MANUAL TRANSMITTAL

2.5.3

Department of the Treasury
Internal Revenue Service

AUGUST 4, 2023

EFFECTIVE DATE

(08-04-2023)

PURPOSE

(1) This transmits revised IRM 2.5.3, Systems Development, Programming and Source Code Standards.

MATERIAL CHANGES

(1) Changed “Signature” from “Nancy Sieger, Chief Information ”Officer to “Kaschit Pandya, Acting Chief Information Officer”

(2) 2.5.3.1.6, Changed title from “Acronyms and Terms” to “Terms and Acronyms”

(3) 2.5.3.1.7, Added Related Resources:

- Todd Baginski and Pat Dunn, Microsoft PowerApps Canvas App Coding Standards and Guidelines
- Kashif Salim Bawany, Microsoft MS Federal Cloud & Customer Experience Team. Governance for Power Platform: Management Models, November 01, 2021.
- Matthew Devaney, PowerApps Collections Cookbook: No Ads, No Fluff, Just Power Apps Stuff,
- Summit Bajracharya, PowerApps Performance: 40 Way to Improve Your App, *https://summitbajracharya.com.np/40-ways-to-improve-your-powerapps-performance/*, October 11, 2020
- Microsoft Training - Describe how to Build Applications with Microsoft Power Apps *Describe how to build applications with Microsoft Power Apps - Training | Microsoft Learn*.

(4) 2.5.3.3 (5), Added Power **Platform programming** in this paragraph.

(5) 2.5.3.12, Added subsection for Microsoft Power Platform

(6) Changed information in the following table:

IRM Subsections	Corrections
2.5.3.2 (2) 2.5.3.2.1.1 (3)f 2.5.3.3.8.1.2 (6)b 2.5.3.4.4 1 2.5.3.5.1 (2) 2.5.3.5.11.4 (6) 2.5.3.7.2.2 (2)d 2.5.3.7.3.6 (2) 2.5.3.9.8 (2)	Added a comma after ‘e.g.’.
2.5.3.3.8.1.3 (1)b	Added space between ‘i.e.,’ and ‘network’
2.5.3.10.2.3 (1)	Removed extra space and comma
Exhibit 2.5.3-18	Added a comma after ‘e.g.’ for Refactor.

Any line marked with a #
is for **Official Use Only**

(7) Changed the word “section ”to “subsection” to ensure the correct terminology is referenced. for the following:

IRM Subsections	Corrections

2.5.3.2.1.1(1)
2.5.3.2.1.2(1)
2.5.3.2.1.3(1)
2.5.3.3(1)
(2), 2.5.3.3.5(1)
2.5.3.3.8.1(1)
2.5.3.4.2(1)
2.5.3.4.3.1(1)
2.5.3.5(1)
2.5.3.7(1)
2.5.3.7.3.9(1)
2.5.3.8(1)
2.5.3.9(1)
2.5.3.9.9(1)
2.5.3.9.9.12(12)
2.5.3.9.9.13.8(1)
2.5.3.10.2.1(1)
	Changed the word “section” to “subsection” to ensure the correct terminology is referenced.

(8) 2.5.3.12.1 (1),(2), Added subsection for Microsoft Power Platform Overview and corrected spelling of business login″ with ″business logic″.

(9) 2.5.3.12.1 (3), Added ″see image in Figure 2.5.3-93 because the sentence was incomplete.

(10) 2.5.3.12.1 (4) a, Changed personalized reports for mobile devises and desktops.″

(11) 2.5.3.12.2, Added subsection for IRS Application Lifecycle Management and Power Platform and changed the words ″being about to drag″ to ″being able to drag″ and change the words ″business login″ to ″business logic″. for more sentence clarity

(12) 2.5.3.12.2.1, Added subsection for IRS Application Lifecycle Management and Power Platform Overview

(13) 2.5.3.12.1 (5) d, Subscriptions - Created a table for Power apps subscriptions

(14) 2.5.3.12.1 (6), “Changed Power Automate Maker Portal” to “Power Apps Maker Portal” because it was not the correct component name.

(15) 2.5.3.12.2.1.1, Added subsection for IRS Microsoft 365 (M365) Power Platform Dedicated Environment Requests Process

(16) 2.5.3.12.2.1.1 (1), Change to “Use consistent naming conventions for variables, collections and objects to ensure application makers or developers can discern the type, purpose, and scope of each. See examples in Figure 2.5.3-91 and Figure 2.5.3-92” for more clarity.

(17) 2.5.3.12.2.2, Added subsection for IRS Application Development with Power Apps.

(18) 2.5.3.12.2.2.1, Added subsection for IRS Canvas PowerApps Coding Standards Best Practices.

(19) 2.5.3.12.2.2.1.1, Added subsection for PowerApps Apps Code Naming Conventions.

(20) 2.5.3.12.2.2.1.2, Added subsection for Microsoft Power Apps for Canvas Apps: Comments.

(21) 2.5.3.12.3, Removed subsection for Microsoft Power Platform Apps Connectors Overview because the information is included in subsection 2.5.3.2.

(22) 2.5.3.12.3.1, Added subsection for Power Platform Apps Coding Standards.

(23) 2.5.3.12.3.2, Added subsection for Power Platform Apps - Best Practices for Performance Optimization.

(24) 2.5.3.12.4, Added IRS Power Automate Flow Coding Standards Best Practices.

(25) Removed Figure 2.5.3-84, Power Business Intelligence (BI) Suite of Tools table and added to IRM 2.5.11 because information was not relevant to this IRM.

(26) Added Figure 2.5.3-85, Power Apps: Building Blocks for Low Code Development table.

(27) Added Figure 2.5.3- 86, IRS Power Subscriptions image.

Power Automate: Framework of Flows

Component	Descriptions
Power Automate Maker Portal	A web browser-based tool used by a developer to create Power Automate Cloud Flows. This tool is hosted in Azure as part of the Power Platform service.
Cloud Flow	A cloud flow works against a cloud data source. Cloud flows come in three basic types: a. Automated Cloud: Flows are triggered by an event (such as email arrival or item change). b. Instant Cloud Flow: This is triggered on a user interface (such as a button) c. Scheduled Cloud Flow: This is triggered based on a pre-defined schedule or time interval.
Power Automate Desktop	A client application used by a developer to create Power Automate Desktop Flows. This tool can store flows in Azure, but can also store the flow as a definition file.
Desktop Flow	A desktop flow automates workstation-based tasks and can combine both local and cloud data sources.
Connectors	Connectors, as described in 2.5.3.12.1 (4) d) above, allow the developer to connect to data sources from within the flow.

(28) Added Figure 2.5.3- 87, Power Virtual Agents (PVA) - Intelligent Virtual Agent table.

(29) Added Figure 2.5.3-88, Dataverse Data Collection Process table.

(30) Added Figure 2.5.3-89, Dataverse Data Collection Process graphic

(31) Added Figure 2.5.3-90, IRS and M365 Comparison of Available Environments table.

(32) Added Figure 2.5.3-91, PowerApps Canvas App Code Naming Conventions table.

(33) Added Figure 2.5.3-92, PowerApps Canvas App Object Naming Conventions.

(34) Added Figure 2.5.3-93, Microsoft Power Apps for Canvas Apps: Comments table.

(35) Added Figure 2.5.3- 94, Apps Home Screen Names graphic.

(36) Added Figure 2.5.3- 95, Naming Convention for Screen Names

(37) Added Exhibit 2.5.3-25, IRS Power Platform Available Environments table.

(38) Changed Exhibit 2.5.3-25, Row: Business Unit Environments - Sandbox Column: Environment Description Item c, current verbiage was a duplicate of item b. Replaced with ″Same restrictions as production Business Unit Environment ″.

EFFECT ON OTHER DOCUMENTS

IRM 2.5.3 dated July 20, 2022, is superseded.

AUDIENCE

The audience intended for this transmittal is personnel responsible for engineering, developing, or maintaining Agency software systems identified in the Enterprise Architecture. This engineering, development, and maintenance includes work performed by IRS management, Information Technology government employees and contractors.

Kaschit Pandya
Acting, Chief Information Officer

2.5.3
Programming and Source Code Standards

Table of Contents

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

Exhibits

#

#

#

2.5.3.1
(08-04-2023)
Program Scope and Objectives

(1) **Purpose**: This Internal Revenue Manual (IRM) establishes standards and guidelines to promote the development of maintainable, portable, reliable software applications in all service used and approved programming languages as outlined in this IRM.

(2) **Audience**: This guidance applies to all IRS senior leadership, Information Technology (IT) managers at all levels. Also included are personnel responsible for: engineering, developing, or maintaining agency software systems identified in the Enterprise Architecture. This engineering, development, and maintenance include services performed by both government employees and contracts.

(3) **Policy Owner**: The current policy owner is the Chief Information Officer (CIO).

(4) **Program Owner**: The current program owner is the Director, Technical Integration Organization (TIO).

(5) Primary stakeholders:

a. Application Development (AD) - AD staff, management, and contractual employees
b. Developers - government and contractual employees
c. Engineers - government and contractual employees
d. IRS IT managers
e. Quality Assurance (QA) - IRS QA staff, managers, and contractual employees

2.5.3.1.1
(08-04-2023)
Background

(1) In response to the June 2018 Government Accountability Office report **(GAO-18-298) Information Technology (IT): IRS Needs to Address Significant Risks to Tax Processing- Investments Performance and Risks**, IRS Information Technology (IT) is continuously improving the performance of IRS Major IT Investments which includes IRS mainframe systems, using legacy programming languages: Common Business Oriented Language (COBOL), Assembler Language Code (ALC), Java programming, etc.

(2) IRS AD has established this IRM as well as risk management strategies and guidance to identify, analyze, mitigate, monitor risks and provide quality performance with system development standards.

2.5.3.1.2
(08-04-2023)
Authority

(1) Treasury Directive 85-1, Department of the Treasury Information Technology (IT) Security Program

(2) IRM 10.8.1, Information Technology (IT) Security, Policy and Guidance

(3) IRM 10.8.6, Information Technology (IT) Security, Application Security and Development

(4) NIST 800-115, Technical Guide to Information Security Testing and Assessment

(5) NIST 800-53 Revision 5: Security and Privacy Controls for information Systems and Organizations

(6) IRM 2.5.1 System Development, establishes the System Development program for the IRS

(7) Government Accountability Office, (GAO)

(8) Treasury Inspector General Tax Administration (TIGTA)

(9) Presidential American Technology Council, 2017

(10) Administrator of the General Services Administration (GSA)

(11) Federal Information Security Modernization Act (FISMA) of 2014

(12) Director of Office of Management and Budget (OMB)

(13) Secretary of the Department of Homeland Security (DHS)

(14) Secretary of Commerce for Modernization of Federal IT

(15) Federal Information Processing Standards (FIPS) Pub 73, Guidelines for Security of Computer Applications

(16) 21st Century Integrated Digital Experience Act (IDEA), December 2018

(17) Federal Risk and Authorization Management Program (FedRAMP), 2011

2.5.3.1.3 (08-04-2023) Roles and Responsibilities

(1) **Application Development's chain of command** include:

a. **Commissioner**: Oversees and provides overall strategic direction for the IRS. The Commissioner's and Deputy Commissioner's main focus is for the IRS's services programs, enforcement, operations support, and organizations. Additionally, the Commissioner's vision is to enhance services for the nation's taxpayers, balancing appropriate enforcement of the nation's tax laws while respecting taxpayers' rights.

b. **Deputy Commissioner, Operation Support (DCOS)**: Oversees the operations of Agency-Wide Shared Services: Chief Financial Officer, Human Capital Office, Information Technology, Planning Programming and Audit Oversight and Privacy, and Governmental Liaison and Disclosure.

c. **Chief Information Officer (CIO)**: The CIO leads Information Technology, advises the Commissioner on Information Technology matters, manages all IRS IT resources, and is responsible for delivering and maintaining modernized information systems throughout the IRS. The Deputy Chief Information Officer for Operations assists the Chief Technology Officer (CTO).

d. **Application Development (AD), Associate Chief Information Officer (ACIO)**: The AD ACIO reports directly to the CIO; oversees and ensures the quality of: building, unit testing, delivering, and maintaining integrated enterprise-wide applications systems to support modernized and legacy systems in the production environment to achieve the mission of the service.

e. **AD Deputy CIO (DCIO)**: The AD Deputy ACIO reports directly to the AD ACIO and is responsible for:
- Leading all strategic priorities to enable the AD Vision, IT Technology Roadmap and the IRS future state
- Executive planning, and management of the development organization which ensures all filing season programs are developed, tested, and delivered on-time and within budget

(2) **Application Development**: Responsible for building, testing, delivering, and maintaining integrated information applications systems, e.g., software

solutions, to support modernized systems and production environment to achieve the mission and objectives of the service. Additionally, AD does the following:

a. Works in partnership with customers to improve the quality of the IRS information systems, products, and services.
b. Maintains the effectiveness and enhance the integration of IRS installed base production systems and infrastructure while modernizing core business systems and infrastructure.
c. Establishes and maintains rigorous contract and fiscal management, oversight, quality assurance, and program risk management processes to ensure that strategic plans and priorities are being met.
d. Provides quality assessment/assurance of deliverables and processes.
e. Creates oversight support of enterprise modernization goals in coordination with Information Technology HR initiatives and policy.
f. Responsible for delivering filing season projects, and implementing Economic Stimulus changes.
g. AD has the following Domains:
    - Compliance
    - Corporate Data (CD)
    - Customer Service (CS)
    - Data Delivery Service (DDS)
    - Delivery Management; Quality Assurance (DMQA)
    - Identity & Access Management (IAM)
    - Internal Management (IA)
    - Submission Processing (SP)
    - Technical Integration Organization (TIO)

(3) **Director, Compliance**: Provides executive direction for a wide suite of Compliance domain focused applications and oversee the IT Software Development organization to ensure the quality of production ready applications.

a. Directs and oversees a unified cross-divisional approach to compliance strategies needing collaboration pertaining to the following:

- Abusive tax avoidance transactions needing a coordinated response
- Cross-divisional technical issues
- Emerging issues
- Service-wide operational procedures

(4) **Director, AD Corporate Data**: Directs and oversees the provisioning of authoritative databases, refund identification, notice generation, and reporting.

(5) **Director, Customer Service**: Directs and oversees Customer Service Support for service and communication with internal and external customers and providing taxpayers with self-service online capabilities.

a. Customer Service Domain’s applications and systems provide:
    - Tax law assistance
    - Taxpayer education
    - Access to taxpayer account data
    - Maintenance of modernized information systems that meet the customer’s needs for researching, updating, analyzing, and managing taxpayer accounts
b. Services to internal and external customers are provided through five primary means:

- Centralized Contact Centers (for telephone, written, and electronic inquiries)
- Self-service applications (via the telephone and Internet)
- Field Assistance (for walk-in assistance)
- Web Services
- Management of Taxpayer Accounts

(6) **Director, Data Delivery Services**: Oversees and ensures the quality of data with repeatable processes in a scalable environment. The Enterprise Data Strategy is to transform DDS into a data centric organization dedicated to deliver Data as a Service (DaaS) through:

- Innovation - new methods, discoveries
- Renovation - streamline or automate
- Motivation - incent and enable individuals

(7) **Director, Delivery Management & Quality Assurance (DMQA)**:

- Executes the mission of DMQA by ensuring AD has a coordinated, cross-domain, and cross-organizational approach to delivering AD systems and software applications
- Reports to the AD ACIO and chairs the AD Risk Review Board.
- Chairperson, Configuration Control Board, see IRM 2.5.1.2.2.2
- Government Sponsor, Configuration Control Board, see IRM 2.5.1.2.2.2

(8) **Director, Identity & Access Management (IAM) Organization**: Provides oversight and direction for continual secure online interaction by verifying and establishing an individual’s identity before providing access to taxpayer information “identity proofing” while staying compliant within federal security requirements.

(9) **Director, Internal Management**: Provides oversight for the builds, tests, deliveries, refund identification, notice generation, and reporting.

(10) **Director, Submission Processing**: Provides oversight to an organization of over 17,000 employees, comprised of: a headquarters staff responsible for developing program policies and procedures, five W&I processing centers, and seven commercially operated lockbox banks. Responsible for the processing of more than 202 million individual and business tax returns.

(11) **Director, Technical Integration Office**: Provides strategic technical organization oversight ensuring applicable guidance, collaboration, and consolidation of technical integration issues and quality assurance for the Applications Development portfolio.

(12) **Information Technology (IT), Cybersecurity**: Cybersecurity manages the IRS IT Security program in accordance with the Federal Information Security Modernization Act of 2014 (FISMA) with the goal of delivering effective and professional customer service to business units and support functions within the IRS. These procedures are done as the following:

a. Provide valid risk mitigated solutions to security inquisitions.
b. Respond to incidents quickly and effectively in order to eliminate risks/threats.
c. Ensure all IT security policies and procedures are actively developed and updated.

d. Provide security advice to IRS constituents and monitor IRS robust security program for any required modifications or enhancements.

2.5.3.1.4
(08-04-2023)
Program Management and Review

(1) The Enterprise Program Management Office (EPMO) is responsible for the delivery of integrated solutions for several of the IRS's large, scaled programs. EPMO plays a key role in establishing change, configuration, release plans, and implementing new information system functional capabilities.

(2) The EPMO is the primary partner with the business for programs under their purview and collaborates with IT delivery partners (AD, ES, EOps, and the other ACIO areas) to deliver required capabilities. This structure positions each organization to maintain a strong core function to optimize their operations.

2.5.3.1.5
(08-04-2023)
Program Controls

(1) The Enterprise Program Controls (EPC) Office is the lead for EPMO Information Technology enterprise-wide program management functions and assists with the Applications Development (AD) organization in cross domain support for a variety of program management disciplines. The EPC office is comprised of seven sections: Business Operations, Program Support Services, Investment & Contract Management, Program Oversight & Reporting, Communications & Organization Readiness, Enterprise Transition Management Office, and Technical Integration.

(2) The controls established in this Internal Revenue Manual (IRM) apply to Service personnel responsible for developing or maintaining the Service's application systems or software applications, identified in the IRS Enterprise Architecture. Service personnel who contract for development or maintenance of these systems/software applications must ensure contracts comply with these controls.

2.5.3.1.6
(08-04-2023)
Terms and Acronyms

(1) See Exhibit 2.5.3-17 for Terms and Acronyms

(2) See Exhibit 2.5.3-18 for Terms and Descriptions

2.5.3.1.7
(08-04-2023)
Related Resources

(1) The following are supplement references on the development of maintainable, portable, reliable, and secure software applications.

- IBM Enterprise COBOL for z/OS Programming Guide Version 6.2, SC27-8714-01, Second Ed. (March 2019)
- National Institute of Standards and Technology Interagency/Internal Report (NISTIR). 8397 Guidelines on Minimum Standards for Developer Verification Software, October 2021 *https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8397.pdf* ,October 2021
- NIST Special Publication (SP) 500-98 Planning for the Software Validation, Verification and Testing, 2008
- NIST SP 800-53 Rev 5
- NIST SP 800-53A *https://doi.org/10.6028/NIST.SP.800-53Ar5*
- Open Web Application Security Project (OWASP) NIST Special Publication (SP) 500-234 Software Verification Process. *Reference information for the software verification and validation process (nist.gov)* Code Review Guide 2.0. *https://owasp.org/www-pdf-archive/OWASP_AlphaRelease_CodeReviewGuide2.0.pdf*

- Coding Standards and Best Practices for Python Code Quality *https://www.zenesys.com/blog/python-coding-standards-best-practices-for-python-code-quality*, Satish Mehta, June 24, 2021
- Andrew Showers and Salles Viana. Good Programming Practices
- Seacord, Robert C., Software Engineering Institute (SEI) CERT C Coding Standard. “98 Rules for Developing Safe, Reliable, and Secure Systems”. Second Ed
- ISO/IEC 9899.2011 ISO/IEC, Programming Languages-C, 3rd ed.
- Brian W. Kernighan and P. J. Plauger, The Elements of Programming Style, ISBN: 0070342075
- IBM High Level Assembler for z/OS Language Reference V1R6
- IRM 2.5.12 - System Development, Design Techniques and Deliverables
- IRM 2.16.1 - Information Technology, Enterprise Life Cycle, ELC Guidance
- IRM 2.120.7 - Solution Engineering, Solution Design Process
- IRM 2.127.2 - Information Technology, Testing Process and Procedures
- IRM 10.8.1 - Information Technology , Security, Policy and Guidance
- IRM 10.8.6 - Information Technology, Security, Application Security and Development
- Nancy Stern, Alden Sager and Robert A. Stein, Assembler Language Programming, ISBN: 0–471–88657–2
- Nancy Stern and Robert A Stern, Structured COBOL Programming, ISBN 0-471-29987-1
- Jay Ranade and Alan Nash, The Elements of C Programming Style, ISBN 0070512787
- IBM Assembler System Standards, Chapter 1- 8 (IRS-defined)
- IRS Document 12384, C++ Programming Standards
- Java Programming Language, Oracle Technology Network/Java
- Java Assembly Package, *https://docs.oracle.com/javase/tutorial/deployment/jar/index.html*
- Java Programming Package and Import Statements, *https//docs.oracle.com/javase/tutorial/java/package/usepkgs.html*
- Java Programming Input Validation and Data Sanitation, *http://docs.oracle.com/javase/8/docs/api/java/lang/RuntimesException.html*
- How to write documentation comments and available tags, *https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html*
- Java primitives - *https://cs.fit.edu/~ryan/java/language/java-data.html and https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html*
- The Open Web Application Security Project (OWASP) *https://www.owasp.org*
- For additional Assembler Standards and References, see Exhibit 2.5.3-19
- C# Corner. What Is .NET Core. *https://www.c-sharpcorner.com/article/whati-is-doc-net-core/*
- Todd Baginski and Pat Dunn, Microsoft PowerApps Canvas App Coding Standards and Guidelines
- Kashif Salim Bawany, Microsoft MS Federal Cloud & Customer Experience Team. Governance for Power Platform: Management Models, November 01, 2021.
- Matthew Devaney, Matthew Devaney, Microsoft PowerApps Collections Cookbook: No Ads, No Fluff, Just Power Apps Stuff, *https://www.matthewdevaney.com/powerapps-collections-cookbook/*

- Summit Bajracharya, PowerApps Performance: 40 Way to Improve Your App, *https://summitbajracharya.com.np/40-ways-to-improve-your-powerapps-performance/*, October 11, 2020
- Describe how to Build Applications with Microsoft Power Apps *Describe how to build applications with Microsoft Power Apps - Training | Microsoft Learn*

2.5.3.2
(08-04-2023)
Federal Government Application Standards Guidance

(1) The Federal Information Security Modernization Act of 2014 (FISMA) was passed for providing a framework with better information security controls over information resources, supporting Federal Government operations and assets. The IRS must be in compliance with the National Institute of Standards and Technology Interagency/Internal Report (NISTIR). 8397 Guidelines on Minimum Standards for Developer Verification Software, October 2021 *https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8397.pdf*, October 2021. The recommended techniques are:

- Code-based or Static Analysis
- Fuzzing
- Good Software Development Practices
- Threat modeling

(2) IRS applications must also be in compliance with federal standards, e.g., NIST SP 800.53 Rev 5 Security and Privacy for Information Systems and Organizations *https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final" title="NIST SP 800.53 Rev 5">https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final*. Some key security controls are:

- Event logging
- Identify Proofing
- Information Spillage
- Incident Handling
- System Development Life Cycle
- System Documentation, Source Code

(3) For NIST SP 800.53A Revision 5 “Assessing Security and Privacy Controls in Federal Information Systems and Organizations” *https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53Ar5.pdf*. Some key security controls are as follows:

- Insider threats
- Software application security (including web applications)
- Cross domain solutions
- Advanced persistent threats
- Industrial/process control systems
- Personally Identifiable Information Processing and Transparency

(4) For information on IRS Information Technology Cybersecurity controls see IRM 10.8.1 Security, Privacy and Assurance, IT Security, Policy and Guidance and IRM 10.8.6 - Security, Privacy and Assurance, IT Security, Application Security and Development.

#
#
#
#

#
#
#
#
#
#
#

#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#

#
#

#
#
#
#
#

#
#
#
#
#
#

#

#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#
#

#
#
#
#
#
#

#
#

#
#
#
#

#
#

#
#
#

#
#

#
#
#

#
#
#
#
#
#
#
#

#
#
#
#

#

#

#
#
#
#

#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#
#
#

2.5.3.3
(08-04-2023)
General Programming

(1) This subsection of the IRM is based on specific enterprise platforms and languages - These IRS standards and guidelines pertain to application program development and documentation efforts.

(2) The objective of this subsection is to promote the development of programs that are reliable, modular, easily maintainable, and as portable as possible.

(3) New software tools for application development and decision support may supplement and/or replace traditional design and programming techniques. Commercially acquired software packages may reduce development time by eliminating “detailed” design and programming activities.

(4) Recommend doing an analysis of application security and system dependencies before the decision of using Commercial Off-The-Shelf (COTS) software packages for developing software.

(5) The scope of this directive is service-wide. This includes software developed by contractors where the guidelines apply to Assembler Language, COBOL, C Language, C++ programming, C# programming, Python programming, .NET, Java programming, and Power Platform Power apps.

2.5.3.3.1
(08-04-2023)
Goals

(1) The primary goal of structured programming is to produce working programs that are: modular, accurate, and self-documenting, so that they are easily read and maintained by someone other than the original author.

(2) Structured programming includes the following activities:

- Developing specifications for the logic of each module
- Writing structured code to implement the logic of the module
- Using a structured testing methodology that gradually creates a working program as each module is introduced into the application system

2.5.3.3.2
(08-04-2023)
Basic Principles

(1) Structured programming employs the use of limited syntax (constructs) for source code, single-entry/single-exit modules, and top-down development.

(2) Base the logic of each module on various combinations of control structures. The three basic constructs are Sequence, Selection (If-Then-Else), and Repetition (Do-While)/(Test-First). Two optional constructs include Repetition (Do-Until)/(Test-Last) and Selection (Case).

(3) Exhibit 2.5.3-4 depicts a flowchart and Structure diagram for each construct. The actual implementation of these structures will vary according to the requirements of the particular language being used.

(4) Ensure that each module has only one entry point to and one exit point from the module.

(5) Partition and organize each module, program, and application system into a hierarchical structure. Structure charts, module specifications, and structured code are part of the design of a system.

#
#

#
#
#

2.5.3.3.3
(08-04-2023)
Design Specifications

(1) Various tools are commonly used to communicate and transition design specifications to source code. These tools are as follows:

a. Structure charts
b. Module specifications

(2) IRM 2.5.12 System Development, Design Techniques and Deliverables, provides comprehensive standards and guidelines regarding structure charts and module specifications during design.

2.5.3.3.4
(08-04-2023)
Software Development and Project Planning

(1) Software development is a complex process which requires proper planning and execution of all initiatives/tasks for the successful completion of the software development project

(2) A useful software development project plan must include as follows:

#
#
#
#
#
#
#
#
#
#

#

#

#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#
#
#

#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#

2.5.3.3.5
(08-04-2023)
Documenting, Testing, and Debugging Source Code

(1) This subsection addresses services that must be performed regardless of the language or platform selected.

2.5.3.3.5.1
(08-04-2023)
Documenting Code

#
#

#
#

#
#

#

#
#
#
#

#
#

#
#
#
#

2.5.3.3.5.2
(08-04-2023)
Developer Testing and Debugging Code

(1) Review, analyze and test the code for consistency, correctness, clarity, and completeness according to IRS coding standards and NISTIR 8397.

(2) Test the software according to IRM 2.127.1 Testing Standards and Procedures; and IRM 2.127.2 IT Test Policy, and Information Technology Testing Process and Procedures.

(3) Ensure the software functions as it was planned, and is free from vulnerabilities by performing the minimum standards for developer testing as follows:

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#

2.5.3.3.6
(08-04-2023)
Selecting Programming Languages

(1) For new projects, select the appropriate programming language based on IRS standards, executive leadership mandates, engineering requirements, and Enterprise Architecture's recommendations.

2.5.3.3.7
(08-04-2023)
Data Controls

#
#

#
#
#
#

#

#
#

#
#

#
#

#
#

#
#
#

#

#

#
#

#
#

#
#
#
#
#
#
#
#
#
#
#
#

2.5.3.3.7.2
(08-04-2023)
Programming Considerations for Data Controls

(1) Integrate controls into the development effort. The types of controls, and the amount of detail are dependent upon the size and complexity of the application system.

(2) Weigh each development effort based on the following operational considerations:

- The amount of operator intervention
- Multi-file/multi-cartridge processing
- Checkpoint/restart capability
- The file ID on all internal reports
- Back-up of control file
- Initialization of working storage and output buffers with spaces and zeros
- Run to run balancing

#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#
#
#
#

#
#

#
#

#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#

#
#
#
#
#
#
#

2.5.3.3.7.5 (08-04-2023) Including Data Controls

(1) Include computer generated control lists with record counts by file, file number and name, money amounts and tape/disk numbers.

#
#
#
#

#

(4) Include computer generated cartridge numbers on all control lists:

#
#

(5) Computer generated hard copy control output for all runs.

#
#

#

#
#

(7) Assign a unique identifier to each cartridge file.

#
#
#

#
#
#

#
#
#
#
#

#
#
#
#
#
#

#
#
#
#
#
#

#
#
#
#
#

#
#

#
#
#

#
#

#

#
#

#
#

#
#
#

#
#
#
#

#
#
#
#
#
#
#
#
#
#

#

#
#

#

#
#
#
#

#

#
#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#

#
#
#

#
#
#
#
#

#

#
#

#
#
#

#
#
#
#

#
#

#
#
#

#
#
#
#
#

2.5.3.4
(08-04-2023)
COBOL Programming

(1) This subsection establishes controls to ensure COBOL programs are reliable, maintainable, and portable.

2.5.3.4.1
(08-04-2023)
COBOL Overview

(1) The Common Business-Oriented Language (COBOL) is a high-level computer programming language developed during 1959 created for its portability and readability of programs as normal English instead of machine language. COBOL is known best for processing large quantities of business data through record and data structure methodology. For example, a record clusters heterogeneous data: ID, name, age, and address into a single unit. A committee of computer manufacturers, users, and U.S. government organizations created CODASYL (Committee on Data Systems and Languages) to establish, oversee the language standard to ensure COBOL's portability across dissimilar systems.

(2) The controls prescribed are applicable to all IRS COBOL programs whether they are developed by the IRS or outside vendors for the IRS.

2.5.3.4.2
(08-04-2023)
COBOL Basic Principles

(1) The development of structured COBOL programs in accordance with this subsection is dependent on structured design.

(2) Structured COBOL code is the implementation of the logic depicted in module specifications. Module specifications directly correspond to the modules shown on the structure chart.

(3) Structure charts, and therefore module specifications and structured code, are based on a top-down design of the application system. Each of the modules that constitute a structure chart should have a single entry point and a single exit point. The logic of each of the modules is based on various combinations of the three control structures: sequence, selection, and iteration.

(4) These principles have been established with the understanding that COBOL programs are not always maintained by the original author. All structured programs will have the same visual format. Only the most common formats are discussed.

2.5.3.4.3
(08-04-2023)
COBOL Structured Programming

(1) Structured programming is comprised of three logical structures:

a. **Sequence structure**: In a sequential structure, the commands are executed in sequence. The flow of the program is to complete one instruction and then drop down and execute the next instruction and then the next until something terminates the sequence such as the end of a paragraph.

b. **Selection structure**: In a selection structure the processing is dependent on a condition that is being tested. In COBOL, the selection structure is usually accomplished with an IF or an EVALUATE (the implementation of the case structure in COBOL) or with an implied IF such as the AT END clause in the READ statement.

c. **Iteration structure (LOOP STRUCTURE)**: The iteration structure causes something to be executed over and over again until some condition terminates the repetition. Additional information is as follows:

- This is the looping structure that has been used in all programs
- There are two basic structures that a language may implement: “Do-While” and “Do-Until”

Note: The difference between the two structures is when the condition is tested. In the “Do-While” structure the condition is tested before the loop is executed while in the “Do-Until” structure the condition is tested after the loop has been executed. This means that with the Do-While structure there is a possibility that the loop will never be executed.

- **The PERFORM...UNTIL**: Is similar to the “Do-While ”structure because the condition is tested before the loop is executed.

#
#
#
#
#
#

#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#

#
#
#
#
#
#

#

#

#
#
#
#
#

#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#

#

2.5.3.4.3.3
(11-26-2001)
COBOL Environment Division

(1) Start each main clause (for example, the SELECT clause) in column 12.

(2) Start each sub-clause in column 16.

#
#
#

#
#
#

#

#
#

#
#
#

#

#

#

#
#

#
#

#
#
#

#
#

#
#
#
#

#

#
#

#
#
#
#
#
#
#
#
#

#
#
#
#

#
#

#
#
#

#
#

#

#
#
#

#

#

#

#
#

#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#

#
#

#
#

#
#

#
#
#

#

#

#

#

#

#

#

#
#

#
#
#

#
#
#

#

#

#
#

#
#

#
#

#
#
#
#

#
#

#
#
#
#
#

#
#
#

#

#

#

#

#
#

#

#
#
#

#

#

#
#

#
#
#

#
#
#

#

#

#
#

#
#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#

#
#

#

#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#

#
#

#
#

#
#
#
#
#

#
#

#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#

[redacted]

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

[redacted]

2.5.3.4.4
(08-04-2023)
COBOL Compile Run-Time Warning Messages

(1) Currently IRS applications hosted on IRM Mainframes, and have migrated to Enterprise COBOL compiler version 6.2. When COBOL programs are written they have to be compiled into object-code from source-code in order to be read by the computer. Some compiler warnings are more severe than others. The following warnings messages identified must be cleared before source-code can be moved into production:

- **Warning messages like**: **IGZ0279W, IGZ0316W and IGZ0318W** - will display for new COBOL compiler with the first two characters “UL” or “UO” (PROCGRP) with, e.g., **UL2NCL versus. DB2NCL, ULNBL versus NCNB**), (ADD INITCHECK?)
- **Example Warning message**: **IGZ0279W** The value data-item-value of data item data-name at the time of reference by statement number verb-number on line line-number in program program-name failed the “NUMERIC” class test or contained a value larger than the “PICTURE” clause as detected by the “NUMCHECK” compiler option. See Exhibit 2.5.3-22 Compiler 6.2 Warnings for more examples.

#
#
#

#
#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#
#
#
#

#
#

#
#
#

#

#
#

#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#
#
#
#

#
#
#
#

#

#

#
#

#
#

#
#
#

#

#
#

#
#
#
#
#
#
#

#
#

#
#

#
#
#
#
#
#
#
#

#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#

#
#
#
#
#

#
#
#
#
#
#

#
#

#
#
#
#
#
#

#
#
#
#

#

#
#
#

#
#

#
#

#
#

#
#

#
#

#
#
#
#
#
#

#
#
#

#

#

#

#

#

#
#
#

#
#

#
#
#
#

#
#
#
#
#
#

#
#
#
#
#

#
#

#
#
#
#

#
#
#

#
#
#
#

#
#
#
#

#
#
#

#
#

#
#
#
#

#

#
#
#
#

#
#
#
#

#
#

#
#

#

#
#

#

#

#
#

#

#
#

#

#

#

#
#
#
#
#
#

#

#

#
#
#

#

#
#

#

#
#
#
#

#
#
#
#

#
#

#
#
#
#

#

#
#
#

#

#
#

#
#

#
#

#

#
#
#
#

#
#
#
#
#
#

#

#

#

#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#

#
#
#
#
#
#

#
#
#
#

#
#
#
#
#
#
#
#

#
#
#
#
#

#
#
#

#
#

#
#

#
#
#
#
#

#
#
#

#

#

#
#
#
#

#
#

#
#
#

#
#
#
#

#
#
#
#
#

#
#
#
#
#

#
#

#
#
#

#
#

#
#
#
#
#
#
#

#

#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#

#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

[redacted]

[redacted]

#
#
#
#
#

#
#
#
#

#
#
#

#
#
#

#
#
#

#
#
#
#
#
#
#
#
#

#
#

#
#
#
#

#
#
#

#

#
#
#

#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#

#
#
#
#

#
#

#
#
#

#
#
#

#
#
#
#

#
#
#

#

#

#

#

#
#
#
#
#

#
#
#
#

#
#
#

#
#
#
#
#
#

#
#

#
#

#
#
#
#

#
#
#
#

#

#

#
#

#
#
#

#
#

#
#

#
#
#
#

#
#

#
#
#
#

#

#
#

#
#
#

#

#
#

#

#

#
#

#
#

#
#
#

#

#

#

#
#
#
#

#

#
#
#

#

#

#

#

#
#

#

#
#
#

#

#
#
#

#
#
#

#

#

#

#

#

#

#

#

#
#
#
#
#

#
#
#
#
#

#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#

#
#
#

#
#
#
#
#

#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#

#
#

#
#

#

#
#

#
#
#
#

#
#
#
#
#
#
#

#
#

#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#

#
#

#
#
#

#

#
#

#
#
#
#

#
#
#
#
#
#
#

#
#

#
#
#
#
#
#

#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#

#
#
#
#

#
#
#

#
#
#

#

#
#
#
#
#
#
#

#
#

#
#
#

#

#
#
#
#
#
#
#
#

#
#

#
#
#
#

#
#
#

#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#

#

#
#
#
#
#
#
#
#

#
#

#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#

#
#
#
#
#

#
#
#
#

#
#

#
#
#

#

#
#
#
#

#
#
#

#
#

#
#

#
#
#
#
#
#

#
#

#
#
#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#

#

#
#
#
#
#
#
#
#
#

#
#

#
#
#
#
#
#

#
#

#
#

#
#

#
#

#
#
#
#
#
#

#
#
#

#

#
#

#
#

#
#

#
#

#

#
#
#

#
#

#
#
#
#
#

#

#
#
#
#

#
#
#
#
#
#

#
#
#
#
#
#

#
#

#

#
#
#
#

#
#
#
#
#
#

#
#

#
#

#

#
#
#
#
#

#

#
#

#
#
#

#
#
#

#
#
#

#
#

#
#
#

#
#
#
#
#

#

#
#

#

#
#

#

#
#
#
#
#

#
#
#

#
#
#
#
#

#
#
#
#
#
#
#
#

#
#
#
#
#
#

#
#
#

#
#
#
#

#
#
#
#

#
#
#

#

#
#
#

#
#
#
#
#

#

#
#
#
#

#
#
#

#
#
#

#
#

#
#
#

#
#

#
#
#
#

#
#

#
#
#
#

#
#
#
#

#

#
#
#

#

#
#
#

#
#
#

#
#
#
#
#

#

#
#
#

#
#

#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#

#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#

#
#
#

#
#
#
#
#
#

#
#
#
#

#
#
#
#

#
#

#
#
#
#

#
#
#

#

#
#

#
#
#

#
#

#
#
#
#
#

#
#
#
#

#

#

#
#

#

#

#

#
#

#
#
#
#

#
#
#
#
#
#
#
#

#
#
#
#

#
#

#
#

#
#

#
#
#

#
#
#
#
#

#
#
#

#
#

#
#
#
#
#
#
#
#
#
#
#

#

#
#
#
#
#
#
#
#
#
#

#

#
#
#

#
#
#

#
#

#

#

#

#
#

#

#
#

#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#

#
#
#
#

#
#

#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#

#
#
#
#
#
#
#
#
#

#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#
#
#
#

#
#
#
#
#
#

#
#

#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#

#
#
#
#

#
#
#
#
#

#
#

#
#
#
#

#
#
#

#
#
#
#
#

#
#

#

#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#

#
#

#
#

#
#
#
#

#
#
#
#
#
#
#

#

#
#
#

#
#
#

#
#

#
#
#
#
#

#
#
#
#
#
#
#
#
#
#

#

#

#

#
#
#

#
#
#

#
#
#
#

#
#
#

#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#

#
#
#

#
#

#
#
#
#
#

#
#
#
#

#

#
#
#
#
#
#

#
#
#

#
#

#

#
#
#

#

#

#

#

#

#
#

#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#

#

#
#

#
#
#
#
#

#
#

#

#

#

#

#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

2.5.3.9 (08-04-2023) Java Programming Language

(1) This subsection of the IRM provides controls to ensure Java programs are reliable, maintainable, and portable. The controls established are applicable to all Java programming projects whether they are developed by IRS government or contract employees.

2.5.3.9.1 (08-04-2023) Java Programming Overview

(1) Java programming is a robust general-purpose computer programming language that has the ability of running several programs, or parts of a program in parallel. This allows a program to achieve high performance, and throughput.

(2) Java has its own structure, syntax rules, and programming framework which is based on the concept of object-oriented programming, and is designed to have as few implementation dependencies as possible This allows application developers to “write once, run anywhere” (WORA) which means compiled Java code can run on all platforms that support Java without requiring recompiling.

(3) Structurally, Java is comprised of the following:

a. **Package**: This is a namespace mechanism consisting of classes.
b. **Classes**: A user defined template from which objects are created consisting of methods, variables, constants, and constructors.
c. **Object**: Consist of the State (attributes), this behavior is (represented by method as an object) pertaining to the following:
- An object and response of an object with other objects.
- Identity (unique name to an object), and enables one object to interact with other objects.

d. **Java compiler**: Java platform source code is written to .java files for compiling, the compiler checks the developers code against the language’s syntax rules them writes out the bytecode in .class files.

2.5.3.9.2
(08-04-2023)
Program Objectives

(1) Java applications are composed of one or more source files. Each source file must be assigned to a package.

(2) A Java assembly contains one or more related packages.

(3) Java source files must follow the following structure and naming conventions as listed below.

#
#
#
#
#
#
#

#
#

#

#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#

#

#
#
#
#
#
#

#
#
#

#
#
#

#

#
#

#

#

#
#

#

#
#
#

#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#
#
#

#
#
#
#
#
#

#
#
#
#
#

#
#

#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#

#
#
#

#
#
#

#
#
#

#

#

#

#
#
#

#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#
#
#

#

#
#
#

#
#

#
#
#

#
#

#
#
#

#
#

#
#

#
#
#

#
#
#

#
#

#
#

#
#

#

#

#

#
#
#
#
#

#

#
#
#
#
#
#
#
#

#
#

#
#
#

#
#

#
#
#

#
#

#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#

#

#
#
#
#

#
#
#
#
#
#

#
#
#

#
#

#

#
#
#
#

#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#

#
#
#

#
#

#

#
#

#
#
#
#
#

#
#

#
#
#

#
#

#
#

#
#
#

#
#
#

#
#
#

#
#
#
#
#
#
#

#
#
#
#

#
#

#
#
#

#
#

#
#
#

#
#

#
#

#
#
#

#
#
#

#
#
#

#
#

#
#

#
#
#

#
#
#

#
#
#

#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#

#

#
#

#
#
#

#
#
#
#
#
#
#

#
#

#
#
#

#
#

#
#

#
#
#
#

#
#

#
#
#
#
#
#

#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#

#
#
#
#
#
#
#
#

#
#

#
#
#
#

#
#

#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#

#
#
#
#
#
#

#
#
#

#
#

#
#
#

#
#
#

#
#
#
#
#
#
#

#
#

#
#
#

#
#
#

#

#

#
#
#

#
#

#
#
#
#
#

#

#
#
#

#
#
#
#

#
#

#
#

#
#
#

#
#
#
#

#
#
#

#
#

#
#
#
#
#

#
#

#
#
#

#
#

#
#

#
#
#
#
#
#

#
#
#

#
#

#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#

#
#

#
#

#
#
#

#
#

#
#

#
#
#

#

#
#
#
#
#
#

#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#

#
#
#

#
#

#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#

#
#
#

#
#
#

#
#
#
#
#

#
#

#
#

#
#

#
#
#
#
#

#
#
#

#
#
#

#
#
#
#
#
#
#

#
#
#

#
#

#
#
#

#
#
#

#
#

#
#

#

#
#
#
#

#
#
#
#
#
#

#
#

#
#

#
#
#
#

#
#
#

#
#

#
#

#
#

#
#
#

#

#
#

#
#

#
#
#

#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#
#

#
#
#

#
#

#

#
#

#
#
#
#
#

#
#

#
#
#

#
#
#

#
#

#
#

#
#

#

#
#

#

#
#
#

#
#
#
#

#
#

#
#
#

#

#
#

#
#
#
#

#
#
#
#
#
#
#
#
#

#
#

#
#
#
#
#
#
#

#
#

#
#

#
#

#
#

#
#

#
#
#
#

#
#
#

#
#
#

#

#
#
#

#
#
#
#

#
#

#
#

#
#

#

#
#

#

#
#

#

#
#

#
#

#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#

#
#

#
#
#
#

#
#

#
#

#
#

#
#

#
#
#
#
#

#
#

#
#
#

#
#
#

#
#
#
#
#

#
#
#

#
#

#
#
#
#
#
#

#
#

#
#
#
#
#

#
#

#

#
#
#
#

#
#

#
#

#
#

#
#
#
#

#
#
#
#
#

#
#

#

#

#
#

#
#

#
#

#
#

#

#

#
#

#
#

#
#

#
#

#
#

#
#

#
#
#
#
#

#
#
#

#
#
#
#
#

#
#
#
#
#

#
#

#
#

#
#
#

#
#

#
#

#
#

#
#

#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#

#
#
#

#
#
#

#
#

#
#

#
#

#
#
#

#
#
#
#
#
#
#
#

#
#

#
#

#
#

#
#
#
#
#
#
#
#

#
#

#

#
#
#
#
#

#
#
#
#
#

#

#
#

#
#

#
#
#
#

#
#

#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#

#
#
#

#
#
#
#
#
#
#
#

#
#

#
#

#
#

#
#

#
#
#
#
#

#
#
#
#

#
#
#
#

#
#
#

#
#

#
#

#
#

#

#
#

#

#

#
#

#
#

#
#

#

#
#

#
#
#
#
#
#
#

#
#

#
#

#

#
#

#
#

#

#

#
#
#
#
#
#
#
#
#
#

#
#

#
#
#
#
#
#
#
#
#

#
#
#
#
#

#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#

#
#
#
#
#
#
#

#
#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#

#
#
#
#
#
#

#
#

#
#
#

#
#
#
#

#

#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#
#

#
#

#
#

#
#
#
#
#

#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#
#
#
#
#
#
#

#
#
#

#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#
#
#

#
#
#
#
#
#
#
#
#
#

#
#
#

#
#

#
#
#

#
#
#
#
#
#
#
#

#
#
#
#
#

#
#
#
#
#
#
#
#
#
#
#

2.5.3.12 (08-04-2023) **Microsoft Power Platform**

(1) This subsection establishes the policies and guidance for use of Power Platform within the Internal Revenue Service to include standard environments for general support and dedicated environments for specialized applications. Power Platform supports a range of development needs within the IRS.

2.5.3.12.1 (08-04-2023) **Microsoft Power Platform Overview**

(1) Microsoft Power Platform which is built on Azure is a low-code platform that spans all of Azure to include: Microsoft 365 (M365), Dynamics, and standalone applications. Each tool in the suite has a graphical user interface that can be used by any business user or developer without prior experience. These range from low-end (Personal and non-critical group/team collaboration) solutions to high-end (Critical/Premium) solutions.

(2) Microsoft Power Platform is a suite of application development, application connectivity, and business intelligence tools using a no-code/low-code programing language to deliver solutions optimized for cloud environments, specifically in Azure. Power Platform offers the ability to create solutions in a portion of the time that it takes to build them using traditional development methods. The business reason for moving to this solution is cost savings and the speed of software development and integration with M365 services. The time saving for code development comes from being able to drag and drop controls onto a canvas and configure properties without understanding pro-

gramming code or Hyper-Text Markup Language (HTML), knowing how to correctly structure code, or knowing how to implement difficult business logic with code.

(3) See OneSDLC for more information on how to onboard a Power Platform project *https://irsgov.sharepoint.com/sites/OneSDLC/SitePages/Home.aspx*.

(4) Within the IRS Power Platform are the recommendations for three development scenarios with different classes of Power Platform environments provisioned to support each as the following:

a. **Personal and Collaborative (Low-end) Solutions** - These solutions:
   - Enhance productivity at the personal and group levels
   - Range from personal solutions that help manage a user's mailbox or OneDrive storage; SharePoint list forms, or collaborative tools hosted on Teams and SharePoint sites
   - Use standard connectors
   - Are not considered business critical and are scoped to limited, internal audiences
   - May handle Controlled Unclassified Information (CUI) data, provided the solution is identified and authorized in conjunction with the information storage container with appropriate Privacy and Civil Liberties Impact Assessment (PCLIA) and Systems Security Plan (SSP)
   - The owner of the low-end solutions is responsible for lifecycle management

Note: The Information Technology, Enterprise Operations (EOps) is responsible for administering the environment where these solutions are hosted.

b. **Branch and Business Unit Operations (Mid-level) Solutions** - These solutions:
   - Replicate non-critical and operational support processes of a branch or business unit
   - Range from organizational line of business applications such as "front door" activities
   - Use standard Power Platform infrastructure components and will use standard subscriptions
   - May be business critical and be scoped to IRS internal audiences, to include all IRS users
   - May handle Controlled Unclassified Information (CUI) data, provided the solution is identified and authorized in conjunction with the information storage container with appropriate PCLIA and SSP
   - Must not use premium or custom Power Platform infrastructure components; therefore, do not require separate test and trials environments
   - Separate development (sandbox) and trials data sources are required
   - The owner of the mid-level solutions is responsible for lifecycle management and administration of production environments hosting the solutions

c. **IRS Operations Critical and Premium (High-end) Solutions** - These solutions:
   - Are high-end solutions intended for larger audiences, to include IRS-wide dissemination
   - Are solutions to replicate critical and operational support processes for the IRS
   - Range from tax processing support (but short of actual tax process-

ing) to major IRS support operation processes

- May use premium or custom Power Platform infrastructure components (defined in detail below,) and thus may require premium subscriptions
- May be business critical and are scoped to IRS internal audiences
- May handle CUI data, provided the solution is identified and authorized in conjunction with the information storage container with the appropriate PCLIA and SSP
- Require separate development (sandbox) and trials environments
- Can have the development (sandbox) and trials environment administration delegated to the owner of the solution
- The owner of the high-end solutions is responsible for lifecycle management

Note: The Information Technology, Enterprise Operations(EOps) is responsible for maintaining production environments hosting high-end solutions.

(5) Microsoft Azure includes the Power Platform service to host Power Platform tools. The major components of the Power Platform service include:

a. **Power Platform Environments**: These are storage containers for applications, workflows, application data, and solutions. Different environments will have differing characteristics to support particular use cases, such as development sandboxes or production. Environments also may have different components based on the needs of hosted solutions. For more information on the IRS's use of environments, see Exhibit 2.5.3-24.

b. **Microsoft Dataverse**: Dataverse is a Software-as-a-Service (SaaS) that allows data to be integrated from multiple sources into a single store. It stores its data within a set of tables and entities. Dataverse includes business logic, validation, and security features to support the applications. See image of data collection for Dataverse in Figure 2.5.3-84.

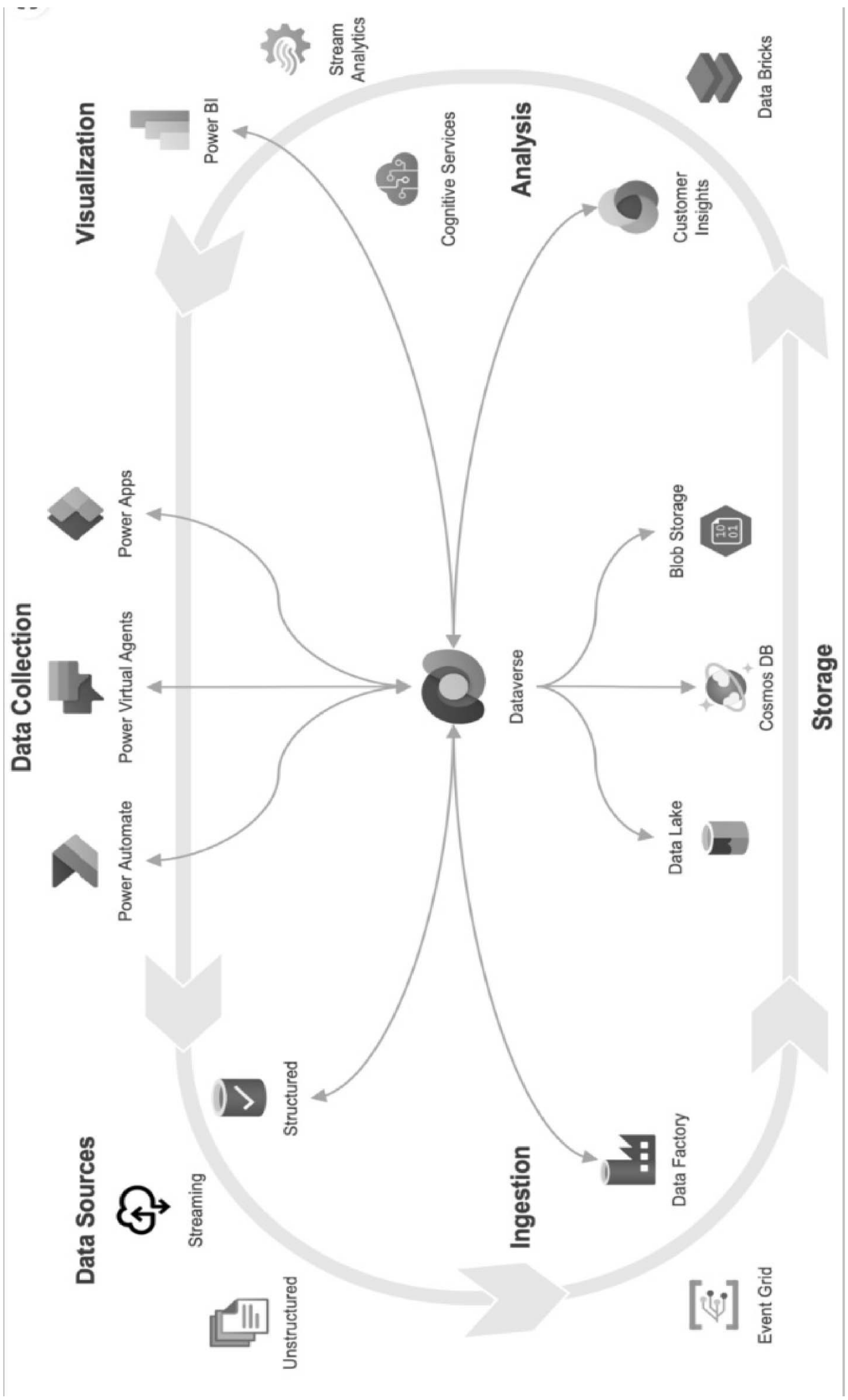

Figure 2.5.3-84

c. See Figure 2.5.3-85 below for types of Dataverse tables:

Dataverse Table Types

Types of Tables	Table Description	Dataverse Table Usage
Standard Tables (Built-In)	These are tables provided by the Dynamics 365 service upon creation of a Dataverse database. These are considered system defined tables.	Used for general purpose or in standard Apps. These include tables referenced as common data sources across Azure and M365, such as Account, Address, Approval, Business Unit, Team, and User. ***Note:*** Business Unit, Team, and User are used as terms relative to the Power Platform solution in this context. See IRM 2.5.3.12.1
Activity Tables (Built-In)	Activity tables are system defined and provide elements to track system and solution defined actions.	Used for specialized purpose or Apps. These include tables used to record activities across Azure and M365 such as: Appointments, Bookings, and Tasks. These record activity events relative to built-in services and features that a developer can use within a Power Platform solution.
Custom Tables	Used for specialized purpose or Apps. These are not considered system tables and require premium subscriptions to access.	Used for general purpose or in standard Apps. These contain application specific data and are defined by the developer of an application.
Virtual Tables	a. A table created as a link to data held in a data source external to Dataverse. b. Virtual tables appear as regular tables, but do not contain the actual data. When accessed, at run time, the virtual table performs a query of the external data source.	a. Built-in virtual tables include Azure Active Directory Users (AAD Users) and Solution History Tables. b. Developers with permission to use external data connections through premium or custom connectors can create virtual tables for specialized use in applications.

Figure 2.5.3-85 This Table Describes the Dataverse Table Types.

d. **Subscriptions**: A subscription is required for all users to access Power Platform tools and solutions. The base, or standard, M365 User subscription allows a user to access tools and solutions limited to M365 components. These are known as Power Platform for M365 subscriptions. Premium Power Platform subscriptions allow users to connect to data and services outside of M365. See Figure 2.5.3-86 for more details.

IRS Power Platform M365 Subscriptions

IRS Subscription Types	Issuances	Granted with Subscriptions
M365 Standard Subscriptions	All IRS Users. (This is the G5-level subscription).	a. Power Apps for M365. b. Power Automate for M365 c. Power BI Pro ***Note:*** The above grants allow users to build and access solutions using data in M365, subject to access controls.
M365 Premium Subscriptions	Individual Users approved for specific applications	a. Power Apps per User OR Power Automate per User. b. The above grant allows a user to access and use Power Apps solutions that use premium components (such as premium connectors or custom Dataverse tables).
M365 Premium Capacity Subscriptions	Applications or solutions that require premium components	These include Power BI Premium, AI Builder, Logic Apps, Power Agents, Power Pages, and Power Apps per App subscriptions.

Figure 2.5.3-86 This Table Describes Power Platform M365 Subscriptions.

e. **Connectors**: These are proxies or wrappers built to interface with specific APIs to allow an underlying service to work with Power Platform. Connectors consist of pre-built actions and triggers. IT Enterprise Operations will maintain a reference list of all allowed, unblocked connectors by category. Connectors come in three main categories as the following:

- **Standard**: These are connectors published by Microsoft for use with Azure and M365, which only require the standard user subscription. Within the list of standard connectors, some are required for M365 services and cannot be blocked (listed as Unblockable in policies). Other connectors can be blocked by policy definitions.
- **Premium**: These connectors can be blocked by policy. Premium connectors require a premium subscription.
- **Custom**: These are connectors built as wrappers to REST APIs; are used for custom APIs, and may be blocked by policy

f. **Data Gateway**: This is a service configured to allow connections between the Power Platform services and on-premises (IRS data centers) data sources. These use premium connectors.

g. **Common Data Model**: A metadata system, leveraging Dataverse and Dynamics to provide a shared metadata model and data language shared across applications and business processes.

h. For more information on “Power Platform Overview and Guidance” view *https://learn.microsoft.com/en-us/power-platform/guidance/*.

(6) Power Platform tools include the following:

a. **Power Business Intelligence (BI)**: A self-service business analytic tool for business intelligence that builds real-time data visualizations, interactive dashboards, and personalized reports on mobile and desktops. Since Power BI is not related to this IRM for programming and coding view IRM 2.5.11.8 for more detailed information.

b. **Power Apps (Application Development)**: Power Apps is a suite of low-code tools providing the ability to use services and connectors to build applications, forms, and data interfaces. Power Apps works with M365 services, Dataverse, and other data sources (such as Microsoft SQL Server and Oracle). Power Apps is optimized to allow rapid development, limiting time and expense. See Figure 2.5.3-87.

Power Apps: Building Blocks for Low Code Development

Component Name	Descriptions
Power Apps Maker Portal	A web browser-based tool used by a developer to create a Power App. This tool is hosted in Azure as part of the Power Platform service.
Connectors	Connectors, as described in IRM 2.5.3.12.1, allow the developer to connect to data sources from within the App.
Canvas Apps	An app development mode that starts with a canvas on which the developer adds elements (fields, tables, forms, media, buttons, etc.), with the data model defined during the application development process with connections.
Model-Driven Apps	An app development mode that starts with a defined data model adding to that forms, charts, and views. Model-Driven Apps depend on Dataverse.
Cards	Micro-Apps that can be shared to other Power Apps or M365 Applications (such as Teams).
SharePoint Form App	A customized form for a Microsoft list or SharePoint library using Power Apps Maker Portal. The form is stored as part of the list or library definition and can only be used by those with access to the list or library.
Teams Embedded App	A Canvas App or Card that is embedded into a Teams channel or conversation.
Teams Dataverse App	A packaged application solution using Dataverse for Teams for data source and embedded into a channel for a team in the Teams user application.

Figure 2.5.3-87 This table describes the Power Apps: Building Blocks for Low Code Development.

c. **Power Automate (Process automation)**: Previously called **Microsoft Flow**, is a low-code tool supporting business logic with creation of workflows using templates, triggers, alerts, actions, and Robotic Process Automation (RPA) quickly without coding. The goal is to perform a set of predefined steps that are repeatable.

Power Automate: Framework of Flows

Component	Descriptions
Power Automate Maker Portal	A web browser-based tool used by a developer to create Power Automate Cloud Flows. This tool is hosted in Azure as part of the Power Platform service.
Cloud Flow	A cloud flow works against a cloud data source. Cloud flows come in three basic types: a. **Automated Cloud Flows:** These are triggered by an event (such as email arrival or item change). b. **Instant Cloud Flow**: This is triggered on a user interface (such as a button). c. **Scheduled Cloud Flow**: This is triggered based on a pre-defined schedule or time interval.
Power Automate Desktop	A client application used by a developer to create Power Automate Desktop Flows. This tool can store flows in Azure, but can also store the flow as a definition file.
Desktop Flow	A desktop flow automates workstation-based tasks and can combine both local and cloud data sources.
Connectors	Connectors, as described in 2.5.3.12.1 (4) d) above, allow the developer to connect to data sources from within the flow.

Figure 2.5.3-88 This table describes Power Platform Power Automate: Framework of Flows.

d. **Power Virtual Agents (Intelligent chatbots)**: A “no-code” graphical interface with Artificial Intelligence (AI) capability for creating intelligent bots to resolve customer issues and automate common questions. The chatbot simulates human conversation through auditory or textual means and can answer specific questions, tasks, and references.

Power Virtual Agents (PVA) - Intelligent Virtual Agent

Virtual Agent Types	Descriptions
Power Virtual Agents Portal	A browser-based tool used by a developer to create a chatbot. This tool is hosted in Azure as part of the Power Platform service.
Linguistic or rule-based chatbots	1. Developer must determine the language for the chatbot. 2. Use ’if or then’ conditions to generate conversational streams.
Menu/Button-based chatbots	1. Displays in the form of menus or buttons. 2. Allows several options to choose from, and depending on what the user clicks on, the bot then prompts another set of options for him to choose.

Virtual Agent Types	Descriptions
Keyword recognition-based chatbots	1. Uses Natural Language Processing (NLP) to serve its users. 2. Users can interact by providing free text input. 3. The chatbot then analyses the user’s text based on keywords as they are keyword-dependent and delivers the best response.
Machine Learning chatbots	Also known as Artificial Intelligence (AI) chatbot responds to questions posed to it in natural language as if it were a real person. **It responds using a combination of pre-programmed scripts and machine learning algorithms**. When asked a question, the chatbot will answer using the knowledge database that is currently available to it. For example the “Alexa” app.

Figure 2.5.3-89 This table describes Power Platform Power Virtual Agents (PVA) - Intelligent Virtual Agent.

e. **Artificial Intelligence (AI) Builder**: The “no-code” Limited - AI Builder enables your business to use intelligence to automate processes from your data in Power Apps and Power Automate by choosing the appropriate prebuilt models, or tailor models as needed.

f. **Azure Logic Apps**: A cloud-based integration Platform-as-a-Service (iPaaS) for creating and running automated serverless workflows that integrate your apps, data, services, and systems (i.e., it can connect different systems across cloud, on-premises, and hybrid environments).

g. **Power Pages**: A secure, enterprise-grade, low-code tool used to create, host, and administer external-facing websites.

Note: Currently the IRS does not have a use case for Power Pages hosting from an Azure/M365 platform.

h. For more information on “Power Platform Overview and Guidance” view website: *https://learn.microsoft.com/en-us/power-platform/guidance/*.

i. For more information on “Azure Logic Apps Overview ”view websites: *https://www.serverless360.com/azure-logic-apps* and *https://learn.micorsoft.com/en-us/azure/logic-apps-overview*.

j. View Figure 2.5.3-89 for the Dataverse data collection process.

2.5.3.12.2 (08-04-2023) IRS Application Lifecycle Management and Power Platform

(1) This subsection establishes Power Platform standards and guidelines for developing consistent high-performing and easily maintainable apps.

2.5.3.12.2.1 (08-04-2023) IRS Application Lifecycle Management and Power Platform Overview

(1) Application Lifecycle Management (ALM) covers the entire life of an application, from initial idea until the end of life instead of taking an iterative, or step-by-step approach to software creation. ALM for Power Platform is the integrated and collaborative approach for the entire lifecycle of your apps and projects.

(2) The IRS has created and implemented a new delivery model "One Software Development LifeCycle (OneSDLC)". The Enterprise Life Cycle (ELC) is no longer feasible for onboarding projects.

(3) Software Development LifeCycle (SDLC) only focuses on the development phase. ALM is more comprehensive than the SDLC because it includes the following:

a. Requirements management
b. Software architecture
c. Development
d. Testing
e. Production
f. Maintenance
g. Change management
h. Continuous integration
i. Project management
j. Deployment
k. Release management

(4) With ALM for Power Platform, developers must implement an environment strategy with three dedicated environments as a minimum: development, production and a test environment. The following must be accomplished for workloads in each environment:

1. Plan & Track
2. Develop
3. Build
4. Define and implement Test Criteria
5. Deploy
6. Operate
7. Improve workloads through monitoring
8. Document the knowledge needed for maintaining all apps

2.5.3.12.2.1.1 (08-04-2023) IRS Microsoft 365 (M365) Power Platform Dedicated Environment Requests Process

(1) Within Power platform, environments serve as containers in which developed solutions, tools, and components are managed. As outlined in section 2.5.3.12.1 (4) a) above the IRS governance of environments allows a variety of types to suit business needs. While the Personal Productivity (Default) and Business Unit environments do not require premium subscriptions, those cannot use premium components of Power Platform. Dedicated solution environments allow for granular governance and controls and support the use of premium components of Power Platform.

(2) A formal request for a Power Platform environment is required for all environments beyond the Personal Productivity (Default) environment as a GetServices request to IT-EOPS. For environments hosting solutions using premium components of Power Platform, additional resource allocation approval may be required; therefore, the requester may be directed to submit additional work requests. See Figure 2.5.3-90 Figure@Exhibit 2.5.3-90:

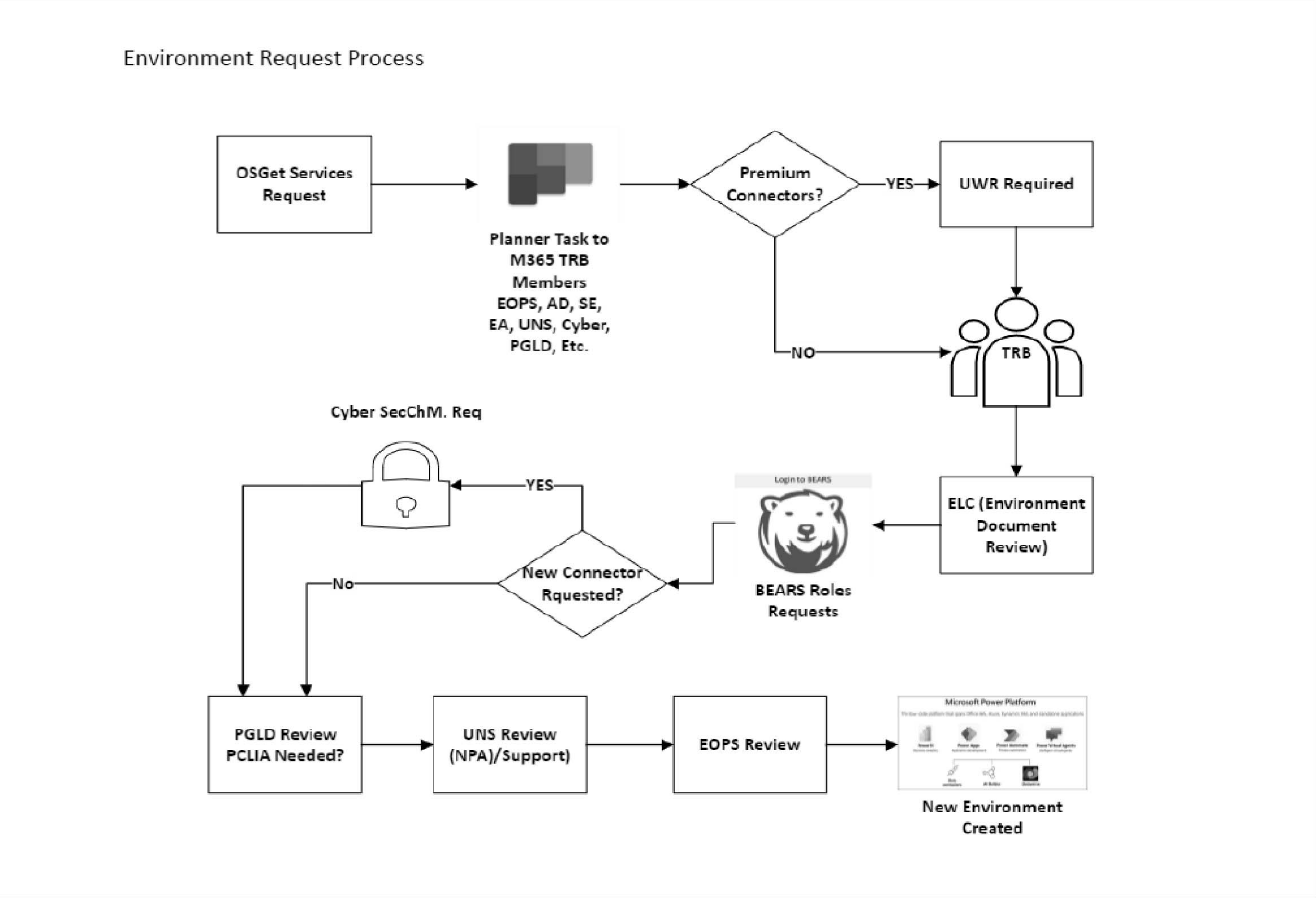

Figure 2.5.3-90

2.5.3.12.2.2 (08-04-2023) IRS Application Development with Power Apps

(1) This subsection establishes Power Apps standards and guidelines for developing consistent high-performing and easily maintainable apps for inexperienced and advanced developers.

2.5.3.12.2.2.1 (08-04-2023) IRS Canvas PowerApps Coding Standards Best Practices

2.5.3.12.2.2.1.1 (08-04-2023) PowerApps Apps Code Naming Conventions

(2) For more information on how to prepare Excel data and naming, see *Canvas apps for enterprise developers, partners, and ISVs - Power Apps | Microsoft Learn*

2.5.3.12.2.2.1.2
(08-04-2023)
Microsoft Power Apps for Canvas Apps: Comments

#

#

2.5.3.12.3 #
(08-04-2023) #

Power Platform Apps - Best Practices for Performance Optimization

#
#
#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

2.5.3.12.4
(08-04-2023)

IRS Power Automate Flow Coding Standards Best Practices

#
#

#
#
#

#
#

#
#
#

#
#
#
#
#
#
#
#
#

#
#

#
#
#
#
#

2.5.3.12.5
(08-04-2023)

Microsoft Power Business Intelligence (BI) Architecture

#
#
#

#
#
#

This Page Intentionally Left Blank

#
#
#

#
#

#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#

Exhibit 2.5.3-2 (08-04-2023)
Java Programming Example of Objects

Comparison of public class ObjectTest and class Point

```java
public class ObjectTest {
public static void main(String[] args) {
Point p1 = new Point(4, 6);
Point p2 = new Point(7, 9);
Point p3 = p2; System.out.println(″Before change:″);
System.out.printf(″Point 1 = (%d,%d)″, p1.getX(), p1.getY());
System.out.printf(″Point 2 = (%d,%d)″, p2.getX(), p2.getY());
System.out.printf(″Point 3 = (%d,%d)″, p3.getX(), p3.getY());
p3.setX(10);
p3.setY(20);
System.out.println(″After change:″);
System.out.printf(″Point 1 = (%d,%d)″, p1.getX(), p1.getY());
System.out.printf(″Point 2 = (%d,%d)″, p2.getX(), p2.getY());
System.out.printf(″Point 3 = (%d,%d)″, p3.getX(), p3.getY());
}
}
class Point {
private int x;
private int y;
public Point(int x, int y) {
this.x = x;
this.y = y;
}
public int getX() { return x; }
public void setX(int x) { this.x = x; }
public int getY() { return y; }
public void setY(int y) { this.y = y; }
}
```

Exhibit 2.5.3-3 (08-04-2023)
Java Programming - Object Test Results

Before change:	After change:
Point 1 = (4,6) Point 2 = (7,9) Point 3 = (7,9)	Point 1 = (4,6) Point 2 = (10, 20) Point 3 = (10, 20)

Exhibit 2.5.3-4 (08-04-2023)

Java Programming Example of “instanceof”

Wrapping Explicit Cast using “Instanceof” Check

```
public class RecastTest {
public static void main(String[] args) {
Dog collie = new Dog();
collie.setName(″Collie″);
System.out.println(collie.getName());
System.out.println(″Leg count: ″ + collie.getLegs());
Animal animal = (Animal)collie;
System.out.println(animal.getName());
// this generates a compiler error
// System.out.println(″Leg count: ″ + animal.getLegs());
if (animal instanceof FourLeggedAnimal) {
System.out.println(″Leg count: ″ + ((FourLeggedAnimal)animal).getLegs());
}
}
}
```

Exhibit 2.5.3-5 (08-04-2023)
Java Programming Example -Subclass

Java Programming Subclass Example

```
class Animal {private String name;
   public String getName() { return name; }
   public void setName(String name) { this.name = name; }
   }
class FourLeggedAnimal extends Animal {
   public int getLegs() { return 4; }
}

   class Dog extends FourLeggedAnimal {
      @Override
   public String getName() { return ″Shaggy ″ + super.getName();
}
}
```

Exhibit 2.5.3-6 (08-04-2023)

Java Programming Example - Enumeration Type

Enumeration Type Example

```
enum Month {

JANUARY(31), FEBRUARY(28), MARCH(31), APRIL(30), MAY(31), JUNE(30), JULY(31),
  AUGUST(31), SEPTEMBER(30), OCTOBER(31), NOVEMBER(30), DECEMBER(31);
    private int days = 0;
    private Month(int days) {
    this.days = days;
    }
          public int getDays(int year) {
        if (days == 28) {
        if (year % 4 == 0 && year % 100 > 0) {
          return 29;
        }
    else if (year % 100 == 0 && year % 400 > 0) {
        return 28;
        }
    else if (year % 400 == 0) {
        return 29;
    }
    else {
        return 28;
    }
  }
    else {
        return days;
        }
      }
  }
public class EnumTest {
    static public void main(String[] args) {
      System.out.printf(″Days in January 2020: %d″, Month.JANUARY.getDays(2020));
      System.out.printf(″Days in April 2020: %d″, Month.APRIL.getDays(2020));
      System.out.printf(″Days in February 1900: %d″, Month.FEBRUARY.getDays(1900));
      System.out.printf(″Days in February 2000: %d″, Month.FEBRUARY.getDays(2000));
      System.out.printf(″Days in February 2019: %d″, Month.FEBRUARY.getDays(2019));
      System.out.printf(″Days in February 2020: %d″, Month.FEBRUARY.getDays(2020));
    }
  }   // Output
Days in January 2020: 31
Days in April 2020: 30 Days in February 1900: 28
Days in February 2000: 29 Days in February 2019: 28
Days in February 2020: 29
```

Exhibit 2.5.3-7 (08-04-2023)
Java Programming Example - Nested Classes

Java Programming Example of Nested Classes

```
import java.util.Optional;
   interface Greeting {
      public String getMessage(String name);
   }

interface Greeting {
      public String getMessage(String name);
}

public class AnonymousClassExample {
      static public void main(String[] args) {
         Greeting englishGreeting = new Greeting() {
               public String getMessage(String name) {
                  Optional nameOptional = Optional.ofNullable(name);
                  return ″Hello ″ + nameOptional.orElse(″Anonymous″);
         }
         };
               System.out.println(englishGreeting.getMessage(″Bob″));
         System.out.println(englishGreeting.getMessage(null));
         }
   }

// Output
Hello Bob
Hello Anonymous
```

Exhibit 2.5.3-8 (08-04-2023)

Java Programming Class Switch Example

Java Programming Class Switch

```
enum Quarter {
Q1, Q2, Q3, Q4;
static public Quarter getQuarter(int index) {
if (index == 1) {
return Q1;
}
else if (index == 2) {
return Q2;
}
else if (index == 3) {
return Q3;
}
else
{
return Q4;
}
}
}
```

```
public class SwitchExample
{
static public void main(String[] args) {
Quarter q1 = Quarter.valueOf(″Q1″);
Quarter q4 = Quarter.getQuarter(4);
displayRequirements(q1);
displayRequirements(q4);
}
```

```
static private void displayRequirements(Quarter currentQuarter) {
boolean taxesDue = false;
boolean runAudit = true;
switch (currentQuarter) {
case Q2: runAudit = false;
break;
case Q1:
runAudit = false;
/* falls through */
case Q3:
taxesDue = true;
break;
case Q4:
break;
}
System.out.printf(″%s: taxesDue=%s, runAudit=%s″, currentQuarter.toString(), taxesDue, runAudit);
}
}
```

Exhibit 2.5.3-8 (Cont. 1) (08-04-2023)
Java Programming Class Switch Example

Java Programming Class Switch

```
// Output
Q1:

taxesDue=true,
runAudit=false
Q4:
taxesDue=false,
runAudit=true
```

Exhibit 2.5.3-9 (08-04-2023)
Java Programming Design Example

Design Example

```
import java.time.LocalDateTime
; public class DesignExample {
private static LocalDateTime firstDate;
private LocalDateTime instanceDate;
// This is the type constructor static {
firstDate = LocalDateTime.now();
}
static public LocalDateTime getFirstDate() {
return firstDate;
}
static public void main(String[] args) {
DesignExample ex1 = new DesignExample();
System.out.printf(″Example 1: First=[%s],Instance[%s]″, DesignExample.getFirstDate(),
ex1.getInstanceDate());
// Put in an execution pause so a date difference can be detected. try {
Thread.sleep(5000);
}
catch (InterruptedException e) {
// TODO Auto-generated catch block e.printStackTrace();
}
DesignExample ex2 = new DesignExample();
System.out.printf(″Example 2: First=[%s],Instance[%s]″, DesignExample.getFirstDate(),
ex2.getInstanceDate());
}
// This is the instance constructor.
public DesignExample() {
instanceDate = LocalDateTime.now();
}
public LocalDateTime getInstanceDate() {
return instanceDate;
}
}

// Output
Example 1: First=[2019-05-30T15:07:08.520],Instance[2019-05-30T15:07:08.566]
Example 2: First=[2019-05-30T15:07:08.520],Instance[2019-05-30T15:07:13.581]
```

Exhibit 2.5.3-10 (08-04-2023)
Java Programming Constructor Example

Java Programming Constructor

```
import java.time.LocalDateTime;
public class DesignExample {
private static LocalDateTime firstDate;
private LocalDateTime instanceDate;
// This is the type constructor static {
firstDate = LocalDateTime.now();
}
static public LocalDateTime getFirstDate() {
return firstDate;
}
static public void main(String[] args) {
DesignExample ex1 = new DesignExample();
System.out.printf(″Example 1: First=[%s],Instance[%s]″, DesignExample.getFirstDate(),
ex1.getInstanceDate());
// Put in an execution pause so a date difference can be detected
. try {
Thread.sleep(5000);
} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
DesignExample ex2 = new DesignExample();
System.out.printf(″Example 2: First=[%s],Instance[%s]″, DesignExample.getFirstDate(),
ex2.getInstanceDate());
}
// This is the instance constructor. public DesignExample() {
instanceDate = LocalDateTime.now();
}
public LocalDateTime getInstanceDate() {
return instanceDate;
}
}
```

// **Output**

```
Example
1: First=[2019-05-30T15:07:08.520],Instance[2019-05-30T15:07:08.566]

Example
2: First=[2019-05-30T15:07:08.520],Instance[2019-05-30T15:07:13.581]
```

Exhibit 2.5.3-11 (08-04-2023)
Java Programming Abstract Properties

Abstract Properties

```
abstract public class Shape {
abstract public double getArea();
}
class Circle extends Shape {
private int radius = 0;
public Circle() {
}
public Circle(int radius) {
this.radius = radius;
}
public void setRadius(int radius) {
this.radius = radius;
}
public int getRadius() { return radius;
} @Override public double getArea() {
return Math.pow(radius, 2) * Math.PI;
}
}

class Rectangle extends Shape {
private int width = 0;
private int height = 0;
public Rectangle() {
}
public Rectangle(int width, int height) {
this.width = width;
this.height = height;
}
@Override public double getArea() {
return width * height;
}
}
class Square extends Rectangle {
public Square() {
}
public Square(int size) {
super(size, size);
}
}
```

Exhibit 2.5.3-12 (08-04-2023)

Java Programming Example of Event Design

Event Design

```
import java.util.ArrayList;
import java.util.EventListener;
import java.util.EventObject;
import java.util.List;
class PropertyEvent extends EventObject {
    private static final long serialVersionUID = 6111548293142751985;
    private String name = ″″;
  public PropertyEvent(Object source, String name) {
    super(source);
    this.name = name;
    }
        public String getName() { return name;
    }
  }
  interface PropertyListener extends EventListener {
      void propertyChanged(PropertyEvent e);
  }
  public class ListenerExample {
    private Listener listeners = new ArrayList<>();
  private String prop = ″″;
      public void addListener(PropertyListener listener) {
      listeners.add(listener);
  }
    public void removeListener(PropertyListener listener) {
    listeners.add(listener);
  }
    public String getProperty() { return prop;
}
public void setProperty(String value) {
    prop = value;
    raisePropertyChangedEvent(new PropertyEvent(this, ″Property″));
    }
    protected void raisePropertyChangedEvent(PropertyEvent e) {
      for (PropertyListener listener : listeners) {
      listener.propertyChanged(e);
    }
  }
```

Exhibit 2.5.3-13 (08-04-2023)
Java Programming Thread example

Thread Example

```
class NirvanaRunnable implements Runnable {
private ThreadExample example = null;
public NirvanaRunnable(ThreadExample example) {
this.example = example;
}
@Override public void run() {
example.guardedJoy();
}
} class EatingRunnable implements Runnable {
private ThreadExample example = null;
public EatingRunnable(ThreadExample example) {
this.example = example;
} @Override public void run() {
example.notifyJoy();
}
} public class ThreadExample {
private boolean joy = false;
static public void main(String[] args) {
ThreadExample example = new ThreadExample();
Thread nirvanaThread = new Thread(new NirvanaRunnable(example));
nirvanaThread.start();
System.out.println(″Started nirvana thread...″);
Thread eatingThread = new Thread(new EatingRunnable(example));
eatingThread.start();
System.out.println(″Started eating thread...″);
} public synchronized void guardedJoy() {
// This guard only loops once for each special event, which may not
// be the event we’re waiting for. while(!joy) {
try {
wait();
} catch (InterruptedException e) {}
}
System.out.println(″Joy and efficiency have been achieved!″);
}
public synchronized void notifyJoy() {
joy = true;
System.out.println(″Reached joy!″);
notifyAll();
}
}

// Output
Started nirvana thread...
Started eating thread...
Reached joy!
Joy and efficiency have been achieved!
```

Exhibit 2.5.3-14 (08-04-2023)
Java Programming Examples of Single Words used for Capitalization Purposes

Java Programming Single Words used for Capitalization Purposes		
Pascal	**Camel**	**Not**
BitFlag	bitFlag	Bitflag
Callback	callback	CallBack
Canceled	canceled	Cancelled
DoNot	doNot	Don’t
Email	email	EMail
Endpoint	endpoint	Endpoint
FileName	fileName	Filename
Gridline	gridline	GridLine
Hashtable	hashtable	HashTable
Id	id	ID
Indexes	indexes	Indices
LogOff	logOff	LogOut
LogOn	logOn	LogIn
Metadata	metaData	MetaData,
Multipanel	multipanel	MultiPanel
Multiview	multiview	MultiView
Namespace	namespace	NameSpace
Ok	ok	OK
Pi	pi	PI
Placeholder	placeholder	PlaceHolder
SignIn	signIn	SignOn
SignOut	signOut	SignOff
UserName	userName	Username
WhiteSpace	whiteSpace	Whitespace
Writable	writable	Writable

Exhibit 2.5.3-15 (08-04-2023)

C Programming Source Code Template

C Programming Source Code Template
/**
//
// Internal Revenue Service
// For Official Use Only
//
// Filename: Filename
// Description: Describe the purpose of the objects in the file,
// followed, in the case of source files, by a list
// of functions whose definitions appear in the file
// Related Files: An identification of any routines or files that
// this file may require
// Restrictions/ Known special cases where the file may not work
// Problems:
//
// Date Modified: YYYY/MM/DD
// Version id: Revision:
// Author: <First Name> <Last Name>
// Locked by: $Locker$
//
// Revision History: Will be provided by ClearCase
**/

Example of C Programming Header Files number 1

C Programming Header Files number 1
/*h*
* H E A D E R F I L E S *
***/
/* System header files */
#include <stdio.h>

Exhibit 2.5.3-15 (Cont. 1) (08-04-2023)
C Programming Source Code Template

C Programming Header Files number 1

```
                              /*h**********************************************************
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
/* User include files */
#include “archive.h”
#include “acdbApi.h”
#include “scdbApi.h”
```

C Programming Example of Defines number 1

C Programming - Defines number 1

```
*              D E F I N E S                    *
/*d**********************************************************
************************************************************/
/* Debug flags */
#ifdef FOR_MAC
              #define SIGALRM 14
#end-if
/* Constants */
#define SUCCESS 0
#ifndef TRUE
              #define TRUE 1
              #define FALSE 0
#end-if
/* Macros */
#define MIN(a,b) ((a)<(b)) ? (a) : (b)
```

Exhibit 2.5.3-15 (Cont. 2) (08-04-2023)
C Programming Source Code Template

C Programming example of TypeDef Number 1

C Programming - TypeDef Number 1

```
/*t**********************************************************
*           T Y P E D E F                                           *
************************************************************/
typedef struct ECT_REG_HEADER_S
{
char *pFirstEntry; /* ptr to 1st registry entry */
int NumEntries; /* number of entries */
} ECT_REG_HEADER_T;
```

C Programming example of Enums number 1

*** E N U M S ***

```
/*e***********************************************************
************************************************************/
                                        enum TAX_FORMS_E
                                        {
                                        F_1040 = 0,
                                        F_1065,
                                        F_941_ELF,
                                        F_941_OLF };
```

C Programming Example of Definitions number 1

*** DEFINITIONS***

```
/*g***********************************************************
************************************************************/
/* External data */
extern char *pStr; /* comments */
extern int GlobalExt; /* comments */
/* Non-static data */
```

Exhibit 2.5.3-15 (Cont. 3) (08-04-2023)
C Programming Source Code Template

```
* DEFINITIONS*
int DataGl; /* comments */
char *pStr; /* comments */
/* Static data */
static int DataGl; /* comments */
static char *pStr; /* comments */
```

C Programming Function Examples number 1

```
* F U N C T I O N P R O T O T Y P E S (alphabetized) *
/*fp**********************************************************
*************************************************************/
int FunctionName1(int par1, char *par2_p);
Void FunctionName2(int par1, char *par2_p);

*
/*f*******************************************************************
* Function Name: FunctionName1
* Description: A description of the major task(s) performed by
* routine. It should be a series of one or more
* simple verb/object statements
* Input parameters : par1 - description
*       par2_P - description
* Output parameters: *par2_p - description
*       Function return - SUCCESS or FAIL
*********************************************************************/
int FunctionName1(int par1, char *par2_p)
{
       /* LOCAL VARIABLES and CONSTANTS*/
       /* FUNCTION BODY */
       return return_code;
}
```

Exhibit 2.5.3-15 (Cont. 4) (08-04-2023)
C Programming Source Code Template

Example of C Function with Multiple Routines

*** Function Name: FunctionName2**

```
/*f************************************************************************
* Description: A description of the major task(s) performed by
* routine. It should be a series of one or more
* simple verb/object statements
* Input parameters : par1 - description
* par2_P - description
* Output parameters: *par2_p - description
************************************************************************/
void FunctionName2(int par1, char *par2_p)
{
/* LOCAL VARIABLES and CONSTANTS*/
/* FUNCTION BODY */
}
```

Exhibit 2.5.3-16 (08-04-2023)
C Language Header File Template

C Language Header File

// Description: Describe the purpose of the objects in the file,

```
/****************************************************************************
//2
// Internal Revenue Service
// For Official Use Only
//
// Filename: Filename
//
followed, in the case of source files, by a list
// of functions whose definitions appear in the file
// Related Files: An identification of any routines or files that
// this file may require
// Restrictions/ Known special cases where the file may not work
// Problems:
//
// Date Modified: Date: YYYY/MM/DD
// Version id: Revision:
// Author: Author: <First Name> <Last Name>
// Locked by: $Locker:$
//
// Revision History: Will be provided by ClearCase
****************************************************************************/
```

C Language Defines Template number 2

```
* D E F I N E S Template                                          *
#ifndef TEMPLATE
#define TEMPLATE
/**************************************************************
**************************************************************/
/* Debug flags */
```

Exhibit 2.5.3-16 (Cont. 1) (08-04-2023)
C Language Header File Template

```
* DEFINES Template *
#ifdef FOR_MAC
#define SIGALRM 14
#END-IF
/* Constants /
#define SUCCESS
0
#ifndef TRUE
#define TRUE 1
#define FALSE 0
#END-IF
/* Macros */
#define MIN(a,b) ((a)<(b)) ? (a) : (b)
```

C Programing Example of TypeDefs number 2

```
* TYPEDEFS Template *
/**************************************************************
**************************************************************/
typedef struct ECT_REG_HEADER_S
{
char *pFirstEntry; /* ptr to 1st registry entry */
int NumEntries; /* number of entries */
} ECT_REG_HEADER_T;
```

C Programming Example of Enums number 2

```
* ENUMS *
/**************************************************************
**************************************************************/
enum TAX_FORMS_E
        {
        F_1040 = 0,
```

Exhibit 2.5.3-16 (Cont. 2) (08-04-2023)
C Language Header File Template

```
*                                   E N U M S                                   *
          F_1065,
          F_941_ELF,
          F_941_OLF
          };
```

C Programming Example of Functions

```
*         F U N C T I O N P R O T O T Y P E S (alphabetized) *
*********************************************************************/
          #END-IF /* TEMPLATE */
          void FunctionName2(int par1, char *par2_p);
int FunctionName1(int par1, char *par2_p);
/*********************************************************************
```

Exhibit 2.5.3-17 (08-04-2023)
Acronyms and Terms

Acronyms and Terms

Acronym	Terms
ACIO	Assistant Chief Information Officer
ACL	Access Control List
AD	Application Development
AES	Advanced Encryption Standard
ALC	Assembler Language Code
ALM	Application Lifecycle Management
ASCII	American Standard Code for Information Interchange
API	Application Programming Interface
APTCA	Allow Partially Trusted Caller Attribute
ASLR	Address Space Layout Randomization
BI	Business Intelligence
CAS	Code Access Security
CAT	Code Analysis Tool
CCoE	Cloud Center of Excellence
COT	Commercial Off-The-Shelf
CERT	Computer Emergency Response Team
CICS	Customer Information Control System
COBOL	Common Business-Oriented Language
COE	Common Operating Environment
CSRF	Cross Site Request Forgery
CUI	Controlled Unclassified Information
CWE	Common Weakness Exposure
DES	Data Encryption Standard
DPAPI	Data Protection API
EA	Enterprise Architecture
EAR	Enterprise Archive
ECL	Executive Control Language
EUP	Employee User Portal
EOF	End of File
FIT	Final Integration Testing

Exhibit 2.5.3-17 (Cont. 1) (08-04-2023)
Acronyms and Terms

Acronym	Terms
FIPS	Federal Information Processing Standard
GAC	Global Assembly Cache
GAO	Government Accountability Office
GC	Garbage Collector
IBM	International Business Machines Corporation
ISBN	International Standard Book Number
HLASM	High-Level Assembler
HTML	Hypertext Markup Language
I10N	Localization: I, then 10 letters, then N
I18N	Internationalization: I, then 18 letters then N
IC	Internal Controls
IDE	Integrated Development Environment
IT	Information Technology
ISO	International Organization for Standardization
JAAS	Java Authentication and Authorization Service
JCL	Job Control Language
JNI	Java Native Interface
JRE	Java Runtime Environment
JSON	Java Script Object Notation
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
MASM	Meta-Assembler
MSDN	Microsoft Developer Network
NIS	Network Information Services
NLP	Natural Language Processing
NIST	National Institute of Standards and Technology
OMB	Office of Management and Budget
OS	Operating System
OWASP	Open Web Application Security Project
PCLIA	Privacy and Civil Liberties Impact Assessment
PII	Personal Indentifiable Information

Exhibit 2.5.3-17 (Cont. 2) (08-04-2023)
Acronyms and Terms

Acronym	Terms
PNG	Portable Network Graphics
QA	Quality Assurance
QSAM	Queued Sequential Access Method
RTM	Requirement Traceability Matrix
SAT	System Acceptance Test
SDK	Software Development Kit
SQL	Structured Query Language
SSP	Systems Security and Privacy
TLS	Transport Layer Security
UNISYS	UNIVAC Systems Corporation
WAR	Web Archive
VLAN	Virtual Local Area Network
VSAM	Virtual Sequential Access Method
XML	Extensible Markup Language

Exhibit 2.5.3-18 (08-04-2023)
Terms and Definitions

Terms	Definitions
Application Programming Interface	The word Application refers to any software with a diverse function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.
Case Structure	A control structure used when there are numerous paths to be followed depending on the contents of a given field or variable.
Cloud Center of Excellence	An organizational team assembled with the goal to implement the best practices, governance, architecture, and training necessary to ensure efficient cloud adoption across the enterprise.
Commercial Off-The-Shelf	COTS products are designed to be easily installed and to interoperate with existing system components. Almost all software bought by the average computer user fits into the COTS category: operating systems, office product suites, word processing, and e-mail programs as examples.
Conditional Statement	An instruction that use the word **"IF"** to test for the existence of a condition.
Enterprise Archive	A compressed file that contains the libraries, enterprise beans and JAR files that the application requires for deployment.
Extensible Stylesheet Language Transformation (XML)	A language for transforming XML document into other XML documents or other formats such as HTML for web pages, plain text or XSL Formatting Objects which may be converted to other formats, such as PDF, PostScript and PNG supported in modern web browsers.
FedRAMP	Goverment-wide program established in 2011 to provide a cost-effective, risk-based approach for the adoption and use of cloud services by the federal government. FedRAMP empowers agencies to use modern cloud technologies, with an emphasis on security and protection of federal information.
Framework	A set of functions within a system, and how they interrelate.
Government Off-The-Shelf	The product is developed by the technical staff of the government agency where is was created. It is sometimes developed by an external entity, but with funding and specification from the agency.
Lightweight Directory Access Protocol	Open vendor industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. Allows sharing of information about users.
Localization	Modification of any application to allow the support of any global language and related local settings

Exhibit 2.5.3-18 (Cont. 1) (08-04-2023)
Terms and Definitions

Terms	Definitions
High-Level Assembler	IBM's assembler programming language and the assembler itself for the IBM z/OS, z/VM, OS/390, MVS, VM and VSE operating systems. Released June 1992.
Hypertext Markup Language	An application of the Standard Generalized Markup Language, which is the international standard for Markup, and is the primary markup language used to write content on the web.
Internal Controls	Management controls that provide reasonable assurance that obligations and cost are in compliance with applicable laws, funds, property; and other assets are safeguarded against waste, loss, unauthorized use or misappropriation.
Internationalization	Internationalization is the process of designing and developing your software or mobile application product so it can be adapted and localized to different cultures, regions, and languages. Internationalization ensures your software is localizable. This is usually done by software developers and engineers.
International Organization for Standardization	The International Organization for Standardization is an independent, non-governmental organization, the members of which are the standards organizations of the 164 member countries. It is the world's largest developer of voluntary international standards and facilitates world trade by providing common standards between nations. Over twenty thousand standards have been set covering everything from manufactured products and technology to food safety, agriculture and healthcare.
Java Bean	An object-oriented programming interface that allows you to build re-usable applications or program building blocks called components. Java Bean can be deployed in a network on any major operating system platform.
Java Native Interface	Programming framework that enables Java code running in a Java Virtual Machine to call and be called by a native application (programs specific to a hardware and operating system platform and libraries written in other languages e.g., C, C++ and Assembler.
Java Runtime Environment (JRE)	Also known as Java Runtime is part of the Java Development kit that contains: Java Virtual Machine(JVM), Java Platform core classes, and supporting Java platform libraries.
Java Script Object Notation	A text-based human readable interchanged format used for representing simple data structures and objects in Web browser based code.

Exhibit 2.5.3-18 (Cont. 2) (08-04-2023)
Terms and Definitions

Terms	Definitions
Natural Language Processing	Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language. NLP draws from many disciplines, including computer science and computational linguistics, in its quest to fill the gap between human communication and computer understanding.
Java Virtual Machine(JVM)	JVM has two functions: to allow Java programs to run on any device or OS, (known as the “Write Once, run Anywhere” principle), and to Manage and optimize program memory.
Personal Indentifiable Information	Any representation of information that permits the identity of an individual to whom the information applies to be reasonably inferred by either direct or indirect means. Further, PII is defined as information: (i) that directly identifies an individual (e.g., name, address, social security number or other identifying number or code, telephone number, email address, etc.)
OWASP	The Open Web Application Security Project is a nonprofit organization focused on improving the security of software. OWASP provides impartial information about AppSec to individuals, corporations, universities, government agencies, and other organizations worldwide.
Privacy and Civil Liberties Impact Assessment	An analysis of how information is handled: to ensure conformance to applicable legal, regulatory, and policy requirements regarding privacy. To determine the risks and effects of creating, collecting, using, processing, storing, maintaining, disseminating, disclosing, and disposing of PII maintained in an electronic information system, and to examine and evaluate privacy controls for an information system and alternative processes for handling PII to mitigate potential privacy concerns.
Portable Network Graphics	A raster graphics file format that supports loss-less data compression, PNG was created as an improved replacement for Graphics Interchanged Format (GIF)
Program	A set of instructions that operate on input data and convert it to output.
QSAM	An access method that, (a) an extended version of the basic sequential-access method and (b) forms a queue of (i) input data blocks that are awaiting processing or (ii) output data blocks that have been processed and are awaiting transfer to an output unit, an auxiliary storage unit, or an external storage unit.
Refactor	Altering an application’s source code without changing its external behavior. The purpose of code refactoring is to improve some of the nonfunctional properties of the code, e.g., readability, complexity, maintainability, and extensibility.

Exhibit 2.5.3-18 (Cont. 3) (08-04-2023)
Terms and Definitions

Terms	Definitions
Systems Security and Privacy	Security is about protecting data from malicious threats, whereas privacy pertains to using data responsibly. Privacy is more complex than security, involving protection of sensitive data in both electronic and physical forms.
Transport Layer Security	TLS is an Internet Engineering Task Force cryptographic protocol designed to provide secure data sent between applications over the internet.
VSAM	An IBM disk file storage scheme first used in s/370 and virtual storage. VSAM comprises three access methods: keyed sequential data set (ksds).
Web Archiving	Web archiving is the process of collecting portions of the World Wide Web, preserving the collections in an archival format, and then serving the archives for access and use.
z/OS	A 64-bit operating system for IBM mainframes, produced by IBM, and derived by successor OS/390.

Exhibit 2.5.3-19 (08-04-2023)
Assembler Language Code (ALC) Standards and References

- IBM Systems Standard Manual Version 5
- BCPA 40 ALC Student Guide
- IRS Messages and Codes Edition 2
- High Level Assembler for z/OS & z/VM & z/VSE Language Reference Version 1R6
- High Level Assembler for z/OS & z/VM & z/VSE Programmer's Guide Version 1R6
- UNISYS ClearPath OS2200 Meta-Assembler (MASM) Programming Reference Manual Level 6R3J
- UNISYS ClearPath OS2200 Executive Control Language (ECL) and FURPUR Reference Manual
- Assembler H Version 2 Application Programming Guide , SC26-4036
- Assembler H Version 2 Application Programming Language Reference, SC26-4037
- MVS JCL Reference, GC28-1352
- MVS/XA Linkage Editor & Loader User's Guide, GC26-4143
- MVS/XA Message Library: System Messages Vol 1, GC28-1376
- MVS/XA Message Library: System Messages Vol 2, GC28-1377
- MVS/XA Message Library: System Codes, GC28-1157
- MVS/XA Data Administration, GC26-4149
- MVS/XA Data Administration: Utilities, GC26-4150
- MVS/XA Data Administration: Macro Instruction Reference, GC26-4141
- MVS/XA Utilities Messages , GC26-4021
- MVS/XA TSO Terminal User's Guide, GC28-1274
- TSO/E TSO Command Language Reference, GC28-0646
- ISPF/PDF Program Reference, SC34-2139

Exhibit 2.5.3-20 (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

Exhibit 2.5.3-20 (Cont. 1) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

			#
			# # # # # #
			# # # # # # # # # # # # # #
			# # # # # #
			# #
			# # # # # # # # # # # #
			# # # #

Exhibit 2.5.3-20 (Cont. 2) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

Exhibit 2.5.3-20 (Cont. 3) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

Exhibit 2.5.3-20 (Cont. 4) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

			#
			# # # # # # # # # #
			# # # # # #
			#
			# # # # # # # # # # # # # # # # #
			# # # # #
			# # #
			#

Exhibit 2.5.3-20 (Cont. 5) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

			#
			# # # # # # # # # #
			#
			# # #
			#
			# # # # # #
			#
			# #

Any line marked with a #
is for **Official Use Only**

Exhibit 2.5.3-20 (Cont. 6) (08-04-2023)
IT Application Security and Development - IRS and OWASP Standards/Guidelines

Exhibit 2.5.3-20 (Cont. 7) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

			#
			# # # # # # #
			# # #
			#
			# # # # #
			# # # # #
			# # # # #

Exhibit 2.5.3-20 (Cont. 8) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

Exhibit 2.5.3-20 (Cont. 9) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

			#
			# # # # # # # # # # # # # # # # #
			#
			# # # # #
			# # # #
			# # # # #
			# # # # # # # #
			# # #

Exhibit 2.5.3-20 (Cont. 10) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

			#
			# # # # # # #
			# # #
			# # # # #
			# # #
			# # # # #
			# # # # #
			# # # # #
			#
			# # #

Exhibit 2.5.3-20 (Cont. 11) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

Exhibit 2.5.3-20 (Cont. 12) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

			#
			# # # # # # # # # # #
			# #
			# # # # # #
			# # # #
			# # # # # #
			# # #
			# #
			# # #
			# # #
			# # #

Exhibit 2.5.3-20 (Cont. 13) (08-04-2023)
IT Application Security and Development - IRS and OWASP Standards/Guidelines

Exhibit 2.5.3-20 (Cont. 14) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

Any line marked with a #
is for **Official Use Only**

Exhibit 2.5.3-20 (Cont. 15) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

Exhibit 2.5.3-20 (Cont. 16) (08-04-2023)

IT Application Security and Development - IRS and OWASP Standards/Guidelines

			#
			# # #
			# # # # # #
			# # # # #

#

Exhibit 2.5.3-21 (08-04-2023)

COBOL Compiler 6.2 Runtime Warning Messages

Exhibit 2.5.3-21 (Cont. 1) (08-04-2023)
COBOL Compiler 6.2 Runtime Warning Messages

#
#
#

#
#
#
#
#
#
#
#
#

#

Exhibit 2.5.3-22 (08-04-2023)

#

	# #
1.	#
2.	#
3.	#
4.	# #
5.	#
6.	#
7.	#
8.	#
9.	# #

#

#

#

#

#

#

#

#

Exhibit 2.5.3-24 (08-04-2023)

IRS Power Platform Available Environments

Exhibit 2.5.3-24 (Cont. 1) (08-04-2023)
IRS Power Platform Available Environments

Exhibit 2.5.3-24 (Cont. 2) (08-04-2023)
IRS Power Platform Available Environments

Exhibit 2.5.3-24 (Cont. 3) (08-04-2023)
IRS Power Platform Available Environments
