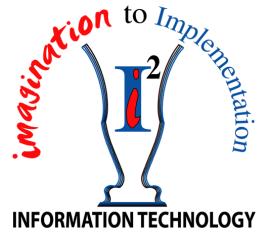


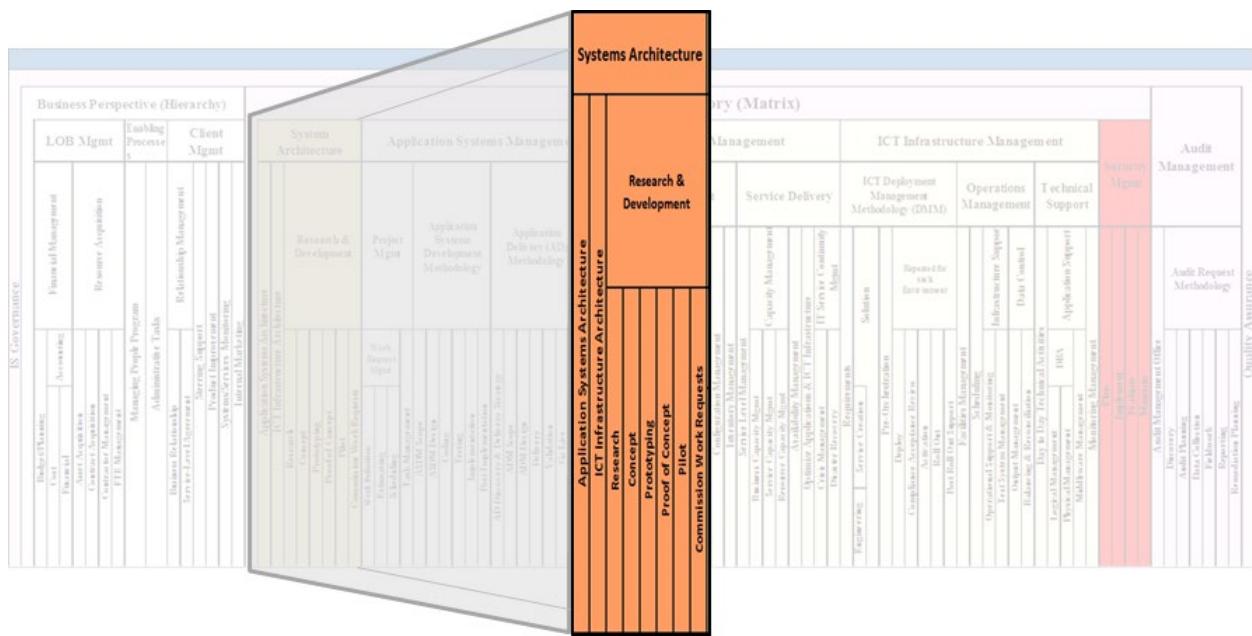


South Carolina

*BlueCross BlueShield of South Carolina
is an independent licensee of the
Blue Cross and Blue Shield Association*



Systems Architecture



Information Systems Standards Manual

Table of Contents

Table of Figures

List of Tables

Chapter 1 General Architectural Standards

1.1 Strategic Innovation Framework Review

1.2 Solution Detail Review Process

 1.2.1 SDRP Criteria

1.3 SDRP Committee

1.4 System Master Index

 1.4.1 Hardware Record Standards

 1.4.2 Software Record Standards

 1.4.3 Architectural Classification

 1.4.4 Source Code Storage Location

 1.4.5 System Documentation Location

 1.4.6 Presentation Layer Documentation

1.5 Vendor Software Usage

 1.5.1 Vendor Alert Process

Chapter 2 Free and Open Source Software

2.1 Overview

2.2 Open Source Life Cycle

 2.2.1 I/S Acquisition of FOSS for Evaluation

 2.2.2 Usage of FOSS

Chapter 3 Application Systems Architecture

3.1 Strategic Application Architecture Committee

3.2 Application Roadmaps

Table of Contents

Printed forms of this document are uncontrolled and may be obsolete. It is the responsibility of the user to ensure that any printed copy is the same revision as the online version.

Copyright ©2023, BlueCross BlueShield of South Carolina. All rights reserved. BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association.

- 3.2.1 Roles for Application Roadmaps for Vendor Products
- 3.2.2 Application Roadmaps Overview
- 3.2.3 Application Roadmap Content Standards
- 3.2.4 Vendor Application Roadmap Approval
- 3.2.5 Aggregated Application Roadmap Approval
- 3.2.6 Storage of Aggregated Application Roadmaps
- 3.2.7 Roadmaps Work Initiation

Chapter 4 ICT Infrastructure Architecture

- 4.1 Strategic Technology Architecture Committee
- 4.2 Technology Roadmaps
 - 4.2.1 Roles for Technology Roadmaps
 - 4.2.2 Technology Roadmaps Overview
 - 4.2.3 Technology Roadmap Content Standards
 - 4.2.4 Roadmap Approval
 - 4.2.5 Storage of Roadmaps
 - 4.2.6 Roadmaps Work Initiation

Chapter 5 Research and Development

- 5.1 Overview
- 5.2 Variability in the R&D Methodology
- 5.3 Similarity in the R&D Methodology
- 5.4 R&D Phases

Chapter 6 Workstation Hardware/Software Procedures

- 6.1 Requesting Workstation Hardware/Software

Chapter 7 Concepts & Techniques

- 7.1 Concepts
 - 7.1.1 Architectural Concepts

Table of Contents

Printed forms of this document are uncontrolled and may be obsolete. It is the responsibility of the user to ensure that any printed copy is the same revision as the online version.

Copyright ©2023, BlueCross BlueShield of South Carolina. All rights reserved. BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association.

7.2 Common Platform Techniques

- 7.2.1 Single Source Application Design
- 7.2.2 Application Cloning
- 7.2.3 Regional Processing Number (RPN) Driven Code
- 7.2.4 Modules by Function
- 7.2.5 CICS Screen Design Consideration for All Input Channels
- 7.2.6 User-Centered Design
- 7.2.7 Concept Diagrams
- 7.2.8 Informational Database
- 7.2.9 Paperless Processing
- 7.2.10 Workflow Automation
- 7.2.11 Document Archival
- 7.2.12 Software Stubbing

7.3 Host Platform Techniques

- 7.3.1 CICS Application Security
- 7.3.2 CICS Application Transaction Router Processing
- 7.3.3 SuperRouter Processing
- 7.3.4 Minimizing the Use of Escape in Aps
- 7.3.5 Common I/O and Edit Modules
- 7.3.6 TIOA Utilization
- 7.3.7 CICS Temporary Storage Queue (TSQ) Usage
- 7.3.8 DB2 PLANBINDs and PACKBINDs
- 7.3.9 History Generating Applications and DB2 Databases
- 7.3.10 IMS Database Audit Trail and Purge Processing
- 7.3.11 IMS Checkpoint/Restart Processing
- 7.3.12 Handling Mixed Case Characters in Online Host Applications

Table of Contents

Printed forms of this document are uncontrolled and may be obsolete. It is the responsibility of the user to ensure that any printed copy is the same revision as the online version.

Copyright ©2023, BlueCross BlueShield of South Carolina. All rights reserved. BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association.

7.3.13 Large Working Storage Areas

7.3.14 IBM DB2 Analytics Accelerator (IDAA)

7.4 Non-Host Platform Techniques

7.4.1 Non-Host Server Logging Process

Table of Contents

Printed forms of this document are uncontrolled and may be obsolete. It is the responsibility of the user to ensure that any printed copy is the same revision as the online version.

Copyright ©2023, BlueCross BlueShield of South Carolina. All rights reserved. BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association.

Table of Figures

Figure 3-1 Architecture Specification Process

Figure 4-1 Architecture Specification Process

Figure 5-1 Research and Development (R&D) Phases

Figure 5-2 Variability in the R&D Methodology

Figure 7-1 Concepts

Figure 7-2 Clients Leveraging Modular Sub-functions

Figure 7-3 Reuse of Single Source Code

Figure 7-4 Reuse of Common Code by Clients and Input Channels

Figure 7-10 Automated Document Generation System (ADGS) Concept Diagram

Figure 7-11 Full Copy Overlay of Informational

Figure 7-12 Apply Selected Changes to Informational

Figure 7-13 Mirror Databases

Figure 7-14 Paperless Processing Example

Figure 7-15 Example Workflow Diagram

Figure 7-16 Software Stub — The software stub sits between the consumer and provider applications and will respond as if it is the provider. The consuming application believes that it is actually communicating with the provider application.

Figure 7-17 Presentation System Calling a Web Service — The software stub sits between the presentation application and the Web Service. The software stub will respond to the presentation application as if it were a live response from the Web Service.

Figure 7-18 COBOL Program Calling a Web Service via MQ — The software stub sits on the MQ Queue Manager and will respond to the MQ message sent by the Host program as if it was a response from the Web Service.

Figure 7-19 Presentation System Scenario-Based Test Data — The software stub sits between the presentation application and the Web Service. Based on rules, the software stub will allow the Web Service or Virtual Data respond. Allowing for more complete testing of edge cases.

Figure 7-20 CICS Application Router Flow

Figure 7-21 SuperRouter Flow

Figure 7-22 Common I/O and Edit Modules as Subprograms

Figure 7-23 I/O (Access) Modules as Subprograms

Figure 7-24 Edit Modules as Subprograms

Figure 7-25 Simple Type Table Sample

Figure 7-26 Complex Type Table Sample

Table of Figures

Printed forms of this document are uncontrolled and may be obsolete. It is the responsibility of the user to ensure that any printed copy is the same revision as the online version.

Copyright ©2023, BlueCross BlueShield of South Carolina. All rights reserved. BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association.

List of Tables

Table 1-1 SDRP Criteria (All LOBs)

Table 1-2 Software Record Ownership

Table 2-1 Free and Open Source Software Terminology

Table 3-1 Release Categories

Table 5-1 R&D Phases

Table 7-2 Commercial Systems RPNs

Table 7-3 PGBA RPN/Plan Codes

Table 7-4 History Generating Applications and DB2 Databases Types

Table 7-5 Macros Used for Checkpoint/Restart Technique

Chapter 1 General Architectural Standards

1.1 Strategic Innovation Framework Review

Strategic Innovation Framework Review (SIFR) is a governing committee that reviews and discusses external influencing factors and their impact on technical strategy and direction. Research is done using the Innovation Thinking Framework, resulting in a white paper that is presented and reviewed with the CIO. Upon approval, a formal Research and Development (R&D) effort or a normal work effort can be initiated.

1.2 Solution Detail Review Process

The Solution Detail Review Process (SDRP) consists of triggering the involvement of the SDRP Committee to evaluate Work Requests based on Client Management- and Line of Business (LOB) Manager-defined SDRP criteria. The Work Requests are monitored by the Estimating Team and the SDRP administration. This involvement may occur upon initiation of a Work Request, when significant change controls have occurred, or any other event outlined within the SDRP criteria. In addition, the SDRP is triggered as part of the proposal development process for the I/S-related deliverables of the proposed solution.

The following benefits are derived from the SDRP:

- Enterprise Architect involvement and oversight determination
- Confirmation of involved application and infrastructure components
- Agreement of proposed solution design concepts
- Identification of other development activity overlaps that can be leveraged across LOBs
- Identification of potential Client sensitivities
- LOB Management and CIO awareness of any cost or technical sensitivities

The SDRP Committee may request an (Initial) Concept Diagram, as part of the HLE development. When the SDRP is triggered after the initiation of the Work Request, the evaluation will review the current design concepts, reconfirm the involved concept diagrams, and review all assumptions.

The SDRP will include the involvement of any staff member that is deemed necessary. When invoked, the SDRP review must be completed within three (3) business days.

1.2.1 SDRP Criteria

The SDRP criteria are determined by Line of Business (LOB) Managers. The table below (Table 1-1) lists the criteria by activity and qualifiers:

SDRP Criteria (All LOBs)	
Activity	Qualifiers
WDD/HLE	<p>Size of Effort:</p> <ul style="list-style-type: none"> • Any Work Estimate $\geq 1,000$ hours • RD-Funded Work <p>Scope of Effort:</p> <ul style="list-style-type: none"> • New Product or Vendor • BCA or Federal/State Mandate • Any effort with Corporate Planning & Strategic Services (CPSS) or vendor involvement • Vendor Alerts — New or significant vendor software upgrades
FLE/RFx	All FLEs or RFxs
ASM Efforts	<p>Executive Oversight:</p> <ul style="list-style-type: none"> • Effort reports to CIO

SDRP Criteria (All LOBs)

Activity	Qualifiers
Significant Scope Change:	
ASM Change Controls	Significant Scope Change: <ul style="list-style-type: none"> • Work Estimate at the end of Scope or exceeds 150% of HLE or FLE • Identification of a new application requiring an additional architect • Addition of a new or significant change (as defined by an Architect) to a user interface (web, desktop, CICS, Vendor) • New Platform/Technology Service • Any inclusion of Cloud platform or solution (Private or Public) • Addition of new or significantly changing database • Addition of CPSS or vendor involvement

Table 1-1 SDRP Criteria (All LOBs)

1.3 SDRP Committee

The charter of the Solution Detail Review Process (SDRP) Committee is to review:

- Those Work Requests that meet the SDRP criteria.
- Any other activity on an as-needed basis identified by senior management.

The purpose of this committee is to ensure that the work efforts reviewed have architecturally sound development solutions that can be accurately estimated and leveraged for future efforts.

1.4 System Master Index

The System Master Index (SMI) contains an inventory of both the hardware types that comprise the infrastructure platforms and all the software that is approved to execute on these platforms. In addition, any software that is specifically restricted from use will be listed on SMI with a Support Status of *Not Authorized*.

SMI provides identifying information for each of the hardware and software elements that comprise our systems and defines the elements classification within the overall I/S Architecture. SMI also provides information on who provides support for each element.

SMI is an index in that it identifies where to find additional information such as source code, system documentation and Disaster Recovery Plans.

Systems are often related to each other. In many cases, subsystems or an internally developed code base is created to support another system. These are identified using a parent/child relationship in SMI. Child systems rely on the parent to exist and have no function without the parent.

1.4.1 Hardware Record Standards

Technology Owners are the *record owners* for hardware entries. They are responsible for ensuring that all hardware elements that comprise their assigned platforms are listed on SMI. The Technology Owner should submit new hardware entries in SMI through BOIT to be approved by the Platform Owner and Enterprise Architecture.

The following fields are required in SMI for hardware:

- Name
- Description
- Technology Owner
- Computing Platform
- Computing Platform Detail

Hardware records on SMI must have the architectural classification Category of *Technical Infrastructure Architecture* and a Technology Stack of *Hardware*.

1.4.2 Software Record Standards

All software approved for use by BlueCross BlueShield of South Carolina (BlueCross) and its subsidiaries is required to have SMI records. This includes all purchased software, freeware, in-house developed software, and software that is externally hosted by a vendor.

In addition, all system prefixes on the CDMSP.VSYSIDNZ table must have corresponding records in SMI.

Any software not specifically listed on SMI must be approved prior to installation. In-house developed software gains approval during the development process. Vendor software is approved through the Vendor Alert Process. Refer to *Systems Architecture > General Architectural Standards > Vendor Software Usage >Vendor Alert Process* for details.

1.4.2.1 Record Ownership

The *Software Status* field on the SMI contains the following values:

- **Standard Supported** — Software that is approved for use and is supported by I/S.
- **End-User Supported** — Software that is approved for use and is supported by an End-User. The End-User contact will be specified. Refer to *Adaptive Change > Standards for Non-I/S Areas > End-User Computing > End-User Supported Software* for details on the End-User support responsibilities.
- **Not Authorized** — Software that is specifically restricted from use on any BlueCross computing system.

To ensure that responsibility for software is clearly identified, Record Owners are assigned in order to provide accountability. The table below (Table 1-2) is used to identify the software record owner for the items in the SMI.

Software Record Ownership

Record Owner	Support Status	Architectural Category	Software Type
Client Advocate	End-User Supported	N/A	All software
System Manager (or delegated System Expert)	Standard Supported	Business Systems Architecture or IT Business Systems Architecture	Vendor Software
System Manager (or delegated System Expert)	Standard Supported	Business Systems Architecture or IT Business Systems Architecture	Internally Developed
System Manager (or delegated Technology Owner)	Standard Supported	Technical Infrastructure Architecture	All software

Table 1-2 Software Record Ownership

The Record Owner for all software hosted at CDS is the System Manager on the SMI record.

Any prefix on the CDMSP.VSYSIDNZ table that is not classified as one of the above categories is classified as *Administrative*, and the Record Owner is the manager that supports the software. For example, an application may have separate prefixes for its test jobs, and these prefixes will be classified as *Administrative*.

Record Owners are responsible for:

- Ensuring that the software is properly classified within the architecture.
- Ensuring that the SMI records are kept up to date on an ongoing basis by:
 - Submitting requests in SMI in BOIT to add new records when new software is installed on any computing platform.
 - Submitting necessary update requests in SMI in BOIT or approving changes requested by others.
- Conducting a 90-day review of the information in the SMI and attesting to the accuracy of the record.

The instructions on how to complete an SMI update request are available on the Enterprise Architecture **SharePoint Online** site.

Updates to records to transfer responsibility require the consent of both the incoming party and the outgoing party (or their management). This applies to the following fields:

- Manager Code
- Technology Owner
- Architect

The Record Owner is responsible for obtaining this approval prior to submitting an update request in SMI in BOIT. If agreement cannot be reached, SMI.ADMIN should be notified, and the Process Audit area will facilitate the resolution.

An Enterprise Architect can authorize changes to the Architect assigned to an SMI record.

1.4.3 Architectural Classification

All SMI entries for Standard Supported software must specify the Computing Platform and Computing Platform Detail on which the software executes and the following:

- Applications in the categories of Business Systems Architecture or IT Business Systems Architecture must include the architectural major and minor.
- Software categorized as Technical Infrastructure Architecture must include the Technology Stack and component layer for the software.

The Record Owner is responsible for determining the architectural classification and ensuring that the applicable Platform Owner or Architect agrees with the classification before submitting new records or updates in SMI in BOIT. In order to ensure accuracy, Enterprise Architect will provide concurrence on all architectural classifications.

Architectural classification is not required for software designated as End-User Supported or Not Authorized. The category, Technology Stack and component layer will all receive values of *N/A* for *Not Applicable*. The Computing Platform and Computing Platform Detail are still required entries.

1.4.4 Source Code Storage Location

Records for Standard Supported software with a category of Business Systems Architecture or IT Business Systems Architecture must specify which Software Configuration Management database houses the source code (or executable) as well as the location of the software within the Software Configuration Management database. The Record Owner is responsible for providing this information in SMI in BOIT.

1.4.5 System Documentation Location

Records for Standard Supported software with a category of Business Systems Architecture or IT Business Systems Architecture must specify where the applicable system documentation is housed and the name of the Disaster Recovery Plan for the software. The Record Owner is responsible for providing this information in SMI in BOIT.

1.4.6 Presentation Layer Documentation

Record Owners for Standard Supported Voice Response Unit (VRU)- and browser-based software must also provide the following information:

- Constituents — The groupings of people for whom a given Presentation Layer was created
- Application name
- Application function
- Data Source Type — Indicates where the data is obtained for the identified function within the identified application
- Data Source Name — Examples include what a program is being called, DB2 table names, MQ channels, CICS transactions, etc.
- Transaction Description — Narrative description of the CICS transaction if one was identified as the Data Source
- Data Business Partner when Data Source Type is LINK

1.5 Vendor Software Usage

Vendor software refers to application and infrastructure software that is purchased under a commercial license from a third-party vendor. Vendor software usage requirements include the following:

- All vendor software (applications and infrastructure) in use must have an active support agreement with a vendor. The agreement should hold the vendor accountable for the management of the open source software license and security risk within their products.
- All vendor software in use must have an accompanying entry in the System Master Index (SMI).

Acquisition and use of vendor software requires approval from the Chief Information Officer (CIO) prior to installation. Upgrades to vendor software may require approval from the Solution Detail Review Process (SDRP) Committee prior to installation. For additional information, please refer to *Systems Architecture > Free and Open Source Software > Solution Detail Review Process > SDRP Criteria*. The Vendor Alert Process is a mechanism for requesting approval for both.

In order to properly manage our inventory of internally developed code, code or configuration files developed by BlueCross BlueShield of South Carolina (BlueCross) in support of Vendor software should have a separate SMI entry. The Vendor should be listed as “BCBSSC,” and the entry should be listed as a child to the Vendor Software SMI entry.

1.5.1 Vendor Alert Process

The purpose of the Vendor Alert is to communicate to all interested parties what vendor software is being investigated and what existing vendor software is being considered for upgrades. This provides the interested parties the opportunity to provide input to the evaluators on past experiences with the software, other packages that may perform the same function, or that they have a need for the same type of product that is being evaluated.

A current listing of all software products for which I/S has been made aware and where an active evaluation is in progress is maintained in the Enterprise Architecture Board. Items on the Vendor Alert list are initially reviewed by an Enterprise Architect. This review helps to ensure that the product meets the business requirements and that there is not an existing product that can be leveraged. This review also helps to determine if the proposed product should or should not be internally developed (Build versus Buy decision). If the Enterprise Architect makes a recommendation to proceed with a vendor product evaluation, the item is sent to the CIO via the Research & Development (R&D) Committee.

The Vendor Alert process also includes the steps to assign a Vendor Owner to new I/S vendors. Vendors with \$10,000 or more in anticipated annual dollars spent are assigned a Vendor Owner. There are three levels of vendor owners: Direct Report Vendor Owner; Vendor Owner; and 2nd Level Vendor Owner, who serves as a backup for the Vendor Owner. Client Management will assign the Vendor Owners based on the Vendor Owner Assignment Guidelines published on My e-Work > Fiscal Services > Vendor Owner Documents.

The Vendor Alert/Application Upgrade Notification is available via the I/S Lighthouse Enterprise Architecture EA Services page (I/S Lighthouse > Information Systems Areas > Systems Architecture > EA Services).

For guidelines and a description of Vendor Ownership, please search My e-Work for the term *vendor owner*.

Chapter 2 Free and Open Source Software

2.1 Overview

Free and Open Source Software (FOSS) refers to application or infrastructure code or software not internally developed by BlueCross BlueShield of South Carolina (BlueCross) and is available free of cost. Source code for FOSS is made available publicly for collaborative development. Free and Open Source is made available under licenses by the copyright holder that define the acceptable use of the software.

Like vendor software, FOSS license, security and operational risk must be managed. Unlike vendor software, management of the risk associated with the use of FOSS code is the responsibility of BlueCross. For applications that embed FOSS, this risk is inherited by the application. Accountability for monitoring and remediation of risk due to the use of FOSS is the responsibility of the assigned Technology Owner/System Expert or End User Contact of the technology or application in SMI.

Software obtained under commercial license is not considered FOSS but vendor software under the terms of that license.

Vendor software is generally preferred over FOSS for software proposed to support core business functions. If a third-party solution exists, approval should be obtained through the Vendor Alert Process.

FOSS falls into three categories: Open Source Solutions, Open Source Development Libraries and Open Source Runtime Libraries. These are defined in Table 2-1 below along with some additional FOSS-related terminology.

Free and Open Source Software Terminology

Term	Definition
Contribution	Contribution refers to the adding of code to open source communities.
Distribution	Distribution refers to software physically made available for installation outside the BlueCross computing environment. For example, publishing a mobile application to a mobile application store.
Open Source Development Libraries	Open Source Development Libraries are components of Open Source Software that are used by software developers at build time to perform specific tasks. These components are not packaged with the final application. Examples of Open Source Development Libraries include JUnit (automated unit testing), eslint (code quality checks), and Cobertura (code coverage).
Open Source Index (OSI)	Open Source Index is the system of record for capturing FOSS as <i>authorized</i> or <i>not authorized</i> . It serves the same purpose as the System Master Index (SMI) for Open Source Libraries.

Free and Open Source Software Terminology

Term	Definition
Open Source Review Board (OSRB)	Under the Enterprise Architect Office, the OSRB approves Open Source Software for use in system development.
Open Source Runtime Libraries	Open Source Runtime Libraries are components of Open Source Software that are included by software developers at execution to perform specific functions within the code. These components are packaged with the final application. Examples of Open Source Runtime Libraries include Apache Commons Lang (language library), Angular (web application framework), and Log4J (logging library).
Open Source Solutions	Open Source Solutions are components of Open Source Software that are installed or extracted onto a system to perform a very specific task.

Table 2-1 Free and Open Source Software Terminology

2.2 Open Source Life Cycle

2.2.1 I/S Acquisition of FOSS for Evaluation

For evaluation, FOSS Development and Runtime Libraries must be obtained and stored in the unapproved repository by version number with license information. Direct download from the internet to developer workstations is prohibited.

Requests for FOSS (new or major version upgrades) approval will follow the Open Source Review Board (OSRB) process.

Open Source Solutions will follow the Vendor Alert process with OSRB review as needed.

Free and Open Source Solutions will be listed in the System Master Index (SMI) with the appropriate status.

2.2.1.1 Open Source Review Board

Request for FOSS should be submitted to the Open Source Review Board (OSRB).

The OSRB will ensure that controls are in place to mitigate risk and promote effective usage of Free and Open Source Software (FOSS) within BlueCross. The OSRB will maintain approved FOSS license types and acceptance criteria.

The OSRB will be responsible for communication and approval of exceptions up through the CIO.

OSRB Members

An **Enterprise Architect** will chair the OSRB to ensure that the OSRB is compliant with policy, criteria and architecture to mitigate corporate risk.

A FOSS subject matter expert will stay current in industry FOSS to ensure that policy, criteria, and approvals mitigate security, license and operational risk.

The Security Operations representative and a Security, Risk and Compliance Assurance (SRCA) representative will stay current on FOSS technologies to ensure that policy, criteria and approvals mitigate security, risk and compliance to controls.

The Information Technology Business Systems Architect (ITBS Architect) will ensure that systems effectively support the FOSS request.

The IT Asset Specialist will participate in license compliance activities.

As Needed OSRB Members

The BlueCross Law Department will be engaged by the OSRB as necessary for license categorization. The Law Department evaluates FOSS licenses, determines the license family categorization and guides FOSS license waivers and remediation.

I/S Governance will be engaged when updates to corporate policy or I/S Standards are necessary.

2.2.1.2 License Families

BlueCross categorizes licenses into three families: Permissive, Weak Copyleft (Weak Reciprocal), and Strong Copyleft (Reciprocal).

The OSRB will maintain a table of the specific licenses arranged by category for use during the evaluation.

2.2.1.3 FOSS Approval

All software obtained under an Open Source license must be approved by the OSRB. All Open Source Solutions must have approval from the OSRB and may include approval up through the CISO, CAO/CTO and the CIO as deemed necessary.

2.2.2 Usage of FOSS

ITBS will store Open Source Development Libraries and Open Source Runtime Libraries in the appropriate repository for Open Source binaries. System Developers will be able to use the authorized FOSS from those repositories.

Usage of FOSS must comply with the terms of the license. However, even if allowed under license, FOSS may not be modified and placed into operational use without approval from the CIO via the OSRB.

BlueCross staff may not contribute code to open source communities on behalf of BlueCross without CIO approval via the OSRB.

Only approved FOSS libraries may be placed into operation use.

Chapter 3 Application Systems Architecture

3.1 Strategic Application Architecture Committee

The Strategic Application Architecture Committee (SAAC) plays an important part in BlueCross System Architecture processes. The Architects work with the SAAC to review and approve the Vendor Application Roadmaps (Figure 3-1). These Roadmaps contribute to the overall Enterprise Roadmap. Where needed, the SAAC interacts with the Solution Detail Review Process (SDRP) Committee and the Research & Development (R&D) Committee in the pursuit of new or repurposed use of applications. Refer to *Systems Architecture > ICT Infrastructure Architecture > Strategic Technology Architecture Committee* for additional information.

The Strategic Application Architecture Committee is chaired by the Chief Application Officer. The committee's charter is to set the strategic application direction for the company.

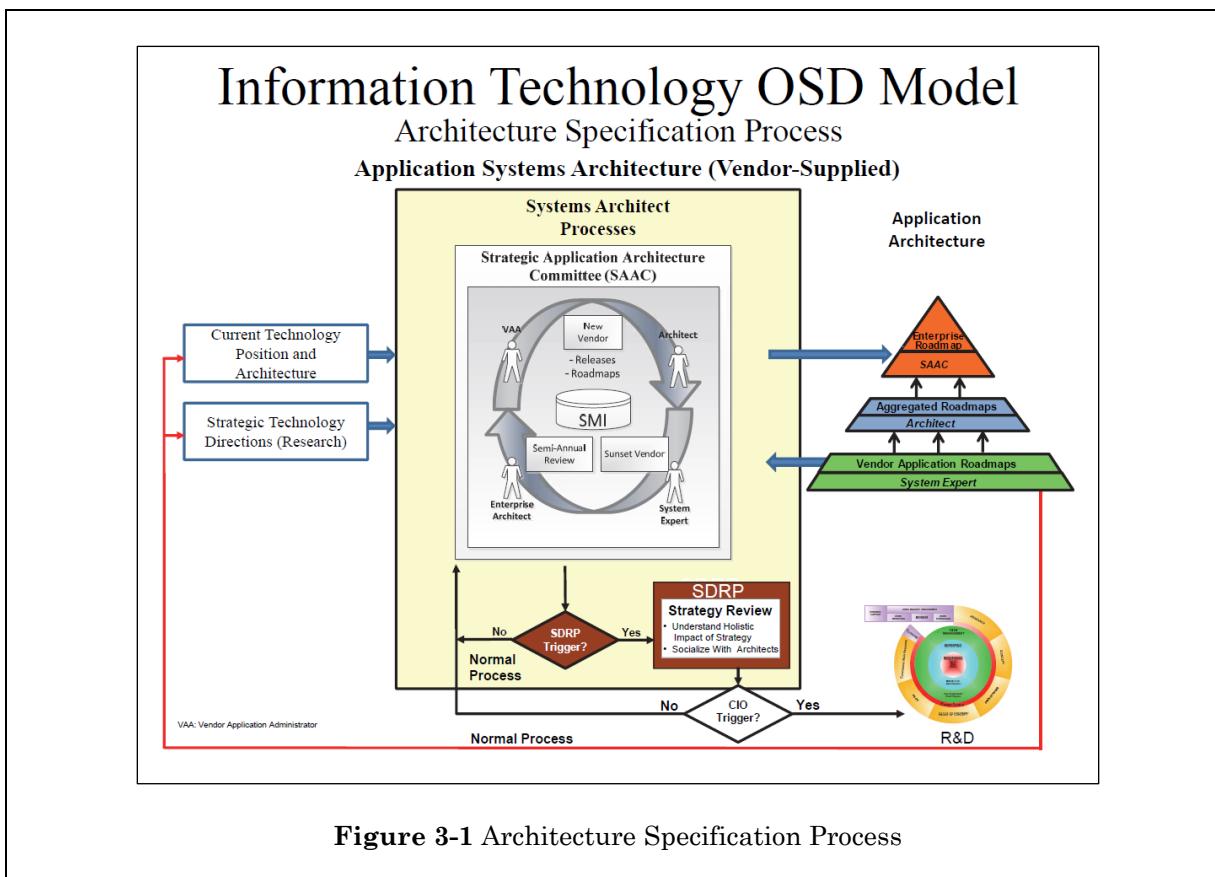


Figure 3-1 Architecture Specification Process

3.2 Application Roadmaps

3.2.1 Roles for Application Roadmaps for Vendor Products

In keeping with the specialization of Information Systems, there are three Roles that directly support the Vendor Application Roadmap process. These are the Vendor Application Administrator, the System Expert and the Architect.

Refer to *Enabling Processes > Managing People Program > IT Roles* for a detailed description of these Roles.

3.2.2 Application Roadmaps Overview

Application Roadmaps are an important vehicle for collaborative planning and assist in providing critical information to drive tactical and strategic business application decisions. The Aggregated Application Roadmap is a plan, outlined by system architectural and functional classifications, that matches both short- and long-term strategic goals with specific solutions that meet business goals. The roadmap provides information to make better application investment decisions by identifying critical business applications, gaps and identifying ways to leverage business investments.

A roadmap has three major uses:

- It helps reach a consensus on implementation recommendations for future releases and enhancements.
- It provides a mechanism to help forecast strategic business application development.
- It provides a framework to help plan and coordinate tactical business application development.

It does this by:

- Identifying business impacts that will drive technology or business feature selection and/or development decisions.
- Outlining vendor's long-term development plan and future enhancements that align with strategic business.
- Incorporating vendor release information with development planning information.
- Including planned release and support schedules.

There are three types of Application Roadmaps used within the I/S organization.

- Vendor Product Release Information
 - The assigned Vendor Application Administrator and/or System Expert collect information from the software vendor regarding product releases and associated maintenance life cycle as input for Application Roadmap planning.
- Vendor Application Roadmap
 - The System Expert, with System Manager oversight, uses the vendor product release and maintenance life cycle information to recommend the strategic and tactical plan of action for the application.

- Aggregated Application Roadmap
 - The assigned Architect curates the Aggregated Application Roadmap, which reflects approved action plans for related applications with functional or technical dependencies based on BlueCross support and business needs in a consolidated view.

3.2.3 Application Roadmap Content Standards

3.2.3.1 Vendor Product Release Information

Vendors control the format and content of their release information as well as the update frequency. The goal is to have information that clearly describes the vendor's intentions for the next 24 months. The Vendor Application Administrator and/or System Expert remains in touch with the vendor to ensure they have the most up-to-date version. At minimum, a semiannual contact is required. The Vendor Product Release Information is used as input into the Vendor Application Roadmap and Aggregated Application Roadmap.

3.2.3.2 Vendor Application Roadmap Content

The Vendor Application Roadmap data is captured for each SMI software entry. Planned vendor release information is correlated to installed version information, and significant milestone events are noted. Roadmap data serves as documentation for the recommendation regarding the strategic direction for the use of the vendor's product and for the Strategic Application Architecture Committee decision regarding that recommendation.

The Vendor Application Roadmap data may include:

- General Information — Product demographic information used to identify the application and facilitate aggregation, including name, business use and description.
- Specific Product Information — Current and Planned Vendor release information in order to support milestones and drivers for migration and to identify existing versions, known dependencies and internal impacts related to software and infrastructure.
- The Subjective Analysis — Recommendations regarding internal direction including industry trends, organizational impact, time line and expected level of effort.

For Business Applications, Vendor Application Roadmaps include premise-based vendor software applications that adhere to scheduled releases. Vendor applications require inclusion in an Aggregated Application Roadmap based on the *Release Category* field (Table 3-1) associated with that application in SMI and will be reviewed semiannually.

Release Categories

Release Category	Description	Roadmap Required	Review Frequency
True Release	Software updates available on planned releases, which include feature enhancements	Yes	Semiannual

Release Categories

Release Category	Description	Roadmap Required	Review Frequency
Time Bound	Software updates for reference data, routinely available on a quarterly, semiannual, annual schedule	No	
Continuous Improvement	Software updates published on a weekly or monthly basis in insertion releases	No	
No Roadmap	Vendor software for which no release schedules are planned and/or published	No	
Discretionary	At the Architecture area's discretion	Yes	Semiannual

Table 3-1 Release Categories

3.2.3.3 Aggregated Application Roadmap Content

The Aggregated Application Roadmap includes a subset of information on the individual roadmaps presented together in relevant views based on overall strategic vision or business drivers.

3.2.4 Vendor Application Roadmap Approval

The Vendor Application Roadmap, compiled by the System Expert, requires Architect approval recorded in the roadmap tool. Any exceptions to the Vendor Application Roadmap approval process must be approved by an Enterprise Architect.

As normal work activity causes a Vendor Application Roadmap update or on a semiannual basis, the Architect reviews the data and aggregates this information for Strategic Application Architecture Committee review.

3.2.5 Aggregated Application Roadmap Approval

The review and approval of the Aggregated Application Roadmap is completed during the quarterly Strategic Application Architecture Committee meetings. The Architect will present recommendations based on the Aggregated Application Roadmap and other pertinent business drivers. The review is the opportunity for discussion and concurrence on the recommendations made by the Architect.

The required participants in the review of the Aggregated Application Roadmap are:

- Vendor Application Administrator and/or System Expert

- Architect
- Enterprise Architects
- Chief Application Officer

The documentation required for the Aggregated Application Roadmap Review is:

- Aggregated Application Roadmap
- Supporting Documentation as needed (e.g., Vendor Release Information, Vendor Application Roadmap data, Industry Trends, etc.)

3.2.6 Storage of Aggregated Application Roadmaps

The SharePoint Online site designated for Aggregated Application Roadmaps and links to any supporting documentation is *ISADM / SAAC / Working Documents / Roadmaps*.

3.2.7 Roadmaps Work Initiation

Once the Strategic Application Architecture Committee has approved the changes to the Aggregated Application Roadmap, the System Manager includes the approval as documentation for funding requests for related work efforts.

Chapter 4 ICT Infrastructure Architecture

4.1 Strategic Technology Architecture Committee

The Strategic Technology Architecture Committee (STAC) plays an important part in the BlueCross System Architecture processes. This committee works with the Platform Owners to review and approve the Platform Roadmaps and uses these Roadmaps to produce the overall Enterprise Roadmap. Where needed, the STAC interacts with the Solution Detail Review Process (SDRP) Committee and the Research & Development (R&D) Committee in the pursuit of new or repurposed technologies (Figure 4-1).

The STAC is chaired by the Chief Technology Officer (CTO). The committee's charter is to set the strategic technological direction for the company and develop the Enterprise Roadmap.

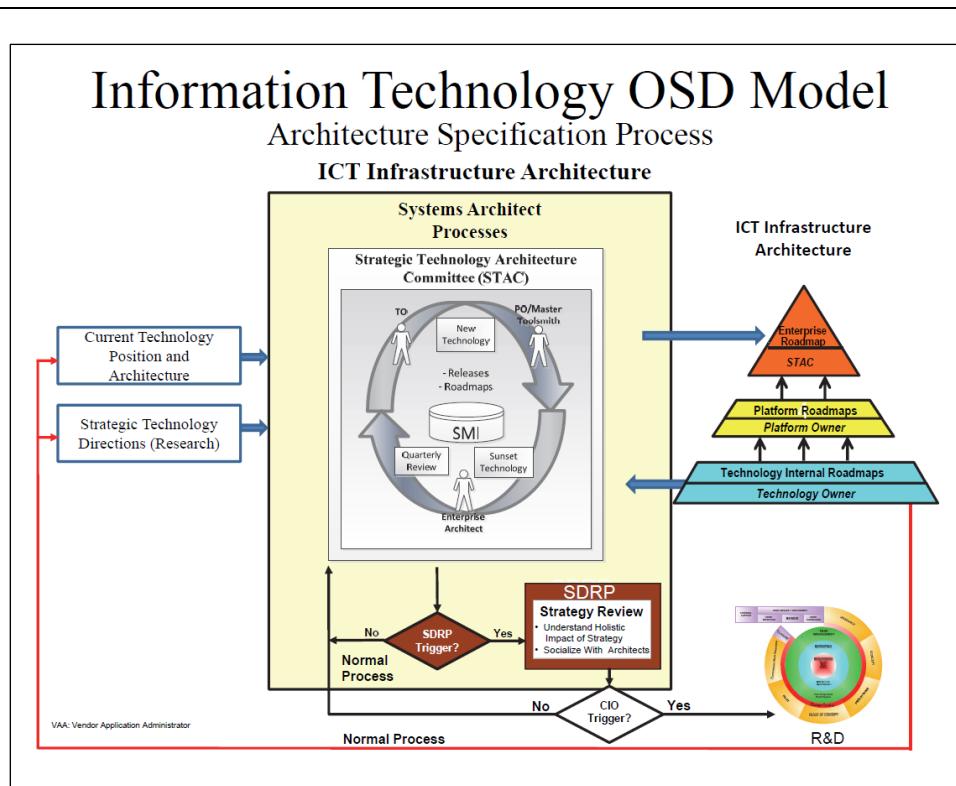


Figure 4-1 Architecture Specification Process

4.2 Technology Roadmaps

4.2.1 Roles for Technology Roadmaps

In keeping with the specialization of Information Systems staff, there are three Roles defined for the Technology Roadmap process. These are the Technology Owner, the Platform Owner and the Master Toolsmith.

Refer to *Enabling Processes > Managing People Program > IT Roles* for a detailed description of these Roles.

4.2.2 Technology Roadmaps Overview

Technology Roadmaps are an important vehicle for collaborative planning and assist in providing critical information to drive tactical and strategic technology decisions. A Technology Roadmap is a plan that matches both short-term and long-term goals with specific solutions to help meet those goals. The Roadmap provides information to make better technology investment decisions by identifying critical technologies, technology gaps and identifying ways to leverage technology investments. This applies both to hardware technologies as well as to infrastructure software.

A Roadmap has three major uses:

- It helps reach a consensus about a set of needs and the technology required to satisfy those needs.
- It provides a mechanism to help forecast changes to the technology.
- It provides a framework to help plan and coordinate the technology life cycle.

It does this by

- Identifying requirements that will drive technology selection and life cycle decisions.
- Determining and selecting technology alternatives to satisfy requirements.
- Generating and initiating the implementation of a plan to develop and deploy appropriate technology.

There are four types of Roadmaps used within the I/S organization:

- Vendor Product Release Information
 - The related vendor develops the Vendor Roadmaps, which describe the vendor's short- and long-term plans.
 - The assigned Technology Owner is responsible for obtaining the Vendor Roadmaps from the vendor.
- Technology Roadmaps
 - The assigned Technology Owner develops the Technology Roadmaps, which describe the BlueCross proposed short- and long-term plans for the related technology. The Technology Roadmaps also take into consideration the Vendor Roadmap but adapt it to BlueCross support and business needs.

- o The assigned Platform Owner is responsible for reviewing and approving the Technology Roadmaps.
- Platform Roadmaps
 - o The assigned Platform Owner develops the Platform Roadmaps, which describe the BlueCross proposed short- and long-term plans for the relevant technologies on their assigned platform (Host, Non-Host, Data/Voice Network, Workstation, z/Linux, Storage, Print).
 - o The Platform Owner and the Strategic Technology Architecture Committee are responsible for reviewing and approving the Platform Roadmaps.
- Enterprise Roadmap
 - o The Strategic Technology Architecture Committee assembles the Enterprise Roadmap, which is the aggregation of all the Platform Roadmaps.
 - o The Strategic Technology Architecture Committee is responsible for scheduling review sessions with the CIO to gain overall approval of the Enterprise Roadmap.

4.2.3 Technology Roadmap Content Standards

4.2.3.1 Vendor Product Release Information

Vendors control the format and content of their release information as well as the update frequency. The goal is to have information that clearly describes the vendor's intentions for the next 24 months. The Technology Owner remains in touch with the vendor to ensure they have the most up-to-date version. At minimum, a semiannual contact is required. The Vendor Product Release Information is used as input into the Technology Roadmap and Platform Roadmap.

4.2.3.2 Technology Roadmap Content

The Technology Roadmap data is captured for each SMI technology entry. Planned vendor release information is correlated to installed version information, and significant milestone events are noted. Roadmap data serves as documentation for the recommendation regarding the strategic direction for the use of the vendor's product and for the Strategic Technical Architecture Committee decision regarding that recommendation.

The Technology Roadmap data may include:

- General Information — Product demographic information used to identify the application and facilitate aggregation, including name, business use and description.
- Specific Product Information — Current and Planned Vendor release information in order to support milestones and drivers for migration and to identify existing versions, known dependencies and internal impacts related to software and infrastructure.
- The Subjective Analysis — Recommendations regarding internal direction including industry trends, organizational impact, time line and expected level of effort.

4.2.4 Roadmap Approval

The Technology Roadmap and the Platform Roadmap require approval. The following describes those approval processes.

4.2.4.1 Technology Roadmap Approval

As normal work activity causes a Technology Roadmap update or when six months have elapsed, the Technology Owner presents their updated Technology Roadmap to the appropriate Platform Owner or Master Toolsmith for approval that is recorded in the roadmap tool.

Any exceptions to the Technology Roadmap approval process must be approved by the Strategic Technology Architecture Committee.

Technology Roadmap Review

The review of the Technology Roadmap is the opportunity for discussion and concurrence on the recommendations made by the Technology Owner.

These are the required participants in the review of the Technology Roadmap:

- Technology Owner
- Platform Owner or Master Toolsmith

The documentation required for the Technology Roadmap Review is:

- Technology Roadmap
- Supporting Documentation as needed (e.g., Vendor Release Information, Aggregated Application Roadmap data, Industry Trends, etc.)

4.2.4.2 Platform Roadmap Approval

As normal work activity causes a Platform Roadmap update or when six months have elapsed, the Platform Owner presents their updated Platform Roadmap to the Strategic Technology Architecture Committee for approval.

Platform Roadmap Review

The review of the Platform Roadmap is the opportunity for discussion and concurrence on the recommendations made by the Technology Owner and the Master Toolsmith, in the case of Tools.

These are the required participants in the review of the Platform Roadmap:

- Strategic Technology Architecture Committee
- Platform Owner or Master Toolsmith

The documentation required for the Platform Roadmap Review is listed below:

- Platform Roadmap
- Supporting Documentation as needed (e.g., Vendor Release Information, Aggregated Application Roadmap data, Industry Trends, etc.)

4.2.5 Storage of Roadmaps

The SharePoint Online site designated for Technology Roadmaps and links to any supporting documentation is *ISADM / STAC / Working Documents / Roadmaps*.

4.2.6 Roadmaps Work Initiation

Once the Strategic Technology Architecture Committee has approved the changes to the technology, the Platform Owner submits a Work Request through the G&A and ICT Client Management organization to authorize the change to the infrastructure as identified by the Platform Roadmap. The Platform Owner will serve as the customer for the Work Request to make the authorized change to the infrastructure. These Work Requests are evaluated during the budget process or as needed.

Chapter 5 Research and Development

5.1 Overview

Research and Development (R&D) is the set of processes related to the investigation of new or repurposing existing infrastructure, business systems applications or I/S processes, hereafter referred to as "the idea."

The R&D processes determines if the proposed idea will work in BlueCross BlueShield of South Carolina's environment, identifies the benefits, possible constraints, and the potential impact to the current support and maintenance activities and other considerations.

Each idea that is considered in the Tech R&D briefing requires Chief Information Officer (CIO) approval. This approval may occur after initial evaluation is performed by the System Architecture area. An Enterprise Architect is engaged for all R&D efforts to serve as the "visionary" and provide guidance to the team.

As the R&D activities reach completion, the CIO will make the determination to implement or reject the idea. If it is approved to move forward, the RD sponsors will provide a recommendation about how to implement the idea that was proposed.

The R&D phases diagrammed here (Figure 5-1) are described later in this section.

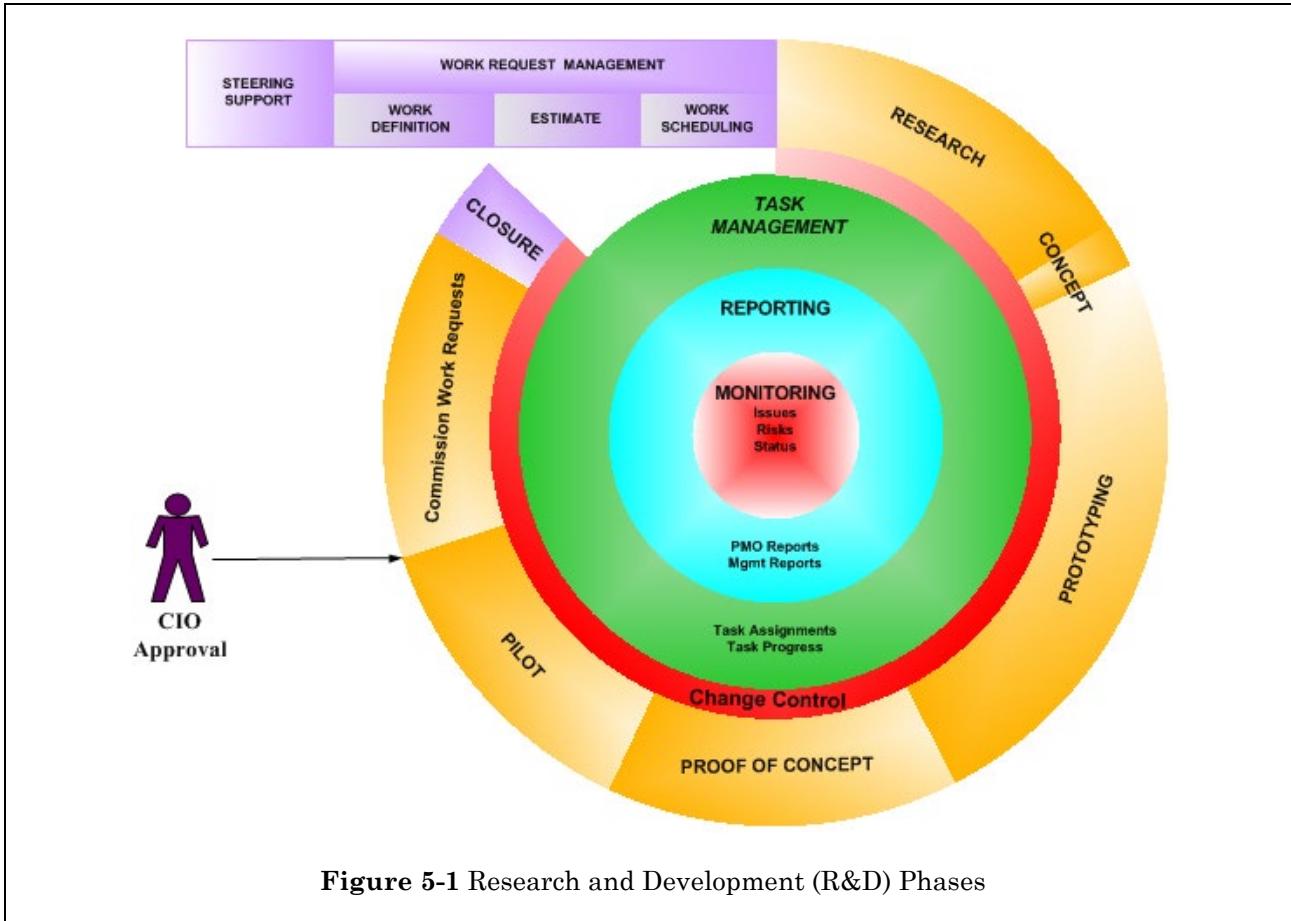


Figure 5-1 Research and Development (R&D) Phases



NOTE The R&D methodology is not related to the funding of Work Requests (for example, those Work Requests designated with an “RD” prefix). Furthermore, Work Requests generated to fully implement the idea of an R&D effort must follow the System Development Methodology. Refer to *Application Systems Management > System Development Methodologies* for details.

5.2 Variability in the R&D Methodology

Though the Research & Development methodology has the framework of a waterfall style methodology, it is more fluid and dynamic. As illustrated in the diagram below (Figure 5-2), the R&D Work Request can skip any phase or can restart any previous phase. As with any other Work Request, it can also be stopped at any point. At the end of each phase, the results will be analyzed and the Enterprise Architect or CIO will provide direction if the Work Request should continue. If so, they will set the direction for what needs to be accomplished in the next phase.

It can be expected that R&D activities may, by their nature, encounter numerous changes in direction and requested deliverables. This level of variability will result in a larger than normal number of date changes or other expectations. Each change must be tracked and followed the same way as any other Work Request.

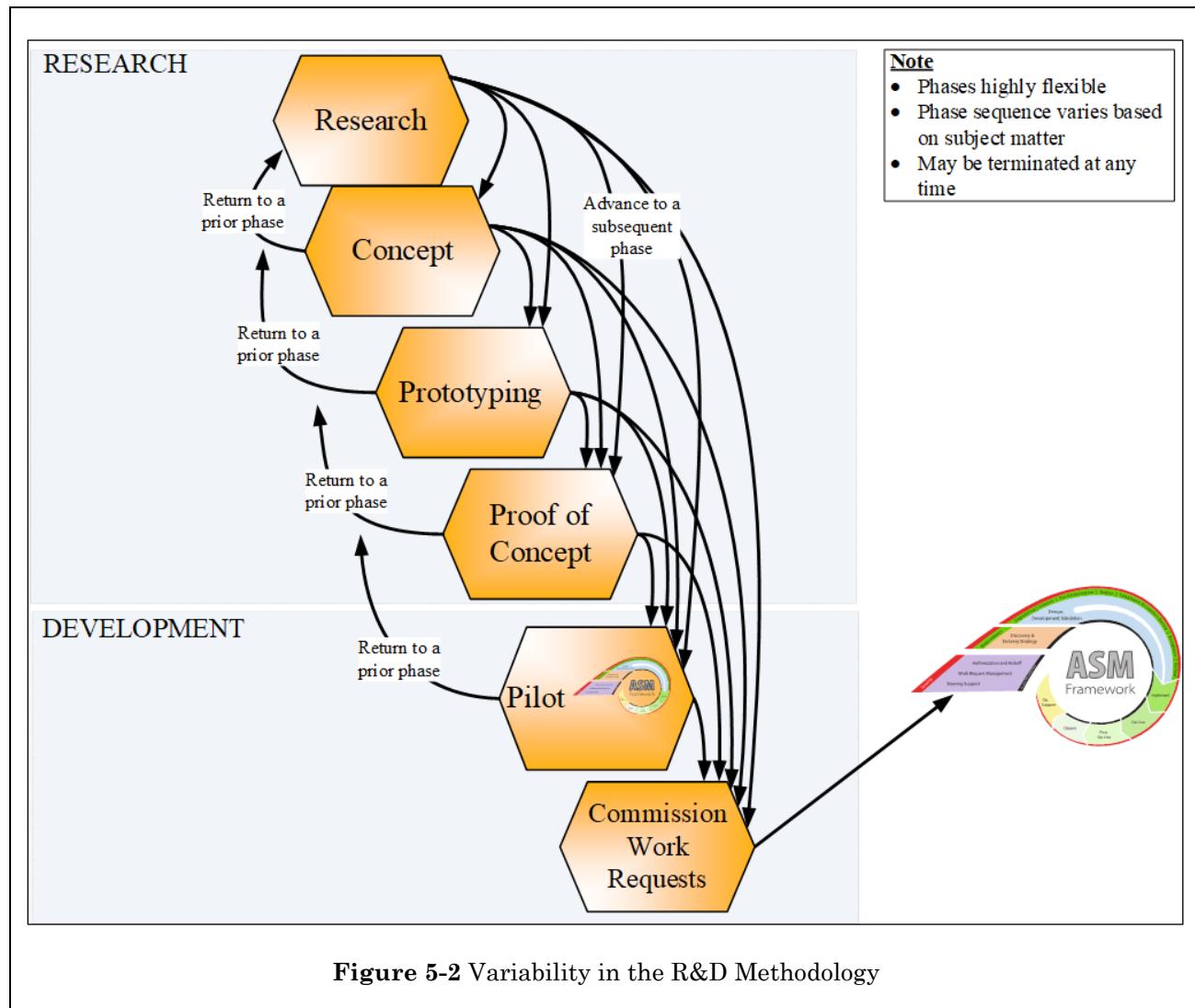


Figure 5-2 Variability in the R&D Methodology

5.3 Similarity in the R&D Methodology

The usual Work Request standards apply:

- Status Reporting
- Resource Management
- Change Control for date changes and established expectations
- Artifacts stored in the **SharePoint Online (SPO)** Work Request site

There are not any standard artifacts required for all R&D Work Requests. The artifacts for each effort need to be determined by the Work Request R&D Team (R&D Team) based on what makes sense for that effort. The Enterprise Architect or CIO will generally specify what the deliverable is for the end of each phase. Many of the same artifacts and deliverables for the System Development Methodology may need to be developed by the R&D Team who will also develop status reports to document the results of the research as it progresses.

The R&D processes deal with items that are unknown and are still in the idea stage. Consequently, it is impossible to predict what is needed to be investigated or documented to understand the idea.

R&D Work Requests are tracked in the **Management Practices System (MPS)**, and the usual tools within MPS are available to the Project Leader and I/S Management. The use of the Lineup Card is not required, though it may be helpful to the R&D Team.

5.4 R&D Phases

The phases of the R&D Methodology are defined below (Table 5-1).

R&D Phases

Phase	Description
Research	Process to study and establish facts about technology, business systems architecture, or I/S process for I/S products or services, and evaluate I/S strategic direction.
Concept	Process of defining a concept, I/S product or service based on the Research.
Prototyping	Process of creating a model and the simulation of the relevant aspects of an I/S product or service to evaluate the viability of a proposed solution.
Proof of Concept	Process of gathering and documenting sound evidence that the proposed solution related to an I/S product or service is feasible and viable.
Pilot	Process of conducting an initial limited roll out of a final solution in a production or operational environment to validate the solution is working as designed.
Commission Work Requests	Process of commissioning the Work Requests to fully implement the I/S product or service. If the implementation of the Work Request is immediate, the R&D Team will transition their knowledge and experiences to the next Work Request Team.

Table 5-1 R&D Phases

Chapter 6 Workstation Hardware/Software Procedures

6.1 Requesting Workstation Hardware/Software

To request a review of new hardware for the physical workstation environment or software for physical and logical workstation environment (Refer to *Technical Standards – Infrastructure > User Interfaces > Workstation Hardware/Software Standards* for a list of hardware and software.), complete the Self-Service Form *New Hardware/Software Review*. This form can be found on the Technology Support Center Self-Service under Service Request under the Category *Information Systems Services*. A Service Request will be generated and routed to ICT Non-Host Platform Technologies for review and coordinated by the chairman of the Workstation Infrastructure Subcommittee (WISC). Upgrades to existing software are considered new and must follow this process as well.



NOTE All requests for non-standard hardware or software will be reviewed and approved by the WISC and a representative of I/S Client Management before the product evaluation begins.

Upon receipt of a request, the requester will be contacted to gain more information about the product requirements, which will be used to help review and approve the request. (See Note above.) Product research and evaluation will be performed to ensure that there are no initial concerns with bringing the product into our environment. Upon approval of the WISC's initial evaluation, the requester will be contacted to get approval prior to the product being ordered by I/S Procurement. This is to ensure the customer's cost analysis has been completed. When the product comes in, it will be delivered to the WISC Chairman who will oversee the testing of the product.

Hardware (Physical Workstation Environment) — Based on the results of the compatibility test and risk assessment, the hardware will either be installed per the requester's instructions or the requester will be notified of any problems encountered. If the test fails, the WISC will work with the requester to locate a suitable replacement.

Software (Physical and Logical Workstation Environment) — Based on the results of the compatibility test and risk assessment, the software will be installed per the requester's instructions for functionality testing or the requester will be notified of any problems encountered. If the software passes all tests, additional copies or licenses of the software will be purchased and installed. If either test fails, the WISC will work with the requester to locate a suitable replacement.

The length of time the hardware/software review will take is dependent upon many variables; therefore, the committee is unable to provide a set time frame for each product until the initial review has been completed.

Variables include:

- Product Type — hardware or software
- Delivery time frame of initially approved product
- Problems encountered during validation

Since the introduction of new hardware for the physical workstation environment and software products results in a change to the BlueCross BlueShield of South Carolina physical and logical workstation environment, the WISC monitors requests that pertain to these types of items.

Chapter 7 Concepts & Techniques

7.1 Concepts

7.1.1 Architectural Concepts

7.1.1.1 Overview

BlueCross BlueShield of South Carolina has seven documented Architecture Concepts that are based upon the System Architecture. Our System Architecture is defined as:

A set of architectural patterns and principles for Business Application Systems development and integration which address and utilize application design concepts such as modularity (structured systems), loose coupling (I/O separation, multiple operating environments), reuse (leveraging), encapsulation (MQ Series, XML, SOAP), and flexibility (data and rules driven) in creating the desired business system solution(s) needed to meet our Client's business requirements.

The definition of a Concept is "A general idea derived from specific instances or occurrences." Our Architecture Concepts are listed below:

- Modularity
- Loose Coupling
- Reuse
- Flexibility
- Encapsulation
- Balancing and Reconciliation
- Data Management

7.1.1.2 Detailed Definitions

Modularity

Modularity consists of an approach that systematically decomposes a problem to be solved with software by dividing the problem into smaller subproblems. Each sub-problem is then analyzed, and a solution for the subproblem is obtained. The solutions of all subproblems are then combined to solve the overall problem. This process of implementing a structured design is called structured programming. The structured design approach is also known as modular programming. The typical structured program is subdivided into modules wherein a *main module* invokes the other modules as needed. In APS and COBOL the modules are called *Paragraphs*. Modularity is closely related to the principle of encapsulation (see Encapsulation below.)

Loose Coupling

Coupling is a measure of the strength of the relationship between modules. Loose coupling indicates a high degree of module independence. Loose coupling allows programming objects to come from completely different development processes and operate on completely different platforms, yet still function together.

Cohesion of a single module/component is the degree to which its responsibilities form a meaningful unit; higher cohesion is better. A first-order principle of software architecture is to increase cohesion and to reduce coupling.

Reusability

Reusability is the characteristic of a module/component that allows it to be used in more than the application for which it was created. When code is broken into modules according to their functions it has the potential of being reused.

The use of Application Program Interfaces (APIs), applying the principles of modularizing software, loose coupling components, and the encapsulation of software elements will greatly help to increase the reusability of the software (see Encapsulation below.)

Flexibility

The ability to extract more value from things you already have, flexibility in programming has to do with the ability to support a multitude of customers and platforms with a single version of code. Flexibility is accomplished by using variable data or rules instead of “hard coding.”

- Rules are ideally stored in Relational Databases such as DB/2 because of a much larger collection of domains. Examples of this are the way the RULEs database is used to allow flexibility and the way a single program can be associated with different customer data using collection ids that are set within the program based on data read in from CARDLIBs.

Encapsulation

Sometimes referred to as data hiding, encapsulation provides a form of *logical data independence*; i.e., the implementation of a type can be changed without need to change any of the programs that use that type, thereby protecting the application programs from implementation changes in the lower layers of the system.

- APS DB commands provide a degree of encapsulation in that a VSAM record, a DB2 row or an IMS segment can all be accessed using the DB-OBTAIN statement/verb. However, the data can be accessed outside of the commands as well.

Balancing and Reconciliation

Balancing — The act of accounting for all inputs, outputs and exceptions related to a process at a distinct and reoccurring point in time.

Reconciliation — The act of balancing all balancing results across all inter-related processes at a distinct and reoccurring point in time.

Data Management

Data Management is the logical and physical management of data and databases. Data Management includes data modeling, data handling, and data transport.

- Data modeling addresses the techniques used to design optimal database structures for operational and informational business requirements.
- Data handling is the approach used in applications to read, add, update and delete data on databases including the ability to retain history and produce an audit trail.
- Data transport defines techniques for moving data within the operational environments and from the operational to the informational environment.

For each of the Architecture Concepts, Architecture Techniques exist. An Architecture Technique is defined as “A systematic procedure by which a complex task is accomplished or a best practice approach.”

7.2 Common Platform Techniques

7.2.1 Single Source Application Design

7.2.1.1 Concepts Involved

- Modularity
- Reuse
- Loose Coupling
- Flexibility
- Encapsulation
- Balancing and Reconciliation
- Data Management

7.2.1.2 Operating System Platform

Host/Non-Host

7.2.1.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects
- Solution System Designers
- System Designers

7.2.1.4 Required Skills/Knowledge

N/A

7.2.1.5 Related Techniques

All Techniques

7.2.1.6 Overview

Single Source Application Design (SSAD) is the technique that employs all of BlueCross BlueShield of South Carolina's (BlueCross') architecture concepts in an integrated approach to create applications (Figure 7-1). SSAD results in enhanced quality and maintainability, reduced development effort and shortened time line from design to implementation. The goal of SSAD is to leverage and re-use code, to 'build code once and use it many times'. SSAD is used by BlueCross to build shared applications and process common functions for multiple platforms, clients, LOBs, etc. As BlueCross' business continues to diversify, Single Source Application Design becomes increasingly vital to enhancing existing functions and building new functionality into our core systems.

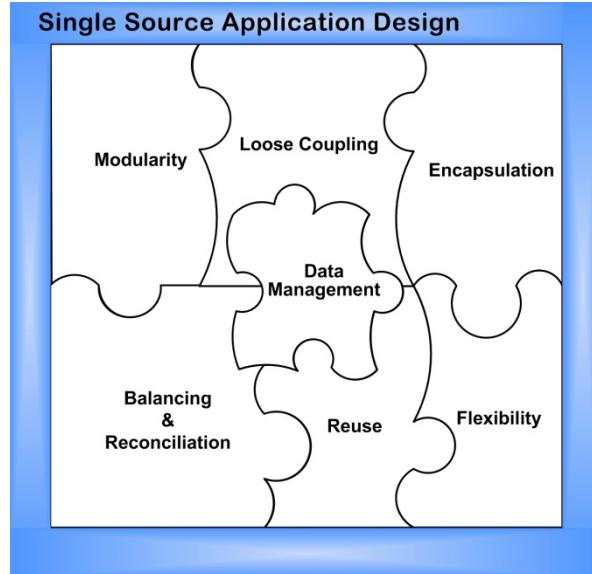


Figure 7-1 Concepts

Single Source Application Design may be conceptualized by considering the interrelated design concepts as puzzle pieces that must all be combined to achieve the final goal of an integrated single source solution. The degree to which any single concept is employed will depend on how the application is designed. All concepts must be carefully considered and employed in the design process to complete the SSAD puzzle.

7.2.1.7 Detail Discussion

The SSAD technique dictates decomposition of business and technical requirements into functions and sub-functions, creating a design that is a collection of logical units of work. SSAD requires constructing sub-functions into distinct units of work independent of each other. Sub-functions isolate the processing for each unit of work, allowing it to be easily documented, targeted for maintenance and testing, and leveraged for re-use in other programs.

Separating application requirements into units of work, SSAD constructs the application in a modular fashion and combines the sub-functions to achieve requirements for each piece of the application. The technique employs variable-driven processes, creating programs that modify execution based on the data passed to the unit of work, for example, Regional Processing Number (RPN) driven logic to minimize hard coding.

SSAD incorporates guidelines for data modeling, data handling and data transport. These Data Management design tools direct the use of data stores that are optimized to meet business requirements, accessible through common components and integrated into corporate standards for operational and informational usage.

Since SSAD enables multiple clients to execute the same set of source code, Balancing and Reconciliation must be maximized to ensure the integrity of variable data through the entire system. Balancing must

allow for reporting at the client level to be meaningful to the client reviewing data processed by shared code.

SSAD requires use of the following BlueCross' concepts (See Concept Overview for details):

- Modularity
- Encapsulation
- Loose Coupling
- Reuse/Leveraging
- Flexibility
- Balancing and Reconciliation
- Data Management

7.2.1.8 Modularity, Encapsulation, Loose Coupling and Reuse

To realize the benefits of Single Source Application Design, function-based modules or components must be designed for reuse across platforms, input channels and applications. SSAD uses the concept of **Modularity** to describe building applications from logical sub-functions. To use the Modularity concept, one must systematically decompose a function into logical sub-functions representing meaningful units of work. One can build modules for each unit of work that can be combined together to complete the function.

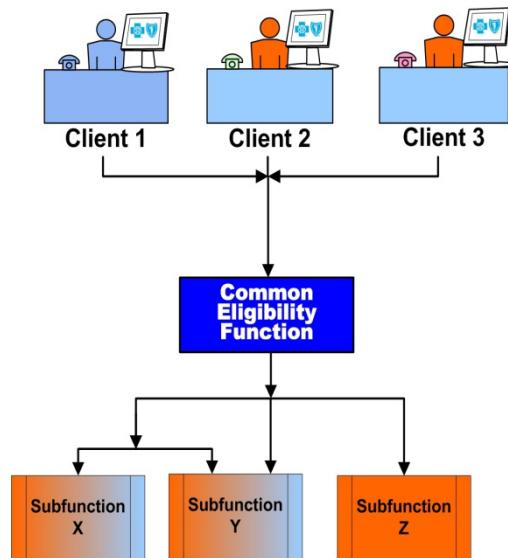


Figure 7-2 Clients Leveraging Modular Sub-functions

For example, the eligibility process combines a large number of sub-functions such as date routines and database access modules that perform separate units of work. The eligibility process (Figure 7-2) independently executes each sub-function in a modular approach to solve the problem and compose the end result. When an enhancement is applied to the eligibility process, the isolation of each sub-function enables them to be modified without impacting other logic in the program.

7.2.1.9 Loose Coupling

This is the concept of building sub-functions that are not tied to logic outside their unit of work, thereby enhancing ‘module independence’ and reuse of the sub-functions. **Loose Coupling** maximizes reuse across multiple clients, reduces system maintenance, and simplifies testing by focusing change in a single module without impacting other modules.

7.2.1.10 Encapsulation

This concept describes insulating data sources from changes that occur to sub-functions, modules, components, applications, etc. **Encapsulation** provides a means of performing the unit of work without any need of understanding the logic that performs the unit of work.

For example, the encapsulation of the claims systems via MQ Series insulates the consumers of the system from internal changes. When change occurs, the interface and data sent via MQ remain relatively constant, thus ‘hiding’ the logic and changes of the claims system from the system’s clients as if it were a “black box.”

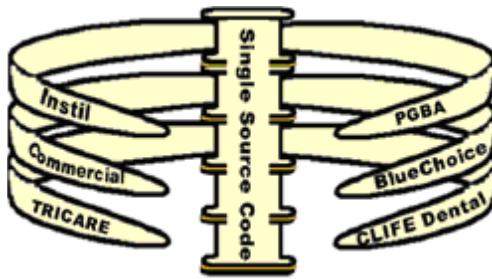


Figure 7-3 Reuse of Single Source Code

7.2.1.11 Reuse

This concept describes the process of leveraging the same code for multiple input channels and clients (client, RPNs, platforms, etc.). The SSAD technique can be compared to a backbone and rib cage, where the backbone contains the reusable code (**Reuse Concept**) and applications that are leveraged to support all client and business functions. When properly developed and applied, the backbone reduces the system’s overall maintenance requirements and streamlines new client implementation.

An example of **Reuse** is found in the common eligibility process, where requests may be received through multiple input channels such as the Web, VRU, EDIG, SELG/BELG (Commercial Eligibility), and for multiple clients such as Commercial Business, InStil, or BlueChoice HealthPlan (Figure 7-4). Single source eligibility code allows access from multiple platforms and input channels. **Reuse** of the common process significantly reduces the need to clone code, since modules/functions are executed using one set of code that performs the same functions regardless of input channel or commercial client.

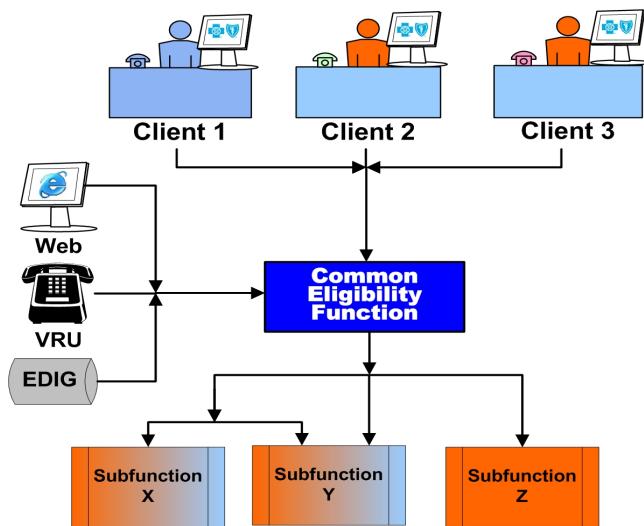


Figure 7-4 Reuse of Common Code by Clients and Input Channels

7.2.1.12 Flexibility

SSAD employs the **Flexibility Concept** using data-driven logic to allow functions, modules, and components to be deployed for multiple clients, thus removing the need to hardcode logic or clone modules.

While many functions are common among all clients, others are client-specific. An analysis of Figure 7-3 shows the ribs representing data-driven logic that is specific to each client. This data-driven logic greatly enhances **Flexibility** and the opportunity for reuse of common modules across multiple clients.

7.2.1.13 Balancing and Reconciliation (B&R)

The overarching concept of Balancing and Reconciliation is critical to the success of an SSAD application. An SSAD application must:

- Balance and reconcile data passing through shared code at the client level in order to create reports meaningful to the client.
- Insure integrity of processing including any batch of numerous transactions and cross-platform data transfers.

7.2.1.14 Data Management

SSAD uses the concept of **Data Management** to ensure the integrity of each client's data while being accessed through a common I/O within the single source application. Data is logically or physically segregated by key structure, as needed, to meet performance requirements and to ensure each client has security access to only their data.

7.2.1.15 Single Source Application Design Considerations

- Testing Complexities
- Negative Testing
- Coordination of Releases
- Processing Environments
- File Structures (relates to flexibility)
- Collection ID

7.2.1.16 Benefits of Single-Source Code

- Quicker Deployment
- System Marketability
- Reduction in Cost of Maintenance and Development
- Enhancement to core system is available to all clients
- Centralized Staff to Maintain Systems

7.2.1.17 Techniques that make Single Source Application Design Reality

- Regional Processing Number (RPN) Driven Code
- Application Cloning
- Modules by Function
- Common I/O and Edit Modules
- Automated Balancing & Reconciliation
- Concept Diagrams
- Informational Databases

7.2.1.18 Existing ISSM Standards

- Refer to *Security Management > Information Security Management > Application and Data Security Control Standards* for further information.
- Refer to *Technical Standards — Applications > Application Development Support Tools > Host Application Languages* for additional information.
- Refer to *Technical Standards — Applications > Application Coding > Content Manager Moves Process* for information on Leveraged System Non Host (LS Non-Host) Languages.
- Refer to *Technical Standards — Applications > Print Operations > ICT Print Operations and Corporate Mail Services > InfoPrint Workflow Testing Procedure* for additional information.

7.2.2 Application Cloning

7.2.2.1 Concepts Involved

- Loose Coupling
- Reuse
- Modularity

7.2.2.2 Operating System Platform

Host/Non-Host

7.2.2.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.2.2.4 Required Skills/Knowledge

N/A

7.2.2.5 Related Techniques

- Regional Processing Number (RPN) driven logic
- Common Input/Output (I/O) and Edit modules
- Modules by Function

7.2.2.6 Overview

An **Application System** is defined as a group of interacting, interrelated, or interdependent computer programs designed for a specific business task or use.

Copying an existing **Application System** to allow it to be modified and used by a different client is considered “Application Cloning.” BlueCross I/S has a strategy to build applications that are leverageable and reusable so that multiple clients can share them. Copying (cloning) is not the same as sharing. Sharing code reduces maintenance by keeping one set of code for all clients. The Enterprise Architect Office (EAO) must approve all requests to clone an application.

Functions and modules within applications should be built in a modular fashion that encourages reuse and minimizes the need to clone. When new applications are built, reuse will be considered during the Design Phase to ensure data structures and processing within the application positions it for additional clients. The decision and approval process to clone individual programs rests within the application area. Typically, the systems expert within the area will make this decision with input from their Architect.

7.2.2.7 Areas Impacted

All programming areas should be very familiar with this concept.

7.2.2.8 Application Cloning Approval

Application cloning should have EAO approval and final approval from the Chief Enterprise Architect.

7.2.2.9 Detailed Information

While application cloning must have EAO approval, it should be noted that individual programs, data structures and common includable code can be cloned with approval from the application Systems Designer (SD) and Architect. However, before cloning individual elements, the basic I/S strategy to build leverageable and reusable Application systems should always be reviewed. Exceptions should be documented with justification and submitted as part of the cloning approval request. It should be further noted that JCL Procedures (PROCs) should not be cloned. Instead symbolics should be used. An example of this is found in the AMMS batch cycle. While there is one set of application code to run AMMS for all clients, there is different JCL (JOBS) that executes the same PROCs.

7.2.3 Regional Processing Number (RPN) Driven Code

7.2.3.1 Concepts Involved

- Reuse
- Flexibility
- Loose Coupling

7.2.3.2 Operating System Platform

Host and Non-Host

7.2.3.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects
- Solution System Designers
- System Designers

7.2.3.4 Required Skills/Knowledge

N/A

7.2.3.5 Related Techniques

- Common I/O and Edit modules
- Minimize the use of Hard Coding
- Application Cloning
- CICS Application Security
- Modules by Function

7.2.3.6 Overview

BlueCross BlueShield of South Carolina (BlueCross) has been able to provide technical solutions to other companies and subsidiaries due to BlueCross' ability to share application systems. A means for sharing the systems is accomplished by assigning a Regional Processing Number (RPN) or a Plan Code to delineate a block of business. RPN is used in Commercial Systems and Plan Code is used in PGBA Systems. The RPN/Plan Code field is carried on most of the file/database structures, batch programs and online transactions as a key data element. Utilizing a RPN/Plan Code as a high order key allows BlueCross to:

- Share or customize business logic
- Share file structures, isolate data
- Control security

When projects result in development of new data structures, batch programs or online transactions, RPN/Plan Code logic should be included. If initially for one client only, the development effort should always assume the possibility of multiple clients. Utilizing the RPN/Plan Code allows the application system to be coded once and leveraged multiple times. The goal with RPN/Plan Code logic is to bring in new business quickly with minor changes (customization) or no change at all.

7.2.3.7 Detailed Discussion

When is it necessary to assign a new RPN/Plan Code?

As new projects reach the end of the requirements/scope phase certain business decisions have been made that should guide the team in knowing if a new RPN/Plan Code is necessary. These types of projects are usually bringing in new blocks of business.

Who decides if a new RPN/Plan Code is needed?

The Architect gathers all relevant information from the Solution System Designer and then meets with the Enterprise Architects to obtain a final decision.

Questions related to RPN/Plan Code Assignment?

Before the Architect meets with the Enterprise Architects, the following questions must be answered:

- Can the primary business logic be shared with other clients?
- Can the support systems/files be shared?
- Will the data be physically separated?
- Who will provide back office support? (Internal staff vs. External staff)
- Who will have security to update support files? (Internal staff vs. External staff)
- Who will have security to update primary files? (Internal staff vs. External staff)

For some projects it will be obvious when a new RPN/Plan Code is needed. If a new TRICARE region is brought up, it would require a new Plan Code. If a new Blue Plan were brought up (like Wisconsin), it would require a new RPN. In any case, the Enterprise Architects make the final call.

Sharing or Customizing Logic (Reuse & Flexibility)

Use of a RPN/Plan Code allows programs to drive logic based on the client. Multiple clients may share primary logic with variations as needed. Screen edits against reference tables are a good example. The table structure is shared. The logic to validate a CICS entry field against the reference table is shared, but the values in the table are different. This allows one client to have “A,” “B,” and “C” as valid values while another client has “X,” “Y,” and “Z.” Using the RPN/Plan Code to read the table will only return the values appropriate for the client.

Shared File Structures/Isolate Data (Reuse, Security)

Sharing logic for data inquiry and/or maintenance is possible because the file structure is identical from one client to the next. The primary variation is the RPN/Plan Code. Data files utilizing a RPN/Plan Code can be physically or logically separated. Physical separation is preferred. There are times when data for more than one RPN/Plan Code is contained in the same file/database (logically separated). In those instances clients would not be given access directly to the file. For DB2, a “view” would be created for the client to access. For flat files, an extract file would be built using the RPN/Plan Code to select the records belonging to a particular client. The client would then have access to the extracted file.

Control Security (Security)

Systems that are shared often have their own security utilizing the RACF ID. This is an additional security layer on top of RACF control. In those systems they connect a user’s RACF ID to all the RPNs or Plan Codes they are allowed to access. This prevents one client from accessing another client’s data.



NOTE If a RACF ID has been granted security access to multiple RPNs or Plan Codes, the application must provide a means of allowing the user to switch RPNs or Plan Codes without I/S or RACF Admin intervention.

7.2.3.8 Existing ISSM Standards

- Refer to *Systems Architecture > Concepts & Techniques > Common Platform Techniques > Regional Processing Number (RPN) Driven Code*. The Data Dictionary and the RPN-to-LOB cross-reference table must be updated as new RPNs are created.
-



NOTE A reference to the corporate Data Dictionary is included. The Data Dictionary element ID for the RPN is DD-21183. Per the ISSM the project team must update the Data Dictionary as new RPNs are assigned. The Plan Code element ID is DD-16352. For TRICARE, the definition reads “The Plan Code Field Is Used to Allow Regional Processing.”

7.2.3.9 Commercial Systems

In Commercial Systems, reference is made to an “Alias” RPN. An alias RPN is used for special processing in the AMMS claims adjudication system. In some cases the AMMS processing RPN and the alias RPN are the same value. The value in the alias RPN is used when directing I/O reads to support files, such as TMCS and PIMS. CES Business Sector reference table details the appropriate RPN to be

used in each support system as well as the CES RPN for that business sector. Below is a list of the Commercial Systems RPNs (Table 7-2).

Commercial Systems RPNs

Client	AMS Processing RPN	AMMS Alias RPN	CES RPN	LCAS RPN
<i>Current:</i>				
BlueCross	“001”	“001”	“001”	“001”
Instil	“001”	“006”	“001”	“006”
BlueChoice	“035”	“035”	“035”	“035”
— LSV Partners				
HNNY HealthNow Med Adv	“001”	“107”	“107”	“107”
Federal Bureau of Prisons	“001”	“108”	“108”	“108”
<i>Prior Clients:</i>				
GE	“001”	“005”	“005”	“005”
Florida Combined Life (FCL)	“001”	“037”	“001”	“037”
Deseret MedAdv	“001”	“100”	“100”	“100”
Michigan MedAdv	“001”	“101”	“101”	“101”
Health Net Med Adv	“001”	“103”	“103”	“103”
VAHERO	“001”	“109”	“109”	“109”
Wisconsin	“002”	“002”	“002”	“002”
Sunstar	“035”	“036”	“---”	“036”
FEP BCA Hosting	“009”	“009”	“009”	“009”
BCBS Western NY(HNNY)	“701”	“701”	“701”	“701”
BlueShield NE NY(HNNY)	“701”	“702”	“702”	“702”
HNNY(Central NY)	“701”	“703”	“703”	“703”

Table 7-1 Commercial Systems RPNs

7.2.3.10 PGBA Systems

PGBA Systems assigns Plan Codes to distinguish a TRICARE Region. In the past, the Plan Code identified the Prime Contractor.

Consolidated data that is not owned by any Prime Contractor uses Plan Code “999.” An example of this would be the TMA zip cross-reference that defines the geographic area assignments by ZIP Code and state to a region. This is a single source of data supplied by TMA, the actual Government body administering TRICARE Contracts, and does not belong to any specific Prime Contractor. Below is a list of the PGBA RPN/Plan Codes (Table 7-3):

PGBA RPN/Plan Codes

Prime Contractor	Region	RPN	Plan Code
<i>Current</i>			
HNFS	North	“810”	“810”
Humana	South	“820”	“820”
UnitedHealthcare	West	“860”	“860”
Consolidated		“999”	“999”
<i>Previous</i>			
Humana Gold		“700”	“700”
Sierra	1	“540”	“540”
Humana	Puerto Rico	“920”	“920”
Anthem	2,5	“530”	“530”
Humana	3,4	“570”	“570”
Triwest	7,8	“590”	“590”
HNFS	9,10,12	“CHP”	“560”
BlueCross	FI	“CHP”	“550”
BlueCross	FI-MH	“CHP”	“600”
Aetna	CA/HI CRI	“CHP”	“580”

Table 7-2 PGBA RPN/Plan Codes

7.2.4 Modules by Function

7.2.4.1 Concepts Involved

- Reuse
- Modularity
- Encapsulation

7.2.4.2 Operating System Platform

Non-Host/Host

Concepts & Techniques > Common Platform Techniques

Printed forms of this document are uncontrolled and may be obsolete. It is the responsibility of the user to ensure that any printed copy is the same revision as the online version.

Copyright ©2023, BlueCross BlueShield of South Carolina. All rights reserved. BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association.

7.2.4.3 Intended Audience

- System Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.2.4.4 Required Skills/Knowledge

Programming Skills

7.2.4.5 Related Techniques

- CICS Application Security
- CICS Application Transaction Router Processing
- Common I/O and Edit Modules
- Application Cloning
- Regional Processing Number (RPN) Driven Code

7.2.4.6 Overview

The concept of structured design (modular programming) is emphasized and encouraged at BlueCross BlueShield of South Carolina (BlueCross). Modular programming can be used to break up a large program into manageable units and to create code that can be easily reused.

The Information Technology standard industry definition of the word **module** is “A portion of a program that carries out a specific function and may be used alone or combined with other modules of the same program.”

The typical structured program is divided into paragraphs or routines that are invoked from a mainline process as needed. Individual programs/modules can also be developed and called from a program to perform a common function. Isolating unique functions into common paragraphs and/or common programs/modules provides clarity and the ability to reuse and leverage code.

Examples of common programs/modules by function include the following:

- Corporate date module (Transcentury)
- Database I/O modules
- MQ API

Examples of common paragraphs by function would include the following:

- Date formatting
- CICS PFKEY usage
- CICS Application Transaction Router processing with an APSSTUB
- APS user macros and CICS screen scrolling



NOTE To reduce complexity and increase maintainability, paragraphs should contain between four (4) and 80 lines of code. Anything less than four (4) lines should not be a paragraph.

7.2.4.7 Non-Host Modularity

The concept of modularity extends to object-oriented development on the Non-Host platform through the related concepts of Encapsulation and Reuse achieved by functional inheritance. Object-oriented programming (OOP) is a highly modular programming methodology that focuses on data rather than processes, with programs composed of self-sufficient modules (objects).

A major feature of OOP is Encapsulation, which ensures good code modularity by building strong cohesion between data and local functions including I/O functionality. Strong cohesion causes a corresponding reduction in coupling between higher order functions and data processed through the functions. Encapsulation hides the complexity of data access and low order functions, allowing development of business logic in separate modular programs. The result of Encapsulation is a simplification of coding and maintenance.

In OOP, cohesive base functions can be extended into new derived functions through the use of inheritance. Using inheritance allows construction of a new function that incorporates (inherits) the properties of the base function. Inheriting characteristics of a base function improves modularity by reusing the logic of the base function, eliminating redundant code and minimizing maintenance impacts.

7.2.4.8 Detailed Discussion

Common Paragraphs/Modules

BlueCross uses both paragraphs and called modules to perform specific functionality. Below is an example of both:

CRHPP101 – HIPAA 270/271 Processing program

The main procedure within the CRHPP101 program is structured (modular) to perform paragraphs.

```
PROC    CRHPB101-COMMAREA
        PERFORM 0050-INITIALIZE
        PERFORM 0100-MAIN-PROCESS
        PERFORM 9999-TERMINATION
```

The main paragraphs then perform subparagraphs for required functions.

```
PARA    0100-MAIN-PROCESS
        PERFORM 0200-EDIT-INQUIRY-RTN
        PERFORM 0300-ELIGIBILITY-INQUIRY
        IF P100-BENEFIT-REQUEST
        PERFORM 0400-BENEFIT-INQUIRY
```

PERFORM 0500-FORMAT-RESPONSE

The subparagraphs then call common programs/modules by function.

```
PARA    0200-EDIT-INQUIRY-RTN
        % IF &AP-GEN-DC-TARGET = 'CICS'
        CALL WS-CRHPP110 USING DFHEIBLK
            DFHCOMMAREA
            ...CRHPB101-COMMAREA
PARA    0300-ELIGIBILITY-INQUIRY
        % IF &AP-GEN-DC-TARGET = 'CICS'
        CALL WS-CRHPP120 USING DFHEIBLK
            ...
            DFHCOMMAREA
            ...CRHPB101-COMMAREA
```

The example above is divided by functions into paragraphs that make the program easy to follow and maintain. It also provides the ability to reuse application code.

The program CRHPP110 is also called in programs CRHPP100 and CRHPP121.

The program CRHPP120 is also called in program CRHPP100.

Common Paragraphs (via APSSTUB)

BlueCross uses APSSTUBs to perform specific includable code and paragraphs within programs.

A primary advantage of using an APSSTUB is that the module requiring the common function will NOT have to CALL (exit the program) another program/module. By executing the function within the executable module, the program will perform more efficiently, since the overhead of the CALL is removed. This is especially true if the code is performed repetitively within a program.

A disadvantage of using an APSSTUB for a common function is that when there are changes in the APSSTUB all modules referencing the APSSTUB must be re-generated (compiled) to pick up the changes made to the APSSTUB.

Generally, APSSTUBs contain common code that is frequently referenced by more than two programs and seldom changes. Below is an example:

The CRHPP416 program performs paragraph 9211-REFORMAT-TO-SCREEN-DATE to reformat a date of service.

```
IF SPCH-FROM-DATE-SRVC NUMERIC AND
... SPCH-FROM-DATE-SRVC > ZEROES
    WS-WORK-DATE          = SPCH-FROM-DATE-SRVC
    WS-WORK-DATE-R         = WS-WORK-DATE
    TRC-CONVR-FROM-DATE-9 = WS-WORK-DATE-R
```

```
TRC-CONVR-FROM-DATE-MASK = 'CCYYMMDD'  
PERFORM 9211-REFORMAT-TO-SCREEN-DATE  
CRHPPM16-DOS-FROM      = WS-SCRN-DATE  
ELSE  
CRHPPM16-DOS-FROM      = SPACES
```

The paragraph 9211-REFORMAT-TO-SCREEN-DATE and the code to reformat a date is coded in the APSSTUB CRHPS001.

```
PARA 9211-REFORMAT-TO-SCREEN-DATE  
(related code for date reformatting follows)
```

The APSSTUB CRHPS001 is included in the CRHPP416 program.

The CRHPP416 module must perform the 9211-REFORMAT-TO-SCREEN-DATE paragraph eleven (11) times to satisfy the various date display requirements.

The CRHPP416 module also performs the paragraph 9201-GET-SYSTEM-DATE. The 9201-GET-SYSTEM-DATE is a paragraph in the APSSTUB CRHPS001.

The APSSTUB CRHPS001 contains several common date routines and is also included in these programs: CRHPP400, CRHPP401, CRHPP402, CRHPP405, CRHPP407, CRHPP408, CRHPP409, CRHPP410, CRHPP411, CRHPP412, CRHPP413, CRHPP414, CRHPP415, CRHPP416, CRHPP417, CRHPP418, CRHPP419, CRHPP420, CRHPP421, CRHPP422, CRHPP423, CRHPP426, CRHPP427, CRHPP428, CRHPP429, CRHPP430, CRHPP431, CRHPP432, CRHPP433, CRHPP434, CRHPP435, and CRHPP461.

Common Code (Via AMB User Rules)

BlueCross uses AMB user rules to perform specific functionalities within programs.

AMB user rules are defined in the BC.NDVR.PROD1.USERMACS dataset. Programmers can reference this file to determine if any of the user rules could be used in a program.

To access a user rule, the AMB application must contain the member name from the USERMACS dataset. AMB user rules allow code to be conditionally inserted within the program, which reduces redundant coding. While development and maintenance of AMB user rules are the responsibility of the appropriate application development areas, ITBS Host retains final approval of all new or changed user rules.

Generally, AMB user rules contain common code that is frequently referenced by more than two programs and seldom changes. Below is an example.

BCEXPS — Is an online user rule used to move the date and time to the map.

Add BCEXPS to the APSAPPL application and depending on where you need this user rule add the statement \$BCEXPS followed by the map name. All AMB user rules must begin with a dollar \$ sign.

```
PARA 8000-SEND-SCREEN
```

```
*****
/* SEND SCREEN
*****
$BCEXPS TMCSM62
```

Non-Host Modularity

- *WebServiceConsumerTemplate* — The *WebServiceConsumerTemplate* class (Java object-related element) is intended to provide a common pattern for sending and receiving data to SOAP-based web services. This is a component of the Service Consumer Interface (SCI) common module for I/S Java applications that consume SOAP web services through the Communication Hub.

7.2.4.9 Existing ISSM Standards

- Refer to *Technical Standards — Applications > Application Coding > AMB/COBOL > Other Considerations or User Rules Guidelines*.
- Refer to *Technical Standards — Applications > Application Development Support Tools > Host Application Languages > AMB/COBOL Online Considerations*.
- Refer to *Technical Standards — Applications > Application Coding > Language Specific Standards > Java*.

7.2.5 CICS Screen Design Consideration for All Input Channels

7.2.5.1 Concepts Involved

- Reuse
- Flexibility

7.2.5.2 Operating System Platform

Host/Non-Host

7.2.5.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.2.5.4 Required Skills/Knowledge

- CICS
- APS
- TIOA

7.2.5.5 Related Techniques

- TIOA Utilization
- Single Source Application Design

7.2.5.6 Overview

The purpose of this technique is to ensure that the design of CICS screens facilitate efficient and effective interactions for all input channels. **Input channels** refer to Presentation Layer Applications such as Intranet Desktop Applications, Internet Secure Web Applications and Interactive Voice Response (IVR) applications.

This is important at BlueCross BlueShield of South Carolina (BlueCross) because its core applications are written once on the Host and leveraged many times by all input channels. The CICS screen designers at BlueCross must always consider the various types of input channels and provide the most direct and efficient path to CICS screen data.

7.2.5.7 Detailed Description

The following are “Global Design” considerations for developing new CICS screens.

Call String/TIOA (Direct Call Paths)

All CICS screen transactions should be programmed to allow access via a **Call String** (TIOA). The use of the TIOA (Terminal Input Output Area) to pass required data to an application can enhance performance (see related techniques: TIOA Utilization).

CICS Screen Attribute Considerations

Only the **Dark** (hidden value), **Color** and **Highlight** (Blink, Reverse Video, Underline) screen attributes should be used in CICS screen design at BlueCross.

The 3270 CICS emulators handle screen attributes differently within Desktop Applications, and screen scraping tools may not recognize the other CICS screen attributes.

Screen Messages

To leverage screen messages across all types of input channels, the messages displayed on CICS screens must be descriptive and user-friendly.

New screen messages must not include variables (i.e., having SSN or RACF ID strung into the message).

EESM's messages must always contain text to indicate successful or not successful. Blanks should never be a valid message.

Screen Field Considerations

All screens that access DB2 tables and scroll must use the DB2 count command and display the current page number and the maximum number of pages.

All screen fields must be initialized.

Stringing multiple fields together into one field should be minimized.

In addition to the “Global Design” considerations, CICS analyst must consider the direct user interactions with the 3270 CICS emulators. There are two basic types of direct user data entry.

1. Head-Down Entry (Push Data)

This type of direct user data entry emphasizes the user’s data entry speed. The user will rapidly enter screen data from a form or other type of document, and will seldom look at the screen while entering the data. CICS screen layouts should mimic the organization of the source data (form/document) to assist the user in speed and efficiency.

Heads-Down Data Entry screens are usually designed to place as many data fields as possible on a screen, which minimizes the total number of screens required to enter a unit of work. The user will not have to spend time paging to another screen to enter more data. Also, minimizing the number of screens executed by the presentation layer will maximize performance for that input channel.

On Heads-Down Data Entry screens, using abbreviated field labels and codes is a method to make more room available on the CICS screen for data fields. If a code is used on a screen, the definition of the code must be on an external reference table, such as GT Assist, RULEs, or an application DB2 table.

2. Inquiry/Update (Ask/Answer) Data Entry

This type of direct user data entry does not emphasize the user’s data entry speed. With this type of data entry, the screen, rather than a form or document, should act as a guide to the user. Field labels, returned data and screen messages should be highly descriptive and self-explanatory so they can be leveraged from the CICS screen to the presentation layer.

7.2.5.8 Existing ISSM Standards

- Refer to *Technical Standards — Applications > Application Coding > Online Screen Standards*.

7.2.6 User-Centered Design

7.2.6.1 Concepts Involved

- Modularity
- Loose Coupling
- Reusability
- Flexibility

7.2.6.2 Operating System Platform

Non-Host/Host

Concepts & Techniques > Common Platform Techniques

Printed forms of this document are uncontrolled and may be obsolete. It is the responsibility of the user to ensure that any printed copy is the same revision as the online version.

Copyright ©2023, BlueCross BlueShield of South Carolina. All rights reserved. BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association.

7.2.6.3 Intended Audience

- Architects
- Solution System Designers
- System Designers
- Business System Analysts
- System Analysts
- Usability Designers
- System Experts
- Software Designers
- Software Developers
- Test Designers, Testers

7.2.6.4 Required Skills/Knowledge

Knowledge of usability principles

7.2.6.5 Related Techniques

N/A

7.2.6.6 Overview

User-centered design (UCD) is a design approach that focuses on the people who will use the product or system — the end users. UCD is a design philosophy and process where the needs, wants, and limitations of end users are considered at every phase of the design process. The product or system is designed for the people who will use it, rather than making end users mold to a product or system designed with limited end user consideration.

Along with designing and implementing solutions for internal users, BlueCross BlueShield of South Carolina (BlueCross) develops products and systems which allow members, beneficiaries, providers, agents, shoppers, and benefits coordinators/group administrators to self-serve. Health care reform, business drivers such as transparency and consumer-driven health, and the need to be a low-cost provider of health claims services means there is a demand for self-service solutions for all constituents. While UCD is important for the design of internal applications, it becomes even more critical to success in achieving optimal self-service solutions.

UCD must always be conducted as a process that adheres to the agreed upon business requirements. It must consider and balance what can be achieved within the architecture and performance standards. The architectural concepts of modularity, loose coupling, reusability, and flexibility support a user-centered design approach.

BlueCross BlueShield of South Carolina uses several UCD techniques. Those techniques include participatory design, field studies, affinity diagramming, card sorting, low- and high-fidelity prototypes, usability testing, and heuristic evaluations.

7.2.6.7 Detailed Discussion

A UCD approach involves active participation of real users and iteration of design solutions. The emphasis is on user tasks, goals, and experience. The process focuses on cognitive factors such as learning, memory, problem-solving, etc. Designs are evaluated in terms of usability.

Usability is a **quality attribute** that assesses how easy user interfaces are to use. The word “usability” also refers to methods for improving ease-of-use during the design process.

Usability has five quality components:

- **Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the design?
- **Efficiency:** Once users have learned the design, how quickly can they perform tasks?
- **Memorability:** When users return to the design after a period of not using it, how easily can they reestablish proficiency?
- **Errors:** How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- **Satisfaction:** How pleasant is it to use the design?

“There are many other important quality attributes. A key one is **utility**, which refers to the design’s functionality: Does it do what users need? **Usability** and **utility** are equally important: It matters little that something is easy if it’s not what you want. It’s also no good if the system can hypothetically do what you want, but you can’t make it happen because the user interface is too difficult. To study a design’s utility, you can use the same user research methods that improve usability.” (Jakob Nielsen, “Usability 101: Introduction to Usability,” *Alertbox Newsletter* (January 4, 2012). <http://www.nngroup.com/articles/usability-101-introduction-to-usability/>

At BlueCross, the following techniques are used at various stages in the user-centered design process.

Participatory Design

This is a method in which analysts, developers, business representatives, and end users work together to design a solution.

Benefits

- Give users a voice in the design process, thus increasing the probability of a usable design.
- Enable technical and non-technical participants to participate equally.
- Provide an opportunity for analysts/developers to meet, work with and understand their users.
- Provide a forum for identifying issues.
- Provide an opportunity to get or enhance user buy-in.

Field Studies

This is a method in which analysts interview users and observe them in their environment. Data from field studies helps drive the design.

Benefits

- Aids in understanding a user's workflow.
- Aids in learning how the user interacts with a design in his/her own environment.
- Identifies the ways that users interact with an application, which can be very different than the analyst or management thought.

Affinity Diagramming

This method is used to sort large amounts of data into logical groups. It is used to analyze and organize findings from field studies, requirements gathering, or usability testing.

Benefits

- Simple and cost effective technique for soliciting ideas from a group and obtaining consensus on how information should be structured.
- Low-tech method that is fast and flexible.

Card Sorting

Card Sorting is a technique for exploring how people group items, so that you can develop structures that maximize the probability of users being able to find items. This method is similar to Affinity Diagramming, but is conducted with one user at a time. This method is appropriate when you have identified items that you need to categorize.

Benefits

- Easy and cheap to conduct.
- Enables you to understand how “real people” are likely to group items.
- Identifies items that are likely to be difficult to categorize and find.
- Identifies terminology that is likely to be misunderstood.

Low- and High-Fidelity Prototypes

Low-fidelity prototypes are usually paper and pencil sketches used to clarify requirements and enable draft interaction screen designs to be very rapidly simulated and validated.

Benefits

- Potential usability problems can be detected at a very early stage in the design process before any code has been written.
- Promotes communication between designers and users.
- Paper prototypes are quick to build and refine, thus enabling rapid design iterations.
- Only minimal resources and materials are required.

High-fidelity prototypes are usually an online format that is a working prototype with lots of detail and functionality. They are very close to what will most likely be the end product design.

Benefits

- Can perform user validation that assesses usability in detail and allows you to make strong conclusions about the design.

- Potential usability problems can be detected before any code has been written.
- Promotes communication between designers and users.

Heuristic Evaluation

Heuristic evaluation is a technique for finding usability problems with a user interface. Trained analysts inspect the interface and apply a set of “heuristics,” broad guidelines that are generally relevant. The importance of the identified issues and problems are ranked, then prioritized and fixed.

Benefits

- Provides quick feedback to designers or analysts.
- Ensures the design conforms to established guidelines, which helps to promote compatibility with similar systems.
- Focus is given to key usability issues such as navigation, design strategy and consistency.

Below are ten general principles for user interface design. They are called “heuristics” because they are more in the nature of rules of thumb than specific usability guidelines.

- **Visibility of System Status**
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- **Match between System and the Real World**
The system should speak the user’s language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. It should follow real-world conventions, making information appear in a natural and logical order.
- **User Control and Freedom**
Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Supports undo and redo.
- **Consistency and Standards**
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- **Error Prevention**
Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
- **Recognition Rather Than Recall**
Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- **Flexibility and Efficiency of Use**
Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- **Simplicity (Minimalist Design)**
Dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- **Recognize, Diagnose, and Recover From Errors**
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

- **Help and Documentation**

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Usability Validation

This method is a rigorous usability evaluation of a prototype or working system under realistic conditions to identify usability problems and to compare measures such as success rate, task time and user satisfaction with requirements. (This is not the same as customer system validation.)

Benefits

- Find out if a design is usable.
- Identify elements that work and don't work.
- Get insight on how to improve a design.
- Shows the true interaction between the user and the system.

7.2.6.8 Summary

User-centered design enables BlueCross to deliver products and systems that provide high-quality user experiences. The quality of a customer's interaction with the company can make or break the relationship. The more that is done to optimize the customer's experience when interacting with our products and systems, the more likely the experience will be a positive one.

7.2.6.9 Existing ISSM Standards

N/A

7.2.7 Concept Diagrams

7.2.7.1 Concepts Involved

Reuse

7.2.7.2 Operating System Platform

Host/Non-Host

7.2.7.3 Intended Audience

- Architects
- Solution System Designers
- System Designers
- System Analysts
- Business System Analysts
- System Experts

7.2.7.4 Required Skills/Knowledge

- System Design and Analysis Skills
- Visio
- PowerPoint

7.2.7.5 Related Techniques

Single Source Application Design

7.2.7.6 Overview

A Concept Diagram is a one-page diagram that depicts the inputs, outputs and process flow of the solution. It also depicts the system components, infrastructure components and ownership of the solution. Concept Diagrams are a deliverable for the Discovery and Delivery Strategy and Design phases and are typically created in PowerPoint or Visio. The Initial Concept Diagram created in Discovery and Delivery Strategy is updated during the Design phase as needed and becomes the diagram of record for the solution.

A one-page Concept Diagram is always required. If more detail is required, additional pages can be attached. Concept Diagrams are also used to drive the design and development process and ensure the design team can explain the solution that is being designed to solve the client's business requirements.

7.2.7.7 Detail Discussion

Each Concept Diagram is unique to the business problem it is depicting to solve. However, each has the same purpose: depict a solution that solves a client's business problem. While there is not a magical way to create a Concept Diagram, the following guidelines should be considered when creating a new one.

Basic Approach

- Does a base Concept Diagram already exist for the Application System? If so, leveraging an approved diagram and notating the changes in the diagram is usually the best approach. Use "call out" boxes to describe the changes where necessary.
- If a base diagram does not exist, copying a diagram is not recommended, since this approach stifles creativity. Always start from a blank sheet of paper and create the base diagram for the application system.
- Think failure points - where could data be lost?
- Ensure all transfers of data are balanced using ABRS or Events
- Avoid technical jargon
- Use words on lines to describe transactional processes or data passed
- All diagrams must:
 - o Flow left to right, top to bottom.
 - o Identify the end users, the required inputs and output channels, the required processing, and what outputs and external partners are required.
 - o Show separation of environments (client vs. Non-Host vs. Host)

- o Show the method of transport required to move data. For example, two boxes that represent a process that communicates via MQ should show MQ as the transport mechanism.
- o Ensure all integration points are depicted.

Presentation Approach

- Use of color is recommended to note ownership.
- Use of swim lanes is recommended to show separation of infrastructure.
- Avoid diagonal lines and lines that cross.
- Minimize page connectors.
- Use legends to define colors of boxes (new vs. existing) technology and line flows.
- Use of colored lines is recommended to note the difference in transport (MQ vs. Web Service).

7.2.7.8 Existing ISSM Standards

- Refer to *Application Systems Management > Application Systems Management Framework > Discovery and Delivery Strategy Phase > Requirements Documentation*.

Attached Example (Figure 7-10):

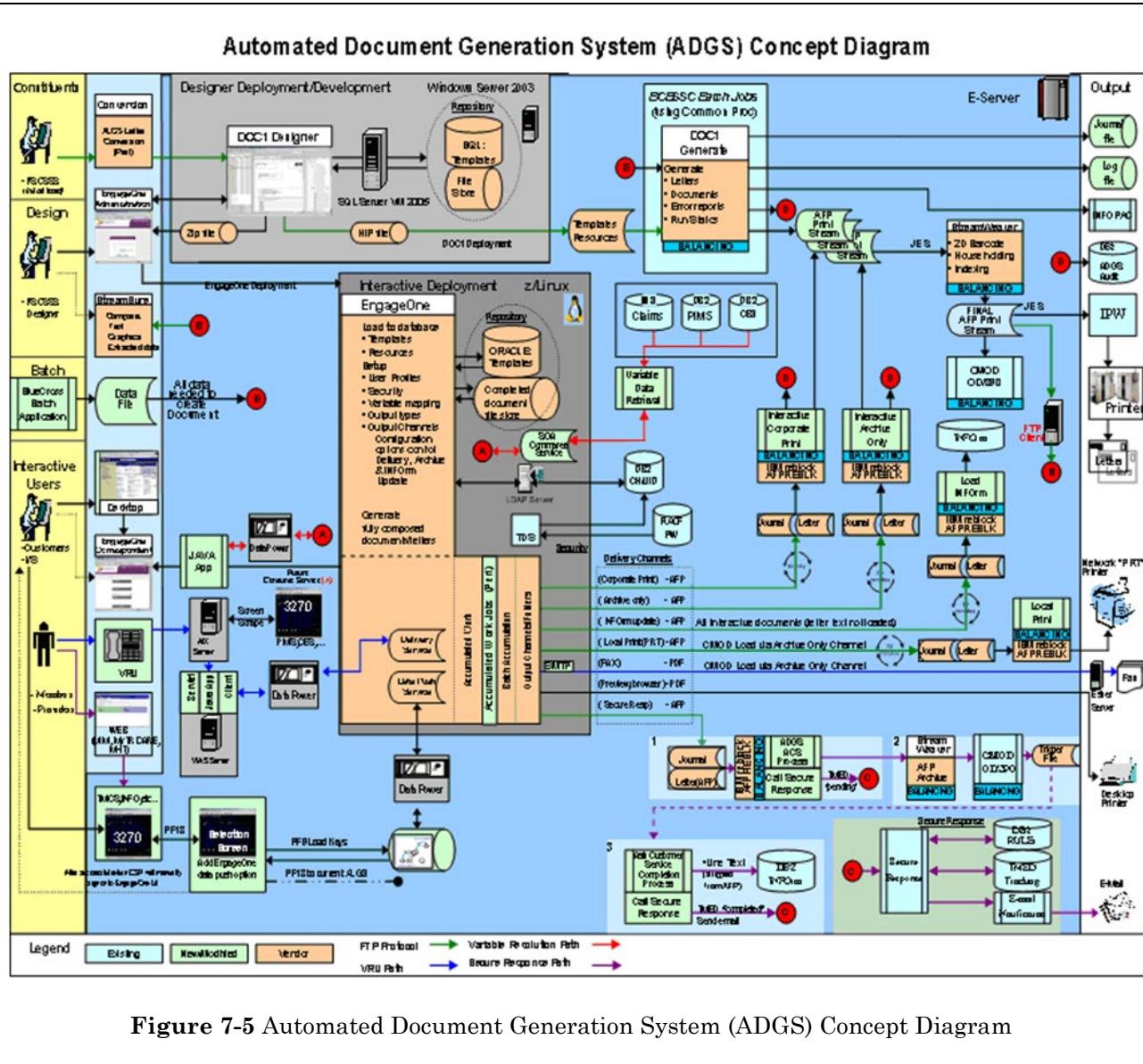


Figure 7-5 Automated Document Generation System (ADGS) Concept Diagram

7.2.8 Informational Database

7.2.8.1 Concepts Involved

- Flexibility
- Reusability

7.2.8.2 Operating System Platform

Host/Non-Host

Concepts & Techniques > Common Platform Techniques

Printed forms of this document are uncontrolled and may be obsolete. It is the responsibility of the user to ensure that any printed copy is the same revision as the online version.

Copyright ©2023, BlueCross BlueShield of South Carolina. All rights reserved. BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association.

7.2.8.3 Intended Audience

- System Analyst
- System Designer
- System Expert

7.2.8.4 Required Skills/Knowledge

Programming Skills

7.2.8.5 Related Techniques

N/A

7.2.8.6 Overview

Creation of Informational databases is a standard for all BlueCross BlueShield of South Carolina (BlueCross) systems. Informational databases are created from an Operational environment into a reporting environment which allows users to run ad hoc queries for reporting. BlueCross utilizes this technique to prevent reporting activities from degrading production processes. BlueCross uses several approaches to accomplish the creation of Informational databases.

7.2.8.7 Detail Discussion

The following approaches are used for creating Informational databases; they can be used for IMS, DB2, Oracle, and SQL Operational database structures. BlueCross uses DB2 as the database structure for all informational reporting, regardless of which Operational database structure is used. DB2 is used for informational reporting because it provides flexibility. Using ad hoc queries, customers can easily access their data. The use of any other Informational database structure than DB2 must be approved by the Enterprise Architects.

Approach 1 — Full Copy Overlay of Informational

This approach should be used when the data on the Informational database needs to be a snapshot of the production data for a given time period. The size of the database should be considered before using this approach, since all of the data is being overlaid on the Informational table, regardless of whether the data has changed or not. (See Figure 7-11.)

DB2 Database Structure

When the Operational database structure is DB2, this approach will only require the DBA area to set up an UNLOAD process and a LOAD REPLACE process using DB2 utilities.

Pros:

1. This approach can be accomplished by using existing utilities instead of application development.

Cons:

2. Using this approach on very large databases can be very time consuming.
3. This approach limits the amount of data available for reporting to the same timeframe as the Operational system.

Non-DB2 Database Structure

When the Operational database structure is Non-DB2, this approach will require an application program to unload the Non-DB2 structure and create the required DB2 format needed for the Informational database. This file can then be loaded to the Informational database using a DB2 utility.

Pros:

1. Since this approach requires an application program, additional logic can be applied to the data before it is loaded to the Informational database.

Cons:

2. Using this approach on very large databases can be very time consuming.
3. This approach limits the amount of data available for reporting to the same timeframe as the Operational system.

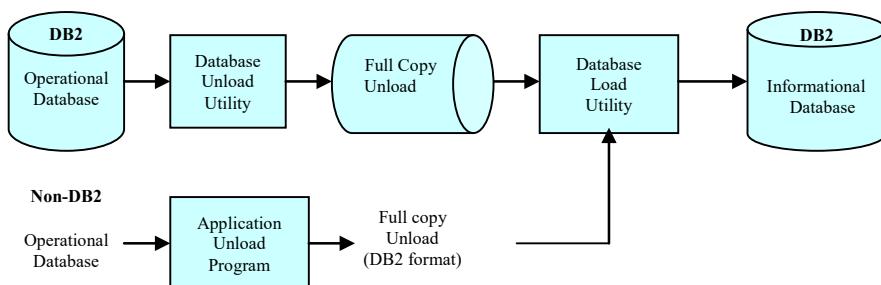


Figure 7-6 Full Copy Overlay of Informational

Approach 2 — Apply Selected Changes to Informational

This approach should be used when the Informational database requires a historical view of data, typically requiring more data than is required for your Operational environment (i.e., Operational environment requires six months of data, whereas the Informational environment requires ten years of data). Applying only changes from the Operational environment to the Informational environment significantly reduces the time to update the Informational databases. (See Figure 7-12.)

Pros:

1. Informational database is not limited to the size of Operational databases.
2. Only records that have changes are applied to Informational databases.
3. This reduces the time needed to update the Informational databases.

Cons:

1. An application program(s) is required to accomplish this approach.

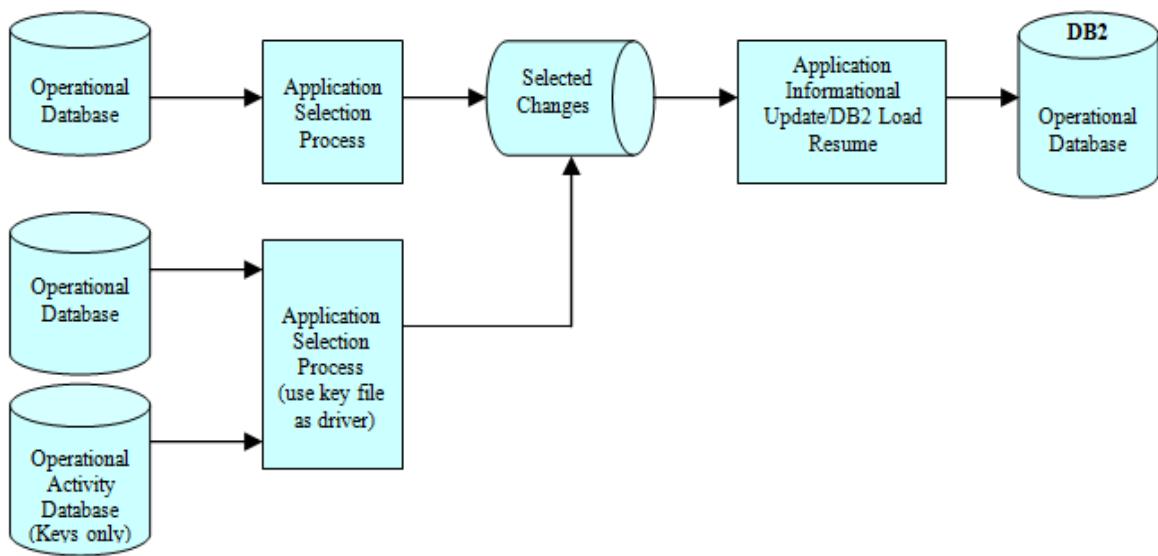


Figure 7-7 Apply Selected Changes to Informational

Approach 3 — Mirror Databases

This approach should be used when the Informational database is required to be a current copy of the Operational database. A Mirror database is an informational copy of an Operational database which is maintained by the Operational system. When changes are applied to the Operational database, these same changes are also applied to the Mirror database. This approach allows the customer to browse the Mirror database in an online mode with current data being available. (See Figure 7-13)

This approach has only been implemented in the WMS system. The use of this approach requires the approval of the Enterprise Architects.

Pros:

1. Informational database is available in a real time mode.
2. No Batch cycle is needed to update Informational databases.

Cons

1. There is overhead associated with the Operational system maintaining additional databases.
2. Performing reporting activities in the Operational environment could possibly have a negative impact on the Operational environment.

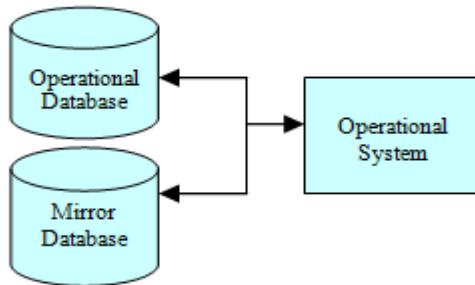


Figure 7-8 Mirror Databases

7.2.8.8 Existing ISSM Standards

- Refer to *Technical Standards — Applications > Application Coding > Online Design Considerations*.

7.2.9 Paperless Processing

7.2.9.1 Concepts Involved

- Modularity
- Loose Coupling
- Reuse
- Encapsulation
- Balancing and Reconciliation

7.2.9.2 Operating System Platform

Host/Non-Host

7.2.9.3 Intended Audience

- Business System Analysts
- Software Designers
- Software Developers
- System Designers
- Solution System Designers
- System Experts
- Architects

7.2.9.4 Required Skills/Knowledge

To understand this concept, a basic understanding of Imaging and Business Process Management in an application development environment is required.

7.2.9.5 Related Techniques

- Workflow Automation
- Document Archival

7.2.9.6 Overview

Paperless Processing occurs when a person works from an image of a document instead of the physical hard copy document itself. Paperless Processing includes both hard copy documents converted to electronic images, and soft copy documents created electronically via software such as MS Office or Enterprise Server applications. Paperless Processing provides numerous benefits related to document management. Paperless Processing provides:

- The ability to control the document/image as soon as it is received by:
 - Eliminating the need to manually share/route a hard copy document.
 - Applying application security to ensure that only authorized personnel can access and view an image.

- A reduction in lost, misplaced or misrouted paper documents by providing electronic document tracking and real-time status capabilities.
- An increase in the accessibility of the document by:
 - Allowing multiple users to access and view the document/image simultaneously.
 - Allowing the document/image to be accessed and viewed from physically different locations.
- A reduction in the general use of paper (e.g., printing additional hard copies).
- A reduction in the amount of physical storage required to store physical paper.

7.2.9.7 Detail Discussion

Paperless Processing involves:

1. **Capturing** a paper document automatically and generating an image of that document.
2. **Importing/uploading** documents/images through the system.
3. **Indexing** key data related to documents/images.
4. **Storing** the document/image in a repository.
5. **Searching, Retrieving and Viewing** the image as required to work with the document/image.

As illustrated in **Figure 7-14**, hard copy documents are usually routed to a department's mail prep area. The documents are prepared for imaging by batching similar document types together. The documents are then fed through a high-speed or low-speed scanner to **capture** them as an image. At the same time as scanning, a document control number (DCN) is assigned to each image. The DCN is a unique number assigned to each scanned document and serves as the image's primary index value.

When a document is faxed, the image is **captured** on the fax server instead of being printed out to a physical fax machine. The fax image also receives a unique DCN.

Documents/images can also be electronically produced by external and internal systems. Reports, letters, and forms are examples of documents generated by both internal and external systems that are never in hard copy form. These documents then use an **importing/uploading** process to bring the document into a system for indexing and storing the documents/images.

Following the capturing or importing step, a document/image can have additional indexes assigned. Indexing can happen systematically/automatically if all of the key data can be assumed or determined (e.g., through system lookups). Indexes can also be assigned manually.

Following the indexing steps, the document/image and associated indexes are stored in a document/image repository to allow users to **search, retrieve and view** the document/image. The document/image can simply be archived or users can utilize the document/image in order to complete tasks in response to the receipt of the document. Multiple users should be able to access the single source document/image concurrently, based on security, rather than waiting on a hard copy document to be passed around consecutively.

Paperless Processing often occurs in conjunction with an automated workflow process.

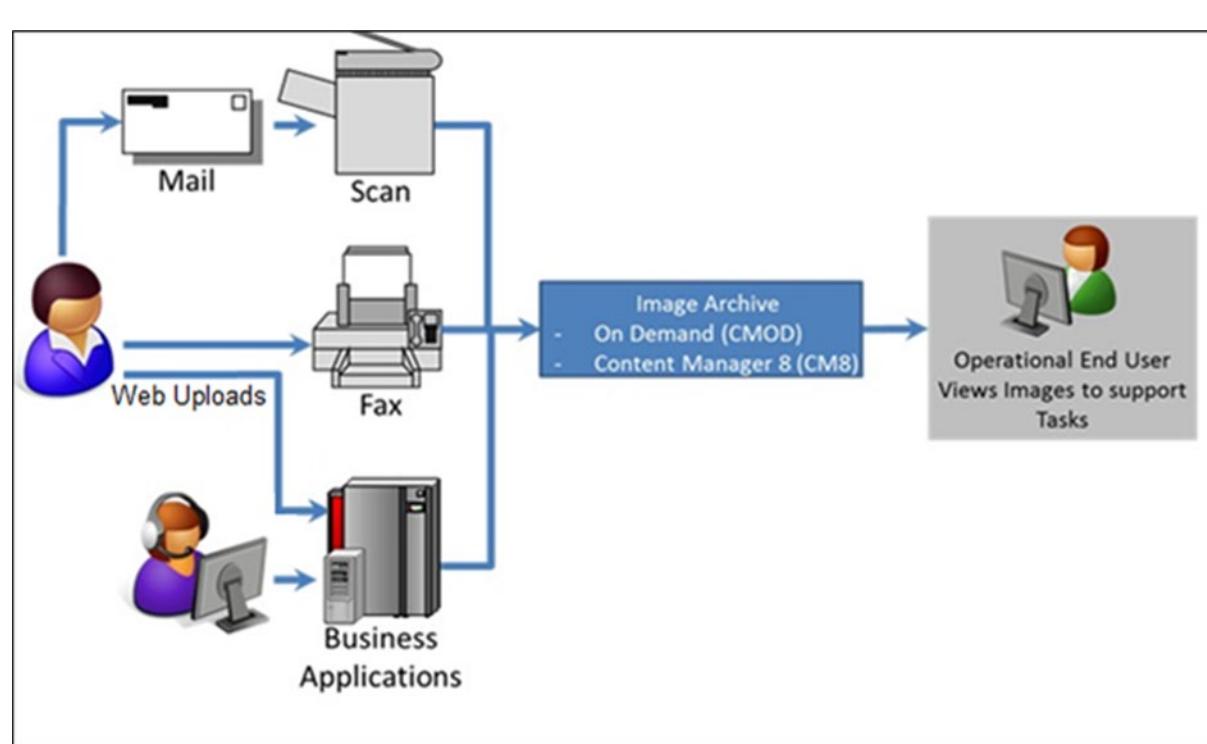


Figure 7-9 Paperless Processing Example

7.2.9.8 Opportunities to Employ Paperless Processing

Paperless Processing can be used to gain efficiencies in many work environments. There are certain triggers that can help identify ideal work environments for Paperless Processing:

- Any environment where people currently handle hard copy documents to complete tasks
- Any environment where people manually move hard copy documents to other people in order to facilitate task completion
- Any location that receives documents through channels where the document is already an electronic/soft copy upon initial receipt (e.g., fax, email attachment, document stored on the network)

7.2.9.9 Optical Character Recognition

Beyond basic scanning, images can also be captured through a process called Optical Character Recognition (OCR). This scanning solution is used when a document is submitted in a static layout, such as a claim form, where data usually populates fields in predictable locations on the form document. OCR can be used to capture the data and send it on to a system for processing. In the example of claims, the data elements from a claim are captured through OCR and sent to the appropriate processing system for adjudication (e.g., AMMS, CMMS). The image is then stored in an archive repository.

7.2.9.10 Tools and Applications for Paperless Processing

The tools employed to enable Paperless Processing are specific to each major system function:

- High-Speed Scanning/OCR — AnyDoc
- Faxing — Esker Deliveryware, FaxBack Net SatisFAXtion
- Autocapture Print Stream — Print to DocFinity
- Document Generation — Automatic Document Generation System (ADGS) batch and real-time, and various Host applications
- Archival — Content Manager On Demand (CMOD), Content Manager 8 (CM8)
- Uploads from full-service and self-service browser-based applications

7.2.10 Workflow Automation

7.2.10.1 Concepts Involved

- Modularity
- Loose Coupling
- Reuse
- Encapsulation
- Balancing and Reconciliation

7.2.10.2 Operating System Platform

Host/Non-Host

7.2.10.3 Intended Audience

- Business System Analysts
- Software Designers
- Software Developers
- System Designers
- Solution System Designers
- System Experts
- Architects

7.2.10.4 Required Skills/Knowledge

To understand this concept, a basic understanding of Imaging and Business Process Management in an application development environment is required.

7.2.10.5 Related Techniques

- Document Archival
- Paperless Processing

7.2.10.6 Overview

Workflow is defined as a sequence of connected steps through which work is done. In an operational area, workflow is often used to describe the process of tasks and steps being completed by people responsible for handling documents or other units of work.

Workflow automation is the implementation of technology to improve the processing of the work through the workflow.

Workflow automation is often implemented in order to gain efficiencies in an otherwise manual process. High-level efficiencies include

- Ability to control the distribution of work through
 - Automation of the movement of work between people
 - Enforcing linear or sequential steps when required
 - Allowing for parallel or simultaneous steps when allowed
 - Ensuring work is processed in a uniform manner
- Ability to systematically capture data for reporting on
 - Productivity data — how fast the work is processed
 - Operational data — the amount and type of work being processed within the system

7.2.10.7 Detailed Features

Workflow automation includes the detailed processes of workflow distribution and task management. Workflow distribution notifies a user of the next available piece of work (based on business rules, key index data, or a combination of both) and guides the user through completing tasks. After the user completes a piece of work, the workflow will automatically move the work to next appropriate step. If more than one person should complete tasks at the same time for one piece of work, the automated workflow will notify all users accordingly. Also, the workflow maintains control over the priority of work within a workload and distributes higher priority items as the process requires.

For automated workloads, the work can be distributed to an individual or a group of people. Automated workflow uses a personal queue to display the work assigned to a single person while a common queue is used for work assigned to a group of people. Business requirements usually drive the type of workload queue used for each business process supported by an automated workflow.

7.2.10.8 Opportunities to Employ Workflow Automation

Workflow automation can be used to gain efficiencies and ensure standard processing in many work environments. There are certain triggers that can help identify ideal work environments for an automated workflow:

1. A work environment where artifacts or resources must be referenced by many people or passed among individuals in order to facilitate task completion.

2. A work environment where service levels must be monitored from the time a piece of work is received in-house to the time the final action is completed.
3. A work environment where the path of task completion for the document needs to be ensured, whether it is linear or parallel in nature.

7.2.10.9 Workflow Automation and Paperless Processing

Work is usually triggered when a document is received. Since paperless processing calls for the processing of documents through use of an image (as opposed to the original hardcopy), workflow automation is typically integrated in a larger system solution that includes capturing, indexing and archiving solutions as well (Figure 7-15).

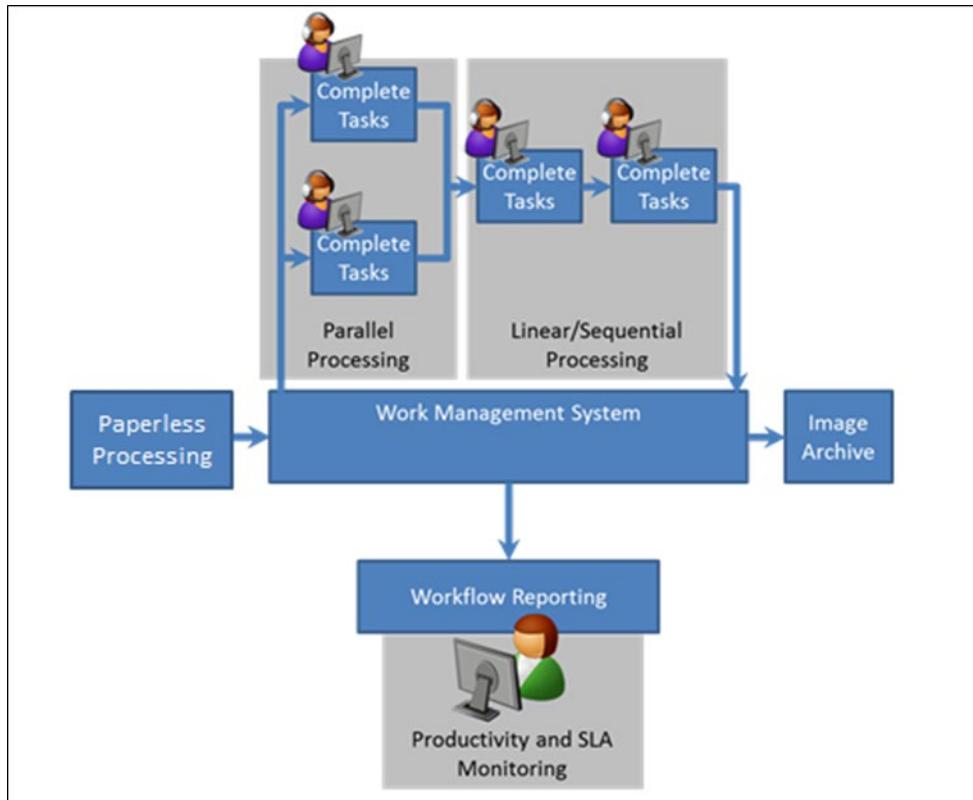


Figure 7-10 Example Workflow Diagram

7.2.10.10 Tools Supporting Workflow Automation

- DocFinity
- iFlow

7.2.11 Document Archival

7.2.11.1 Concepts Involved

- Modularity
- Loose Coupling
- Reuse
- Encapsulation
- Balancing and Reconciliation

7.2.11.2 Operating System Platform

Host/Non-Host

7.2.11.3 Intended Audience

- Business System Analysts
- Software Designers
- Software Developers
- System Designers
- Solution System Designers
- System Experts
- Architects

7.2.11.4 Required Skills/Knowledge

To understand this concept, a basic understanding of Imaging and Business Process Management in an application development environment is required.

7.2.11.5 Related Techniques

- Workflow Automation
- Paperless Processing

7.2.11.6 Overview

Document Archival is the long-term storage of documents. Multiple file types, such as MS Office, PDF, TXT, AFP, HTML, TIFF are stored in an archive repository. These documents can be generated internally as a by-product of an internal system (e.g., remittances, letters, explanations of benefit, etc.) or they can be received from an external entity (e.g., Claims or correspondence from providers). Archiving involves associating relevant key values or indexes with each document when it is stored to enable future search and retrieval needs. Archiving also provides for several additional needs:

- Document storage in a centralized repository for search and retrieval purposes
- Long-term storage space for systems that generate or temporarily house the documents while work on the document is in process
- Repository back-up procedures to provide for Disaster Recovery purposes

- Legal compliancy regarding the company's obligation to retain certain document types

7.2.11.7 Detailed Features

Hard copy documents are imaged (i.e., scanned) so that the document can be stored in an archive repository, which allows for the hard copy to be destroyed. Also, archiving is used for storing any documents or files that are systematically created and then distributed as paper documents, such as a remittance, explanation of benefit or letter.

Indexes are stored with an archived document in order to allow for users to search for it within the document repository. Unique index data, such as a document control number (DCN), should produce a search result of a single document. However, since archiving allows for the storage of multiple indexes in addition to the DCN, multiple documents may display in the search results when a non-unique index value (e.g., Subscriber ID, Date of Service) is used for search criteria.

7.2.11.8 Opportunities to Employ Document Archival

1. Situations where hard copy documents are received, imaged (through scanning), and stored in a file system for future use or referenced after being actively worked.
2. Contractual or legal requirements to store documents for extended or indefinite timeframes.
3. Instances where system output results in a hard copy document being mailed to an external entity. These documents usually need to be available for future use (e.g., displaying for CSR staff via a desktop application or displayed to our customers via a website).
4. Efforts to reduce dependency for large or expensive short-term physical storage space.
5. Efforts to address system performance where short-term online storage has a direct impact on a system's ability to process efficiently.

7.2.11.9 Repositories for Document Archival

- Content Manager On Demand (CMOD) for z/OS
- CMOD for Host document formats
- CM8 for Non-Host document formats
- DocFinity
 - o This repository serves as an interim repository for documents that are processed through the DocFinity workflow system.
- SharePoint Online (SPO)

7.2.11.10 Methods for Document Archival

- Virtual Tape
- Network Share/File Server

7.2.11.11 Tools Supporting Document Archival

- Electronic Data Archive (EDA)
- Message Queue (MQ)
- IBM Content Collector (ICC)

7.2.12 Software Stubbing

7.2.12.1 Concepts Involved

- Modularity
- Reuse
- Loose Coupling
- Flexibility
- Data Management

7.2.12.2 Operating System Platform

Host/Non-Host

7.2.12.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects
- Solution System Designers
- System Designers
- Testers
- Test Designers

7.2.12.4 Required Skills/Knowledge

N/A

7.2.12.5 Related Techniques

- Encapsulation

7.2.12.6 Overview

Software stubbing is the act of creating a software stub. A software stub (Figure 7-16) is a process that serves as substitute functionality when the intended process is unavailable during the Design, Development and Validation Phase. The stub has the interface (I/O) of a real process but is merely an

interactive placeholder used as a stand-in to allow development and unit validation until the intended actual process is available. Software stubs are commonly used as placeholders for routines that still need to be developed or modified. Software stubs are usually built with just enough intelligence for the interface to respond with useful output. Using this technique allows basic unit validation to occur faster, without waiting on delays in a component dependency (such as a Web service, CICS region, MQ message, or even scenario-based test data).

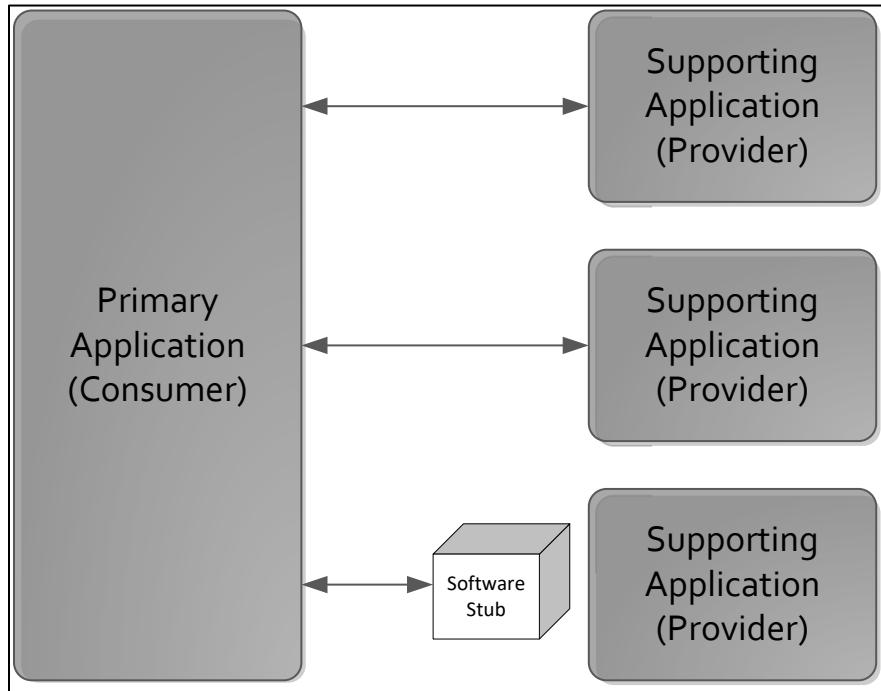


Figure 7-11 Software Stub — The software stub sits between the consumer and provider applications and will respond as if it is the provider. The consuming application believes that it is actually communicating with the provider application.

Software stubbing is most useful in software development and validation. It is not intended to replace the practice of integrated validation. Integrated validation should remain an activity of the Design, Development and Validation Phase of a work effort and should include fully functional interfaces, not software stubs.



NOTE This technique is not to be confused with APSSTUB which is used to perform specific includable code and paragraphs within Host programs. For more information on APSSTUB, refer to *Systems Architecture > Concepts & Techniques > Common Platform Techniques > Modules by Function*.

7.2.12.7 Detailed Features

Software development can occur in parallel as long as there are clearly defined interfaces between each dependent system. A consuming application need not have knowledge of a component's internal implementation; through **encapsulation** it is shielded from the differences between a "fully functional" and a stand-in software stub that returns specific data.

Use of software stubs supports the following aspects of software development and validation:

- Developing Dependent Applications in Parallel
- Test Data Management
- Third Party Services
- Load Testing

Developing Dependent Applications in Parallel

In the world of BlueCross, we build integrated application systems in order to reduce costs and achieve consistency across our distributed or Non-Host systems. In this model, we have three distinct application layers:

- **Application** (for example, Host)
- **Integration** (for example, Web services)
- **Presentation**. The presentation applications consume and rely upon the integration layer which, in turn, depends on systems within the application layer for business logic and data access.

Per Application Systems Development Methodology, during the Design, Development and Validation Phase, designers from all three layers collaborate to produce well-defined code-level artifacts which express a contract of how the software layers will communicate with each other. This approach is intended to unify the solution team under a common design and facilitate parallel development across the three layers.

Parallel development can be hindered when all validation that is done is fully integrated validation. The integration and presentation layers must wait until changes in the application layer are available for consumption. Then the presentation layer has to wait for the integration layer changes to be available.

By using software stubs, each team in the three layers is able to work in parallel because their integration dependencies can be met with software stubs. This assists with shortening the time line required to deliver software solutions for our business customers.

Test Data Management

Putting together test data for integration or presentation layer systems can be a challenge because of the integrated nature of the application layer data stores. It is not as easy as simply loading claims data into AMMS and expecting it to work in a presentation system. The corresponding data also needs to be loaded into the membership system. Depending on the presentation system, other data loads may be required in order to fully validate functionality.

With software stubs, the data is completely controlled by the software stub and it becomes much easier to correlate all of the data across the various systems. This allows integration and presentation layer systems to unit test more scenarios than what is possible through fully integrated layers.

Third-Party Services

As part of an application across all layers, we use third-party integrations. Examples include external membership and eligibility systems such as DEERS and FEP, and external financial institutions. These systems are not controlled by BlueCross and can cause integration issues in test environments. When work is scheduled on those applications, BlueCross is unable to validate with them. Sometimes these business transactions also have a cost associated with using them, such as financial transaction processors.

With software stubs, functional and performance validation can occur without having to coordinate around maintenance windows or downtime. Software stubs also allow us to unit test the third-party services without incurring any costs related to using them.

Load Testing

Our applications are fully integrated from the presentation layer all the way down to the application layer. This means that when a load test is executed, we not only load test the presentation system, we also load test the integration layer as well as the application layer. If a load problem is detected, the teams from all layers undergo extensive research to determine where the problem exists.

Software stubs provide an opportunity to execute early load testing because all layers of the solution can be load tested independently. A presentation layer application load test against software stubs would no longer load test integration or application layers. Any issues that came up would belong to the presentation layer application.

Decreasing the scope of load testing to only the layer that needs to be load tested simplifies testing. Simplification encourages more frequent load tests. With increased frequency of load tests, we increase confidence in the application quality.



NOTE Full stack load testing still needs to be done without stubs. Stubs should be used only to load test the individual pieces.

Examples

Consider the following examples on how software stubs could be used for software development and validation.

Example 1 — Presentation System Calling a Web Service

When a presentation application is being updated to consume a new or modified Web service, a software stub can sit between the two applications to allow for parallel development (Figure 7-17). The presentation application will have what appears to be a functional Web service to use that is provided by the software stub. The presentation application can continue to develop and validate changes while the Web service is still being developed. When the Web service is completely developed, validated, and deployed, the software stub can be shutdown to allow the presentation application to consume the actual service.

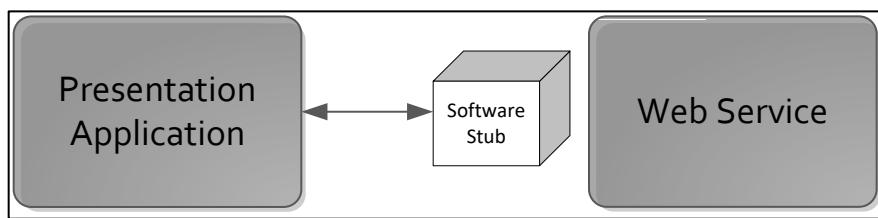


Figure 7-12 Presentation System Calling a Web Service — The software stub sits between the presentation application and the Web Service. The software stub will respond to the presentation application as if it were a live response from the Web Service.

Example 2 — COBOL Program Calling a Web Service via MQ

Host programs can call Web services through MQ messages. If the Host program needs to access a Web service that is unavailable because it is still being developed, then a software stub can be put together and put into place on the MQ Queue Manager (Figure 7-18). The software stub will intercept the traffic intended for the Web service and respond back to the Host program as if it were the service. Once the Web service is complete and ready for use, the software stub can be shutdown to allow the Host program to consume an actual service.

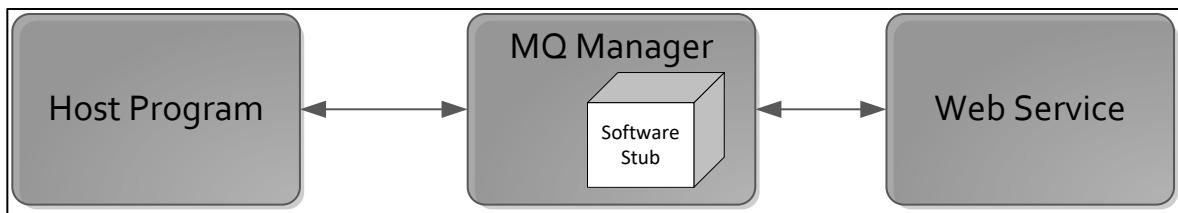


Figure 7-13 COBOL Program Calling a Web Service via MQ — The software stub sits on the MQ Queue Manager and will respond to the MQ message sent by the Host program as if it was a response from the Web Service.

Example 3 — Presentation System Scenario-Based Test Data

When validating presentation applications, sometimes the data that is needed can be hard to find or sometimes it does not exist. The software stub can be used to route traffic between actual Web services and virtual data (Figure 7-19). This allows for certain transactions to be processed by the actual systems while others pull their response from the virtual data. Filling in the test data gaps with virtual data will allow applications to be more thoroughly validated than with standard integrated validation.

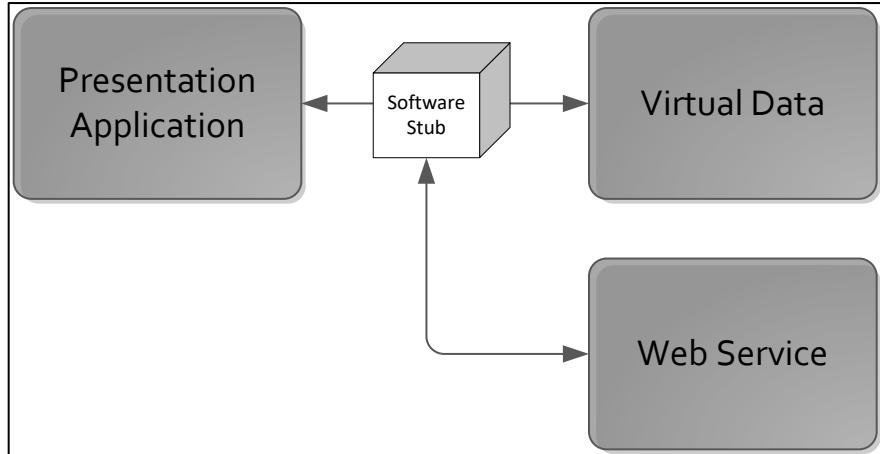


Figure 7-14 Presentation System Scenario-Based Test Data — The software stub sits between the presentation application and the Web Service. Based on rules, the software stub will allow the Web Service or Virtual Data respond. Allowing for more complete testing of edge cases.

7.2.12.8 Existing ISSM Standards

N/A

7.3 Host Platform Techniques

7.3.1 CICS Application Security

7.3.1.1 Concepts Involved

- Modularity
- Loose Coupling
- Reuse

7.3.1.2 Operating System Platform

Host

7.3.1.3 Required Skills/Knowledge

- CICS
- APS
- RACF

7.3.1.4 Related Concepts

- CICS Application Transaction Router Processing
- Common I/O and Edit modules
- Modules by Function
- Temporary Storage Queues

7.3.1.5 Overview

BlueCross BlueShield of South Carolina (BlueCross) utilizes the Customer Information Control System (CICS) application security systems and IBM's Resource Access Control Facility (RACF) security product to control access to all of its CICS applications. While RACF is used to handle CICS transaction level security, BlueCross CICS Application Security systems are needed when security is needed at the function/access level and/or field level and include all aspects of security for a CICS RACF User ID access including:

- Who Am I? RACF User ID access.
- Systems/Program Access — Levels of access (add/update/inquiry/void) a RACF User ID has.
- Field Level Access — Inquiry vs. update a RACF User ID has.
- A complete audit trail that meets Corporate Audit standards related to setting up and granting access to RACF User IDs.

7.3.1.6 Detailed Process

Who Am I? RACF ID Access

After RACF security is validated at the CICS transaction level, the RACF User ID Demographic security DB2 table is accessed first within the CICS Security system. This table serves as the entry point into security and is required so that all other security tables can be updated. Without a record loaded into this table, a RACF User ID cannot access the system.

Systems/Program Access

After the RACF User ID Demographic security DB2 table is accessed and validated, the Systems/Program Access DB2 Security table is accessed to ensure the RACF User ID has access to the program that is being accessed. Access level is typically controlled at the following levels:

- 0 — No access
- 1 — Inquiry only
- 2 — Inquiry/Add access
- 3 — Inquiry/Add/Update access
- 4 — Inquiry/Add/Update/Void access
- 5 — Full access

Security Field Level Control

Specific application code is required before field level security can be integrated into the application. The customer's business requirements will drive the security access needed. This type of access does require additional programming to ensure that the necessary security I/O module is called throughout the system.

An example of this type of access includes filtering out sensitive health programs information (e.g., mental health and chemical dependent) from individual RACF User IDs' workloads.

Security Audit Trail

In order to meet Corporate Audit standards, every function performed within the security system must be captured in a DB2 Audit repository such as:

- Adds to all tables.
- Any change to any security table field, to include a before and after image of the record.
- A date/timestamp must be included.
- The RACF User ID of the security administrator performing the task must be included.
- On average, security audit trails are retained within the system for ten years.

Code Reference example:

INTRD010

Common I/O Used by the Security System

I/O modules are typically integrated into the following areas within a CICS application:

- Router called (TMCSP000 call TMCSD024) — Program level access
- Program called (THMCP001 calls TMCSD024) — Field Level Access

Code Reference example:

TMCS024

Preloading Security

In order to optimize the performance of the CICS application's security system, security access information can be read when the CICS application's first program is executed and then passed through the system via temporary storage queues. This reduces the overhead of reading DB2 each time the security access information is needed.

7.3.2 CICS Application Transaction Router Processing

7.3.2.1 Concepts Involved

- Loose Coupling
- Reuse

7.3.2.2 Operating System Platform

Host

7.3.2.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.3.2.4 Required Skills/Knowledge

- CICS
- APS

7.3.2.5 Related Techniques

- INTRP000 — the CICS SuperRouter
- TIOA Utilization
- Plan and Package Binds (future)
- Temporary Storage queues
- CICS Application Security

7.3.2.6 Overview

BlueCross BlueShield of South Carolina (BlueCross) utilizes CICS application transaction routers (CICS routers) to handle CICS navigation between CICS transactions. The CICS router is specific to a CICS

application and usually ends with “P000” as the last 4 characters of the program name. All new application CICS routers must comply with this naming convention. The concept of CICS router processing relates to loose coupling and reuse. CICS routers ensure processing used to handle CICS navigation between programs and COMMAREA management is handled by one common program per CICS application. As part of navigation management, the CICS router manages and maintains certain standard PF-Keys used for navigation such as PF3, PF12 and clear key. The CICS router updates several reserved areas within the CICS COMMAREA for navigation including an array of programs accessed that is used for PF3- Back logic. It is also used as the “go-between” for CICS application security ensuring that the RACF accessing the program has security to render the next CICS screen within the navigation. Refer to *Systems Architecture > Concepts & Techniques > Host Platform Techniques > DB2 PLANBINDs and PACKBINDs* for further details.

In addition to navigation and COMMAREA Management, CICS routers are utilized to ensure the following:

1. Standardization of application keys and data passed within the applications via the COMMAREA. The COMMAREA size should be 4000 bytes or less so that customized coding is not required for the SuperRouter (INTRP000). See section titled “SuperRouter Processing” in this Chapter for additional information.
2. Common entry into CICS programs for all invocation modes (transaction invoked, program invoked, screen invoked) to ensure the code to handle such processing is in one common program and not cloned across many programs.
3. Capture of the CICS Terminal Input Output Area (TIOA).
4. Temporary Storage Queue Management.

7.3.2.7 CICS Application Transaction Routers Setup

1. Setup of new CICS transactions within an existing CICS application: Before new CICS transactions can be used in a CICS router, they must be established in the CICS regions they will execute in both test and production regions. Usually, the team lead or system expert will handle getting the transactions established. Additionally, the system expert will need to be notified to ensure the necessary modifications are made to the CICS router for all new CICS transactions.
2. Setup of a New CICS Application: When a new CICS Application is being built, a new CICS router is required. The new CICS router should not be built from scratch, but instead an existing CICS router should be cloned (copied and modified) from an existing CICS router. During the design phase, the technical design document will need to be approved the Enterprise Architect Office.

7.3.2.8 CICS Application Transaction Routers Detail

1. The CICS router uses the COMMAREA to standardize the data keys and fields passed between CICS programs. This ensures all CICS programs within the CICS application have access to all data keys and fields needed for accessing databases.
2. Upon execution, invocation mode is checked. Since the CICS entry is setup to point to the CICS router, the CICS router will always execute first before control is passed to the new program and it executes. This is very important to understand especially when debugging a program using SmartTest. See Figure 7-20 below:

CICS Application Router Flow — The capture CICS Terminal Input Output Area (TIOA) is handled in the CICS router for CICS programs to provide a common input access point for the

CICS application. The data is then loaded into the COMMAREA and passed to the associated CICS program.

3. Temp Storage Queue Management — Since the CICS router is invoked first upon screen invocation, the CICS router also serves as a common area to delete temporary storage queues for each application transaction when certain keys are pressed such as: clear key, PF3, PF12.
4. Application Security Interface — There are 2 key interface points within CICS routers to ensure application security is executed. They are in the transaction-invoked paragraph and the program-invoked paragraphs.
5. Common APSSTUBs — Most CICS routers use a CICS router APSSTUB. It contains key paragraphs used for CICS router processing. Examples include TMCSS001, AMMSS001 and RULES001.
6. CICS Router Examples:

TMCSP000

AMMSP000

RULEP000

7. Router Navigation and Standards.

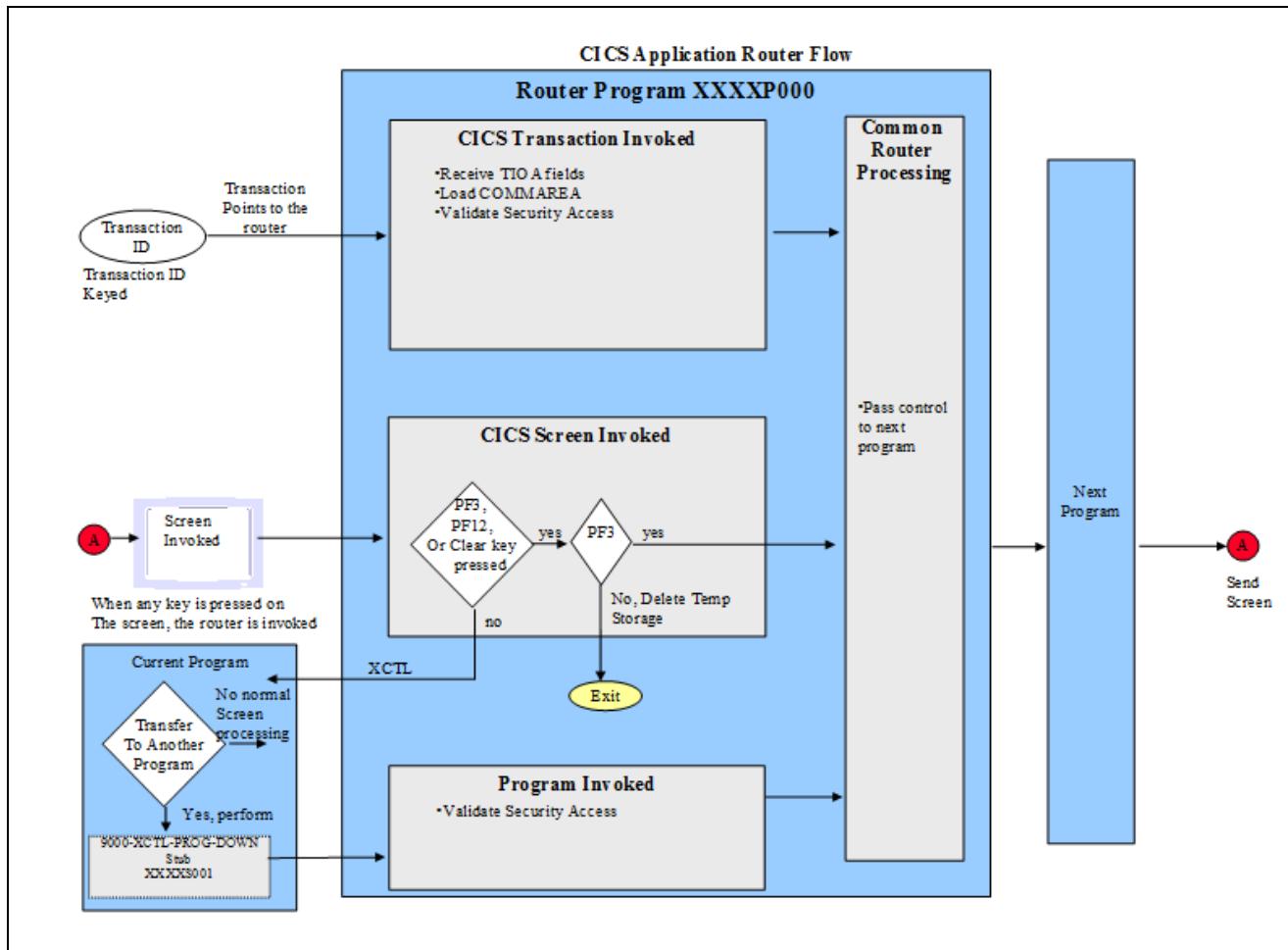


Figure 7-15 CICS Application Router Flow

7.3.2.9 Existing ISSM Standards

- Refer to *Technical Standards — Applications > Application Coding > Online System Design Considerations* for additional information.

7.3.3 SuperRouter Processing

7.3.3.1 Concepts Involved

- Modularity
- Loose Coupling
- Reuse

7.3.3.2 Operating System Platform

Host

7.3.3.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.3.3.4 Required Skills/Knowledge

- CICS
- APS

7.3.3.5 Related Concepts

- CICS Application Transaction Router (CICS Router) processing
- Temporary Storage queues (future)

7.3.3.6 Overview

BlueCross BlueShield of South Carolina utilizes a cross-region router (INTRP000) known as the “SuperRouter” to handle navigation between CICS application transaction routers moving from one Application Owning Region (AOR) to another in a seamless manner. It maintains and manages application data during the transfer from one application to another and passes application keys and data fields within the CICS start data area to ensure a seamless interface exist between applications. Due to the cross-region processing required to validate the SuperRouter, all validation must be done in the CICS Terminal Owning Regions (TORs) through all test environments. QA validation is required for

all SuperRouter changes. This is done to ensure the SuperRouter is functioning correctly and that control and data is being transferred from one AOR to another while passing through the CICS TOR.

The SuperRouter handles the following:

- Standardization of application database keys and data fields (currently a maximum length of 500 characters) passed between CICS Routers via the CICS Start command.
- Allows for a seamless interface between disparate applications with minimal coding.
- Handles up to 4000 characters of COMMAREA data for each specific application. Any COMMAREA data greater than 4000 characters will be truncated thus requiring all application database keys and fields passed to be in the first 4000 characters of the COMMAREA.
- Retains the COMMAREA data from the calling application by writing the calling applications COMMAREA data to a temporary storage queue before passing control to the next application so that the calling application can be returned to its state before the transfer occurred.
- Uses the TCTUA to store a system navigation indicator to determine whether or not the control is being passed to or returning from the application.
- Allows for application database keys and fields to be updated by the “transferred to program” and returns the updates to the application that initiated the transfer.
- **SuperRouter details:**
 - Setup of new transactions: Before new transactions can be used in the SuperRouter, they must be established as SuperRouter transactions in the CICS regions that they will execute in (both test and production regions). The System Expert of Leveraged Systems (LS) - Host handles all setup of SuperRouter transactions with the assistance of Tech Support. All SuperRouter transactions used to cross between AORs must start with the letter “I.” This is done to help Tech Support manage and monitor the SuperRouter transaction activity.
 - When to Use: The SuperRouter is needed anytime one CICS program needs to navigate via a CICS XCTL or a CICS Start Command to another CICS program that is not part of the current application. An example of this would be when navigation is required to one of the claims systems (AMMS or CMMS) from the managed care system (TMCS). Since the claims systems use a different CICS Router in a different CICS AOR, the SuperRouter would be needed to pass control and data back and forth between the two applications.
 - TCTUA Definition and usage: The Terminal Control Table Terminal Entry User Area (TCTTEUA) is an optional user defined work area that is provided as an extension of the Terminal Control Table (TCT) in static storage to pass information between terminal inputs. The TCTTEUA is associated with a particular TCT Terminal Entry (TCTTE). The BlueCross TCTTEUA is 255 bytes long of which 30 bytes is reserved for SuperRouter Navigation. Tech Support owns and approves all updates to the TCTUA.
- SuperRouter Requirements:
 - Name of the CICS Router to pass control to.
 - Name of the transaction to start within the application.
 - Size and copybook name of the COMMAREA for the CICS Router.
 - Name of the program that needs to be executed within the application.
 - What application owning region the application will execute in (test and production).
 - What data fields need to be passed to the application.
 - What data fields need to be passed back to the initiating program.
 - What CICS Routers the new application will need to transfer to and from.
- SuperRouter things to watch:

- When the SuperRouter is being migrated into production, it is very important to ensure that the APSDATA or Copylib COMMAREA for all applications are the same in production as they are in test. If these are out of sync, data can be passed incorrectly.
- When debugging using SmartTest, realize the SuperRouter executes twice for each pass.
 - First, it executes in the initiating AOR, to start and pass control and keys to the TOR via the CICS Start Data area and CICS Start command.
 - Second, it executes in the receiving AOR to do a CICS Retrieve command to gather the keys from the CICS Start Data area before passing keys and control to the receiving CICS Router.
- If navigation between applications no longer works, contact the Leveraged Systems (LS) Systems Support.
 - Since the SuperRouter uses the COMMAREA of other CICS applications, the LCAS Host Systems Expert must be notified if there are any updates made to a COMMAREA that uses SuperRouter processing. Otherwise, data can be passed incorrectly.
- Temp Storage usage.
 - Temporary storage is used to save the initiating program's Commarea before control is passed to the next application.
- Code Example:
 - **INTRP000**
- The SuperRouter is maintained by Leveraged Systems -Host, system expert. All changes should be coordinated through this area.
- SuperRouter Flow (see Figure 7-21).
- [Current Applications utilizing SuperRouter Functionality](#)
Since there are already several applications utilizing the SuperRouter functionality, the programming staff should always review the attached spreadsheet to determine what existing functionality can be reused.
- Router Navigation and Standards.

7.3.3.7 Existing ISSM Standards

- Refer to *Technical Standards — Applications > Application Coding > Online System Design Considerations* for additional information.

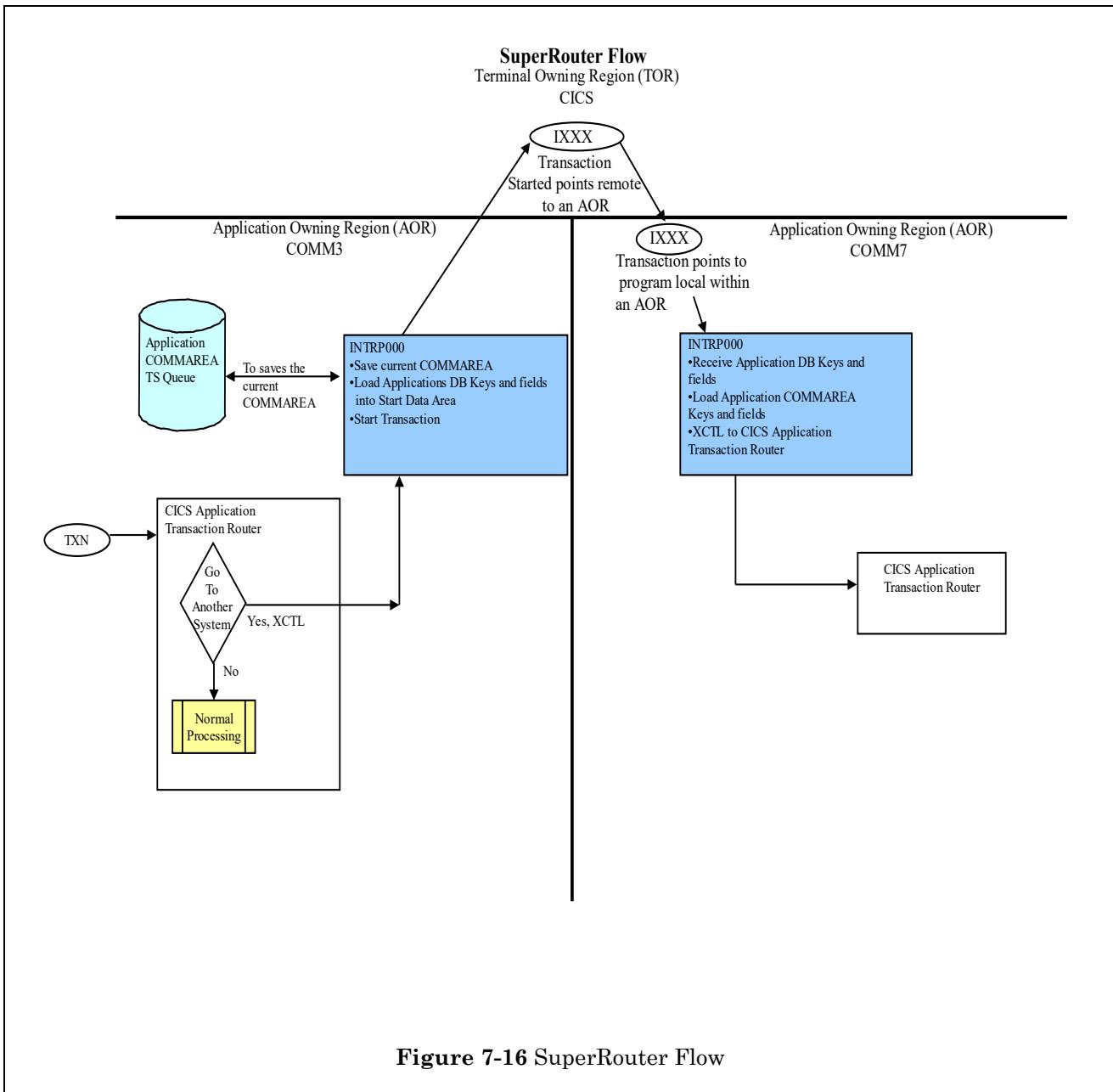


Figure 7-16 SuperRouter Flow

7.3.4 Minimizing the Use of Escape in Aps

7.3.4.1 Concepts Involved

- Modularity
- Reuse
- Flexibility

7.3.4.2 Operating System Platform

Host

7.3.4.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.3.4.4 Required Skills/Knowledge

APS programming structures

7.3.4.5 Overview

BlueCross BlueShield of South Carolina needs to minimize the use of **ESCAPE** within APS programming. When used “improperly,” **ESCAPE** causes difficulty in understanding a program’s flow, creates confusion in debugging program defects, and promotes unnecessary paragraph branching (when used at the beginning of a paragraph). **ESCAPE** is an APS S-Cobol structure that generates a COBOL GO TO statement which directs execution to the exit of the current paragraph. The improper use of the APS **ESCAPE** structure arises when **ESCAPE** is:

- Used in conjunction with an APS looping process.
- Used at the beginning of a paragraph to detect that the logic in the paragraph should not be executed.
- When Escape is used excessively within a paragraph.
- Used as the last instruction within the paragraph.

When **ESCAPE** is used “properly,” it can simplify code and allow the programmer to exit logic that does not need to be executed. Examples are below:

- Use **ESCAPE** to support leveragability within a program.
- Use **ESCAPE** “within the body” of a paragraph to exit the paragraph.
- Use **ESCAPE** to exit “complex” logic that would require restructuring of workable code.

7.3.4.6 Detailed Discussion

When using **ESCAPE**, APS generates a “GO TO para-EXIT” statement. (Para being the paragraph name in which the **ESCAPE** resides). This instruction directs execution to the exit of the paragraph. Once this instruction is executed, processing resumes with the statement following the statement that performed the paragraph; except if the **ESCAPE** occurs in the main logic paragraph in which an Exit Program is executed. (See Appendix A — APS Generation of **ESCAPE**.)

The following documents the “improper” use of the APS **ESCAPE** structure (See Appendix B — Improper Usage):

- Used in conjunction with an APS looping process:

EXAMPLE: Within the APS **REPEAT** structure

- Used at the beginning of a paragraph to detect that the logic in the paragraph should not be executed:

EXAMPLE: Within an **IF** statement at the beginning of a paragraph to bypass processing logic



NOTE An exception to this usage occurs when the paragraph is called from 3 or more locations within the program to support leveragability. The use of **ESCAPE** within the **IF** statement at the beginning of the paragraph is then permissible

- Overly used:
 - EXAMPLE: Within multiple independent **IF** statements
 - Used as the last instruction within the paragraph:

As a general rule, the use of the **ESCAPE** structure should be minimized. For a list of permissible **ESCAPE** usages, see Appendix C — Best Practices.

7.3.4.7 Appendix A — APS Generation of **ESCAPE**

Below is an example of an APS Source listing using the **ESCAPE** structure.

```
IF  BYPASS-ENTRY
  ELSE
    IF  WS-HOLD-COMP-CD NOT = WS-INQ-COMP-CD OR
      ... WS-HOLD-DIV-CD  NOT = WS-INQ-DIV-CD
      GENERATE DIV-TOTAL
    ELSE-IF  WS-HOLD-DEPT-CD NOT = WS-INQ-DEPT-CD
      GENERATE DEPT-TOTAL
    ELSE-IF  WS-HOLD-EMP-ID NOT = WS-INQ-EMP-ID-NO
      PERFORM 2100-RESET-HOLD-FIELDS

      PERFORM 2200-CALCULATE-TIME

      PERFORM 8400-READ-INQ-RSP-FILE
      .
```

```
*****
* CALCULATE TIME AND ADD TO CORRESPONDING BUCKET.
*****
2200-CALCULATE-TIME

INITIALIZE WS-ELAPSED-DAYS
...
WS-ELAPSED-TIME

PERFORM 2300-CALCULATE-DAYS

IF WS-ELAPSED-DAYS < 0
    DISPLAY 'INQ BYPASSED - RSP DATE < THAN LOG DATE'
    ESCAPE

IF WS-ELAPSED-DAYS = 0
    IF WS-RSP-TIME-HHMI < WS-INQ-LOG-TIME
        DISPLAY 'INQ BYPASSED - RSP TIME < THAN LOG TIME'
        ...
        WS-INQ-LOG-TIMESTAMP
    ESCAPE

ADD +1 TO WS-DIV-TOTAL
ADD +1 TO WS-DEPT-TOTAL
ADD +1 TO WS-EMPL-TOTAL
ADD +1 TO WS-FINAL-TOTAL
```

When compiled, APS generates the following code and the **ESCAPE** structure is converted into a GO TO statement.

```
IF BYPASS-ENTRY
    CONTINUE
ELSE
    IF WS-HOLD-COMP-CD NOT = WS-INQ-COMP-CD OR
        WS-HOLD-DIV-CD NOT = WS-INQ-DIV-CD
        PERFORM DE-DIV-TOTAL
    ELSE IF WS-HOLD-DEPT-CD NOT = WS-INQ-DEPT-CD
        PERFORM DE-DEPT-TOTAL
    ELSE IF WS-HOLD-EMP-ID NOT = WS-INQ-EMP-ID-NO
        PERFORM 2100-RESET-HOLD-FIELDS
```

```
END-IF  
END-IF  
END-IF  
PERFORM 2200-CALCULATE-TIME  
THRU 2200-CALCULATE-TIME-EXIT  
END-IF  
PERFORM 8400-READ-INQ-RSP-FILE ← (STEP 3: Processing resumes)  
.  
*****  
* CALCULATE TIME AND ADD TO CORRESPONDING BUCKET.  
*****  
2200-CALCULATE-TIME.  
    INITIALIZE WS-ELAPSED-DAYS  
        WS-ELAPSED-TIME  
  
    PERFORM 2300-CALCULATE-DAYS  
    IF WS-ELAPSED-DAYS < 0  
        DISPLAY 'INQ BYPASSED - RSP DATE < THAN LOG DATE'  
        GO TO 2200-CALCULATE-TIME-EXIT ← (STEP 1: generated "GO TO para-EXIT")  
    END-IF  
    IF WS-ELAPSED-DAYS = 0  
        IF WS-RSP-TIME-HHMI < WS-INQ-LOG-TIME  
            DISPLAY 'INQ BYPASSED - RSP TIME < THAN LOG TIME'  
            WS-INQ-LOG-TIMESTAMP  
            GO TO 2200-CALCULATE-TIME-EXIT ← (STEP 1: generated "GO TO para-EXIT")  
        END-IF  
    END-IF  
    ADD +1 TO WS-DIV-TOTAL  
    ADD +1 TO WS-DEPT-TOTAL  
    ADD +1 TO WS-EMPL-TOTAL  
    ADD +1 TO WS-FINAL-TOTAL  
2200-CALCULATE-TIME-EXIT ← (STEP 2: generated para-EXIT)  
EXIT.
```

7.3.4.8 Appendix B — Improper Usage

Used in Conjunction with an APS Looping Structure

This example illustrates the use of **ESCAPE** in conjunction with the APS **REPEAT** structure. By adding more and more escapes within the repeat loop, multiple exit points are created which goes against structured programming.

```
0200-SEARCH-RESPONVP
REPEAT VARYING RVP-X FROM 1 THRU 2000
    IF WS-RVP-DIV (RVP-X) (1:2) > TNQ-DIV-CODE (1:2)
        ESCAPE
        IF WS-RVP-DIV (RVP-X) (1:2) = TNQ-DIV-CODE (1:2)
            TRUE RESPONVP-DATA-FOUND
            ESCAPE
WS-PREV-DIV = WS-CURR-DIV
```

When compiled, APS generates the following code and the programmer can now easily see that the **ESCAPE** structure is directing execution to the exit of the current paragraph.

```
0200-SEARCH-RESPONVP.
    MOVE 1 TO RVP-X
    PERFORM WITH TEST BEFORE
    UNTIL RVP-X > 2000
        IF WS-RVP-DIV (RVP-X) (1:2) > TNQ-DIV-CODE (1:2)
            GO TO 0200-SEARCH-RESPONVP-EXIT
        END-IF
        IF WS-RVP-DIV (RVP-X) (1:2) = TNQ-DIV-CODE (1:2)
            MOVE TRUX TO RESPONVP-DATA-FOUND-FLG
            GO TO 0200-SEARCH-RESPONVP-EXIT
        END-IF
        ADD 1 TO RVP-X
    END-PERFORM.
    MOVE WS-CURR-DIV TO WS-PREV-DIV
0200-SEARCH-RESPONVP-EXIT.
    EXIT.
```

Recommended Alternative: A less confusing method of processing would be to incorporate a flag that would indicate and document when processing would exit giving one exit point from the loop processing.

Used at the Beginning of a Paragraph

In the following example the **ESCAPE** structure is used within the **IF** statement at the beginning of the paragraph causing unnecessary paragraph branching. The only purpose of the **PERFORM** is to exit the paragraph.



NOTE This paragraph is called from only one location within the program.

```
0100-PROCESS
    IF WHHB-WIP-REC-TYPE = '05'
        PERFORM 0200-PROCESS-HEADER
    ELSE
        IF WHHB-WIP-REC-TYPE = '50'
            WIP-LINE-ITEM-BASE = WIP-HEADER-HISTORY-BASE
        PERFORM 0300-PROCESS-LINES
        .
        .
        .
0300-PROCESS-LINES
DISPLAY '0300-PROCESS-LINES' WLIB-CLAIM-NO
... 'LI-' WLIB-LI-NO

IF NOT PROCESS-CLAIM
    ESCAPE
IF WS-PROCESS-CLAIM NOT EQUAL WLIB-CLAIM-CNTL-NO
    DISPLAY '** INPUT CLAIM FILE OUT OF SEQUENCE'
    ... WS-PROCESS-CLAIM, ' ' WLIB-CLAIM-CNTL-NO
```

Recommended Alternative: In this case, placing the condition within the calling program should be considered as an alternative method. This approach would promote branching to paragraphs only on an as needed basis as well as minimize the usage of the **ESCAPE** structure.

```
0100-PROCESS
    IF WHHB-WIP-REC-TYPE = '05'
        PERFORM 0200-PROCESS-HEADER
    ELSE
        IF WHHB-WIP-REC-TYPE = '50'
            WIP-LINE-ITEM-BASE = WIP-HEADER-HISTORY-BASE
            IF NOT PROCESS-CLAIM
            ELSE
                PERFORM 0300-PROCESS-LINES
```

0300-PROCESS-LINES

```
DISPLAY '0300-PROCESS-LINES ' WLIB-CLAIM-NO  
..      'LI-' WLIB-LI-NO  
  
IF WS-PROCESS-CLAIM NOT EQUAL WLIB-CLAIM-CNTL-NO  
    DISPLAY '** INPUT CLAIM FILE OUT OF SEQUENCE'  
    WS-PROCESS-CLAIM , ' ' WLIB-CLAIM-CNTL-NO
```

APS ESCAPE Overly Used in a Paragraph

In the following example, the **ESCAPE** structure is overly used.

0220-LOAD-OVERRIDE-D0E17

```
IF WS-DEFER-OVERRIDE-1 = SPACES  
    WS-DEFER-OVERRIDE-1 = '0E17D'  
    ESCAPE  
IF WS-DEFER-OVERRIDE-2 = SPACES  
    WS-DEFER-OVERRIDE-2 = '0E17D'  
    ESCAPE  
IF WS-DEFER-ADDTL-OVR-1 = SPACES  
    WS-DEFER-ADDTL-OVR-1 = '0E17D'  
    ESCAPE  
IF WS-DEFER-ADDTL-OVR-2 = SPACES  
    WS-DEFER-ADDTL-OVR-2 = '0E17D'  
    ESCAPE  
DISPLAY 'CLAIM ' WPCI-CLAIM-NO ' WAS NOT OVERRIDEN.'  
ESCAPE
```

0230-LOAD-WPCI-RECORD

Recommended Alternative: The **EVALUATE** or **ELSE-IF** structures should be considered as an alternative method. By using the **EVALUATE** structure, the code is easier to read and maintain.

0220-LOAD-OVERRIDE-D0E17

```
EVALUATE TRUE  
    WHEN WS-DEFER-OVERRIDE-1 = SPACES  
        WS-DEFER-OVERRIDE-1 = '0E17D'  
    WHEN WS-DEFER-OVERRIDE-2 = SPACES  
        WS-DEFER-OVERRIDE-2 = '0E17D'
```

```
WHEN WS-DEFER-ADDTL-OVR-1 = SPACES
    WS-DEFER-ADDTL-OVR-1 = '0E17D'
WHEN WS-DEFER-ADDTL-OVR-2 = SPACES
    WS-DEFER-ADDTL-OVR-2 = '0E17D'

WHEN OTHER
    DISPLAY 'CLAIM ' WPCI-CLAIM-NO ' WAS NOT OVERRIDEN.'
```

0230-LOAD-WPCI-RECORD

Used as the Last Instruction within the Paragraph

In the following example the **ESCAPE** structure is not required since the **IF** statement is the **last** instruction in the paragraph. The **ESCAPE** is only being used to document the program flow when the 'IF conditions' are not met.

```
CHECK-DATA
    PERFORM PARAGRAPH-1
        ADD 1 TO READ-COUNTER
    PERFORM PARAGRAPH-2
    .
    .
    .

PARAGRAPH-1
    ADD 1 TO PROCESS-COUNTER
    IF WS-DEFER-ADDTL-OVR-1 = SPACES
        WS-DEFER-ADDTL-OVR-1 = '0E17D'
    ELSE-IF WS-DEFER-ADDTL-OVR-2 = SPACES
        WS-DEFER-ADDTL-OVR-2 = '0E17D'
    ELSE
        ESCAPE

PARAGRAPH-2
    ADD 1 TO PARAGRAPH-2-COUNTER
```

7.3.4.9 Appendix C — Best Practices

- Use **ESCAPE** to support leveragability within a program.

- Use **ESCAPE** “within the body” of a paragraph to exit the paragraph.
- Use **ESCAPE** to exit “complex” logic that would require restructuring of workable code.

7.3.5 Common I/O and Edit Modules

7.3.5.1 Concepts Involved

- Modularity
- Loose Coupling
- Reuse
- Flexibility
- Encapsulation

7.3.5.2 Operating System Platform

Host/Non Host

7.3.5.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects
- Solution System Designer

7.3.5.4 Required Skills/Knowledge

Basic programming skills

7.3.5.5 Related Techniques

- CICS Application Security
- Modules by Function
- Application Cloning
- Regional Processing Number (RPN) driven logic

7.3.5.6 Overview

BlueCross BlueShield of South Carolina advocates the use of the structured design approach to programming. The structured design approach or modular programming allows you to take larger more complicated tasks and break them down into smaller sub problems. The task of solving these sub problems is achieved through the use of modules. A module is a portion of a program that carries out a specific function and may be used alone or combined with other modules. Two such modules used are common I/O (update and access) and Edit modules. Common I/O and edit modules may also be placed in a subprogram or an APSSTUB. This allows modules to be used by multiple programs that may share

common files or edits. When writing a module, it should be coded as loosely coupled as possible. This will make the module more flexible and reusable.

Advantages to using a common I/O or edit module are as follows:

- Programs follow the concepts of structured programming.
- Modules are reusable and can be leveraged (code it once use it many times).
- Centralized Data Access.
- Reduces maintenance.

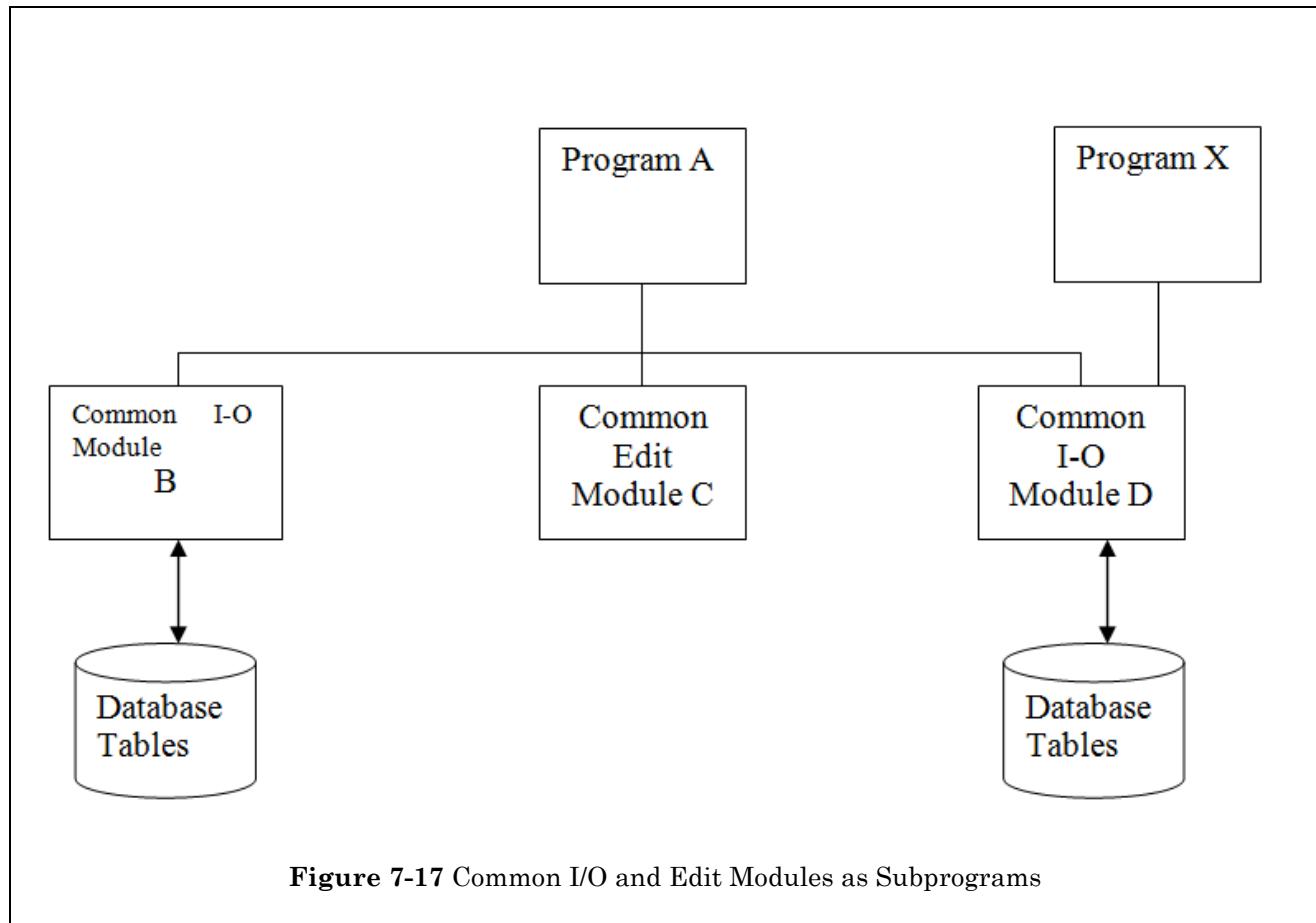


Figure 7-17 Common I/O and Edit Modules as Subprograms

Common I/O and Edit modules can be subprograms that perform a specific function within a system (Figure 7-22). They can be written and compiled separately from the main (calling) program.

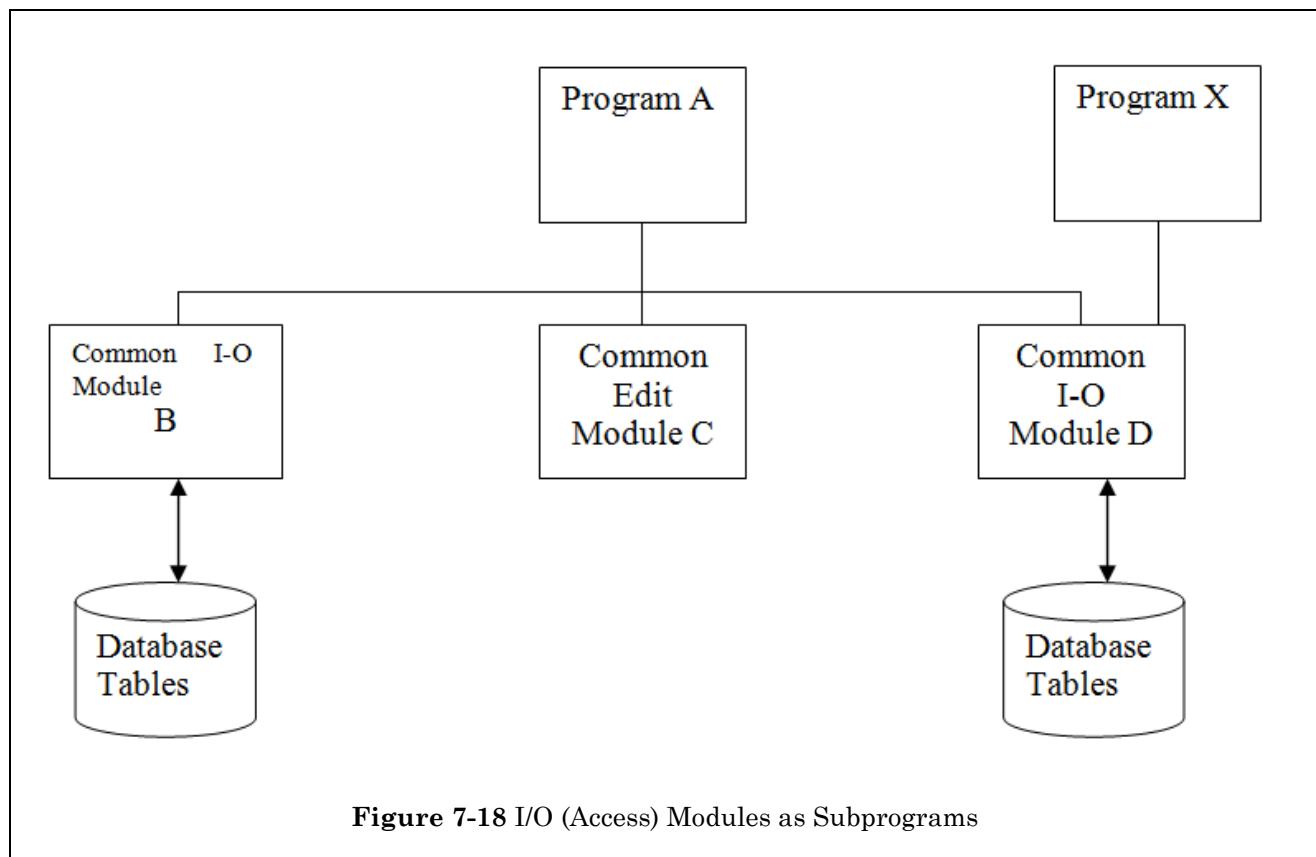
Since the programs are compiled separately, if a change occurs to the process in the called module without changes to the data passed, only the called module would have to be recompiled instead of compiling every module that would otherwise perform the process. Subprograms bring all the advantages of modularity and reuse that a functional paragraph brings to a program. In the example above, program A is the main program and it calls three subprograms (B, C and D). Programs B, C, and D may likewise be called and reused by other programs (program X) that require the same function. This practice can reduce redundant coding, minimize maintenance and reduce database dependencies. Placing this code into a separate module also allows for both batch and online programs to reuse the same code.

(See section titled “Compiling subprogram for both BATCH & ONLINE” in this Chapter.

For performance reasons, common I/O modules are NOT recommended to obtain data for CICS workload programs (CICS programs that display multiple database records on a CICS screen). It is more efficient to place the database calls within the CICS workload programs to reduce the paging it takes to call the common I/O program multiple times.

7.3.5.7 I/O (Access) Modules as Subprograms

When possible common I/O modules (as subprograms) (Figure 7-23) should be used to access and update a database. This practice will give different programs a common way to communicate with the database. All programs accessing the data will use the common I/O module to obtain data. If changes to the database occurred, the calling program will remain unchanged unless the data exchanged between the 2 programs change. In the example below only the common I/O module would need to be modified and generated.



7.3.5.8 Existing ISSM Standards

Refer to *Technical Standards — Applications > Application Coding > IMS Programming Considerations* for additional information.



NOTE Common I/O and edit modules are used to access (read) and update (add, void, update, and delete) data. Manipulation of data should be minimized to ensure the integrity of the encapsulation process.

7.3.5.9 Edit Modules as Subprograms

Edit modules should be subprograms (Figure 7-24) when multiple programs use the same edits. In the example below, programs A and B have an identical edit check; therefore, the edit module was created to perform the edit check and is called by both programs. If a change occurs in the way the data is edited, only the edit module will need to be updated unless the data exchanged between the 2 programs change. One example of an edit module used at BlueCross BlueShield of South Carolina is CORPL207. The purpose of this module is to emulate the functionality of the TRANSCENTURY module (TRCENGIN) through a program that calls the TRANSCENTURY process. The TRCENGIN is also an edit module. It handles other types of date manipulation such as adding and subtracting days from a year. Without it each program that needs some type of date conversion would have to develop its own process for calculating dates.

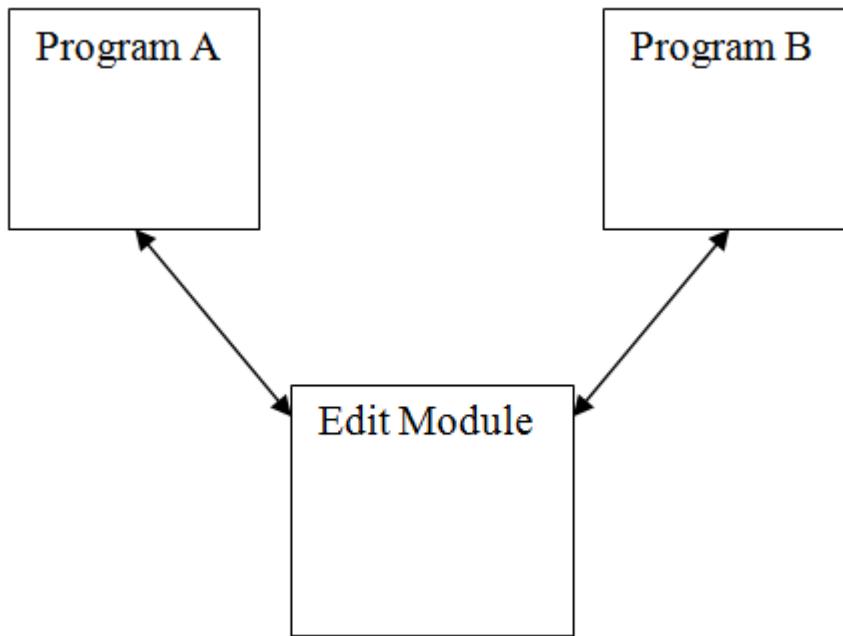


Figure 7-19 Edit Modules as Subprograms

7.3.5.10 Compiling Subprogram for Both Batch & Online

When a common I/O or edit module is to be used as both batch and online it must be compiled with a type of XAPS, XAPSDB2 or XAPSIMS. This will create both an EXECLIB or LOADLIB and a CICSLIB. The processor group will determine if an EXECLIB or LOADLIB will be created.

Since some of the code in a common I/O or edit module may be specific to batch programs or online programs, you will need to identify the code with the following if statement:

```
% IF &AP-GEN-DC-TARGET = 'CICS'  
    CALL WS-PROGRAM-ETKSDC00 USING DFHEIBLK, DFHCOMMAREA,  
    .. ETKSBC00-COMMUNICATION-AREA  
% ELSE  
    CALL WS-PROGRAM-ETKSDC00 USING  
    .. ETKSBC00-COMMUNICATION-AREA
```



NOTE In the example above the CICS version needs the DFHEIBLK, and DFHCOMMAREA but the batch version does not therefore the batch version would fail on compile if not for the 'IF statement'. All other code in the program is shared by both batch and online.

7.3.5.11 Common I/O and Edit Paragraphs/Routines as an APSSTUB



NOTE The use of an APSSTUB is not recommended.

I/O and edit routines could also be placed in an APSSTUB. This allows the code to be used by multiple programs similar to the way subprograms can be reused. However, when a program uses an APSSTUB the code is pulled into the program at compile time. If there is a change to the APSSTUB, all programs using it must be recompiled to reflect the change. If the APSSTUB is used in multiple programs, this could result in an extraordinary amount of overhead recompiling the programs; therefore, the use of an APSSTUB should be limited.

7.3.6 TIOA Utilization

7.3.6.1 Concepts Involved

- Reuse
- Flexibility

7.3.6.2 Operating System Platform

Host

7.3.6.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers

Concepts & Techniques > Host Platform Techniques

Printed forms of this document are uncontrolled and may be obsolete. It is the responsibility of the user to ensure that any printed copy is the same revision as the online version.

Copyright ©2023, BlueCross BlueShield of South Carolina. All rights reserved. BlueCross BlueShield of South Carolina is an independent licensee of the Blue Cross and Blue Shield Association.

- System Experts
- Architects

7.3.6.4 Required Skills/Knowledge

- APS programming
- CICS knowledge

7.3.6.5 Related Techniques

- CICS Application Transaction Router Processing
- Common I/O and Edit Modules (APSSTUB)

7.3.6.6 Overview

To leverage legacy systems and provide application flexibility, BlueCross BlueShield of South Carolina (BlueCross) utilizes the Terminal Input Output Area (TIOA). The TIOA is a storage area that is associated with a 3270 terminal within CICS. It is set up by storage control and is chained to the Terminal Control Table Terminal Entry (TCTTE) as needed for terminal input/output operations.

Many Host CICS transactions at BlueCross use the TIOA to receive data keys and fields to be entered on a 3270 screen before the transaction can perform the requested function. This is typically done to create a “fast path” processing for the user allowing them to bypass a CICS Menu or screen processing.



NOTE When invoking a Host CICS transaction, Presentation Systems applications use the TIOA area to pass required transaction data to enhance performance by reducing the number of CICS screens the process has to traverse.

7.3.6.7 Detailed Discussion

The technical requirements for using the TIOA at BlueCross include:

Application Data Structure and APSSTUB

An APSSDATA structure is typically used to define the TIOA area and the TIOA parameter delimiters. An APSSTUB is typically used to parse the data.

The APSSDATA structure **RM70I004** is an example of a TIOA data structure and is used in conjunction with APSSTUB **RM70S001**. RM70S004 receives the TIOA area and parses it WS-TIOA-PARM1 thru WS-TIOA-PARM4.

Application Code for Parsing and Valuing the TIOA Area

Application code is required to parse the parameters passed in the TIOA area and to move the parameters to a structure (COMMAREA) that can be passed to the requested transaction. When there

are multiple parameters passed in the TIOA area, a delimiter must be used to determine where one parameter has ended and another has begun. Generally, a comma “,” or a slash “/” are used as a TIOA parameter delimiter. However, applications using a TIOA must consider what data can be passed in the parameters. If commas and/or slashes can be in the parameters (data sent), the delimiter cannot include a comma or a slash and another value must be used for the delimiter. However, spaces are never to be used as the delimiter.

CICS Application Transaction Router

When a CICS Application Transaction Router (CICS Router) is used, the TIOA processing is handled within the CICS Router. The CICS router must include the APSSDATA structure that defines the TIOA area and an APSSTUB for parsing the TIOA.

Example from application router RM70P000:

```
WHEN 'RMCL'
      IF EIBCALEN > 0
      ELSE
          PERFORM 1525-RECEIVE-TIOA(RMCA-CLAIM-NO,
                                         ...
                                         RMCA-PLAN,
                                         ...
                                         WS-DUMMY,
                                         ...
                                         WS-DUMMY)
          WS-NEXT-PROGRAM = 'RM74P005'
```

In the example above, when the transaction RMCL is invoked (and there is no COMMAREA) the paragraph 1525-RECEIVE-TIOA is performed to receive the TIOA area.

The TIOA parameters passed to the RMCL transaction (RMCA-CLAIM-NO, RMCA-PLAN) have been predetermined based on the requirements of the RMCL transaction. The RM70P000 router has been coded to receive up to 4 TIOA parameters (WS-TIOA-PARM1, WS-TIOA-PARM2, WS-TIOA-PARM3, WS-TIOA-PARM4).

```
PARA    1525-RECEIVE-TIOA(-WS-TIOA-PARM1,
                           ...
                           -WS-TIOA-PARM2,
                           ...
                           WS-TIOA-PARM3,
                           ...
                           -WS-TIOA-PARM4)
```

The RMCL transaction only needs 2 (RMCA-CLAIM-NO, RMCA-PLAN) of the 4 TIOA parameters that are available. A working storage field, WS-DUMMY, is used as a placeholder where a parameter has been defined, but is not used for a requested transaction.

Where a transaction is invoked and there are no TIOA parameters passed, the router will not perform the paragraph 1525-RECEIVE-TIOA paragraph to receive the TIOA area.

Example:

```
WHEN 'RMBT'
      WS-NEXT-PROGRAM = 'RM70P990'
```

Length of the TIOA

The length of the TIOA (TIOAL) should be set to a value that is slightly larger than the average input message length for the terminal. The maximum value that may be specified for the TIOAL is 32767 bytes. Real storage can be wasted if the TIOAL value is too large for most terminal inputs in the network. If the TIOAL is smaller than most initial terminal inputs, excessive GETMAIN requests can occur resulting in additional processor requirements (unless the TIOAL is zero).

7.3.6.8 Areas impacted by the use of the TIOA

Presentation Application Systems (PAS) must be notified when there are new or changed TIOA parameters for CICS screens as screen scraping may need to be affected.

7.3.7 CICS Temporary Storage Queue (TSQ) Usage

7.3.7.1 Concepts Involved

- Data Management
- Loose Coupling
- Modularity

7.3.7.2 Operating System Platform

Host

7.3.7.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.3.7.4 Required Skills/Knowledge

- CICS
- APS

7.3.7.5 Related Techniques

- Modules by Function
- CICS Application Transaction Router Processing
- SuperRouter Processing

7.3.7.6 Overview

“Temporary Storage is the primary CICS facility for storing data that must be available to multiple transactions. Data items in temporary storage are kept in queues whose names are assigned dynamically by the program storing the data. A temporary storage queue containing multiple items can be thought of as a small data set whose records can be addressed either sequentially or directly, by item number. If a queue contains only a single item, it can be thought of as a named scratch pad area.”
(Excerpt from *CICS Application Programming Guide*, IBM United Kingdom Laboratories Limited, Hampshire, England, 1994, Chapter 2.)

1. CICS application integration
2. TSQ management through CICS Application Router Processing

7.3.7.7 Detailed Discussion

CICS Application Integration

To provide seamless integration for CICS applications, BlueCross uses temporary storage queues to save the COMMAREA of the “calling” CICS application before control is passed to the “transferring to” CICS application. This processing is done so that when the end user presses the F3 Key or Return Tab, the “calling” CICS application is placed back into its original state thus providing a seamless CICS to CICS application interface. This processing is handled within the SuperRouter to isolate and centralize the management of the TSQ.

TSQ Management through CICS Application Router Processing

BlueCross uses the CICS Application Router to provide a common module for the management of TSQs ensuring that all TSQs within a CICS application are deleted when the CICS application is exited.

Other Considerations

When using temporary storage queues, BlueCross requires that the ‘LENGTH OF’ statement be used to dynamically determine the length of the queues to prevent any length errors when writing or reading queues.

7.3.7.8 Existing ISSM Standards

- Refer to *Technical Standards — Applications > Application Development Support Tools > Host Application Languages > AMB/COBOL Online Considerations* for additional information.

7.3.8 DB2 PLANBINDs and PACKBINDs

7.3.8.1 Concepts Involved

- Flexibility
- Data Management

7.3.8.2 Operating System Platform

Host

7.3.8.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.3.8.4 Required Skills/Knowledge

- APS or COBOL programming
- DB2

7.3.8.5 Related Techniques

N/A

7.3.8.6 Overview

The bind function in DB2 analyzes SQL and formulates data navigational instructions known as “access paths”. The very nature of a relational DBMS makes the bind process a necessary and critical function. This is because SQL is “free form”, meaning that you specify “what” data is needed but you cannot directly specify “how” to retrieve the data. In other words, SQL is coded without embedded navigational instructions. Bind terminology includes plans, packages, collection ids, version ids, timestamps, contokens and DBRMs. A bind is invoked via the DB2 bind statement after the DB2 precompile and the program compile steps are complete.

Package binds are always done for programs which access DB2 tables. This process establishes a relationship between an application program and its relational data. A bind must be successfully completed before a DB2 program can be executed. The package bind uses the DBRM from the DB2 precompile and the previously created PACKBIND source that has been stored in ENDEVOR. The parameters of the bind source member, such as qualifier, owner, ENDEVOR region and DB2 subsystem determine how and where a package is bound. The collection ID is a user defined name and allows a package to be bound to multiple physical tables with different qualifiers having the same physical column layout. The ‘SET CURRENT PACKAGESET’ SQL statement is used to allow application modules to dynamically swap between collection IDs that are defined within the package so that program code can be shared across multiple clients.

Planbinds are done at the “driver” level for the unit of work. It establishes all of the DB2 programs (packages) and collection ids that can be executed within the batch or online unit of work. Generally, the plan name would be the driver program name for batch and the CICS transaction name for online. The plan bind uses previously created PLANBIND source that has been stored in ENDEVOR.

At BlueCross BlueShield of South Carolina (BlueCross), it is the responsibility of the application programming staff to create PLANBIND and PACKBIND source members for DB2 modules. ENDEVOR is the change management system used to move, generate, and bind plans and packages through test regions, QA, and into production. ENDEVOR is also the mechanism that is used to ensure that the appropriate back binds, or binds to test regions that are defined in the PACKBIND source are completed when an element is moved to QA.

7.3.8.7 Detailed Discussion

PACKBINDs

The use of generic PACKBIND source is encouraged at BlueCross. Generic PACKBIND source is used by default by ENDEVOR and contains the bind statements for all collection IDs and qualifiers associated with a particular application. Generic PACKBIND source exists for most ENDEVOR sub-systems that contain DB2 modules and is recommended for any new sub-systems defined to ENDEVOR that will contain DB2 modules. The use of generic PACKBINDs enables application areas to standardize the use of collection IDs used in packages, thus allowing for the use of common modules or program stubs for setting the collection ID. The use of generic PACKBINDs also reduces maintenance time and effort while minimizing the occurrences of bind errors.

Although the use of generic PACKBINDs is preferred, there are times when a specific PACKBIND must be used to support a DB2 module. For example, if the table being accessed by a module does not exist in all DB2 sub-systems defined by the generic PACKBIND, or if a different application area is accessing a DB2 module using a different collection ID than is defined in the generic PACKBIND, a specific PACKBIND must be built for that DB2 module.

ENDEVOR uses the following order in determining which PACKBIND to use:

1. Program name (specific)
2. First 4 characters of program name (generic; currently available for TRICARE only)
3. ENDEVOR sub-system (generic)

Collection IDs

There are four different approved methods for setting the SQL SET CURRENT PACKAGESET statement using a collection ID.

Table Driven I/O Module

The table driven I/O module (CISID300) is used to determine the COLLECTION ID based on RPN, APPLID (from CICS region or batch CARDLIB input) and table set (application system that owns the data or subset of the application data) and a macro (\$BC-CISI-PACKAGE) to call CISID300 and perform the PACKAGESET SQL.

Use of the table driven I/O module eliminates the need to modify any programs for a new line of business. The PACKBIND source and the table (COLLECTION_ID_REF) need to be updated only if new table qualifiers are used. If a new RPN is added that will exist on the same physical tables as existing business, programs would already be bound for that qualifier and only the table updates to point to the existing collection IDs would be needed.

While this approach does use a macro, the macro itself simply establishes the correct working storage, calls CISID300 and performs the PACKAGESET SQL. The macro would not need to be modified to add new business, so the overhead of recompiling modules to pick up macro changes are eliminated.

It should be noted that CISID300 does not incur DB2 overhead with every call. For CICS, the table is loaded into the temp storage queue the first time CISID300 is called when the CICS region comes up and collection IDs are acquired from there. For batch, the entire table is loaded into working storage the first time it is called or linked and for any subsequent calls in the same step only the working storage is used.

I/O Module

Another method of an I/O module that is not table driven is to set the collection ID for online or batch by passing the data from the calling program; no DB2 table is used with this method. The DB2 module calls the I/O module, setting the collection ID based on the RPN, Plan Code or APPLID that is passed. The advantage to an I/O module is that only one module is changed for any new Line of Business. Usually a change is not required if the module is dynamically creating the collection ID based on RPN, Plan Code or APPLID.

APSSTUB or APS Macro

Another method for setting the collection ID is the use of APSSTUB or APS macros. The disadvantage to this is that if a change is made to the APSSTUB or APS macro, then all modules using the APSSTUB or APS macro will need to be recompiled. This is very time consuming and not the preferred technique.

CARDLIB

Another method is for a DB2 batch program to read in a CARDLIB that contains the collection ID to set the package. When a new Line of Business is added, the module is not changed, but a new CARDLIB for the collection ID is added.

The use of hardcoded logic directly to set the collection ID directly into a module is contrary to the BlueCross BlueShield of South Carolina (BlueCross) concepts of reusability, flexibility and loose coupling. Hard coding data directly into a module results in non-reusable code and requires the module be changed whenever there are additions or changes to the embedded data.



NOTE When using any one of these methods, the PACKAGESET should be reestablished upon return from any called module because the called module may have reset the collection ID to something different.

PLANBINDS

The approach for building PLANBIND source differs when dealing with batch modules versus CICS online modules. For batch processing, a PLANBIND must exist for each standalone module or driver module that accesses DB2. For CICS online processing, a PLANBIND must exist for each transaction that has DB2 access. PLANBINDS for CICS online modules that access DB2 are built using the name of

the transaction that will execute the bind. All PACKBINDS for DB2 modules that are called as part of the execution of an online transaction must be included in the package list for that plan.

When building PLANBIND source for a stand-alone batch DB2 module, the name of the PLANBIND will match the name of the DB2 module. The package list included in the PLANBIND will only include the name of the standalone module. For batch DB2 sub-modules, the name of the PLANBIND must be built using the name of the driver module. The package list will contain the name of any specific DB2 sub-modules that need to be executed.

The requirements for building PLANBIND source for XAPS DB2 modules are the same as those for building PLANBIND source for batch and CICS online modules. Care must be taken to ensure that both the online and batch components of the XAPS element have the appropriate PLANBIND source generated.

See Appendix A below for references to examples of PLANBIND and PACKBIND source.

7.3.8.8 Existing ISSM Standards

- Refer to *Technical Standards — Applications > Application Coding > AMB/COBOL > Programming Guidelines* for additional information.

7.3.8.9 Appendix A

The following examples can be found in the production ENDEVOR libraries.

PLANBIND Examples

Stand-alone batch DB2 module:	BC.NDVR.PROD1.PLANBIND(CISID169)
Module calling batch DB2 sub-modules:	BC.NDVR.PROD1.PLANBIND(MM08P002)
Online DB2 module:	BC.NDVR.PROD1.PLANBIND(RMGP)

PACKBIND Examples

Generic PACKBIND	BC.NDVR.PROD1.PACKBIND(PIRS)
Module-specific PACKBIND	BC.NDVR.PROD1.PACKBIND(RM78D100)

7.3.9 History Generating Applications and DB2 Databases

7.3.9.1 Concepts Involved

- Data Management
- Loose Coupling

7.3.9.2 Operating System Platform

Host

7.3.9.3 Intended Audience

- System Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.3.9.4 Required Skills/Knowledge

- APS
- DB2

7.3.9.5 Related Techniques

- Database Audit Trail
- Purge Processing

7.3.9.6 Overview

The History Generating Applications and DB2 Databases technique at BlueCross BlueShield of South Carolina describes how to define applications and DB2 databases that allow for the storage of the most current active data along with the information to be available for processing along with the history of changes. All operational DB2 tables at BlueCross should be history generating and have a purge process to archive data.

7.3.9.7 Detail Discussion

The History Generating Applications and DB2 Databases technique requires the database and the application to work together to provide the most current information and the history of changes. The appropriate fields must be available in the database for the application to perform this technique. The application controls the add/update process, which will define the inquiry process so the correct information will be returned when queried. All I/O performed against these DB2 tables should use highly cohesive, loosely coupled access modules.

There are two types of History Generating Applications and DB2 Databases (Table 7-6). The simple type is used when there can only be one active row for a primary key on the table. The complex type is used when there can be more than one active row for a primary key on the table. There is a variation of these types of applications and databases where the active most current data is kept on a table that is physically separate from the table that retains the complete history. (In the PIRS example of this technique, the “history” table can also contain active rows because it is a complex type database table). This approach would be used when there is a large amount of data and the desired database performance cannot be achieved with the active and history data on the same physical table.

Simple Type

In the simple type of History Generating Applications and DB2 Databases, the database must contain a date/time stamp in the primary key and a column to identify the row as active or void (usually `sys_void_sts_ind`). The add/update application must make sure there is only one active row per primary key. Each time a column on a row is updated, a voided history row is created to retain the data prior to the update. A row is identified as active when the `sys_void_sts_ind` is set to ‘N’, and is identified as void when it is set to ‘Y’. The audit information on the active row is also updated to reflect the latest changes.

In a simple type database, voided records may not be used as historical data. When an application calls a module to read a simple type database, it always requests the most current information. If historical information is needed for a specific point in time that may not be the most current, the complex type of database is used.

Complex Type

In the complex type of History Generating Applications and DB2 Databases, the database must contain a date/time stamp in the primary key, the begin and end dates in the row must be valid, and must contain a column to identify the row as active or void (usually `sys_void_sts_ind`). The date/time stamp column allows multiple active rows to exist for a primary key, which allows for multiple begin and end date combinations. To appropriately handle multiple active rows, the add/update application must edit for overlapping begin and end dates to make sure there are not duplicates. Each time a value on a row is updated, rather than setting an end date and building a new active row, a voided history row is created to retain the data history prior to the update. A row is identified as active when the `sys_void_sts_ind` is set to ‘N’, and is identified as void when it is set to ‘Y’. The audit information on the active row is also updated to reflect the latest changes.

When an application calls a module to read a complex type database, it either uses a date to return the row that covers that point in time, or requests the most current information. If the most current information is needed, a sub-select for the maximum begin date is used to return the correct row.

There are complex type History Generating Applications and DB2 Databases at BlueCross that use a sequence number to track history and identify the most current row. Because of the design improvements of the date/time stamp technique, the complex type is the approved method for new system development.

History Generating Applications and DB2 Databases Types

Type	Usage	#Active Rows	Use of Void	Identify Current Row	Identify Row by Date Range
Simple	Current data access, historical data for reference	1	Indicates row is not active or current	Void indicator or set to ‘N’	Not Available
Complex	Current and historical data for access	n	Indicates now is not active	Row has the maximum begin date for key and void indicator is set to ‘N’	Begin date is less than or equal to target date, end date is greater than target date (end date is ‘to’ date not ‘thru’ date)

Table 7-3 History Generating Applications and DB2 Databases Types

Example:

Sample of a simple type table (Figure 7-25) utilizing the technique with the date/time stamp:

RP	PLAN	NO	CD	KEY	SYS		VOID	STS
					TM			
					STMP	BEG	TRM	IND
		820	820	111111111	2005-04-28-11.10.51.442492	2005-01-01	9901-01-01	N
		820	820	111111111	2005-07-07-11.43.50.402058	2005-01-01	9901-01-01	Y
		820	820	111111111	2005-07-07-11.43.53.453290	2005-01-01	9901-01-01	Y
		820	820	111111111	2005-07-07-11.43.47.403806	2005-01-01	9901-01-01	Y

Figure 7-20 Simple Type Table Sample

Sample of a complex type table (Figure 7-26) utilizing the technique with the date/time stamp:

		SYS		SYS SYS		VOID ADD		SYS		SYS SYS		ADD ADD ADD		LST		SYS		SYS SYS		LST LST LST	
RP	PLAN	TM		REC	REC	STS	TM	RACF	TRM	PGM	TM	ID	ID	NME	SIMP	ID	ID	NME	ID	ID	NME
NO	CD	KEY	SIMP	BEG	TRM	IND	SIMP	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID
820	820	1111111111	2005-04-28-111051.442492	2005-01-01	9901-01-01	N	2005-07-07-114353.453290	D882408	LAP%	PIZSD122	9901-01-01-00.0000.000000										
820	820	1111111111	2005-07-07-114350.402058	2005-01-01	9901-01-01	Y	2005-07-07-114347.408806	D882408	LAP%	PIZSD122	2005-07-07-114350.402058	D882408	LAP%	PIZSD122							
820	820	1111111111	2005-07-07-114353.453290	2005-01-01	9901-01-01	Y	2005-07-07-114350.402058	D882408	LAP%	PIZSD122	2005-07-07-114353.453290	D882408	LAP%	PIZSD122							
820	820	1111111111	2005-07-07-114347.408806	2005-01-01	9901-01-01	Y	2005-05-13-12.55.57.541468	D882408	LAP%	PIZSD122	2005-07-07-114347.408806	D882408	LAP%	PIZSD122							
820	820	2222222222	2005-07-07-1145.09.524501	2002-01-01	9901-01-01	N	2005-07-07-1146.27.211445	D882408	LAP%	PIZSD122	9901-01-01-00.0000.000000										
820	820	2222222222	2005-07-07-1145.30.304582	2002-01-01	9901-01-01	Y	2005-07-07-1145.09.524501	D882408	LAP%	PIZSD122	2005-07-07-1145.30.304582	D882408	LAP%	PIZSD122							
820	820	2222222222	2005-07-07-1146.27.211445	2002-01-01	9901-01-01	Y	2005-07-07-1145.30.304582	D882408	LAP%	PIZSD122	2005-07-07-1146.27.211445	D882408	LAP%	PIZSD122							

Figure 7-21 Complex Type Table Sample

The sample application found in ‘bc.ndvr.prod1.apsprog(piz5*)’ provides examples of the code used to support the complex type of this technique.

Purge

1. The DB2 databases where the records that are to be purged, should be backed up before the purge.
2. Records to be purged are written to a sequential file or tape containing the keys to purge.
3. Then, the file containing the keys to purge is read. The records matching the selection are purged, and the records are written to a sequential file or tape. The purged records should also be written to the Disaster Recovery file.
4. For a large amount of purged records, consideration should be made to reorg the databases.
5. A process should also be written to restore any purged records to the operational database.

7.3.10 IMS Database Audit Trail and Purge Processing

7.3.10.1 Concepts Involved

Data Management

7.3.10.2 Operating System Platform

Host

7.3.10.3 Intended Audience

- System Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.3.10.4 Required Skills/Knowledge

- APS
- IMS

7.3.10.5 Related Techniques

- History Generating Applications
- DB2 Database
- IMS Checkpoint/Restart Processing

7.3.10.6 Overview

The IMS Audit Trail and Purge Processing technique at BlueCross BlueShield of South Carolina describes the audit trail process required for all update activity against IMS databases and the purging and archiving of data so it can be recovered. Audit trails are required for all update activity to any IMS database.

7.3.10.7 Detail Discussion

Audit

The process for producing an audit trail when using IMS databases involves using one of the IMS audit databases to record all activity against the database. The audit databases are defined generically so the application area can determine how and when they would be used. A typical use of the database would be writing images of all segments that are added and deleted, and writing a before and after image for updates. The audit databases are unloaded each night. Based on business requirements, it is up to the application area to determine the data retention and archival requirements.

Purge

1. The IMS databases where the records that are to be purged, should be backed up before the purge.
2. Records to be purged are written to a sequential file or tape containing the keys to purge.
3. Then, the file containing the keys to purge is read. The records matching the selection are purged, and the records are written to a sequential file or tape. The purged records should also be written to the Disaster Recovery file.
4. For large amount of purged records, consideration should be made to reorg the databases.
5. A process should also be written to restore any purged records to the operational database.

7.3.11 IMS Checkpoint/Restart Processing

7.3.11.1 Concept Involved

Data Management

7.3.11.2 Operating System Platform

Host

7.3.11.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.3.11.4 Required Skills/Knowledge

- Application Productivity System (APS) Batch
- Information Management System (IMS)

7.3.11.5 Related Techniques

Modules by Function

7.3.11.6 Overview

IMS Checkpoint/Restart is a technique that is used in batch processing to allow the restart of an ABENDED batch job to automatically reposition IMS databases and Generalized Sequential Access Method (GSAM) sequential files to the last successful database commit. BlueCross BlueShield of South Carolina (BlueCross) requires checkpoint/restart for all batch programs that update IMS databases. The use of this technique eliminates operator error by automatically repositioning all files and databases to the last successful commit and reduces run time so the job does not have to be rerun from its beginning.

7.3.11.7 Detail Discussion

For handling the IMS Checkpoint/Restart technique within batch jobs, the IMS databases are accessed by either the Batch Message Processing (BMP) or the Data Language Interface (DLI) protocol.

Batch Message Processing (BMP) allows batch jobs to read and update IMS databases without disrupting processing when an abend occurs because backouts are handled by the IMS Database Control Environment (DBCTL) system. BMP protocol automatically backs out the updates on the IMS databases to the last successful commit when the job abends. It is recommended that BMPs be restarted in a timely fashion (usually less than two (2) hours) after they abend so that log files will not be required in the

restart process. DBA RECOMMENDATION: For most IMS batch processes, BMP should be used instead of DLI (exceptions listed below).

Data Language Interface (DLI) is different from BMP in that its logging is done as part of the batch job instead of in the IMS DBCTL system. A good example of when to use DLI would be IMS database purges or conversions that have extremely large volumes of data to update and log. If an abend occurs in a DLI job that was updating IMS databases, a batch backout job may be needed if indicated in the Job Entry System (JES) log before the job is restarted. A batch backout does an “undo” on the database updates from the time of the abend back to the last commit point. Locks are held on the database updates until the backout completes, so time is of the essence in running the backout.

Macros Used for Checkpoint/Restart Technique

\$UV-CHKPAUTO	This macro sets the checkpoint/restart parameters and should be the first call in the IMS batch program.
\$UV-CHKPINIT	Performs the initial call to checkpoint control using savearea parameters and should be used at the start of processing.
\$UV-CHKPCALL	Performs the checkpoint call to checkpoint control. The application program should make a checkpoint call for a unit of work. The application program should be designed to specify checkpoint interval on a parameter card.
\$UV-CHKPTERM	Performs termination processing. The application should make one call at the end of processing. If the ‘TERM’ is not issued, then the next time the job is run, restart is assumed.

Table 7-4 Macros Used for Checkpoint/Restart Technique

Example of How to Use Macros and How to Set Up the JCL

```

/***** 00010000
/* THIS IS AN EXAMPLE OF READING IN A GSAM FILE AND      * 00020000
/* DELETING SEGMENTS FROM THE IMS DATABASE USING      * 00030000
/* CHECKPOINT RESTART                                00040000
/***** 00050000

00060000
00070000
WS01   WS-COUNTERS-AREA-LTH          PIC S9(5) COMP VALUE +20. 00080000
WS01   WS-COUNTERS-AREA.            00090000
      05  WS-INPUT-COUNT          PIC S9(8) COMP VALUE +0. 00100000
      05  WS-UNKNOWN-SEG-COUNT    PIC S9(8) COMP VALUE +0. 00110000
      05  WS-STMCSPCH-DELETED    PIC S9(8) COMP VALUE +0. 00120000
      05  WS-STMCSPID-UPDATED    PIC S9(8) COMP VALUE +0. 00130000
      05  WS-CKPT-COUNT          PIC S9(8) COMP VALUE +0. 00140000
                                         00150000

```

```

/* INPUT GSAM FILE                                00160000
WS01  DCDMSP50.                                 00170000
      05 WS-PRG-SPSB-KEY.                         00180000
          10 WS-PRG-SBSCR-ID-NO      PIC X(14).    00190000
          10 WS-PRG-RP-NO        PIC X(03).    00200000
          10 WS-PRG-PLAN-CODE-NO   PIC X(03).    00210000
      05 WS-PRG-SPID-KEY      PIC X(004).    00220000
      05 WS-PRG-SPCH-KEY.                         00230000
          10 WS-PRG-SPCH-KEY-DTE   PIC S9(9) COMP-3. 00240000
          10 WS-PRG-SPCH-AUTH-NO  PIC X(13).    00250000
                                         00260000
NTRY                                         00270000
                                         00280000
/***** THIS IS THE BEGINNING ENTRY POINT OF THE PROGRAM. * 00290000
/***** THIS MACRO SETS THE CHECKPOINT/RESTART PARMs       00300000
                                         00310000
                                         00320000
/*THIS MACRO SETS THE CHECKPOINT/RESTART PARMs       00330000
                                         00340000
$UV-CHKPAUTO                                  00350000
                                         00360000
PERFORM 1000-HOUSEKEEPING                    00370000
                                         00380000
REPEAT                                         00390000
    PERFORM 2500-GET-INPUT                     00400000
    UNTIL EOF-INPUT-GSAM                      00410000
        PERFORM 3000-PURGE-AUTHS                00420000
                                         00430000
    PERFORM 9000-TERMINATION                  00440000
                                         00450000
PARA   1000-HOUSEKEEPING                    00460000
/***** THIS PARAGRAPH ISSUES THE INITIAL RESTART CALL TO * 00470000
/*      DETERMINE IF THIS IS A RESTART. IT ALSO INDICATES * 00480000
/*      WHICH AREAS OF WORKING STORAGE ARE TO BE SAVED OR   * 00490000
/*      RESTORED AND INDICATES THE LENGTHS OF THOSE AREAS. * 00500000
/***** THIS PARAGRAPH ISSUES THE INITIAL RESTART CALL TO * 00510000
                                         00520000
                                         00530000
$UV-CHKPINIT WS-COUNTERS-AREA-LTH          00540000

```

	... WS-COUNTERS-AREA	00550000
		00560000
	IF CHKP-CNTL-RESTART	00570000
	DISPLAY K-PROGRAM-RESTART-MSG	00580000
		00590000
	FALSE EOF-INPUT-GSAM	00600000
		00610000
PARA	2500-GET-INPUT	00620000
	/*****	00630000
	/* READ THE GSAM INPUT FILE AND SET PROGRAM FLAGS	*00640000
	/* AT END OF FILE.	*00650000
	/*****	00660000
		00670000
	FALSE RESET-POSITION	00680000
		00690000
	\$IM-GN DCDMSP50	00700000
		00710000
	IF IM-OK	00720000
	ADD +1 TO WS-INPUT-COUNT	00730000
	ELSE-IF IM-END-OF-DATA OR IM-SEGMENT-NOT-FOUND	00740000
	TRUE EOF-INPUT-GSAM	00750000
	ELSE	00760000
	\$BC-UNV-SHOW '2500-READ-INPUT'	00770000
	DISPLAY K-GSAM-READ-ERRMSG	00780000
	DISPLAY 'IM-DB-PCB-STATUS = ' IM-DB-PCB-STATUS	00790000
	\$BC-UNV-ABEND 4000	00800000
		00810000
PARA	3000-PURGE-AUTHS	00820000
	/*****	00830000
	/* PURGE THE HEADER USING AN OBTAIN HOLD THEN AN ERASE	* 00840000
	/*****	00850000
		00860000
	PERFORM 4000-GET-SPCH-HOLD	00870000
	IF IM-OK	00880000
	PERFORM 4100-ERASE-SPCH	00890000
	PERFORM 3500-CALL-CHECKPOINT	00900000
		00910000
PARA	3500-CALL-CHECKPOINT	00920000
	/*****	00930000

```

/* THIS PARAGRAPH INVOOKS THE CHECKPOINT CALL MACRO.          *00940000
/*****00950000
00960000
$UV-CHKPCALL          00970000
00980000
PARA 4000-GET-SPCH-HOLD          00990000
/*****01000000
/* GET UNIQUE TMCS AUTH HEADER SEGMENT (HOLD FOR UPDATE) *01010000
/*****01020000
01030000
DB-OBTAİN REF SPSB-STMCSPSB          01040000
... WHERE SPSB-KEY = WS-PRG-SPSB-KEY          01050000
... REF SPID-STMCSPID          01060000
... WHERE SPID-KEY = WS-PRG-SPID-KEY          01070000
... REC SPCH-STMCSPCH          01080000
... WHERE SPCH-KEY = WS-PRG-SPCH-KEY          01090000
... HOLD          01100000
01110000
IF AB-ON-REC          01120000
$BC-UNV-SHOW-NAMED WS-PRG-SPCH-KEY          01130000
DISPLAY 'IM-DB-PCB-STATUS = ' IM-DB-PCB-STATUS          01140000
$BC-UNV-ABEND 4000          01150000
01160000
PARA 4100-ERASE-SPCH          01170000
/*****01180000
/* DELETE TMCS AUTH HEADER SEGMENT          *01190000
/*****01200000
01210000
DB-ERASE REC SPCH-STMCSPCH          01220000
01230000
IF IM-OK          01240000
ADD +1 TO WS-STMCSPCH-DELETED          01250000
ELSE
DISPLAY K-STMCSPCH-ERASE-ERRMSG          01270000
$BC-UNV-SHOW-NAMED WS-PRG-SPCH-KEY          01280000
DISPLAY 'IM-STATUS      = ' IM-STATUS          01290000
DISPLAY 'IM-DB-PCB-STATUS = ' IM-DB-PCB-STATUS          01300000
$BC-UNV-ABEND 4000          01310000
01320000

```

PARA	9000-TERMINATION	01330000
	/******	01340000
	/* PERFORMS CHECKPOINT TERMINATION PROCESS	*01350000
	/******	01360000
		01370000
	\$UV-CHKPTERM	01380000

JCL SAMPLE

```

//*****
//** THIS IS A SAMPLE BMP JCL READING GSAM FILE *
//*****
//*
//TMCSCA2Q PROC CONDSET=000,           SET CONDITION CODE
//      SYSOUTL='L',
//      MBR1=TMCSDXXX,          APPL PROGRAM NAME
//      PSB1=TMCSDXXX,          PSB NAME
//      IMSID=IMP1,             IMS DBCTL ID
//      IRLM=IRLM,              IRLM
//      AGN=TMCSPROD            APPLICATION GROUP NAME
//*
//*****
//** BMP PROGRAM READING GSAM FILE AND UPDATING IMS DATABASE
//*****
//TMCSS050 EXEC PGM=DFSRRC00,
//      PARM=(BMP,&MBR1,&PSB1,,,
//      N00000,,,,,&IMSID,&AGN,,,'')
//*****
//** INCLUDE BMP FILES
//*****
//      INCLUDE MEMBER=IMSPBMP
//*
//IMSLOGR DD DUMMY
//IMSMON DD DUMMY,
//      DCB=(IMSP.CDMS.LOGBASE,BUFNO=16)
//*
//*****
//** GSAM INPUT FILE
//*****
//CDMSP500 DD DSN=BC.YOUR.GSAM.FILE,DISP=SHR
//*

```

```
*****  
/* CONTAINS CHECKPOINT FREQUENCY EVERY 50 RECORDS  
/*  
/* TMCSXXXX000050  
*****  
//SYSIN      DD   DSN=BC.NDVR.PROD1.CARDLIB(TMCSXXXX),DISP=SHR  
/*  
//D00AERR    DD   SYSOUT=&SYSOUTL  
//IMSERR    DD   SYSOUT=&SYSOUTL  
/*
```

7.3.11.8 Definitions

Checkpoint

Checkpoints (commits) are issued to establish synchronization points throughout an application program. Checkpoints can be issued based on a number of updates or 'n' number of minutes or seconds.

- A checkpoint releases locks on databases being processed and therefore should be issued periodically to allow concurrent processing with other batch and online systems but not so frequently as to add significant overhead to the program.
- A checkpoint call will issue a checkpoint for all IMS databases accessed by that application.
- Thought should be given as to what comprises a unit of work so that restart from the last checkpoint is done at an appropriate place in the process.
- Checkpoints can be "basic" or "extended." Basic checkpoints only commit the database updates and can't be used for restart. Extended checkpoints commit database updates but also allow for repositioning of databases and sequential files during restart. Extended checkpoints also allow for multiple fields to be saved in working storage that may be useful during restart (e.g., keys for repositioning DB2 databases and counters to be used across restarts).

Restart

A restart allows the program to be restarted from the last successful "checkpoint," instead of the program having to be restarted from the beginning.

IMS Checkpoint/Restart must be used in the following application programs:

- **IMS Program**
If a program updates an IMS database, IMS checkpoint/restart MUST be used.

If a program is "read only" to an IMS database AND is considered "long running," then IMS checkpoint/restart may be used. In this case, the usefulness of checkpoint/restart is the restart portion. What is considered "long running" may vary among different applications, but a rule of thumb is two hours (clock time).
- **DB2 (with or without IMS)**
If a program updates a DB2 database AND reads or updates an IMS database, IMS

checkpoint/restart MUST be used.

If a program updates a DB2 database (but does NOT read or update an IMS database), DB2 commits or IMS checkpoint/restart MUST be used.

- **MQ**

Executing IMS and IBM's Message Queue requires that IMS BMP be used for checkpoint/rollback logic. Simple MQ commits and backouts do not function in an IMS composite.

7.3.11.9 Important Notes

- **GSAMs** — It is the application's responsibility to reposition all non-IMS files. Therefore, it is strongly recommended that all sequential files be defined and accessed as an IMS GSAM database so that IMS can handle the repositioning.
 - o GSAMs can be defined as input or output.
 - o GSAMs can be defined as FB or VB.
 - o Output GSAM files must be allocated and cataloged in a step prior to the IMS step.
 - o If using tape for a GSAM, it is recommended that BLKSIZE=0 not be used. Instead, use a BLKSIZE that is valid for the Logical Record Length (LRECL) being used.
 - o If using tape for a GSAM, the word GSAM should be used as a node somewhere in the dataset name. There is an exit in place that will interrogate the dataset name to ensure that the tape dataset retains the correct expiration date after an abend. GSAM dataset name examples include these:
 - TEST.AMMS.S81S.GI.GSAM.EXTRACT.SRT(+1)
 - BC.AMMS.AMM SD105.GSAM.CHCW.AUD(+1)
 - o IMS will not tolerate records being added or removed from the GSAM files between the abend and the restart. Any changes to the GSAM files will cause a U0102 upon restart. If records need to be ignored during a restart, then they should be bypassed with program logic.
 - o B37s on an output GSAM file must be resolved using a KNKYCOPY job (combination of IEBUGENER and SORT) that preserves the record formatting of the file before the IMS program is restarted. The programmer may reference JCL found in BC.NDVR.PROD1.PROCLIB(KNKYCOPY) and follow the instructions in the comments section.
- **DLI LOGs** — IMS jobs that use checkpoint and run as DLI must specify a log file on the IEFRDER DD statement.
 - o It is recommended that dual tape logging (IEFRDER and IEFRDER2) be used in all production DLI jobs that update IMS databases. The Generation Data Group (GDG) base for the production logs should be created by an IMS DBA prior to the first run.
 - o To ensure that the dataset retains the correct expiration date after an abend, there are exits in place to interrogate the log dataset name. The first node of the tape dataset name should start with IMST for test or IMSP for production. It should also include the word LOG in either position 20 or position 23 of the name. Log dataset name examples include these:
 - IMST.AMMS.MM08D001.LOG(+1)
 - IMSP.AMMS.A7.MM08D001.LOG(+1)
 - IMSP.AMMS.A7.MM08D001.LOG2(+1)

- o All IMS logs should be specified with (NEW,CATLG,CATLG) so that the log files are cataloged in the case of an abend and can be used in subsequent backouts and restarts.
- o If a DLI job abends with a U0616 tape error on an IMS log file, an IMS DBA must be contacted to close/copy the log file before a backout can be run.
- If a BMP job abends with a U0102 on the restart, an IMS DBA must be contacted to provide the DBCTL logs needed for restart. This is generally because the job was not restarted in a timely fashion after the abend. The logs will be added to the restart JCL on the IMSLOGR DD statement.
- IMS extended restart will restart the program from the last checkpoint issued prior to the abend. If a program must be restarted prior to that point, then the checkpoint will need to be deleted and the program will be restarted from the beginning. The programmers can delete checkpoints in the test environments, and an IMS DBA must delete the checkpoint in the production environment.
- When running BMP, do not use TIME= or OUTLIM= parameters in the JCL. When a TIME or OUTLIM limit is reached, the job will abend bringing down the IMS DBCTL system.
- Batch jobs processing against DB2P cannot run under BMP processing. They must use DLI processing.

7.3.11.10 Existing ISSM Standards

- Refer to *Technical Standards — Applications > Application Coding > IMS Checkpoint/Restart Guidelines*.
- Refer to *Technical Standards — Applications > Application Coding > IMS Programming Considerations*.
- Refer to *Technical Standards — Applications > Application Development Support Tools > Naming Conventions > COBOL DDNames*.
- Refer to *Technical Standards — Applications > File Design > Enterprise Server — IMS*.

7.3.12 Handling Mixed Case Characters in Online Host Applications

7.3.12.1 Concepts Involved

Flexibility

7.3.12.2 Operating System Platform

Host

7.3.12.3 Intended Audience

- System Analysts
- Software Designers
- Software Developers
- System Experts
- System Designers
- Solution System Designers
- Architects

7.3.12.4 Required Skills/Knowledge

Programming Skills

7.3.12.5 Related Techniques

N/A

7.3.12.6 Overview

The “Handling Mixed Case Characters in Online Host Applications” technique addresses the need to accept and store MIXED CASE values from a CICS 3270 session. Data maintained in 3270 environments is also used for presentation, such as the Web. This data can be stored in MIXED CASE.

There are three different data conversion methods available to a CICS 3270 screen:

- UPPER CASE conversion
- UPPER CASE conversion for TRANSID only
- NO UPPER CASE conversion

BlueCross BlueShield of South Carolina (BlueCross) currently uses two of these methods. The BlueCross Corporate environment has a default setting of “UPPER CASE.” The BlueCross EDC environment has a default setting of “UPPER CASE for TRANSID only.”

This technique defines a single solution regardless of the data conversion method.

In order to accomplish this technique, application changes are needed to override these default settings. Your Business Requirements will determine if this technique is appropriate for your application.



NOTE Because of the potential impact to the application, Enterprise Architect approval is required. Please review with your area Architect and/or Solution System Designer.

7.3.12.7 Detailed Discussion

Our current data storage devices (DB2, IMS, etc.) already have the ability to store MIXED CASE data. In order for the BlueCross environments 3270 sessions to process MIXED CASE data, the application area must override the default setting of the 3270 session.

The application program should be written so that it saves off the original CASE settings so that at the end of the session it can be returned to the default setting. To receive MIXED CASE data the application program must use the “NOUCTRAN” no uppercase translation command. Once this command is issued, all data fields on the 3270 session will accept and process MIXED CASE data.

Applications must programmatically account for any fields that must remain in UPPER CASE. Since the stored data must also be available for retrieval from any 3270 session, all KEY fields must be coded so

that they are stored in UPPER CASE. The application code must also ensure that the 3270 session is converted back to the default setting at the end of their application use.

To ensure the terminal is always in the proper CASE setting, the application program should set the default setting at the top of the program and set the needed Mixed Case setting (No Upper Case Translation) just before sending the map. This approach will ensure the terminal is in the proper case setting even if the application program abends.

Failure to convert the session back to the default setting may create problems for the next user of the 3270 session.

Since many 3270 screens are also used by SOA services, the application area should ensure that the screens which allow MIXED CASE data can also be placed into the MIXED CASE mode by an SOA service.

Coding Examples

```
XXXX-TRANSID-INVOKED.  
* SAVE ORIGINAL UCTRANST.  
    EXEC CICS INQUIRE TERMINAL(EIBTRMID)  
        UCTRANST(TS88I000-ORIG-UCTRANST)  
    END-EXEC
```

In your application programs, you always set back to default at the top of the program.

```
* SET UPPER CASE TRANSLATION AS DEFAULT.  
    EXEC CICS SET TERMINAL(EIBTRMID)  
        UCTRANST(TS88I000-ORIG-UCTRANST)  
    END-EXEC
```

Also, set to NOUCTRAN right before sending the map.

```
EXEC CICS SET TERMINAL(EIBTRMID) NOUCTRAN  
END-EXEC
```

Use the following code to handle fields that must always be in UPPER CASE after the terminal has been set to Mixed Case:

```
MOVE FUNCTION UPPER-CASE (Screen-Variable) TO Screen-Variable
```

7.3.12.8 ISSM Standards

N/A

7.3.13 Large Working Storage Areas

7.3.13.1 Concepts Involved

- Modularity
- Single Source Application Design
- Common I/O and Edit modules

7.3.13.2 Operating System Platform

Host

7.3.13.3 Intended Audience

- System Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.3.13.4 Required Skills/Knowledge

CICS, APS, IBM Websphere MQ

7.3.13.5 Related Techniques

- Loose Coupling
- Reuse
- Flexibility
- Encapsulation

7.3.13.6 Overview

Application programs today require more working storage areas to process the data. Data today is coming in the format of XML or AMB/COBOL copylib that contain record sizes greater than 32K. When processing the data, the programmer needs to be aware of how they are referencing the data by using INITIALIZE statements or moving spaces to large working storage areas. Programmers need to be aware of the impact to CICS response time when using large working storage areas.

7.3.13.7 Detailed Discussion

Initializing Data

The INITIALIZE statement sets selected categories of data fields to predetermined values. It is functionally equivalent to one or more MOVE statements. However, it is inefficient to initialize an entire group unless you really need all the items in the group to be initialized. If you have a group that contains

OCCURS data items and you want to set all items in the group to the same character (for example, space or x'00'), it is generally more efficient to use a MOVE statement instead of the INITIALIZE statement.

MQ Messages

When reading in a MQ message, there is no need to initialize the MQ record if you are reading the message into a working storage area. When reading the MQ record, you should only read in the length of the MQ record for variable length messages.

Example of an INITIALIZE statement not needed:

```
WS01          WS-MQ-TABLE      PIC     X(26136598)
              INITIALIZE WS-MQ-TABLE

CALL MQIP-INTERFACE USING
... DFHEIBLK DFHCOMMAREA
... MQIP WS-MQ-TABLE
```

Performance Considerations

The performance considerations for an INITIALIZE statement on a program that has five (5) OCCURS clauses in the group are as follows:

- When each OCCURS clause in the group contained 100 elements, a MOVE to the group was 8% faster than an INITIALIZE statement of the group.
- When each OCCURS clause in the group contained 1000 elements, a MOVE to the group was 23% faster than an INITIALIZE statement of the group.

Other Considerations

- Avoid an INITIALIZE statement unless the functionality is really needed.
- It is much faster to MOVE SPACES or X'00' to the group.
- If individual fields need to be set to spaces or different types of zero (external decimal, Packed-decimal, numeric-edited), then by all means use an INITIALIZE statement.
- Use GETMAIN as a fixed length area in linkage for large working storage areas.
- After the first INITIALIZE statement, save the area in working storage and use a working storage area for a move instead of an INITIALIZE statement.

7.3.14 IBM DB2 Analytics Accelerator (IDAA)

7.3.14.1 Concepts Involved

- IBM DB2 Analytics Accelerator (IDAA)
- SQL
- Dynamic Queries

7.3.14.2 Operating System Platform

Host

7.3.14.3 Intended Audience

- Architects
- Solution System Designers
- System Designers
- Business System Analysts
- System Analysts
- System Experts
- Software Designers
- Software Developers

7.3.14.4 Required Skills/Knowledge

Dynamic Query Applications, SQL

7.3.14.5 Related Techniques

None

7.3.14.6 Overview

The IDAA for z/OS is a hardware appliance designed for high performance with an optimized database query engine. The appliance is connected to DB2 for z/OS via OSA pipes, which transfer requests and data from DB2 for z/OS to IDAA and results from IDAA to DB2. Queries which meet the criteria to be accelerated will be transferred to the appliance for processing, and will receive a performance boost from acceleration. One of the big benefits of IDAA is to save cost by offloading MIPS from the Host to the appliance.

7.3.14.7 General Principles

Data for Production Application DB2 Tables is loaded to a “shadow copy” of the DB2 table, which resides on IDAA. This data must be refreshed any time that the data is refreshed in DB2P (via utility loads, application programs or ad hoc queries). The refresh of IDAA data is generally done by the DBAs using stored procedures supplied by IBM.

The IDAA supports dynamic queries from SAS, QMF, Easytrieve, Optim, ODBC, Cognos, Business Objects, Rocket Shuttle, StarSQL, and sample unload (DSNTIAUL) with SQL.

The DB2 Optimizer (Optimizer) determines at run time whether or not a query meets the criteria for acceleration, and whether or not the query will benefit from acceleration. If the Optimizer decides that the query will run faster in native DB2, then it will run there. If the Optimizer decides that the query will run faster on the accelerator, then it will ship it to the IDAA.

Sometimes the Optimizer decides incorrectly that a query will run faster in native DB2. If this happens, there are special commands that can be used to force the query to run on IDAA or force a query to run on native DB2. Contact ICT Database Administration for the commands and how to use them.

7.3.14.8 Criteria for Query Acceleration

Queries must meet the following criteria in order to be accelerated:

1. All tables referenced in the query must be loaded and enabled on the appliance.
2. The query must be read only. For ambiguous cursors, you can add the phrase FOR FETCH ONLY to force the DB2 Optimizer to recognize it as read only.
3. Queries which contain a correlated sub-query will run on the device if the DB2 Optimizer can rewrite the query such that it no longer uses a correlated sub-query. If not, the query will not run on the appliance.
4. The query must be a SELECT statement.

NOTE The above list is not complete but is a general guideline. Refer to the IDAA for z/OS User's Manual for further details.



If you don't know if your query is eligible for acceleration, you can find out by running an EXPLAIN on the SQL in question. ICT Database Administration can assist you with this.

7.3.14.9 IDAALIST

ICT Database Administration has created a REXX than can be used to verify if a table is on the IDAA. For more information, see the document titled "IDAALIST for tables on Accelerator" in Outlook under Public Folders > All Public Folders > Legacy TAO Conferences > DATABASE-SUPPORT.

7.3.14.10 Adding Tables to IDAA

If your Production Application table is not on the IDAA, then you can request that it be added to the IDAA by filling out a Database Help ticket on TSC Self-Service. The tables being requested would then be vetted by the DBAs through Tech Support for approval.

If you are designing new tables to support reporting on large volumes of data, you may want to consider requesting that the tables be added to IDAA at implementation. This request can be included in the "Additional Requirements" section of the DB2 Table Request form. The reports in question should be validated during the Design, Development and Validation Phase to determine if acceleration is truly needed.

7.3.14.11 Tips for Using the Accelerator

1. Always use an ORDER BY to get your data in a particular order. (Repeat executions of a query without an ORDER BY will return results in a different order each time.)
2. Add FOR FETCH ONLY to make the query read only.
3. Use the special commands to force the query to IDAA when needed.
4. Use EXPLAIN to see if the query qualifies for acceleration.
5. Use IDAALIST to see if the table/view is on the IDAA.
6. Run your queries after the IDAA refresh is complete.

7.4 Non-Host Platform Techniques

7.4.1 Non-Host Server Logging Process

7.4.1.1 Concepts Involved

- Flexibility
- Balancing
- Reconciliation

7.4.1.2 Operating System Platform

Non-Host

7.4.1.3 Intended Audience

- Systems Analysts
- Software Designers
- Software Developers
- System Experts
- Architects

7.4.1.4 Required Skills/Knowledge

To use this technique basic conceptual knowledge of computer languages is required.

7.4.1.5 Related Techniques

Data Storage on Servers (future)

7.4.1.6 Overview



NOTE The details of the technique below are approved for iPlanet-based applications. A refined technique is currently under development for applications deployed to WebSphere Application Server or z/Linux.

This section will describe baseline procedures for logging processing for Non-Host Servers. A log is a message container object, most commonly a file, but it is not uncommon to find logs to be: database tables, email messages, etc. A log is commonly the output of an application, containing the successful/failed steps.

A log strategy is very important to BlueCross BlueShield of South Carolina (BlueCross), because it allows for the possibility to inject secondary processes against it, like monitoring for instance. Also it

allows the possibility to retrieve execution steps back in time, and understand the possible failures, for as long as the logs are retained.

In order to accomplish this, every Non-Host logging will have to implement the following criteria:

- Applications must create a new log or append to a previous log. Preferred solution is to use a rolling file with specified number of log file retention.
- Scheduled jobs running on Non-Host must not overwrite existing files.
- All program exceptions that are considered errors must incorporate logging facilities, including the program stack if available. If an exception is considered part of normal program execution then the exception should be committed to logs only when the appropriate debug level is set for the application.
- All log entries should have a timestamp generated.
- Capture 3270 screen and save to log file if error occurs while screen scraping screen. Status line and parameters must also be logged.
- User authentication information must not be logged.
- Purge criteria must be considered.

7.4.1.7 Detailed Features

- Applications must create a new log or append to a previous log. Preferred solution is to use a rolling file with specified number of log file retention.

- If Java is used, then log4j is the recommended platform, it will include the capability of having rolling files based on customized criteria.
 - If running on a Unix platform the command statement could be the following:

```
cmd 1>>file-`date \+\%Y\%m\%d\-\%H\%M`.log 2>&1
```

- If running on the Windows platform the command line could be:

```
command 1>>file.log 2>&1
```

- Scheduled jobs running on Non-Host must not overwrite existing files.
 - If system commands are used, avoid to overwrite files by redirecting standard output with the overwrite file feature “>”, instead use the “>>” append to file feature.
 - If log is within the application, then open the file in append mode, ideally you would create a filename string containing the running date yyyyymmdd, so that a new log file is generated every day.
- All program exceptions that are considered errors must incorporate logging facilities, including the program stack if available. If an exception is considered part of normal program execution then the exception should be committed to logs only when the appropriate debug level is set for the application.
 - If Java is the development platform, every try-catch block must contain a log statement, preferably using log4j framework.
 - It will be mandatory to check/log function return codes.
- All log entries should have a timestamp generated.

- If Java is the development platform, the timestamp can be generated through log4j.
- If performance should become an issue, then the timestamp can be moved at a block level, instead of a line.
- Capture 3270 screen and save to log file if error occurs while screen scraping screen. Status line and parameters must also be logged.
- If running ClientSoft, the 3270 screen dump can be generated using the trace facility. This does not require any code changes.
- User password information must not be logged.
- All participant-specific information (PHI) such as name, social security number, tax ID, date of birth, and all other sensitive information must be encrypted if logs reside on server that is accessed from the Internet.
- Logs must be archived from the servers that can be accessed from the Internet.
- Logging level must be configurable to ensure flexibility and ease of use. Recompiling programs to change the logging level needs to be avoided.

 **NOTE** LCAS-NH has a recommended logging level usage standard, but this has not been established as a standard.)

- Logging can be used to load Event Management for Non-Host Balancing and Reconciliation.

Log4J is an Open Source JAVA logging API that is widely used at BlueCross. It provides a robust, reliable, fully configurable, easily extendible, and easy to implement framework for logging Java applications for debugging and monitoring purposes. The recommended logging levels for log4J are:

Debug

Use the DEBUG level priority for log messages that convey extra information regarding life-cycle events. Developer or in depth information required for support is the basis for this priority. Looking at the DEBUG and INFO messages for a given service category should tell you exactly what state the service is in, as well as what server resources it is using: ports, interfaces, log files, etc.

Logging Characteristics

This priority indicates the application requires no attention and no action.

INFO

Use the INFO (not to be confused with BlueCross INFORMs) level priority for service life-cycle events and other crucial related information. Looking at the INFO messages for a given service category should tell you exactly what condition or state the service is in.

Logging Characteristics

This priority indicates the application requires these events to be monitored in some way.

WARN

Use the WARN level priority for events that may indicate a non-critical service error. Recoverable errors or minor breaches in request expectations fall into this category. The distinction between WARN and ERROR may be hard to discern and so it is up to the developer to judge. The simplest criterion is to ask would this failure result in a user support call; if it would, use ERROR; if it would not, use WARN.

Logging Characteristics

This priority indicates the application requires these events to be monitored with some type of action after a specified threshold has occurred.

ERROR

Use the ERROR level priority for events that indicate a disruption in a request or the ability to service a request. A service should have some capacity to continue to service requests in the presence of ERRORS.

Logging Characteristics

This priority indicates the application requires an email or some kind of notification be sent for a review of the problem where action can be taken at a later time.

FATAL

Use the FATAL level priority for events that indicate a critical service failure. If a service issues a FATAL error it is completely unable to service requests of any kind.

Logging Characteristics

FATAL would indicate the application requires an email or page to be sent out for immediate attention where some kind of action is needed.

Log entries should use a standard format.



NOTE A standard logging format is currently being developed.
