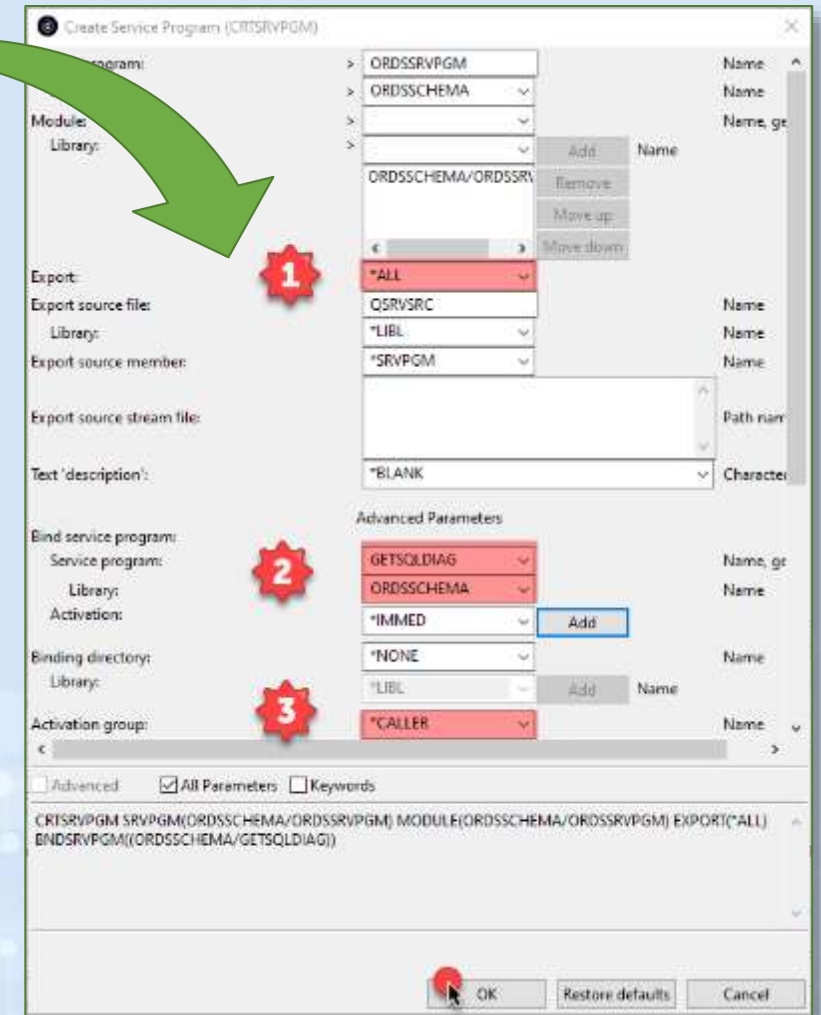
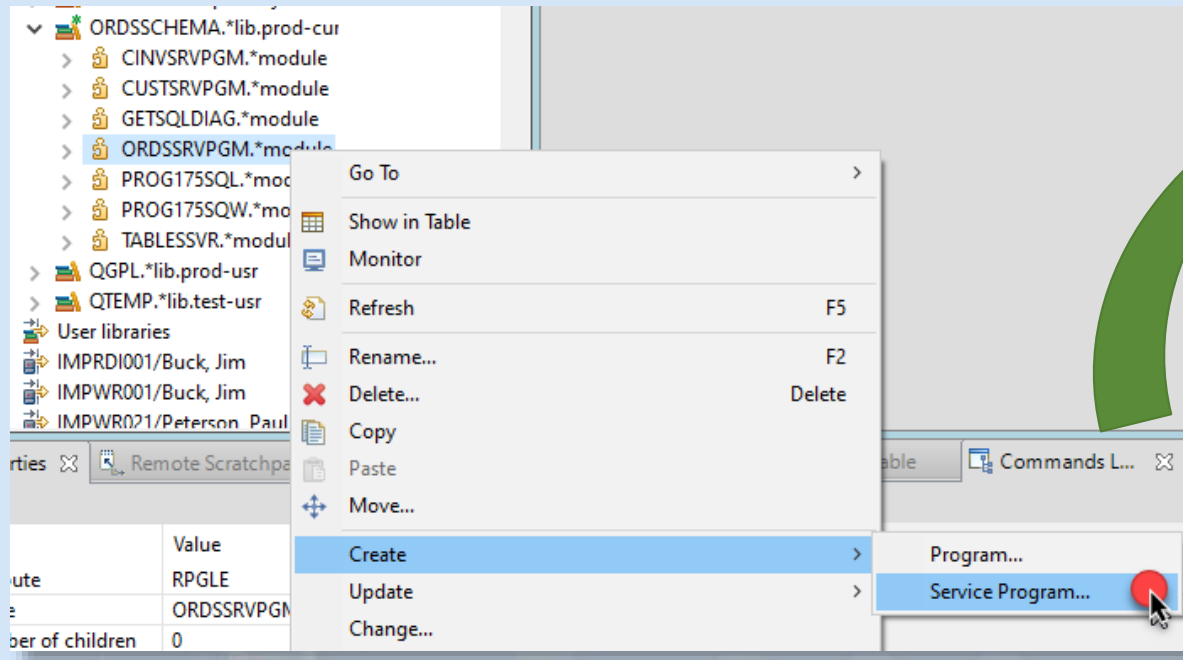
2. Change Compile type to \*Module

3. Click on Ok

Always check that the compile worked under commands log view



# Create ORDSSRVPGM Service Program



## Create ORDSSRVPGM Service Program

Right-click the program module THEN Select Create -> Service program

1. Change export to \*ALL
  2. Add the **YourLib**/GETSQLDIAG Service program
  3. Use the \*CALLER to the ACTGRP parameter
- Click "Ok"



# Create A RUNNABLE Application

## Use a Binding Directory

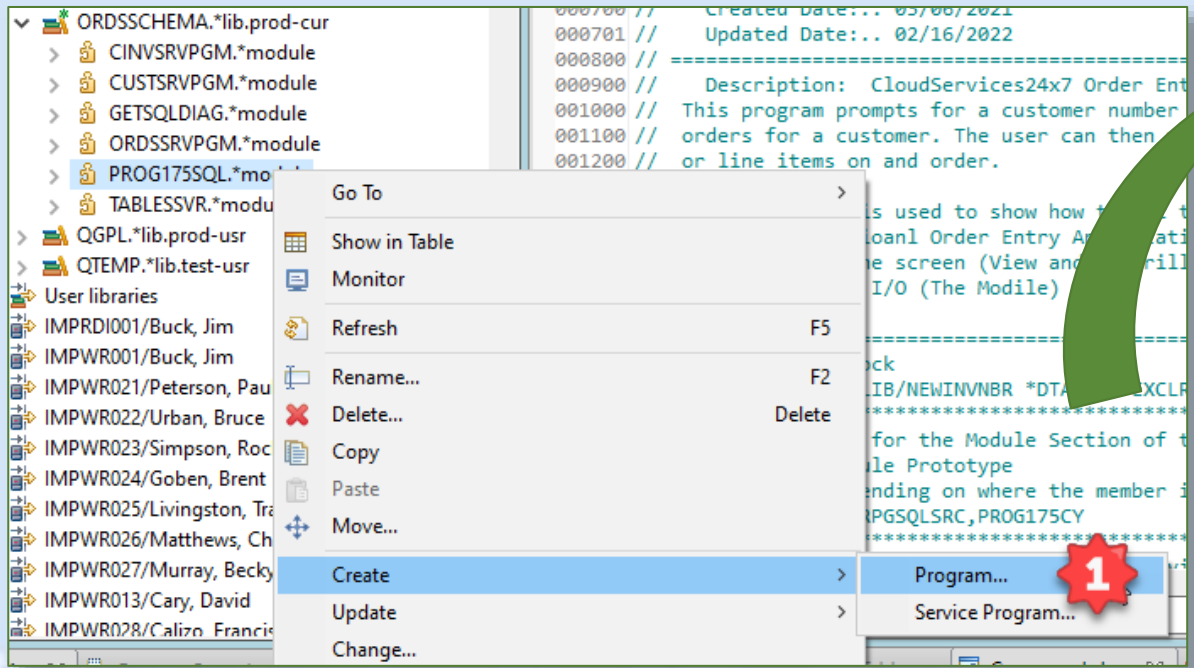
1. Expedites using Service Programs
2. Contains service programs with the required procedures for the Order Entry Application

```
ORDSSCHEMA/ORDERSBIND (*BNDDIR)  PROG175SQL.SQLRPGLE  X
Column 1      Column 1      Replace
.....1.....2.....3.....4.....5.....6.....7.....
000000  Opt Option(*NoDebugIO:*SrcStmt:*NoUnRef) alwnull(*usrctl);
000000  Ctrl-Opt bnmdir('ORDSSCHEMA/ORDERSBIND');
000300  // =====
000500  //   Created By:.... Jim Buck
000600  //   Program Name:.. PROG175SQL
000700  //   Created Date:.. 03/06/2021
000701  //   Updated Date:.. 02/16/2022
000800  //
```

The screenshot shows the IBM i development environment. On the left, a 'Library list' window displays a tree view of libraries, with 'ORDSSCHEMA.\*lib.prod-cur' selected and 'ORDERSBIND.\*bnmdir' highlighted. On the right, a table displays the contents of the 'ORDSSCHEMA/ORDERSBIND (\*BNDDIR)' library.

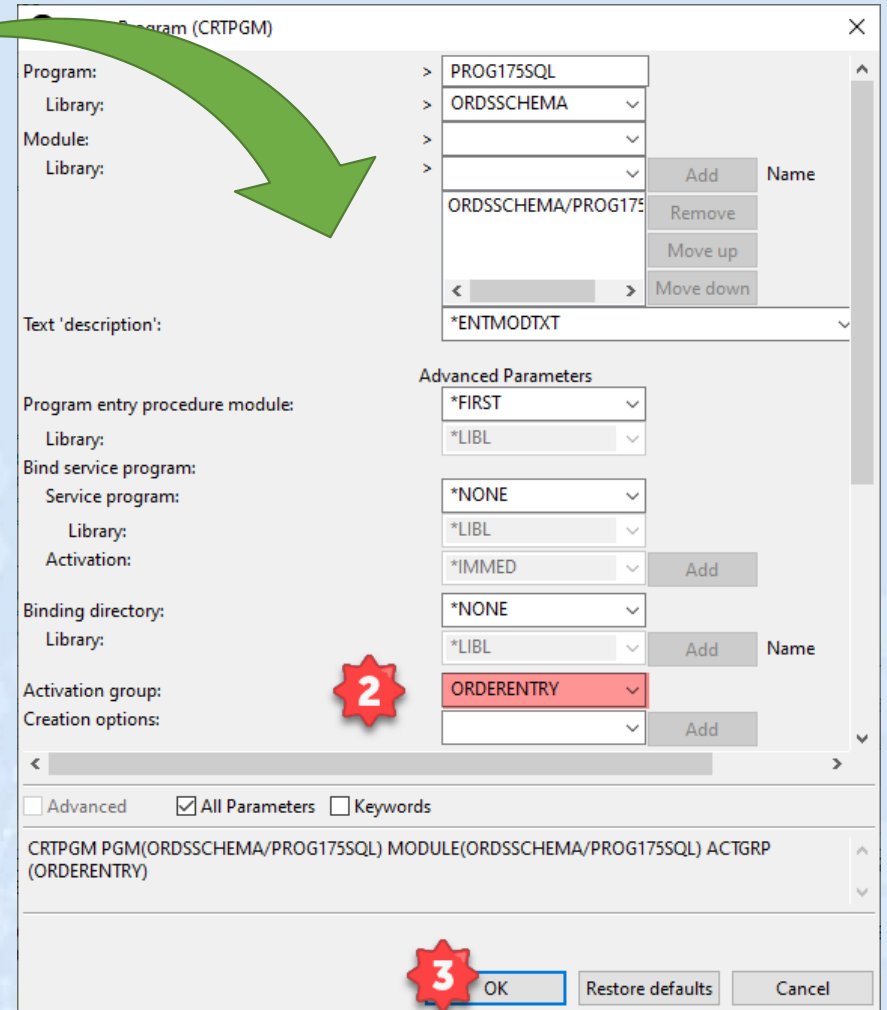
Library	Object	Object type	Activation
ORDSSCHEMA	ORDSSRVPGM	*SRVPGM	*IMMED
ORDSSCHEMA	CUSTSRVPGM	*SRVPGM	*IMMED
ORDSSCHEMA	CINVSrvPGM	*SRVPGM	*IMMED
ORDSSCHEMA	GETSQLDIAG	*SRVPGM	*IMMED

# Create A RUNNABLE Application



## Create PROG175SQL Program

- 1.Right-click the program module THEN Select Create -> Program
- 2.Change ACTGRP to ORDERENTRY
- 3.Click "Ok"



Participants

Badges

Competencies

Grades

IBM i Concepts and Operations

Getting Started

Section 01 - Communicating with the System

Section 02 - Using Control Language

Section 03 - IBM i Objects

Section 04 - Handling Spool Files

Section 05 - Describing a Database File & Logical Files

Section 06- Introduction to Access Client Solutions

# IBM i Concepts and Operations

Dashboard

My courses

IBM i Concepts

RPG Class

Participants

Badges

Competencies

Grades

Programming in RPG ILE

Start Here

Section 01 - Introduction to Programming - Getting Started

Section 02 - Files and Declarations

Section 03 - Controlling Program Workflow & Arithmetic Operations and Functions

Section 04 - File Operations and using SQL

Section 05 - Processing Character Data / Working with Dates

## Welcome!



Announcements



## Programming in RPG ILE

# Welcome!



Announcements

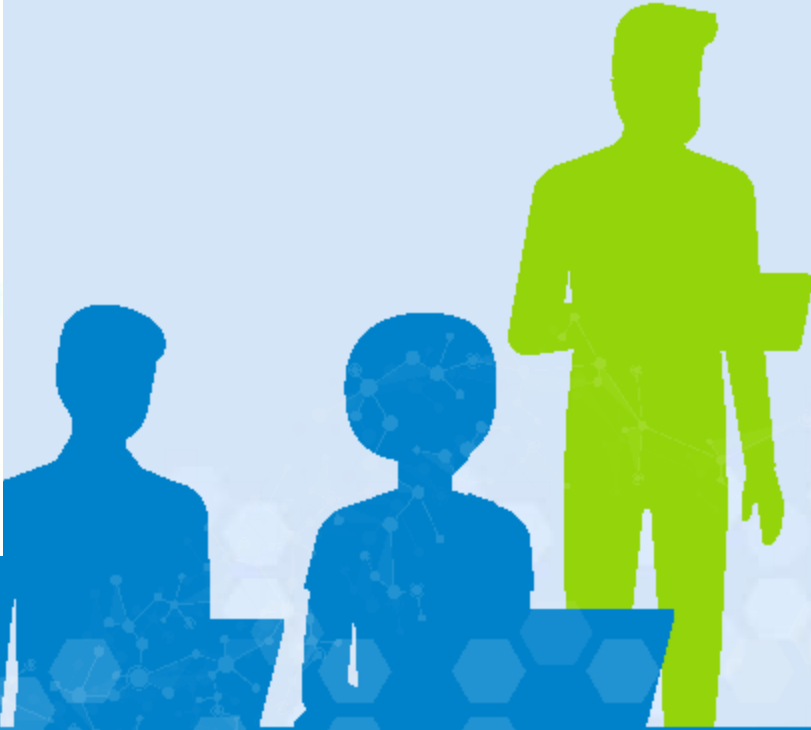
General news and announcements



**New or updated Offerings**  
RDi & Modular Programming  
SQL Queries Workshop



**Complete online Classes**  
Based on my textbooks  
Includes numerous Videos,  
Quizzes, Sample programs & and program assignments



Jim Buck  
Phone 262-705-2832  
[jbuck@impowertechnologies.com](mailto:jbuck@impowertechnologies.com)  
Twitter - @jbuck\_imPower  
[www.impowertechnologies.com](http://www.impowertechnologies.com)

# Any Questions or Comments?