WITN11070108

Document Title: HNG-X Architecture - Counter Business Application

Document Reference: ARC/APP/ARC/0009

Release: Not Applicable

Abstract:

Document Status: APPROVED

This document contains sections that have been identified to POL as comprising evidence to support the assessment of named Acceptance Criteria by Document Review.

This text must not be changed without authority from the FS Acceptance Manager.

Author & Dept: Andy Thomas

(Contributors: Ben Holland, Giacomo Piccinelli, Dave Johns, James Skene, Lee Walton, Jim Sweeting)

External Distribution:

Approval Authorities:

Name	Role	Signature	Date
Amit Apte	HNG-X Chief Technical Officer		
David Court	HNG-X Programme Manager		

Note: See RMGA HNG-X Reviewers/Approvers Role Matrix (PGM/DCM/ION/0001) for guidance.

Documents are uncontrolled if printed or distributed electronically. Please refer to the Document Library for the current status of a document.

WITN11070108

0 Document Control

0.1 Table of Contents

WITN11070108

WITN11070108

WITN11070108

0.2 Figures

0.3 Tables

WITN11070108

0.4 Document History

Version No.	Date	Summary of Changes and Reason for Issue	Associated Change - CP/PEAK/PPRR Reference
0.1	30-OCT-2006	Initial draft	N/A
0.2	07-NOV-2006	Draft for review	N/A
0.3	28-NOV-2006	Issued for formal review. Contains updates following informal review, and resolution of some outstanding issues. Changes from version 0.2 are provided as marked changes.	N/A
0.4	02-FEB-2007	Revised following review and updated for consistency with high-level design.	N/A
1.0	14-FEB-2007	Submitted for Approval. No other changes from version 0.4.	N/A
1.1	22-NOV-2007	Updated for PCI changes, minor changes and clarifications following feedback from design and resolution of further comments on the document.	N/A
1.2	29-FEB-2008	This document has been revised by RMGA Document Management on behalf of the Acceptance Manager to contain notes which have been identified to POL as comprising evidence to support the assessment of named Acceptance Criteria by Document Review. This text must not be changed without authority from the FS Acceptance Manager. This version will not require full review using the RMGA Document Control Process, as agreed between Acceptance Manager and Programme Management.	N/A
1.3	23-JAN-2009	Changes arising from Architectural End-to-End Review. Updated SSL related packages. Updates to support retention of Utimaco VPN in architecture. De-scoped migration to Windows XP. Introduction of Section 0.5 Acceptance by Document Review table containing POL Non Functional Requirements and their section references for Acceptance by Document Review. This version of the document has been released for acceptance of the sections outlined in section 0.5 by the Post Office.	N/A CP4622 CP4549 CP4472 *(DEFFERED)*
1.4	30-MAR-2009	Updated Java Security Section	N/A
1.5	18-JUN-2009	Update acceptance criteria regarding ARC-429. Tidy Up TODO entries Remove Section 3.9 Workflow Subsystem. The solution no longer requires this subsystem in order to support the business requirements. Add new Section 3.9 Process Engine Add acceptance criteria regarding SEC-317 & SEC-3072	N/A
1.6	17-JUL-2009	Add new Section 3.9 Process Engine Update internal references to new section Process Engine in place of Workflow Subsystem. Add acceptance criteria regarding SEC-317 & SEC-3072	N/A
1.7	15-SEP-2009	Updated Circulation List	N/A
1.8	30-SEP-2009	Incorporated Review Feedback	N/A
2.0	01-OCT-2009	Version for approval	N/A
2.1	16th July 2011	Minor changes to encryption for FiTE-2 Crypto Services (section 7.5.2) and Bureau Spot Rates (Section 5.3.2)	CP0510 CP0618
2.2	16th July 2011	Updated document control sections.	N/A
2.3	5th Aug 2011	Updated after internal review.	N/A
3.0	26th Aug 2011	Version for approval	N/A

WITN11070108

0.5 Review Details

This document is subject to Group Review. Mandatory Reviewers should consult the author to determine the details of the associated Group Review before submitting a formal comment sheet.

Review Comments by :	
Review Comments to :	
Role	**Name**
Solution Design / Development	Tariq Arain
Infrastructure Design	Alex Kemp
Head of Service Introduction	Role unfilled
CISO	Ian Howard
Information Governance	Bill Membery
Capacity & Configuration Manager	Mark Brosnan
Optional Review	
Role	**Name**
Security & Risk Team	CSPOA.Security GRO
Architect	Jason Clark
Network Architect	Mark Jarosz
Test Design	Sheila Bamber ; Debbie Richardson
Service Network	Andrew Hemingway
Head of Service Management	Tony Atkinson
LST Manager	Mark Ascott
SV&I Manager	Chris Maving
POL Test Manager	James Brett (POL, JTT)
VI & TE Manager	Not applicable
Testing Manager	Debbie Richardson
SSC	Steve Parker
Business Continuity	Adam Parker
Head of Service Support	Sarah Bull
Infrastructure Delivery Manager	Martin Brett
Integration Team Manager	Vijesh Pandya
Programme Manager	David Court
Integrity Testing	Not applicable
Operational Security	Donna Munro
Core Division	Ed Ashford
Core Division	Andrew Gibson
CTO	Amit Apte
Lead Architect Systems / Estate Management	Ian Bowen
POL Design Authority	Ian Trundell (POL, via RMGA Document Management)
Issued for Information – Please restrict this distribution list to a minimum	
Position/Role	**Name**
Acceptance Manager	David Cooke

Note: See RMGA HNG-X Reviewers/Approvers Role Matrix (PGM/DCM/ION/0001) for guidance.

WITN11070108

0.6 Acceptance by Document Review

The sections in this document that have been identified to POL as comprising evidence to support Acceptance by Document review (DR) are listed below for the relevant Requirements:

POL NFR DR Acceptance Ref	Internal FS POL NFR Reference	Document Section Number	Document Section Heading
SEC-3209	SEC-3298	7.1	Secure Counter Build
SEC-3231	SEC-3304	6.2	Diagnostics and Trace
SEC-3231	SEC-3304	7.5.5	Diagnostic Logs and Trace files
SEC-3231	SEC-3304	7.5.6	PCI
SEC-3233	SEC-3207	6.2	Diagnostics and Trace
SEC-3233	SEC-3207	7.5.5	Diagnostic Logs and Trace files
SEC-3233	SEC-3207	7.5.6	PCI
ARC-429	ARC-429	9.1	Parallel Processing
SEC-3226	SEC-3226		Whole Document
SEC-3226	SEC-3226	7.1.1	Java Security
SEC-317	SEC317	7	Security
SEC-3072	SEC-3073	7	Security

0.7 Associated Documents (Internal & External)

Reference	Version	Date	Title	Source
PGM/DCM/TEM/0001 (DO NOT REMOVE)	1.0	13-JUN-06	Fujitsu Services Post Office Account HNG-X Document Template	Dimensions
PGM/DCM/TEM/0001 (DO NOT REMOVE)			Fujitsu Services Post Office Account HNG-X Document Template	Dimensions
			Sub schedules B3.2, B3.3, B3.4 and B6.2.	HNG-X contract
ARC/SOL/ARC/0001			HNG-X Solution Overall Architecture	Dimensions
ARC/SOL/ARC/0005			HNG-X Counter Training Offices Architecture	Dimensions
ARC/GEN/REP/0001			HNG-X Glossary	Dimensions
ARC/APP/ARC/0001			HNG-X Reference Data Architecture	Dimensions
ARC/APP/ARC/0002			HNG-X Integration Architecture	Dimensions
ARC/APP/ARC/0003			HNG-X Counter Architecture	Dimensions
ARC/APP/ARC/0004			HNG-X Branch Access Layer Architecture	Dimensions
ARC/APP/ARC/0005			HNG-X Online Services Architecture	Dimensions
ARC/APP/ARC/0007			HNG-X Batch Application Architecture	Dimensions
ARC/APP/ARC/0008			HNG-X Branch Database Architecture	Dimensions
ARC/MIG/STG/0001			HNG-X Migration Strategy	Dimensions

WITN11070108

ARC/NET/ARC/0001			HNG-X Network Architecture	Dimensions
ARC/PER/ARC/0001			HNG-X System Qualities Architecture	Dimensions
ARC/PPS/ARC/0001			HNG-X Platform and Storage Architecture	Dimensions
ARC/SEC/ARC/0002			Security Constraints	Dimensions
ARC/SEC/ARC/0003			HNG-X Security Architecture	Dimensions
ARC/SOL/ARC/0005			HNG-X Architecture - Counter Training Offices	Dimensions
ARC/SVC/ARC/0001			HNG-X Support Services Architecture	Dimensions
ARC/SYM/ARC/0001			HNG-X System and Estate Management Architecture	Dimensions
DES/GEN/PRP/0001			Counter UI Technology Evaluation	Dimensions
DEV/MGT/MAN/0001			HNG-X User Interface Construct Guidelines	Dimensions
REQ/CUS/STG/0001			HNG-X Migration Strategy – Agreed Assumptions and Constraints	Dimensions
REQ/CUS/STG/0002			HNG-X Branch Exception Handling Strategy – Agreed Assumptions and Constraints	Dimensions
REQ/CUS/STG/0003			HNG-X Counter Reference Data Delivery Strategy - Agreed Assumptions and Constraints	Dimensions
REQ/CUS/STG/0004			HNG-X Training Strategy (CTO) - Agreed Assumptions and Constraints	Dimensions
SD/DES/005			Horizon OPS Reports and Receipts - Post Office Account Horizon Office Platform Service	PVCS
SVM/SDM/PRO/0017			Transaction Time Benchmarking Process	Dimensions
TST/GEN/STG/0002			HNG-X Testing Strategy	Dimensions
ARC/APP/RTM/0001			Requirements Traceability Matrix for Counter Application	Dimensions
CP4472 (HNG-X CP0041)			XP Counter Descoping *(CURRENTLY DEFERRED)*	
CP4549 (HNG-X CP0098)			Retention of Utimaco VPN - Development and Operational Impacts	
CP4622 (HNG-X CP0156)			PPD Changes for Key management	
DES/APP/HLD/0035			Exceptions And Logging Frameworks High Level Design	Dimensions
DES/APP/HLD/0038			HNG-X Counter Applications: Peripherals Subsystem High Level Design	Dimensions
DES/APP/HLD/0039			HNG-X Counter Applications: Presentation Subsystem High Level Design	Dimensions
DES/APP/HLD/0040			HNG-X Counter Applications: Interaction Subsystem High Level Design	Dimensions
DES/APP/HLD/0041			HNG-X Counter Applications: Business Logic Subsystem High Level Design	Dimensions

WITN11070108

DES/APP/HLD/0042			HNG-X Counter Applications: Business Services Subsystem High Level Design	Dimensions
DES/APP/HLD/0043			HNG-X Counter Applications: Communications Subsystem High Level Design	Dimensions
DES/APP/HLD/0045			HNG-X Counter Applications: Reference Data Subsystem High Level Design	Dimensions
DES/APP/HLD/0046			HNG-X Counter Applications: Business Data Subsystem High Level Design	Dimensions
DES/APP/HLD/0047			HNG-X Counter Application High Level Design	Dimensions
DES/APP/HLD/0058			UCR Document For Banking Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0059			UCR Document For Branch Accounting Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0060			UCR Document For Branch Admin Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0061			UCR Document For Branch Support Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0062			UCR Document For Bureau Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0063			UCR Document For Cash & Stock Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0064			UCR Document For ETU Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0065			UCR Document For In / Out Pay Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0066			UCR Document For Postal Services Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0067			UCR Document For Retail Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0068			UCR Document For Shared Use Case Barrel High Level Design	Dimensions
DES/APP/HLD/0069			HNG-X Automated Payment / Advanced Data Capture (AP-ADC) High Level Design	Dimensions
DES/APP/HLD/0076			HNG-X PDL High Level Design	Dimensions
DES/APP/HLD/0083			HNG-X Recovery High Level Design	Dimensions
DES/APP/HLD/0096			HNG-X Training System High Level Design	Dimensions
DES/APP/HLD/0099			HNG-X Counter Applications: System Use Case High Level Design	Dimensions
DES/APP/HLD/0102			High Level Design Of HNG-X Counter Help System	Dimensions
DES/APP/HLD/0109			Counter Reporting High Level Design	Dimensions
DES/APP/HLD/0110			HNG-X Counter Applications: Transaction Benchmarking High Level Design	Dimensions

WITN11070108

DES/APP/HLD/0858			Counter HLD for CP0510 Bureau TPOS	Dimensions
DEV/APP/AKI/0001			HNG-X DEVELOPER GUIDE FOR JAVA DEVELOPMENTS	Dimensions
DES/GEN/MAN/0001			HLD Designer Guidelines	Dimensions

Unless a specific version is referred to above, reference should be made to the current approved versions of the documents.

0.8 Abbreviations

Abbreviation	Definition
ADC	Advanced Data Capture
AP	Automated Payment
APOP	Automated Payment Out-Pay
BDM	Business Data Manager
BDO	Business Data Objects
BLM	Business Logic Manager
BLO	Business-Logic Object
BRS	Business Requirements Specification
BSM	Branch Session Manager
CCD	Contract Controlled Document
CTO	Counter Training Office
DR	Disaster Recovery
DVLA	Driver and Vehicle Licensing Agency
GUI	Graphical User Interface
JVM	Java Virtual Machine
LDEMs	Logical Device Element Managers
LDM	Logical-Device Manager
LDWM	Logical Device Widget Manager
PAF	Postal Address File
PKE	Public Key
POU	Peripheral-Operation Unit
RDM	Reference Data Manager
RSIM	Remote Services Interaction Manager
RSIs	Remote Service Interfaces
SLT	Service Level Target
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol

WITN11070108

UIAM	User Interaction Agent Manager
UIAs	User Interaction Agents
UIEs	User Interaction Elements
UIW	User Interface Widgets
XML	Extensible Mark-up Language

0.9 Glossary

See also HNG-X Glossary (ARC/GEN/REP/0001).

Term	Definition
Clean Room	The design area where the UI Constructs were originally designed in isolation from the main development for the protection of IPR.
JavaPOS	Java standard for the retail point-of-sale community
SYSMAN	Systems Management application

0.10 Changes Expected

Changes

0.11 Accuracy

Fujitsu Services endeavours to ensure that the information contained in this document is correct but, whilst every effort is made to ensure the accuracy of such information, it accepts no liability for any loss (however caused) sustained as a result of any error or omission in the same.

0.12 Security Risk Assessment

Security risks have been assessed and it is considered that there are no security risks relating specifically to this document.

WITN11070108

1 Introduction

1.1 Overview

This document defines the architecture of the HNG-X Counter Business Application.

The role of Counters in the overall HNG-X System is described in *HNG-X Solution Overall Architecture (ARC/SOL/ARC/001)*.

The position of the Business Applications within the overall Counter Architecture is described in *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

This section of the document explains the background to the HNG-X programme, the document context and scope. The notation is also explained, and key assumptions are identified.

1.2 Background

Post Office Ltd operates in both the retail and financial services industries. The main channel to market for Post Office is a network of approximately 14,000 branches with volumes of up to 28 million customers per week. In addition, Post Office has been expanding the use of the Internet and Call Centres as part of a comprehensive multi-channel strategy.

The objective of the HNG-X programme is to develop a system with structural and operational characteristics that substantially reduce ongoing support and maintenance costs with respect to the current Horizon system.

The overall requirement is that the business capabilities offered by the current system (Horizon) are preserved in the new system (HNG-X). However, a limited number of business capabilities will be revised based on a joint optimisation of business requirements and system properties.

The analysis of the serviceability profile for Horizon has highlighted data management as one of the most significant drivers for cost. The storage of transactional data within counters causes the need for security mechanisms that impact both the structural complexity and the operational performance of the Counter Business Application. In addition, the presence of sensitive data on the counter increases the time, complexity, and ultimately the cost of maintenance procedures.

The HNG-X solution is based on a set of business applications that support a centralised model for data storage. The counters retain operational data (e.g. Reference Data) and business logic, but transactional information is stored directly in the Data Centre.

The counter side of the new applications is based principally on Java technology. The counter hardware is reused from Horizon with the initial migration deploying the new application on the existing Windows NT 4.0 operating system.

Following completion of the HNG-X Release 1 migration, a subsequent operating system upgrade to Windows XP or other operating system may be undertaken.

1.3 Context

Three types of HNG-X architecture document exist:

- The **overall solution architecture** is the top-level description of the HNGX solution, identifying the individual topic architectures.
- A **topic architecture** is the first level of decomposition of the overall solution architecture. Topic architectures include Applications, Platforms, Networking, System Management, Security, System Management, Recovery and Resilience.

WITN11070108

- Where a topic architecture is complex, then a further decomposition into individual **component architectures** may be required. For example the System Management topic architecture could constitute Software Distribution, Monitoring, Remote Access and Diagnosis and Time Synchronisation component architectures. A component architecture identifies the scope for the associated High Level Design.

This document is the **topic architecture** for the **HNG-X Counter Business Application**, and is derived from the outline architecture described in *HNG-X Solution Architecture Outline (ARC/SOL/ARC/0001)*, and will be used as input into the subsequent High Level Designs.

It is closely associated with the document *HNG-X Counter Architecture (ARC/APP/ARC/0003)* which describes the platform onto which the Counter Business Applications are deployed.

The following High Level Design documents can be considered children of this architecture:

Table 1 – Child High Level Design Documents

Document Reference	Title
DES/APP/HLD/0035	Exceptions And Logging Frameworks High Level Design
DES/APP/HLD/0038	HNG-X Counter Applications: Peripherals Subsystem High Level Design
DES/APP/HLD/0039	HNG-X Counter Applications: Presentation Subsystem High Level Design
DES/APP/HLD/0040	HNG-X Counter Applications: Interaction Subsystem High Level Design
DES/APP/HLD/0041	HNG-X Counter Applications: Business Logic Subsystem High Level Design
DES/APP/HLD/0042	HNG-X Counter Applications: Business Services Subsystem High Level Design
DES/APP/HLD/0043	HNG-X Counter Applications: Communications Subsystem High Level Design
DES/APP/HLD/0044	HNG-X Counter Applications: Workflow Subsystem High Level Design
DES/APP/HLD/0045	HNG-X Counter Applications: Reference Data Subsystem High Level Design
DES/APP/HLD/0046	HNG-X Counter Applications: Business Data Subsystem High Level Design
DES/APP/HLD/0047	HNG-X Counter Application High Level Design
DES/APP/HLD/0058	UCR Document For Banking Use Case Barrel High Level Design
DES/APP/HLD/0059	UCR Document For Branch Accounting Use Case Barrel High Level Design
DES/APP/HLD/0060	UCR Document For Branch Admin Use Case Barrel High Level Design
DES/APP/HLD/0061	UCR Document For Branch Support Use Case Barrel High Level Design
DES/APP/HLD/0062	UCR Document For Bureau Use Case Barrel High Level Design
DES/APP/HLD/0063	UCR Document For Cash & Stock Use Case Barrel High Level Design
DES/APP/HLD/0064	UCR Document For ETU Use Case Barrel High Level Design
DES/APP/HLD/0065	UCR Document For In / Out Pay Use Case Barrel High Level Design
DES/APP/HLD/0066	UCR Document For Post Services Use Case Barrel High Level Design
DES/APP/HLD/0067	UCR Document For Retail Use Case Barrel High Level Design
DES/APP/HLD/0068	UCR Document For Shared Use Case Barrel High Level Design
DES/APP/HLD/0069	HNG-X Automated Payment / Advanced Data Capture (AP-ADC) High Level Design
DES/APP/HLD/0076	HNG-X PDL High Level Design
DES/APP/HLD/0083	HNG-X Recovery High Level Design

WITN11070108

DES/APP/HLD/0096	HNG-X Training System High Level Design
DES/APP/HLD/0099	HNG-X Counter Applications: System Use Case High Level Design
DES/APP/HLD/0102	High Level Design Of HNG-X Counter Help System
DES/APP/HLD/0109	Counter Reporting High Level Design
DES/APP/HLD/0110	HNG-X Counter Applications: Transaction Benchmarking High Level Design

1.4 Scope

The scope of this document is limited to the architecture of the business application for the HNG-X Counter platform.

The scope includes references to the hardware (platform and peripherals), to the operating system, and to the management applications and services. The detailed architecture for such elements of the HNG-X Counter System is covered in *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

The Counter Business Application is responsible for realising the following components defined in the *HNG-X Counter Architecture* (in part or in full):

- Fujitsu Interstage Client (PS500.70)
    - Application Server Client (LS600.70)
    - Encrypted Network Communications (LS600.20) *[partial]*
- Counter Business Application (PS500.75)
    - Business Applications (LS500.75)
    - Reference Data Services (LS500.77)
    - Peripheral Controller (LS500.40) *[partial]*
    - Peripheral Maintenance Services (LS500.44) *[partial]*

This document is intended to describe the software infrastructure and architecture that will be the basis for implementing the business requirements of HNG-X within the Counter Business Application.

The application architecture described within sections 2 and 3 of this document is a generic application framework for an application written in Java, following SOA principles, that fits within the target system architecture for HNG-X.

The Counter Business Application provides a set of Business Capabilities and Support Facilities (BCSF) as described in Sub-schedule B3.2 of the HNG-X contract.

In summary the set that are provided by the Counter Business Application are:

Point of Sale Capability;

In / Out Payment Capability;

APOP Facility;

Banking Capability;

DVLA Licensing Capability;

Electronic Top-Up Capability;

Bureau de Change Capability;

Postal Services Capability;

Payment Management Capability (Cash, Cheque, Vouchers, Debit or Credit Cards);

WITN11070108

Cash and Stock Management Capability;

Branch Management Capability (Stock Unit Balancing, Branch accounting, Branch Reports, Reversals and Refunds, Transaction Corrections);

Branch Administration Facility (User Log On / Off, User / Password Management, Stock Unit Creation / Allocation, Provision of Secure Inactivity Time-Out Facilities, Generic User Help System),

Branch Support Facility (Sales Prompts, Bulk Input of transactions, Reference Data, PAF, Message Handling, Audit and Training).

The document does not intend to provide any details of the business requirements themselves; instead the architecture should be seen as the framework within which the business requirements will be implemented.

The non-functional requirements described in sections 4 - 9 and 11 include resilience, stability, performance, manageability, security and testability as they apply to the Counter Business Application. These sections summarise the responsibility of the Counter Business Application in these areas. The full set of requirements for the business application is covered in the relevant business and system requirement specifications. High level design documents will cover the implementation details of the individual business application requirements.

Section 10 covers the Migration from Horizon to HNG-X from the counter perspective. It includes a high level description of the Horizon changes that are specific to the migration of a branch to HNG-X.

In addition, the document addresses the following areas:

- Some of the major technological choices and their rationales (some other choices are detailed in separate Option papers – e.g. *Counter UI Technology Evaluation (DES/GEN/PRP/0001)*).
- Service interface definition and best practices.
- Architectural description and high-level design for each of the major components within the Counter Business Application.
- Practices for implementing individual services within the provided architecture/framework.
- Other areas of consideration, such as recommended testing practices, performance considerations etc.
- Impact of surrounding environment.

Hereafter, use of the term “the application” is used to refer to the Counter Business Application as defined by this architecture.

WITN11070108

2 Architectural Description

2.1 Requirements

2.1.1 Business Requirements

The Business Requirements are summarised within the HNG-X contract under section B3.2 (Business Capabilities and Support Functions). Details are provided within the Post Office Use Case descriptions. In addition, further business requirements can be found within the explicit requirement statements – both from Post Office and from Customer Services.

The Counter Business Requirements are captured in the Telelogic DOORS system, and this system can provide a view of the requirements that apply to the counter.

2.1.2 System Requirements

The Counter System Requirements provide the detailed interpretation of the requirements set that applies to the counter.

The overriding objectives for the application are:

- That this is a front-end system and the user must be kept in mind
- The application must be robust and reliable. The software must not break, and if it does it does so gracefully.

The Counter System Requirements are captured in the Telelogic DOORS system, and this system can provide a view of the requirements that apply to the counter.

2.2 Architectural Objectives

Some objectives derive from objectives of the wider HNG-X System; others are specific to the counter. Note that the objectives covering the legislative and standards requirements are covered in ARC/SEC/ARC/0001.

2.2.1 Business Objectives

The business objectives are to:

- Support current Business Processes.
- Maintain or improve Usability.
- Maintain or improve Robustness.
- Reduce the Total Cost of Ownership.
- Maintain or improve Business Agility (namely product introduction speed).
- Maintain or improve Training capabilities (namely CTOs).

2.2.2 Technical Objectives

The technical objectives are to:

WITN11070108

- Support the Online Model for business transactions.
- Leverage existing as well as new Hardware.
- Improve Reliability and Robustness.
- Improve Modularity.
- Maintain or improve Manageability.
- Comply with security requirements.

2.3 Architectural Principles

The Architectural principles are high-level principles that complement the more specific indications in the counter architecture.

2.3.1 Multiplier Effect

The following factors have to be considered:

- There will be in excess of 30,000 counters in the estate.
- Technical choices that propagate in any way beyond the boundaries of one individual counter must take into consideration the potential of multiplier effects in terms of the impact on the overall HNG-X system.
- Local optimisations may have significant global benefits.
- Local mistakes may cause disastrous global consequences.

2.3.2 Data Driven Approach

2.3.2.1 Business Data

Business processes and transactions are driven by business data that changes over time (e.g. product price, product availability, content of notices). The counter must support change of business data without need for structural/code changes.

2.3.2.2 Business Processes and Transactions

The structure of business processes and transactions can change over time (e.g. to support new regulations). The counter must support change to the structure of business processes and transactions in a way that minimises the need for structural/code changes.

Similar considerations apply to the introduction as well as the elimination of new or existing business processes and transactions. Some capabilities (such as AP-ADC) are provided directly to Post Office to manage business processes and transactions via reference data.

2.3.2.3 System Structure

Structural properties of the counter may need to vary over time.

A balanced approach must be taken in deciding the most appropriate means of change. The approach must consider and contrast aspects such as the frequency and cost of change with the cost and complexity of change capabilities to be introduced upfront.

WITN11070108

A data-driven approach should not be assumed as a default option.

2.3.2.4 Correctness of Data

The data delivered to the counter is assumed to be correct both in terms of content and format.

Only limited capabilities are required on the counter to contain the effect of incorrect data being delivered.

2.3.3 No Transaction Data on the Counter

The counter must store all transaction data centrally. At no point can transaction data be placed in the counter permanent storage (i.e. disk).

If the counter is unable to write data to the central system, the business application must take appropriate steps to handle the exception case. Any recovery data needed must be stored on the central systems.

The application must conform to any business specific security requirements concerning masking or suppression of data when written to diagnostic logs or trace files. See section 7.5.5.

Implicit storage (e.g. via the operating system swap file), does not need to be protected against, but the volume of data that can be exposed in this way must be minimised.

2.3.4 No Network - No Business

It is acceptable for the business capabilities offered by the counter to degrade as a consequence of degradation in network capabilities.

However, capability degradation must be contained and managed within the counter in a way that:

(1) Minimises the impact on end users.

(2) Preserves a consistent state for business data, processes, and transactions.

2.3.5 Cohabitation

The application is not the only application hosted within the counter operating environment. Refer to *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

The application must consider the resource and operational requirements of other applications.

Reciprocity is expected from other applications.

2.3.6 Flexibility

Flexibility and generality normally come at cost in terms of structural and operational complexity.

Flexibility should be retained only where specific benefits can be clearly identified.

2.3.7 Scalability

The volume of activity on the counter (see section 9) is not expected to vary significantly over time.

In accordance with the flexibility principle, the counter should be optimised for current volumes.

WITN11070108

2.3.8 Portability

The priority for the application is to deliver all the expected business capabilities within the specific operating environments identified in this document (see section 9).

However, the portability to other operating environments should be supported wherever there is no impact on the remaining aspects of the solution (namely reliability, usability, management, security, complexity, and cost).

2.3.9 SOA

A Service Oriented Architecture approach should be applied for the modularisation of the counter.

2.3.10 Third Party Software

As a general principle, the use of third-party software (especially open source) should be avoided. See section 4.2.2

Exceptions to this rule may be identified and considered during the design stage, but each must be explicitly approved by the HNG-X Programme and the Post Office.

2.3.11 Security

The architecture security details are covered in ARC/SEC/ARC/0003.

2.4 Technology Choices

The following major decisions have already been made with regard to technology.

2.4.1 Windows NT

The application should be designed to be operating system agnostic. The initial target for HNG-X Release 1 will be to deliver the application onto Windows NT 4.0 Workstation (the current operating system in use under Horizon).

Following completion of the HNG-X Release 1 migration, a subsequent operating system upgrade to Windows XP or other operating system may be undertaken.

2.4.2 Java

The programming language for the majority of the application will be Java.

Other languages such as C can be used where specific features require interface to local / operating system resources which is difficult to implement within Java. However, any such cases must be localised and minimised.

2.4.3 Swing

The User Interface components will be implemented using Java Swing. Within the application, there is an explicit software layer (the Presentation Layer) that handles all user interactions.

WITN11070108

For further information on the rationale behind this decision, refer to the *Counter UI Technology Evaluation (DES/GEN/PRP/0001)*.

2.5 Design Topics

2.5.1 Application Help

To aid usability and to reduce the dependence on printed user manuals, the application will provide a context sensitive help capability.

Help pages will be authored using static HTML, and the context sensitive linkage between business functionality and individual help pages defined in reference data.

The help pages will be resident on the counter, delivered as reference data, eliminating any additional network traffic.

The help browser will be implemented using the JavaHelp API.

For further information, refer to the *High Level Design of the HNG-X Counter Help System (DES/APP/HLD/0102).*

2.5.2 Transaction Recovery

Failures occur in all computer systems, centralised or distributed. The more components that are involved in an application, the greater the probability of a failure occurring. In a distributed system, component failures are generally independent of each other and one failure does not necessarily (or immediately) lead to another.

The Counter Business Application provides basket functionality to which products and services can be added. These basket items represent transactions. The settlement process commits these transactions to the data centre and completes the session. However some transactions may have left state in 3rd Party systems or generated valuable artefacts. Losing these transactions is not desirable and having a recovery system is necessary to ensure the results of a transaction are applied consistently to all resources affected by a transaction.

Recovery after failure requires that information about the transaction and the resources involved survive the failure and is accessible afterwards. This information needs to be held in a durable state-store and held at the data centre, in order to reconstruct the transaction. To this end, the Branch Access Layer will provide services to allow recovery data to be stored at the request of the Counter Business Application.

Upon the next logon at a failing counter, the logon will check for the presence of any recovery data and will invoke relevant recovery scripts.

Recovery itself needs to be resilient to failure and should an error occur during the process, then the process needs to be rerun before the counter can become operational, or the recovery data flagged as outstanding in the event of repeated failures. Recovery should also be entirely predictable, leaving the outcome of the process certain that no incorrect data or status is left in any system, both internally or within 3rd Party systems.

For further information, refer to the *HNG-X Recovery High Level Design (DES/APP/HLD/0083)*.

2.5.3 Auditable Transactions

For the purpose of auditing, certain messages between the Counter Business Application and the Branch Access Layer need to be stored with their integrity guaranteed so that they can be harvested by the HNG-X Audit Servers.

WITN11070108

To this end, the Counter Business Application is able to identify a given message as Auditable.

Auditable messages use a sequence number unique to auditable messages, and should form a dense set. This sequence number is referred to as the Journal Sequence Number (commonly seen in abbreviated form as JSN).

All messages between the Counter Business Application are signed using an asymmetric key created during logon.

Auditable messages when received by the Branch Access Layer are stored verbatim in the BRDB (including the associated signature). The public key required to verify the messages is stored as an auditable message itself, created during the logon process.

In the event that the Branch Access Layer receives a duplicate JSN, this will be treated as a serious error, and Counter Business Application will abort the current user session. Upon logging in again, the JSN will be resynchronised between the Counter Business Application and Branch Access Layer.

For further information, refer to the *HNG-X Counter Application High Level Design (DES/APP/HLD/0047).*

The JSN is also persisted onto the Counter's hard drive, and is used to assist in the recovery process. For further information, refer to the *HNG-X Recovery High Level Design (DES/APP/HLD/0083).*

More details on Audit security requirements can be found in ARC/SEC/ARC/0003.

2.5.4 Transaction Benchmarking Service

A number of Service Level Agreements & Service Level Targets exist that involve the counter. Typically, these will be centred upon the overall time to perform a given transaction, excluding the time spent processing in 3rd party systems outside of Fujitsu's control.

To this end, as the Counter Business Application is the component responsible for initiating these transactions, the performance of the applicable transactions need to be measured from within this component. Where feasible, it should also collate the performance data from other components within the HNG-X solution, so that a consolidated set of performance data can be stored.

This requires that where feasible, performance data should be returned to the Counter Business Application as part of the response message to requests sent to the Branch Access Layer.

After collating the performance data for an individual transaction, the data will be sent to the data centre on a subsequent message.

In addition to the SLA / SLT measurement requirements, "Video Benchmarking" is currently used to test the performance of the Horizon application between versions, to ensure that no significant performance bottle necks are introduced. Under HNG-X, the intention is to cease video benchmarking, and provide suitable in-built timing to allow an equivalent level of analysis to be performed.

This capability will also be provided by the Transaction Benchmarking Service. Note that the data collected by the Transaction Benchmarking Service will only consist of data required for performance analysis.

For further information, refer to the *HNG-X Counter Applications: Transaction Benchmarking High Level Design (DES/APP/HLD/0110).*

2.5.5 Counter Training Offices

Unlike Horizon, the build of a training counter under HNG-X will be exactly the same as a live counter, and prior to installation, could be used in either environment.

During installation, when the counter is associated with a branch, counters will determine from reference data if the branch they are assigned to a CTO branch or a live branch.

WITN11070108

During authentication by users with the branch access layer, both the counter and branch access layer will assert whether the counter belongs to a live or CTO branch and the assertion must be consistent across the platforms. Thereafter, all communication with the branch access layer will be routed to either the live or CTO environment.

For further information, refer to the *HNG-X Counter Training Offices Architecture (ARC/SOL/ARC/0005)* and the *HNG-X Training System High Level Design (DES/APP/HLD/0096)*.

WITN11070108

3 Counter Blueprint

3.1 Overview

3.1.1 Modelling Formalism

In this document, the application architecture is presented as a UML model, with associated natural language commentary.

In the model, packages are used to represent various subsystems of the counter architecture. A subsystem encapsulates a set of closely related technical capabilities required to enable the business capabilities of the counter architecture.

The contents of each subsystem are presented using a number of component diagrams. Components are used to represent capabilities of the application, types of business data, and types of data implied by the required capabilities. For example, product data is represented by a class in the model; also, various specification and configuration data for data-driven capabilities of the architecture are represented as classes.

Components are represented by boxes in class diagrams. Associations between components are represented using lines connecting components with arrowheads indicating the direction of association.

Associations represent permissions for one capability, represented by a component, to make use of another capability or manipulate some data. Associations are also used to represent references between data objects.

The direction of associations is significant. If an association is represented with an arrowhead, it is navigable in only one direction, meaning that the source capability (without the arrowhead) can utilise the target capability (with the arrowhead), but not the reverse. Associations without arrowheads are navigable in both directions.

Where a component is exposed outside of the subsystem, it is associated with a port on the subsystem, represented as a small square with an attached sphere. Where a component uses a port exposed by another subsystem it is associated with a port on the subsystem, represented as a small square with an attached cup.

Because the components in this architecture document represent capabilities, or units of functionality, they will not necessarily be implemented in one-to-one correspondence by classes in a design for the counter architecture. Nevertheless, access relationships, indicated by associations, must be respected in the design. Also, all capabilities must be implemented in the design.

All components in the model should be considered as being abstract. They represent capabilities or data which may require more concrete refinements at design time.

NB: Some of the finer grained detail has been moved from this document to the individual High Level Designs responsible for realising the design of the application.

3.1.2 Subsystem View

The counter can be characterised as a cohesive set of technical capabilities that act as an enabler for the delivery of business capabilities. However, the delivery of business capabilities is not the only driver for the architecture, design, and implementation of the counter system.

The counter system will be managed and maintained over a number of years. In addition, the dynamic nature of business requirements may require the technical capabilities of the counter to evolve. The possibility to identify and isolate technical capabilities within the system is instrumental to the prompt and effective resolution of problems, as well as the cost-effective introduction of change.

WITN11070108

In accordance to mainstream OO and SOA principles, the technical capabilities within the counter are modularised based on the notion of subsystem. In essence, a subsystem encompasses a set of related technical capabilities. The implementation of business capabilities almost invariably requires the coordinated use of different technical capabilities. Hence, interaction is required between subsystems.

WITN11070108

Subsystems encapsulate technical capabilities. In this specification, subsystems are further decomposed into classes representing finer grained capabilities required to deliver the overall capabilities of the subsystem.

There is not necessarily a one-to-one relationship between the subsystems and classes in this specification and technical components in the final implemented system. This relationship will be first established at the High Level Design stage, and then progressively refined.

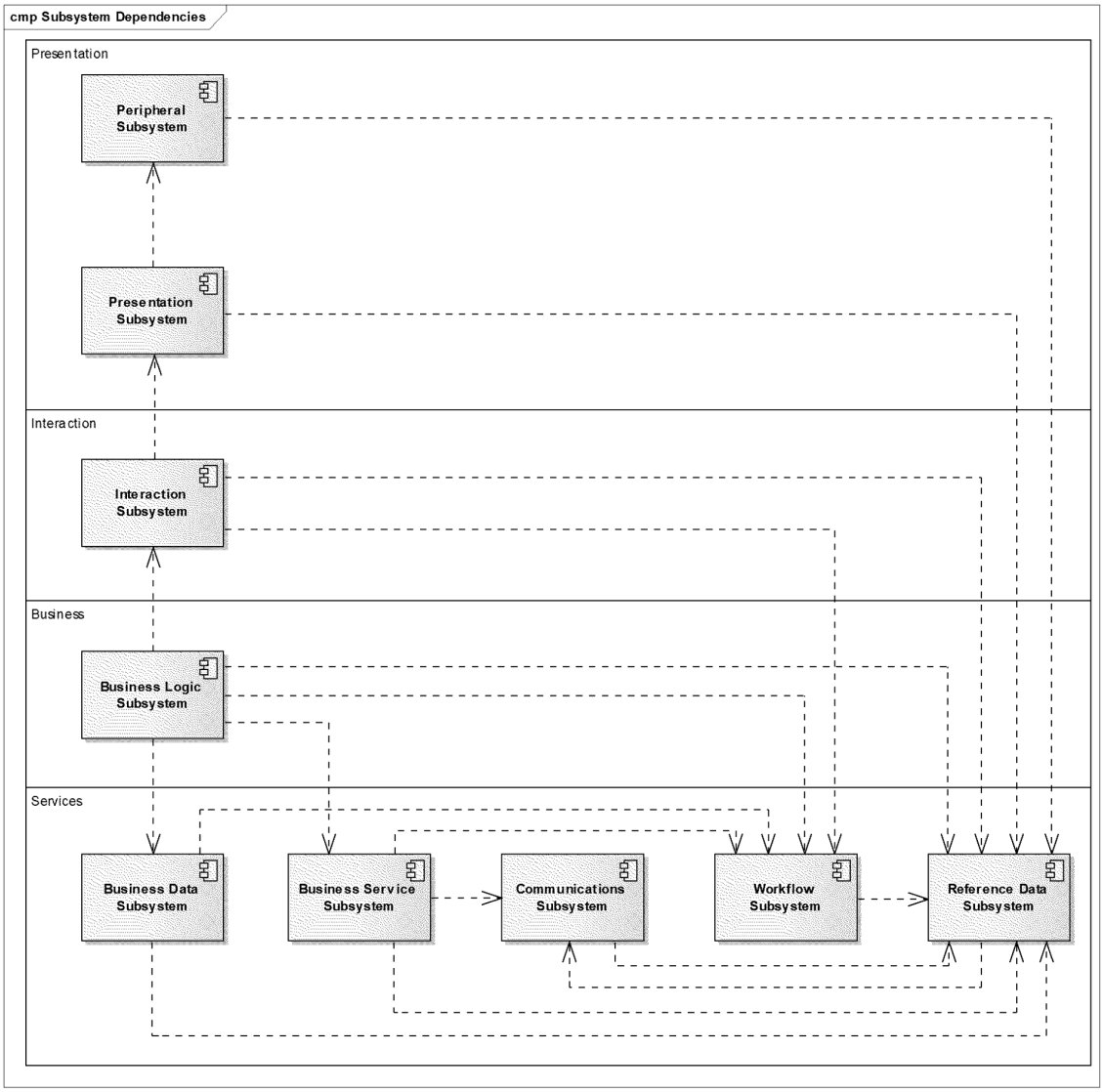


Figure 1: Counter Subsystems

The counter includes 9 main subsystems (see Figure 1):

- Peripheral Subsystem

WITN11070108

- Presentation Subsystem
- Interaction Subsystem
- Business Logic Subsystem
- Business Data Subsystem
- Business Services Subsystem
- Reference Data Subsystem
- Process Engine Subsystem
- Communications Subsystem

In addition, the application includes other application components and infrastructural capabilities. These sit outside of the application subsystems and are accessible by all of the subsystems.

From a layering perspective, the subsystems can be grouped into four layers:

- Presentation Layer (includes Peripherals, Presentation and Communications Subsystems)
- Interaction Layer (includes Interaction Subsystem)
- Business Layer (includes Business Logic Subsystem and Business Data Subsystem)
- Services Layer (includes Business Services, Process Engine, and Reference Data Subsystems)

3.1.3 Active Versus Passive Capabilities

In subsequent sections we decompose the subsystems described above into finer-grained capabilities. In general, capabilities represent a set of behaviours of the application.

We assume that capabilities may interact via synchronous calls. That is, as part of the behaviour of one capability, it may request some behaviour from another capability, suspend its own behaviour until that behaviour completes, then resume its own behaviour.

In the description of subsystem capabilities below a distinction is made between active and passive capabilities of the application.

A passive capability has some behaviour, but this behaviour only ever occurs in response to a synchronous call. For example, the Business Logic Manager in the Business Logic subsystem is a passive capability. It can create Business Logic Objects but will not do so until called by another component.

On the other hand, some capabilities are active, and so may act spontaneously without having been called by another capability. For example, there are several components within the architecture which run on a timed basis. They check a component which is external to the counter application, such as the PIN PAD. If they detect a change in the state of that external component they may start a new business process.

Active capabilities may perform any number of independent sequences of actions, not in response to a synchronous call, simultaneously.

Clearly active and passive objects will map onto some concurrency mechanism in the implementation of this architecture, to be decided at design time.

Java provides a concurrency mechanism in the form of Java “Threads”, which, depending on the Java Runtime environment, may map directly onto the underlying operating system’s threading mechanism.

WITN11070108

3.1.4 Subsystem Start-up / Shut-down

The design shown in this document does not cover facilities within the subsystems to enable start-up / shut-down. These will emerge during the design phase, and will be added during the design process, which can add to this architecture as appropriate.

All subsystems should implement a common interface to control start-up / shut-down.

Design decisions on start-up and shut-down will need to take into account the overall performance characteristics of the hardware and software platform, and in particular the trade-off between run time performance and memory usage of the individual subsystem.

3.1.5 Managed Objects

The design shown in this document adopts a common pattern across the subsystems used to instantiate the various objects exposed by each sub-system.

It is envisaged that this pattern is implemented as an abstract capability that can be used across the subsystems, and specialised through object inheritance for those subsystems where additional capabilities are required.

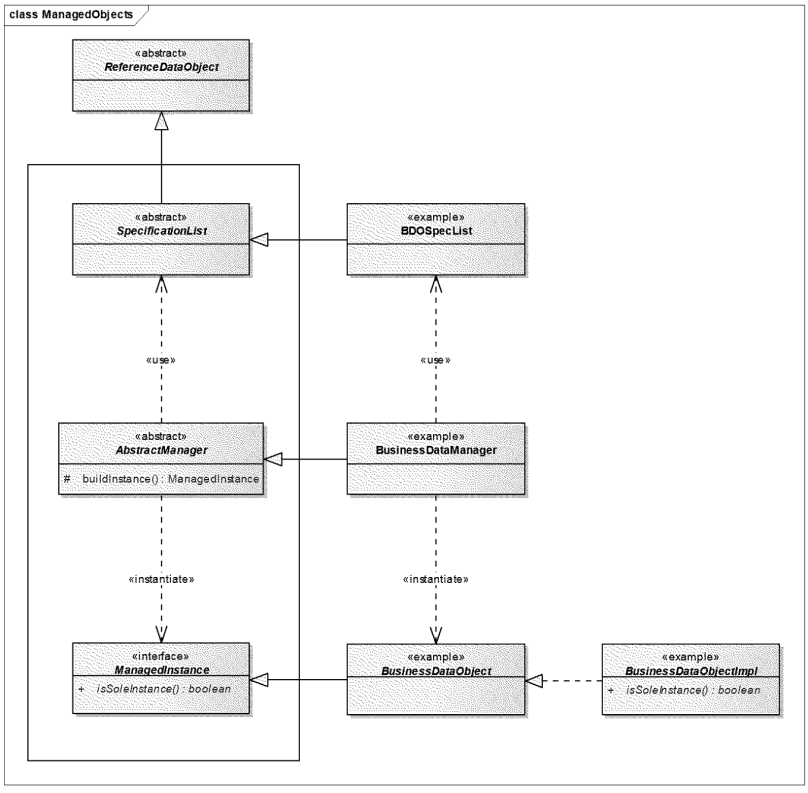


Figure 2: Managed Objects

WITN11070108

3.2 Peripheral Subsystem

3.2.1 Overview

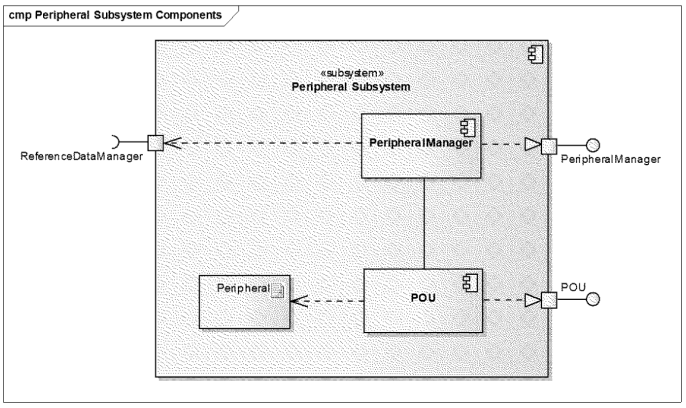


Figure 3: Peripheral Subsystem Overview

The Peripheral Subsystem provides the capability to interact with the counter peripherals programmatically.

The subsystem also ensures that any configuration policies regarding the management of the peripherals are enforced.

The subsystem is decomposed into the following capabilities:

- One Peripheral Manager
- For each peripheral, a Peripheral Operation Unit. This mediates all interactions between the peripheral and its consumers.

Peripheral Operational Unit capabilities may be configured using reference data.

For further information, refer to the *HNG-X Counter Applications: Peripheral Subsystem High Level Design (DES/APP/HLD/0038)*.

WITN11070108

3.2.2 Subsystem Dependencies

The Peripheral Subsystem interfaces with the Reference Data Subsystem to obtain configuration information for Peripheral Operational Units.

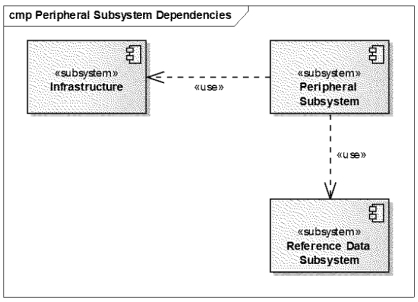


Figure 4: Peripheral Subsystem Dependencies

3.2.3 Components

3.2.3.1 Peripheral Manager

3.2.3.1.1 Description

Peripheral Operational Units are managed by a Peripheral Manager.

The Peripheral Manager activates Peripheral Operational Units for a hard-coded set of peripheral types. When the set of peripherals that the Peripheral Manager recognises is changed, the implementation of the Peripheral Manager will have to change.

3.2.3.1.2 Volume

There will be a single Peripheral Manager capability in the application.

3.2.3.1.3 Lifetime

The Peripheral Manager capability will be created at start-up and its lifetime will be that of the application.

3.2.3.1.4 Interactions

Interactions with the Peripheral Manager will be frequent. It is primarily used by consumers to attach to devices, and temporarily acquire ownership of the device. As all the consumers are very short lived, this will be very frequent (e.g. a new screen means a new menu and this means a new UIW connects to the scanner to listen for scanned tokens).

3.2.3.1.5 Concurrency

The Peripheral Manager will be a passive capability.

WITN11070108

3.2.3.1.6 Configuration

The Peripheral passes the correct configuration data to each Peripheral Operational Unit that it creates. Generally POU configuration data is not reference data.

3.2.3.2 Peripheral Operational Unit

3.2.3.2.1 Description

Each Peripheral Operational Unit has a type specific to the type of peripheral with which it interacts. In general, these types fall into three abstract types:

- Input Peripheral Operational Units; this type allows a consumer to receive input data from the peripheral. The nature of the interface provided to consumers is peripheral-specific. For example, a keyboard peripheral input interface may provide access to streams representing input data; a touch-screen peripheral input interface may provide an event-based interface.
- Output Peripheral Operational Units; this type allows a consumer to direct output data to the peripheral. The nature of the interface provided to consumers is peripheral-specific. For example, a screen may provide access to a data-buffer representing the output-raster; a printer may provide a stream to a serial device.
- I/O Peripheral Operational Units; this type are both input and output Peripheral Operational Units.

Peripheral Operational Units restrict access to the programming environment support for peripheral management, because all manipulation of peripherals should be via Peripheral Operational Units capabilities. This is intended to simplify the testing and debugging of the application.

Peripheral Operational Units also represent the capability to enforce peripheral management policies defined in reference data. For example, the timeout for powering down the screen could be configured by the Peripheral Operational Unit.

The initial known set of peripheral types for which Peripheral Operational Unit capabilities are required is:

- Input peripherals:
  - Barcode Scanner
  - Weigh Scales
- Output peripherals:
  - Slip Printer
  - Rates Board
  - Back-Office Printer
- IO peripherals:
  - Touch Screen
  - PIN Pad
  - LIFT Keyboard

A Peripheral Operational Unit may only communicate with its owner and the underlying peripheral driver.

Where feasible the JavaPOS standard shall be used to allow access to peripherals.

WITN11070108

3.2.3.2.2 The Volume

There will be a Peripheral Operational Unit for each peripheral attached to the counter.

There will be of the order of ten peripherals.

3.2.3.2.3 Lifetime

Peripheral Operational Units will be created at start-up.

3.2.3.2.4 Interactions

Peripheral Operational Units will be accessed frequently, up to several times per second.

This may be in response to peripheral events caused by interactions with the user of the application.

This may also be in response to direct calls from the consumer owning a Peripheral Operational Unit.

3.2.3.2.5 Configuration

Peripheral Operational Units may be configured using Reference Data, to reflect anticipated changes in policy which do not require recoding of the application. For example, the quality of printing provided by the back-office printer is a possible parameter of the *"BackOfficePrinterPOU"*. Another example is the amount of data used to buffer input and output data between consumers of Peripheral Operational Units and the underlying peripherals. Configuration data will be provided to Peripheral Operational Units by the Peripheral Manager.

WITN11070108

3.3 Presentation Subsystem

3.3.1 Overview

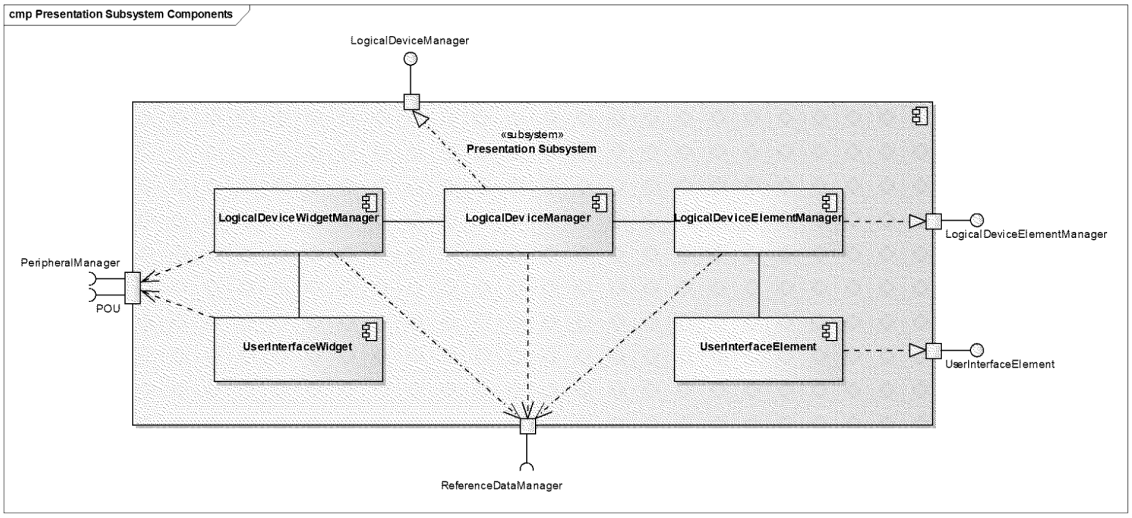


Figure 5: Presentation Subsystem Overview

The Presentation Subsystem encapsulates capabilities related to communicating with the counter user. The capabilities raise the level of abstraction from primitive interactions with peripherals (provided by the Peripheral Subsystem), such as setting a pixel on the screen, to primitive interactions with the user, such as the completion of a form. The Presentation Subsystem provides capabilities in support of the Interaction Subsystem which further raises the level to basic interactions between the user and the counter.

The Presentation Subsystem raises the level of interaction in two phases. In the first, Logical Device Widget Managers provide a logical abstraction of the capabilities of single peripherals or collections of peripherals. The capabilities of the Logical Device Widget Managers are further refined by the definition of a number of types of User Interface Widgets for each logical device, which represent primitives for interaction with the user in the context of the logical device. User Interface Widgets also provide implementation transparency, hiding the implementation of the capabilities of Peripheral Operational Units.

Logical Device Widget Managers establish ownership of Peripheral Operational Units using the Peripheral Manager. Logical Device Widget Managers create User Interface Widgets that interact with the Peripheral Operational Units that they own.

At the second level of abstraction, User Interaction Elements, managed by Logical Device Element Managers provide a business abstraction for interactions, for example in terms of forms, shopping-baskets, and account summaries.

An overall Logical Device Manager encapsulates the capability of the subsystem to grant access to Logical Device Element Managers to User Interaction Agents, which represent capabilities of the Interaction Subsystem able to manage interactions.

WITN11070108

The Presentation subsystem represents the capability to enforce policies derived from Clean Room presentation recommendations for all devices, but in particular for the terminal.

Enforcement of configuration policies at the User Interface Widget and User Interface Element level is represented in the model by the management components, Logical Device Element Managers and Logical Device Widget Managers. Checking of policies for these elements occurs when they are created.

Ideally, policies should not be violated at runtime. Therefore runtime checking may only be required during testing. Alternatively, policy exceptions may be reported to owning capabilities, and eventually handled by an exception-handling mechanism, implemented using the capabilities of the business logic subsystem.

The various capabilities of the Presentation Subsystem may make use of configuration information defined as Reference Data.

Java Swing components should be used for all on-screen presentations, unless a suitable component does not exist. Where no suitable Java Swing components exist, bespoke Java Swing components should be developed.

Pluggable Swing User Interface styles may be used to implement some aspects of User Interface Widget policy.

For further information, refer to the *HNG-X Counter Applications: Presentation Subsystem High Level Design (DES/APP/HLD/0039)*.

3.3.2 Subsystem Dependencies

Logical Device Widget Managers establish ownership of Peripheral Operational Units using the Peripheral Manager exposed by the Peripheral Subsystem. Logical Device Widget Managers create User Interface Widgets that interact with the Peripheral Operational Units that they own.

The Presentation Subsystem also interfaces with the Reference Data Subsystem to obtain configuration information for User Interface Elements and User Interface Widgets.

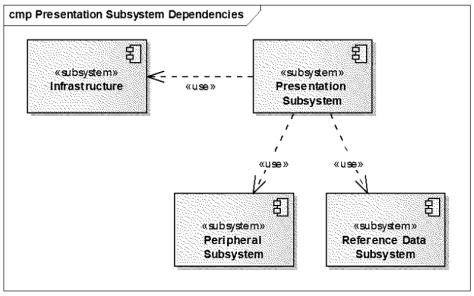


Figure 6: Presentation Subsystem Dependencies

WITN11070108

3.3.3 Components

3.3.3.1 Logical Device Widget Manager

3.3.3.1.1 Description

A Logical Device Widget Manager creates and manages a set of User Interface Widgets to provide a logical abstraction of the capabilities of one or more peripherals.

The set of User Interface Widgets that may be created and managed by a Logical Device Widget Manager will be hard-coded into the application. However User Interface Widgets may be parameterised to effect alter their appearance and behaviour.

A Logical Device Widget Manager represents the capability to enforce a policy regarding aspects of the appearance and behaviour of the User Interface Widgets it manages. A Logical Device Widget Manager for the screen represents in part the capability to enforce the fine-grained policies derived from the Clean Room recommendations. These policies are checked at object creation time.

3.3.3.1.2 Volume

The application will include the capabilities of one Logical Device Widget Manager per logical device. There will be as many or fewer logical devices than peripherals, so there will be of the order of ten Logical Device Widget Managers.

3.3.3.1.3 Lifetime

The lifetime of each Logical Device Widget Manager will be the lifetime of the application.

3.3.3.1.4 Interactions

A Logical Device Widget Manager is required to create User Interface Widgets and perform checking in response to interactions with User Interface Widgets. At peak rate, these will occur frequently, of the order of tens per second. Usually the rate will be much lower.

3.3.3.1.5 Concurrency

A Logical Device Widget Manager is a passive capability.

3.3.3.1.6 Configuration

Widget policies may be configurable using Reference Data; for example, the size and colour of buttons.

3.3.3.2 User Interface Widget

3.3.3.2.1 Description

User Interface Widgets encapsulate the capability to present an interaction element to the user, at a level of abstraction convenient for interacting with a logical device in a business process independent manner. For example, a button is a User Interface Widget. A button is a convenient abstraction in a graphical-user interface, but is independent of the business logic of any particular application.

WITN11070108

User Interface Widgets are created by a Logical Device Widget Manager on behalf of a User Interface Element and associated with one or more Peripheral Operational Units, some of the capabilities of which they may access.

Generally, User Interface Widgets may be of one of three abstract types:

- Output widgets, which deliver information to the user;
- Input widgets, which receive information from the user;
- I/O widgets, which are both Input and Output widgets.

Widgets will be of a type specific to a logical device.

Once instantiated, User Interface Widgets are always owned by a single User Interface Element.

3.3.3.2.1.1 Rendering Model

User Interface Widgets will be designed to meet the requirements of the Clean Room ergonomics and usability rules, and where they are associated with the UI are synonymous with the "sub-constructs" defined within the *HNG-X Construct Catalogue (DEV/GEN/MAN/0003)*. Although they represent a logical view of the capabilities of the device, they will not expose significantly more capabilities than are required to meet these requirements. If, for example, the Clean Room dictates that the screen should be permanently split in two, then widgets for the top and bottom parts of the screen should be implemented.

Where flexibility is included in User Interface Widgets, it should preferably be configured by reference data and not exposed in the interface to the User Interface Element.

A User Interface Widgets for the screen encapsulates all the rendering related data (e.g. position, size, borders) as well as the logic concerning they ways in which such data can change (e.g. size can change only of a factor of +/- 10%).

The rendering logic can take into consideration the state of the User Interface Widget.

The detail of the rendering logic derives from the rules of ergonomics and usability defined by the Clean Room.

3.3.3.2.2 Volume

There are likely to be of the order of tens of User Interface Widgets in existence at any point in time. User Interface Widgets are fine-grained capabilities owned by User Interface Elements, so care should be taken to ensure that User Interface Elements are not maintained for longer than is required, and that they are destructed correctly.

3.3.3.2.3 Lifetime

The lifetime of a User Interface Widget will range from of the order of a second to the lifetime of the application.

3.3.3.2.4 Interactions

Peak interaction with User Interface Widgets will be of the order of tens of interactions per second, initiated by events from Peripheral Operational Units, or by User Interface Elements driven by other active capabilities in the architecture such as interaction agents or business logic. Peak interaction rates will only occur sporadically, and User Interface Widgets may go for long periods experiencing no interaction or very little interaction.

WITN11070108

3.3.3.2.5 Concurrency

A User Interface Widget is a passive capability.

3.3.3.2.6 Configuration

Configuration information for a User Interface Widget will fall into two categories:

- Configuration information provided by the Logical Device Widget Manager. This will be provided when the User Interface Widget is created and will not vary during the lifetime of the User Interface Widget. It represents policy regarding the presentation or behaviour of the widget which is defined in the reference data. For example, this may control the shape and size of a button.
- Configuration information provided by the User Interface Element. This will provide the run-time configuration of the User Interface Widget. For example, this may control the position and labelling of a button.

3.3.3.3 User Interface Element

3.3.3.3.1 Description

At the second level of abstraction of interaction encapsulated by the presentation subsystem, User Interaction Elements are managed by Logical Device Element Managers.

A User Interface Element consists of a set of User Interface Widgets arranged and coordinated for a specific purpose. For example, a panel area, a set of buttons and a text field could be combined to present a dialog to the user.

User Interface Elements are largely static and do not manage the logic of long running interactions with the user. This is handled by the interactions subsystem.

Generally, User Interface Elements will be of one of three abstract types:

- Input; constructed partly of input widgets, the User Interface Element will maintain an Input State, reflecting the data contained in those widgets.
- Output; constructed partly of output widgets, the User Interface Element will maintain some layout information, reflecting the relationship of these widgets and their display states.
- I/O; both an Input and Output User Interface Element.

A User Interface Element will not interact with other User Interface Elements.

A User Interface Element is responsible for basic validation of the values attached to its attributes. Validation encompasses basic relations between attributes.

3.3.3.3.1.1 Rendering model

The rendering of a User Interface Element derives from the rendering of the related User Interface Widgets. The complexity of the relations varies between User Interface Elements. For example, if one of the User Interface Widgets is not visible, the User Interface Element may or may not be considered visible.

The detail of the rendering logic derives from the rules of ergonomics and usability defined by the Clean Room, and is enforced the Logical Device Element Manager for a User Interface Element.

WITN11070108

3.3.3.3.2 Volume

There are likely to be of the order of tens of User Interface Elements in existence at any point

3.3.3.3.3 Lifetime

The lifetime of User Interface Element may vary from a second to the lifetime of the application.

3.3.3.3.4 Interactions

A User Interface Element will issue and receive direct calls to and from its owner.

A User Interface Element will issue and receive direct calls to and from the widgets it owns.

At peak rates, calls to and from User Interface Elements may occur in the order of tens per second, but usually the rate will be lower.

3.3.3.3.5 Concurrency

A User Interface Element is a passive capability.

3.3.3.3.6 Configuration

A User Interface Element may be data-driven to varying degrees. It may have a fixed set of parameters that vary its appearance or behaviour. Alternatively it may be completely described in Reference Data.

3.3.3.4 Logical Device Element Manager

3.3.3.4.1 Description

Logical Device Element Managers represent the capability to create User Interface Elements.

Logical Device Element Managers are created by the Logical Device Manager in one-to-one correspondence with Logical Device Widget Managers.

On request from a consumer, a Logical Device Element Manager constructs a User Interface Element. It provides the User Interface Element with a reference to its corresponding Logical Device Widget Manager capability and instructs it to construct its initial representation.

A Logical Device Element Manager enforces a policy regarding the combination of User Interface Elements on a logical device. For example an Logical Device Element Manager for the screen might support the layering of User Interface Elements to create a windowing effect.

A Logical Device Element Manager for the screen represents in part the capability to enforce the layout policies defined by the Clean Room.

3.3.3.4.2 Volume

The application will include the capabilities of one Logical Device Element Manager per logical device. There will be as many or fewer logical devices than peripherals, so there will be of the order of ten Logical Device Element Managers.

3.3.3.4.3 Lifetime

The lifetime of Logical Device Element Managers will be the lifetime of the application.

WITN11070108

3.3.3.4.4 Interactions

The capabilities of the Logical Device Element Manager will receive direct calls from consumers to create and destroy User Interface Elements. In turn the Logical Device Element Manager will issue direct calls to the Logical Device Widget Manager to create and destroy User Interface Widgets. At peak rate the Logical Device Element Manager will receive calls frequently, of the order of tens per second. Usually the rate will be much lower.

3.3.3.4.5 Concurrency

A Logical Device Element Manager is a passive capability, driven by the behaviour of other capabilities of the system.

3.3.3.4.6 Configuration

Logical Device Element Managers will be configured via Reference Data to control at runtime the set of User Interface Elements that the Logical Device Element Managers can construct.

Logical Device Element Managers are also configured with an element policy against which it may check the behaviour of the current set of User Interface Elements.

The nature of these policies is yet to be determined. However, they are likely to at least specify the mutual exclusion of User Interface Elements. For example, when the Extended Workspace is visible, the *BasketUIE* should not be visible.

3.3.3.5 Logical Device Manager

3.3.3.5.1 Description

A single Logical Device Manager is responsible for constructing Logical Device Widget Managers, Logical Device Element Managers and the one-to-one relationship between them, in response to notification from the Peripheral Manager that the set of peripherals has changed.

The capabilities of Logical Device Widget Managers supporting the set of known peripherals for the counter will be required. Logical Device Element Managers can be generic.

Several peripherals may be combined into a single logical device. For example, the touch-screen and LIFT keyboard could be regarded as a single terminal device.

Logical devices can potentially be aligned with the capabilities of underlying implementation technologies, such as Java Swing.

3.3.3.5.2 Configuration

None required.

3.3.3.5.3 Volume

There will be a single Logical Device Manager capability in the application.

3.3.3.5.4 Lifetime

The lifetime of the Logical Device Manager will be the lifetime of the application.

WITN11070108

3.3.3.1.5 Interactions

The Logical Device Manager will receive direct calls from User Interface Agents to obtain sessions with Logical Device Element Managers. These will occur on the order of tens per minute.

3.3.3.1.6 Concurrency

The Logical Device Manager is a passive capability.

3.3.4 Clean Room Compliance Statement

As of 20-Oct-2006, the application architecture is compliant with the prescriptions provided by the Clean Room in the document *HNG-X User Interface Construct Guidelines (DEV/MGT/MAN/0001)*, as follows:

Subsection 3.1:

> *'Constructs should be made up of sub-constucts.'*
>
> The architecture provides the capabilities of User Interface Elements and User Interface Widgets, corresponding to constructs and sub-constructs.
>
> *'All constructs should be implementable in Swing'*
>
> Swing is consistent with the Peripheral Operational Units / User Interface Widgets / User Interface Elements architecture proposed in this document.

Subsection 3.2:

> *'All constructs should be generic…'*
>
> User Interface Elements may be data-driven, and hence generic.

Subsection 3.3:

> *'There may be more than one construct displayed on the screen at once. On occasion these may need to interact… properties should be defined on the constructs to define what functions can be performed on other constructs'*
>
> Interactions between constructs are possible, and depend to some extent on their type. Interactions between constructs are mediated by the Logical Device Element Managers, and also by controlling processes implemented in the interaction and business logic subsystems.

Subsection 3.4:

> *'Peripheral input has two formats: Passive case…; Active case, where the peripheral input … causes a new action to happen.'*
>
> Realistically there is no difference between these cases. Even in the passive case some action normally has to be taken such as updating a field representation on screen. However, the architecture is conformant in that User Interface Elements may communicate with their controlling User Interface Agents in response to events.

Subsection 3.5:

> This section requires further elaboration, but appears to be special applications of the provisions in the prior subsections. Hence, the architecture is conformant.

Section 4:

> *'Constructs include reference(s) to one or more of the lower-level Sub-constructs in the context of the construct.'*
>
> See the User Interface Element capability.

WITN11070108

'Constructs should not be nestable'

User Interface Elements may not communicate between themselves. They are hence not nestable. Policy for arranging User Interface Elements visually may be enforced by the relevant Logical Device Element Manager.

Section 5:

'Sub-constructs are reference by higher-level constructs'

See section 4, above.

'Sub-constructs should identify the types of input and output fields'

User Interface Widgets are of either Input, Output or I/O types. These types are presumed to provide information on the I/O capabilities of the User Interface Widgets.

WITN11070108

3.4 Interaction Subsystem

3.4.1 Overview

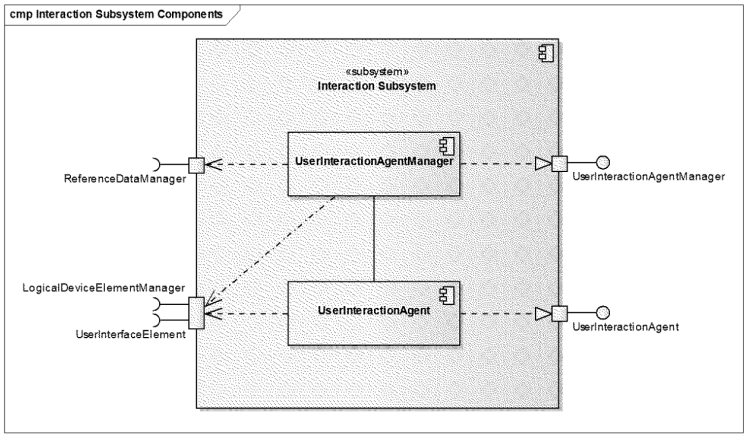


Figure 7: Interaction Subsystem Overview

The Interaction Subsystem encapsulates capabilities relating to interactions between the counter and the user.

Interactions of particular types are managed by User Interaction Agents. Each User Interaction Agent manages a single user interaction.

User interactions are short, potentially reusable interactions with the user.

A single User Interaction Agent Manager encapsulates the capability to create User Interaction Agents.

A User Interaction Agent always has an owning capability, which is the capability that requested the creation of the User Interaction Agent. This owning capability may be another User Interaction Agent. The owning capability is responsible for causing the User Interaction Agent to destruct.

User Interaction Agents can interact with each other. A User Interaction Agent obtains references to other User Interaction Agents by creating new User Interaction Agents, receiving references from its owner or obtaining references from a User Interaction Agent to which it already has a reference.

For further information, refer to the *HNG-X Counter Applications: Interaction Subsystem High Level Design (DES/APP/HLD/0040)*.

WITN11070108

3.4.2 Subsystem Dependencies

User Interaction Agents communicate with the Presentation Subsystem's Logical Device Manager to discover Logical Device Element Managers. They communicate with Logical Device Element Managers directly to obtain User Interface Elements, which they own. Only the User Interaction Agent owning a User Interface Element can communicate with the User Interface Element.

The Interaction Subsystem also interfaces with the Reference Data Subsystem to obtain configuration information for User Interaction Agents.

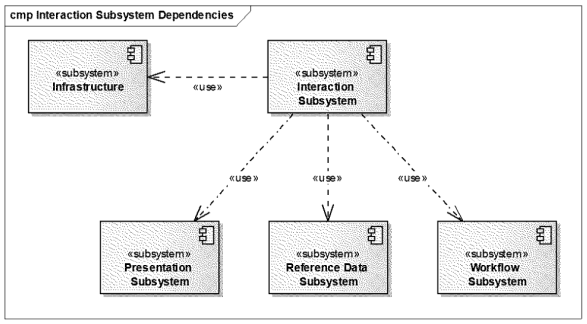


Figure 8: Interaction Subsystem Dependencies

3.4.3 Components

3.4.3.1 User Interaction Agent Manager

3.4.3.1.1 Description

A single User Interaction Agent Manager encapsulates the capability to create User Interaction Agents.

3.4.3.1.2 Volume

There will be a single User Interaction Manager capability in the application.

3.4.3.1.3 Lifetime

The lifetime of the User Interaction Manager is the lifetime of the application.

3.4.3.1.4 Interactions

The User Interaction Manager will be accessed directly or via notifications from consumers and other User Interaction Agents. At peak times this may occur several times per second, but the rate will usually be much lower.

3.4.3.1.5 Concurrency

The User Interaction Manager is a passive capability.

WITN11070108

3.4.3.1.6 Configuration

The User Interaction Manager is configured with Reference Data to enable User Interaction Agents to be instantiated based upon dynamic specification.

The set of User Interaction Agent capabilities that the counter provides can vary according to the Reference Data used.

The User Interaction Manager therefore encapsulates the capability to create User Interaction Agents based on an identifier for the type of User Interaction Agent required and any additional supporting parameters required.

For example, the User Interaction Manager may be capable of implementing User Interaction Agents that are expressed as a PDL script. The specification would identify the PDL script for the User Interaction Agent. The User Interaction Manager would instantiate a generic User Interaction Agent capability and configure it with the PDL Script.

3.4.3.2 User Interaction Agent

3.4.3.2.1 Description

Interactions of particular types are managed by User Interaction Agents. Each User Interaction Agent manages a single user interaction. User interactions are short, potentially reusable interactions with the user alone.

A User Interaction Agents is created by the User Interaction Agent Manager on behalf of another capability, which becomes the owner of the User Interaction Agent.

3.4.3.2.2 Volume

There will be of the order of tens of User Interaction Agents active in the application at any given time.

3.4.3.2.3 Lifetime

The lifetime of User Interaction Agents will vary from a few seconds to the lifetime of the application.

3.4.3.2.4 Interactions

User Interaction Agents may be accessed by direct calls from consumers, User Interaction Elements and other User Interaction Agents. This may occur at a rate of tens of times per minute.

3.4.3.2.5 Concurrency

User Interaction Agents may be passive or active capabilities. This will depend on the type and function of the User Interaction Agent.

If the User Interaction Agent can be implemented to be driven entirely by other active capabilities then this should be preferred.

3.4.3.2.6 Configuration

Much of the behaviour of User Interaction Agents may be configured through Reference Data, and will verify according to the purpose and function of the individual User Interaction Agent.

WITN11070108

3.5 Business Logic Subsystem

3.5.1 Overview

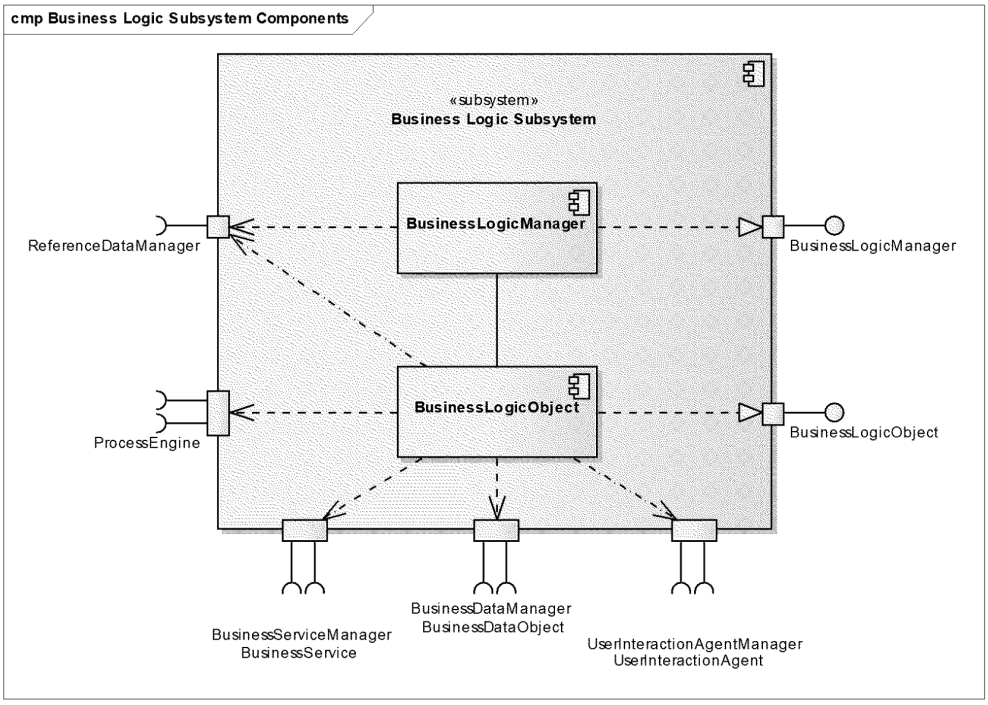


Figure 9: Business Logic Subsystem Overview

The Business Logic Subsystem encapsulates the control capabilities of the application.

A Business Logic Object represents the capability to manage some process or business transaction. For example a top-level Business Logic Object may manage the initial configuration of the application. A lower-level Business Logic Object may manage all activities associated with the sale of a particular product, from the creation of the requisite User Interaction Agents to the invocation of Business Services to interact with the data centre.

The capabilities represented by Business Logic Objects may be implemented to some extent by the capabilities of the Process Engine Subsystem.

The capability to create Business Logic Objects is represented by a single Business Logic Manager.

WITN11070108

3.5.2 Subsystem Dependencies

Business Logic Objects leverage User Interaction Agents for user interaction. One Business Logic Object can use a number of User Interaction Agents and the use is exclusive (one User Interaction Agent can be associated to only one Business Logic Object).

Business Logic Objects leverage and operate upon the business data provided by the Business Data Subsystem. However, all aspects of the lifecycle of business data remain fully the responsibility of the Business Data subsystem.

Business Logic Objects leverage the services provided by the Business Services Subsystem. Reusable computational logic underpinning Business Logic Objects is implemented as discrete services in the Business Services Subsystem. The Business Services Subsystem also provides access to remote services.

The Business Logic Subsystem also interfaces with the Reference Data Subsystem to obtain configuration information for the Business Logic Manager and Business Logic Objects.

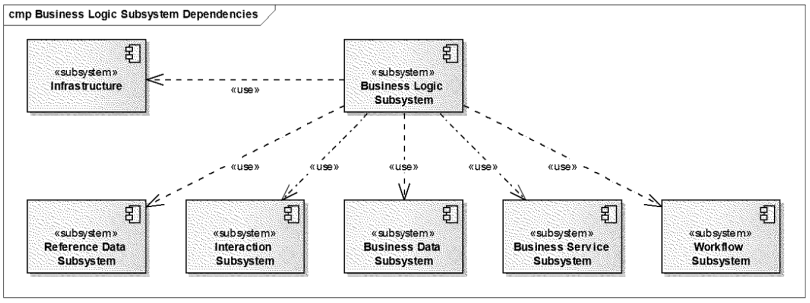


Figure 10: Business Logic Subsystem Dependencies

3.5.3 Components

3.5.3.1 Business Logic Manager

3.5.3.1.1 Description

A single Business Logic Manager encapsulates the capability to create Business Logic Objects.

It may retrieve configuration data from the Reference Data Subsystem for Business Logic Objects.

3.5.3.1.2 Volume

There will be a single Business Logic Manager capability in the application.

3.5.3.1.3 Lifetime

The lifetime of the Business Logic Manager is the lifetime of the application.

WITN11070108

3.5.3.1.4 Interactions

The Business Logic Manager will be accessed directly or via notifications from Business Logic Objects. At peak times this may occur several times per second, but the rate will usually be much lower.

3.5.3.1.5 Concurrency

The Business Logic Manager is a passive capability.

3.5.3.1.6 Configuration

The Business Logic Manager is configured with Reference Data to enable Business Logic Objects to be instantiated based upon dynamic specification.

The set of Business Logic Objects that the application provides can vary according to the Reference Data used to configure the application.

The Business Logic Manager therefore encapsulates the capability to create Business Logic Objects based on an identifier for the type of Business Logic Object required, and a specification, where each specification is Reference Data encapsulating the knowledge of how to instantiate the Business Logic Object.

For example, the Business Logic Manager may be capable of instantiating Business Logic Objects that are expressed as a PDL Script. The specification would identify the PDL Script for the Business Logic Object. The Business Logic Manager would instantiate a generic Process Engine BLO and configure it with the PDL Script.

3.5.3.2 Business Logic Object

3.5.3.2.1 Description

A Business Logic Object represents the capability to manage some process or business transaction. For example a top-level Business Logic Object may manage the initial configuration of the application. A lower-level Business Logic Object may manage all activities associated with the sale of a particular product, from the creation of the requisite User Interaction Agents to the invocation of Business Services to interact with the data centre.

A Business Logic Object enables access to:

- The User Interaction Agent Manager;
- Any User Interaction Agents that it owns or to which it has a reference;
- The Business Data Manager;
- Any Business Data Objects that it owns or to which it has reference;
- The Business Service Manager;
- Any Business Services that it owns;
- The Business Logic Manager;
- Any Business Logic Objects that it owns, or has reference to, and its owner;
- The Reference Data Manager;

WITN11070108

3.5.3.2.2 Volume

There will be of the order of tens of Business Logic Objects active in the application at any given time.

3.5.3.2.3 Lifetime

The lifetime of Business Logic Objects will vary from a few seconds to the lifetime of the application.

3.5.3.2.4 Interactions

Business Logic Objects may be accessed by direct calls from other Business Logic Objects, and by notifications from the objects it consumes. This may occur at a rate of tens of times per minute.

3.5.3.2.5 Concurrency

Business Logic Objects may be passive or active capabilities. This will depend on the type and function of the Business Logic Object.

If the Business Logic Object can be implemented to be driven entirely by other active capabilities then this should be preferred.

3.5.3.2.6 Configuration

Much of the behaviour of Business Logic Objects may be configured through Reference Data, and will verify according to the purpose and function of the individual Business Logic Objects.

WITN11070108

3.6 Business Data Subsystem

3.6.1 Overview

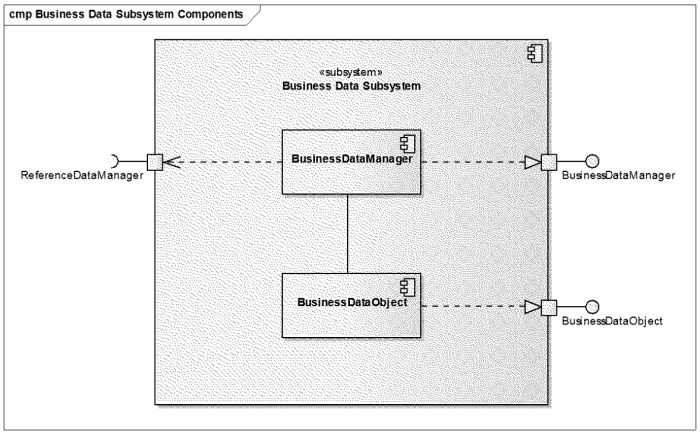


Figure 11: Business Data Subsystem Overview

Business Data Objects encapsulate the capability of the application to provide reusable which manage business state on behalf of Business Services and Business Logic Objects.

The Business Data Manager provides the capability to create Business Data Objects. The Business Data Manager has the capability to interact with the Reference Data Manager to retrieve configuration information and product information to instantiate Business Data Objects. Business Data Objects are generally java types.

WITN11070108

3.6.2 Subsystem Dependencies

The Business Data Subsystem interfaces with the Reference Data Subsystem to obtain configuration information some Business Data Objects.

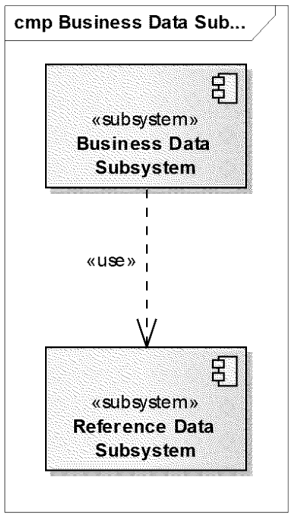


Figure 12: Business Data Subsystem Dependencies

3.6.3 Components

3.6.3.1 Business Data Manager

3.6.3.1.1 Description

A single Business Data Manager encapsulates the capability to create Business Data Objects.

It may retrieve product and configuration data from the Reference Data Subsystem to configure Business Data Objects.

3.6.3.1.2 Volume

There will be a single Business Data Manager capability in the application.

3.6.3.1.3 Lifetime

The lifetime of the Business Data Manager is the lifetime of the application.

3.6.3.1.4 Interactions

The Business Data Manager will be accessed directly or by notification from Business Logic Objects. At peak times this may occur several times per second, but the rate will usually be much lower.

WITN11070108

3.6.3.1.5 Concurrency

The Business Data Manager is a passive capability.

3.6.3.1.6 Configuration

The Business Data Manager encapsulates the capability to create Business Data Objects based on an identifier for the type of Business Data Object required.

Business data objects can be defined in PDL or as Java objects. Business Data Objects tend to be defined as Java objects. PDL based Business Data Objects can be dynamically configured through the use of the Reference Data Download.

3.6.3.2 Business Data Object

3.6.3.2.1 Description

Business Data Objects encapsulate the capability of the application to provide reusable assistance in accessing and gathering business data, and managing transactions to the developers of Business Logic Objects.

This data is expressed as a set of values. A value may be some primitive data or a lightweight object produced by the programming platform. A value may also be another Business Data Objects. Hence Business Data Objects may be nested. However, Business Data Objects do not interact programmatically.

3.6.3.2.2 Volume

There may be of the order of hundreds of Business Data Objects in the system at any given time. Care should be taken to keep the amount of product data accessed concurrently and session data to a minimum.

3.6.3.2.3 Lifetime

Business Data Objects may be associated with an Application Session, a User Session, or a Customer Session, or may be transient in nature and associated with the lifecycle of a Business Logic Object.

Business Data Objects associated with an Application Session with have the lifetime of the application, as they represent data captured associated with the operation of the application.

Business Data Objects associated with a User Session may have a lifetime from minutes to the lifetime of the application, as they represent data captured associated with the currently logged in user, but excluding customer transaction data.

Business Data Objects associated with a Customer Session may have a lifetime likely to last for tens of minutes at most, as they will represent data gathered while interacting with post-office customers, or performing administrative tasks on the counter.

3.6.3.2.4 Interactions

At peak rate a Business Data Object may be updated or accessed several times per second. Usually it will be much lower than this however.

WITN11070108

3.6.3.2.5 Concurrency

Business Data Objects may be active or passive capabilities. Passive capabilities should be preferred if possible.

3.6.3.2.6 Configuration

Much of the behaviour of Business Data Objects may be configured through Reference Data, and will verify according to the purpose and function of the individual Business Data Objects.

WITN11070108

3.7 Business Services Subsystem

3.7.1 Overview

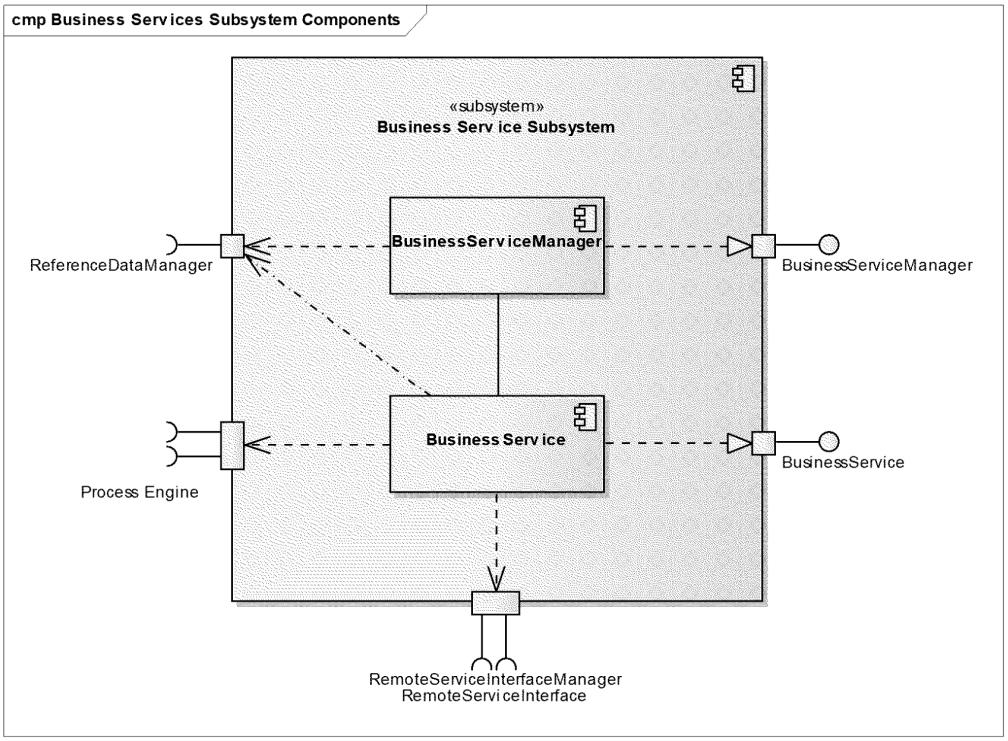


Figure 13: Business Services Subsystem Overview

The business services subsystem encapsulates the capacity of the application to provide reusable services in support of Business Logic Objects, to encapsulate remote services with local services and to provide location transparency.

Business Services are discrete pieces of functionality that may be used by a Business Logic Object. Each Business Service may be used concurrently by exactly one Business Logic Object.

A single Business Service Manager provides the capability to instantiate Business Services.

Business Services may:

- Encapsulate functionality to interact with remote services.
- Be configurable using reference data.

They will always be exactly one of local or remote.

A remote business service encapsulates a remote service.

Business services may be configurable. A business service may be implemented in PDL.

WITN11070108

3.7.2 Subsystem Dependencies

The Business Service Subsystem interfaces with the Reference Data Subsystem to obtain configuration information for the Business Service Manager and Business Services.

The Business Service Manager interacts with the Remote Services Interface Manager to obtain Remote Service Interfaces to associate with remote Business Services. Remote Business Services then interact directly with the Remote Service Interfaces.

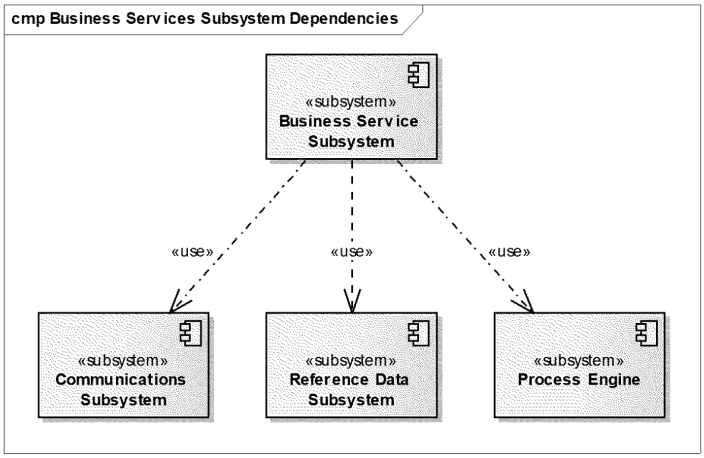


Figure 14: Business Services Subsystem Dependencies

3.7.3 Components

3.7.3.1 Business Service Manager

3.7.3.1.1 Description

A single Business Services Manager encapsulates the capability to create Business Services.

It may retrieve configuration data from the Reference Data Subsystem for Business Services.

It may obtain Remote Service Interfaces from the Communication Subsystem for Remote Business Services.

The Business Service Manager maintains no state.

3.7.3.1.2 Volume

A single Business Service Manager encapsulates the capability to create Business Services.

3.7.3.1.3 Lifetime

The lifetime of the Business Service Manager is the lifetime of the application.

WITN11070108

3.7.3.1.4 Interactions

The Business Service Manager will be accessed directly and via call-backs from Business Logic Objects.

3.7.3.1.5 Concurrency

The Business Service Manager is a passive capability.

3.7.3.1.6 Configuration

Business services can be defined in Java and PDL.

The Business Service Manager encapsulates the capability to create Business Services based on an identifier for the Business Service required.

3.7.3.1.7 Description

Business Services are discrete pieces of functionality that may be used by a Business Logic Object. Each Business Service may be used concurrently by exactly one Business Logic Object.

Business Services do not interact with other Business Services.

Business Services may:

- Encapsulate functionality to interact with Remote Service Interfaces.
- Be configurable using Reference Data.

They will always be either local or remote.

A Remote Business Service encapsulates a remote service. Business Services may encapsulate state.

3.7.3.1.8 Volume

There may be tens of Business Services in existence in the application at any time.

3.7.3.1.9 Lifetime

Business services may have lifetimes ranging from fractions of a second to the lifetime of the application

3.7.3.1.10 Interactions

Business Logic Objects interact with services, using synchronous calls or call-backs, either directly or the process engine. At peak rates, there may be several interactions per second with a single service. Usually the rate will be lower.

3.7.3.1.11 Concurrency

3.7.3.1.12 Implementation

A range of implementation options exist for local services, depending on the extent to which they are data-driven.

At one extreme, the capabilities represented by a particular type of local service may be hard-coded in the application without additional parameterisation.

WITN11070108

More flexibility can be obtained by specifying that some parameters for a hard-coded service should be obtained from reference data.

A large degree of flexibility to modify the behaviour of a local service can be achieved by implementing some of the behaviour using the capabilities of the Process Engine.

Passive services should be preferred, to reduce the load on platform resources, unless the processing that they perform is likely to take a long time, and it is preferable to implement call-back mechanism.

Active services are also possible but not preferred.

3.7.3.1.13 Configuration

Remote services are not configurable

See the implementation section above for a discussion of local services.

3.7.4 Design and Implementation Guidelines

Services provide location transparency as far as the client is concerned (all services appear to be local), but careful consideration of the multiplier principle for the service realisation is required.

WITN11070108

3.8 Reference Data Subsystem

3.8.1 Overview

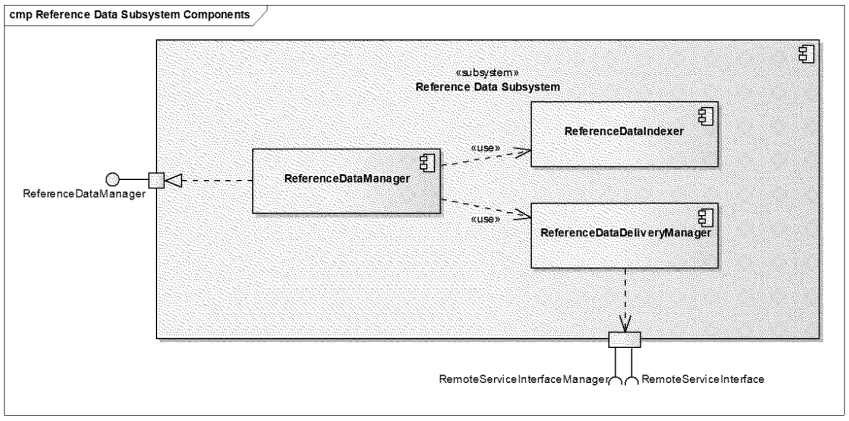


Figure 15: Reference Data Subsystem Overview

The reference data subsystem encapsulates the capabilities to provide a data-centric view of the reference data used by the application, to retrieve reference data from a remote source, and to store and retrieve reference data from a local repository.

A single Reference Data Manager (RDM) encapsulates the operational capabilities of the subsystem. It exposes two interfaces. Its main interface is used for the retrieval of reference data by other counter subsystems. Its control interface may be accessed by Business Logic Objects to control the staging of reference data into the counter.

Because reference data upgrades may result in inconsistent data or behaviour in the counter, BLOs may also cache subsets of the reference data. This is used by areas which need to maintain a value in reference data for some time. For example, the bureau subsystem must maintain a spot rate value between quote and transaction.

The reference data files delivered contain both current and future dated reference data. The Reference Data Subsystem pre-processes the reference data files to obtain the current set of reference data for use by the counter applications.

The application will continue to use an existing reference data set until new reference data is made available by the RDM and the application is in a state to accept it.

A number of configuration and specification data types have been defined in other subsystems. These will be required in addition to data structures supporting the various product types that the counter application will support.

The detailed structure of this subsystem will be defined to a large extent by the overall data-model for the counter.

WITN11070108

3.8.2 Subsystem Dependencies

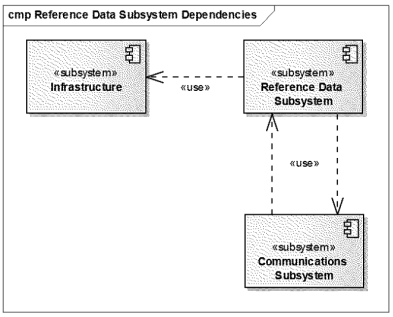


Figure 16: Reference Data Subsystem Dependencies

The RDM interacts with manager components from a number of other subsystems.

3.8.3 Components

3.8.3.1 Reference Data Manager

3.8.3.1.1 Description

A single RDM encapsulates the capability to provide a data-centric view of reference data to other capabilities of the counter application.

Via its control interface, the RDM may be asked to update the current reference data set. All data will be updated immediately.

3.8.3.1.2 State

The reference data manager may maintain state to cache reference data in memory.

3.8.3.1.3 Behaviour

The reference data manager can respond to direct calls to:

- Retrieve reference data
- Refresh reference data from reference data sources

The RDM will interact with other subsystems to provide reference data structures as required.

When providing a reference data structure, the RDM may first retrieve the raw reference data required from the local repository, thus avoiding the need to maintain all reference data in memory. This reflects the capability of the counter application to use its memory resources efficiently.

When RDM refreshed data, the RDM will process the reference data, which contains both current and future dated reference data, to obtain the current set of reference data for use by the counter applications.

WITN11070108

3.8.3.1.4 Volume

A single RDM encapsulates the capability to provide a data-centric view of reference data to other capabilities of the counter application.

3.8.3.1.5 Lifetime

The lifetime of the RDM is the lifetime of the application.

3.8.3.1.6 Interactions

Configuration data may be required whenever a configurable capability is created. Since many capabilities will be configurable, peak rates of interaction with the RDM may be up to tens of times per second.

3.8.3.1.7 Concurrency

The RDM is a passive capability. However, the RDM may receive concurrent requests from other active capabilities.

The RDM should be designed to support concurrent processing as much as possible.

3.8.3.1.8 Implementation

At peak rates the RDM is likely to receive heavy loading.

Mutual exclusion of RDM requests should only be required to a limited extent, as RDM requests will tend to target different reference data.

Refreshing of reference data, which is likely to be an expensive operation, typically occurs at non-peak times, e.g. overnight. However, where a counter has been offline overnight, or there is an emergency distribution of an update, the refresh can occur whilst the application is active.

Reference data caching should be done in a bounded amount of memory. Caching should support temporal locality to increase effectiveness. One possibility would be for the RDM to share configuration data with several active capabilities. Therefore, it would be possible to create new capabilities of a type already active without increasing the memory required to store reference-data configuration.

3.8.3.1.9 Configuration

The amount and policy for caching reference data should be configurable in the RDM.

WITN11070108

3.9 Process Engine

3.9.1 Overview

The Process Engine provides the facility for the counter to interpret and execute business scripts. The scripts are defined in a Process Definition Language (PDL). PDL was developed for the HNGx counter by Fujitsu. The more process based use cases have in general been delivered using PDL.

Changes in the Java application are delivered through an application rollout to the live estate. As PDL is an interpreted language it is delivered to the counters via the reference data delivery method. This is faster than an application rollout.

Therefore the presence of a process engine alleviates the need for the full code base to be delivered to a counter to effect certain business changes.

3.9.2 Subsystem Dependencies

The process engine is used by the Business Logic subsystem and the UI agents.

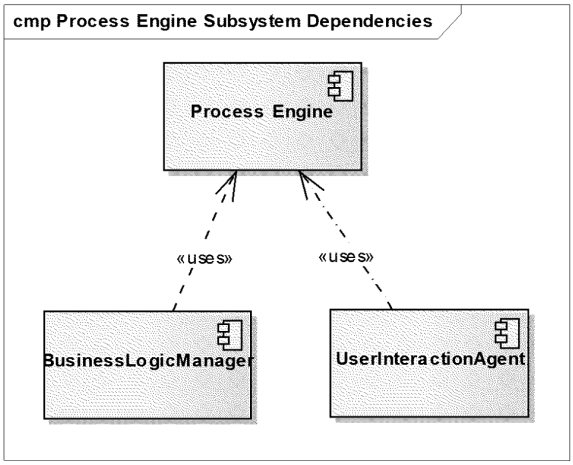


Figure 17 Process Engine Subsystem Dependencies

3.9.3 Components

The core components of the process engine are the scripts, the interpreter and a script context. The script context provides a reference to any state relevant to the script at execution time.

WITN11070108

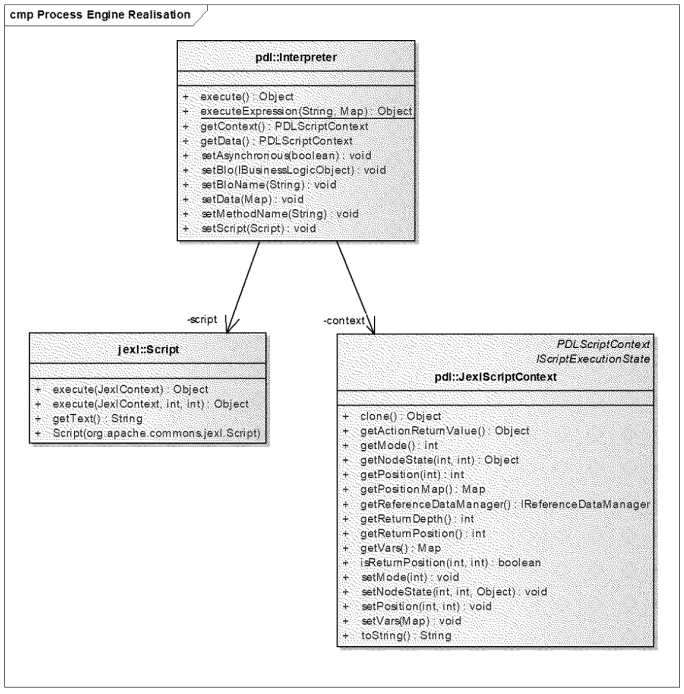
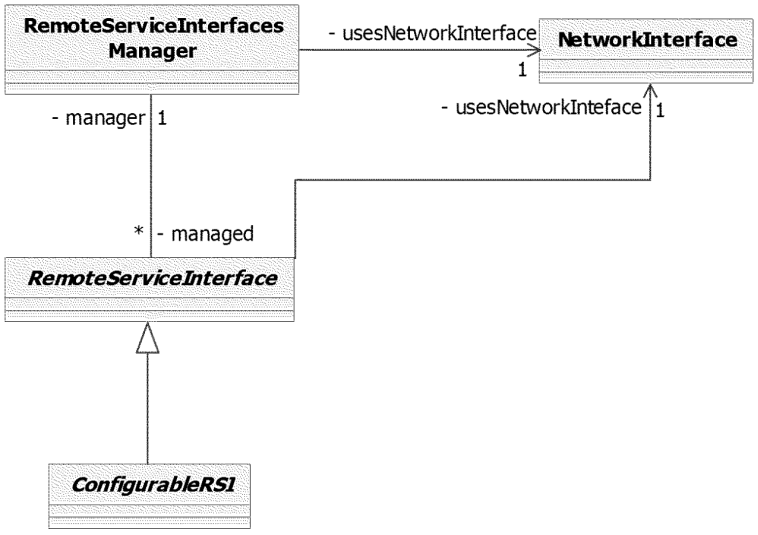


Figure 18 Process Engine

3.10 Communication Subsystem

3.10.1 Overview



WITN11070108

Figure 19: Communication core

The communications subsystem encapsulates the capability to access remote services via a network interface. The communications subsystem is the only subsystem with the capability to use network interface hardware.

The capabilities of the subsystem are distributed across a number of Remote Service Interfaces (RSIs), a Remote Service Interfaces Manager (RSIM) and a network interface component.

Remote Service Interfaces provide stubs for remote services, for example services exposed by the Branch DB.

RSIs can be configured with some policy. The specific policy will be determined at design time and depends to some extent on the type of the remote service.

The set of RSIs required may vary during the lifetime of the counter application. Therefore, RSIs are created based on a list of specifications provided as reference data.

3.10.2 Subsystem Dependencies

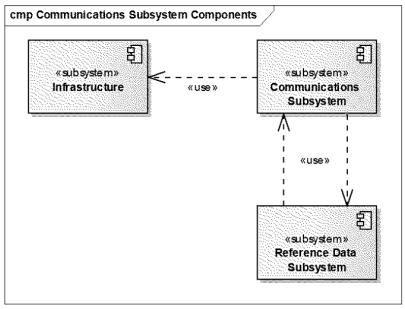


Figure 20: Communications Subsystem Dependencies

RSIs may be accessed via the reference data subsystem, remote business services (represented by RemoteBS in Figure 20) and transfer BDOs. Not all RSIs are represented as remote business-services.

3.10.3 Components

3.10.3.1 Remote Service Interfaces Manager (RSIM)

3.10.3.1.1 Description

A single RSIM encapsulates the capability to create and manage RSIs.

The RSIM receives requests for RSIs.

3.10.3.1.2 State

The RSIM maintains state to:

- List the RSIs that exist and maintain a count of the capabilities referencing them.

3.10.3.1.3 Behaviour

The RSIM may receive direct calls from various manager capabilities to create an RSI if required.

WITN11070108

3.10.3.1.4 Design-time refinements

None required.

3.10.3.1.5 Volume

A single RSIM manages remote service interfaces.

3.10.3.1.6 Lifetime

The lifetime of the RSIM is the lifetime of the application.

3.10.3.1.7 Interactions

The capabilities of the RSIM will be accessed in the order of a few times per second at peak rate, as capabilities are created that require access to RSIs. However, normally the rate will be much lower.

3.10.3.1.8 Concurrency

The RSIM is a passive capability.

3.10.3.1.9 Implementation

No advice.

3.10.3.1.10 Configuration

It will not be suitable to issue multiple references for all services. Therefore the RSIM may be configured to restrict access to RSIs.

3.10.3.2 Remote Service Interface (RSI)

3.10.3.2.1 Description

A remote service interface encapsulates a remote service.

3.10.3.2.2 State

A RSI maintains no state.

3.10.3.2.3 Behaviour

RSIs are created by the RSIM and last until no object has a reference to them.

RSI are destructed by the Java garbage collector.

An RSI may react to direct calls from reference-data remote sources, remote business services (local capability of the Service subsystem), and transfer BDOs to:

- Make remote calls to services

An RSI may react to incoming calls from remote services to invoke calls or callbacks on capabilities attached to the interface.

WITN11070108

3.1.1.1.4 Design-time refinements

The specific set of RSI types must be determined with reference to the services architecture for HNG-X.

3.1.1.1.5 Volume

There will be one RSI for each remote service. There may be of the order of hundreds of RSIs.

3.1.1.1.6 Lifetime

An RSI is usually created by the RSIM and remain in memory until no object has a reference to them.

3.1.1.1.7 Interactions

At peak rate, an RSI may be invoked multiple times per second, to communicate with a service. Usually the rate will be much lower than this.

3.1.1.1.8 Implementation

In practice the role of RSIs is likely to be implemented by the middleware solution chosen to communicate with remote data centres. However, the access rules defined in the interfaces section strictly apply.

3.1.1.1.9 Configuration

RSIs can apply communication policies on a per service basis. For example, an RSI could potentially implement logging.

The network interface component encapsulates the capability to apply policies to network interface traffic. For example, the network interface component could enforce the use of SSL communications for all services.

The RSIM is responsible for creating the remote services interfaces.

RSIs are modelled as abstract capabilities. Specific RSI types will depend on the types of remote services made available to the counter application. This will be determined at design time through a reconciliation of this document with the data-centre architecture.

WITN11070108

4 Platforms

The Counter platforms are described more fully within the *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

From the perspective of the Counter Business Application the following sections highlight the major features of the platform.

4.1 Hardware

4.1.1 Counter PC

There are a range of counter hardware platforms and these will potentially extend over time. The business application must be written to be transportable across the range.

The business application must operate within a limited memory budget in order to support the original Fujitsu Siemens ErgoPro x365/400 Personal Computer, which has a 400MHz processor and 256MB RAM.

4.1.2 Peripherals

The following peripherals will be supported by the business applications:

4.1.2.1 Touch Screen

The Touch Screen monitor allows enables both on-screen and keyboard based interaction with the business application.

All Touch Screens are flat panel displays and support a screen resolution of 800 x 600 pixels with 16-bit colour display. This specification this will be assumed by the application.

Higher colour depths are available, but provide inconsistent results across the range of touch screens used in the estate.

4.1.2.2 Keyboard

The 99-key LIFT keyboard incorporates a Magnetic Swipe and Smart Card reader.

The application will support the Magnetic Swipe reader as an input device.

The application will support the Smart Card Reader as an input/output device.

Usage of the Smart Card Reader by the application is dependent on whether there are any Post Office products to be supported that utilise this device. It will however be supported by the engineering test application, “Europa”; Please refer to the *HNG-X Counter Architecture (ARC/APP/ARC/0003)* for further information.

4.1.2.3 Barcode Scanner

The Bar Code Scanner will be supported by the application as an input device.

4.1.2.4 Counter Printer

The Epson TMJ7100 inkjet printer will be supported by the application. It will run in native Epson mode, rather than the Ithica emulation mode support on Horizon counters.

WITN11070108

The previous Ithica printer will be completely retired from the estate prior to the introduction of HNG-X.

The printer has two ink cartridges; one black and one red. The application will support usage of the different colours. The printer is also capable of storing and applying print overlays such as logos.

Individual report and receipt formats will be as specified in the agreed documents (DES/GEN/SPE/0008, DES/GEN/SPE/0009, DES/GEN/SPE/0010, DES/GEN/SPE/0011). All access to the printer will be through logical devices, for example the tally roll on the counter printer will be regarded as a separate logical device from the cheque or slip printing capability. In addition, there will be separate logical devices for certain types of printing, such as postage labels. This will enable dedicated printers to be added more easily in the future.

4.1.2.5 Report Printer

Each branch has a single back office A4 printer, primarily used to print accounting statements and reports. The application will deliver the appropriate reports to this printer based on the report definitions as specific in the agreed documents (DES/GEN/SPE/0008, DES/GEN/SPE/0009, DES/GEN/SPE/0010, DES/GEN/SPE/0011).

More than once type of printer is used, but all operate as the same logical device by emulation of the 'master' printer type. The application is unaware of the individual device type.

The printer is connected to just one counter at each branch. The counter will be fixed (at the first counter in the branch) and cannot be varied at individual branch level.

4.1.2.6 PIN Pad

A PIN Pad is connected to each counter.

The application will support the PIN Pad for use in specific transactions as covered by the business use cases.

Whilst there is a constraint within the infrastructure management software that PIN Pads must not moved between counters, this will not be assumed by the application.

There are different PIN Pads on live and training counters; the business application will need be aware of the difference, and only allow the appropriate type of PIN Pad to be used in its operating environment.

4.1.2.7 Weigh Scales

Electronic weigh scales are fitted to some counters, and can be connected to either one or two counters using separate serial cables. The business application will support weigh scales as an optional device, and will assume that each scale has a separate physical connection to a counter.

The business application will support the different models of weigh scale as defined in *HNG-X Counter Architecture* (ARC/APP/ARC/0003).

4.1.2.8 Exchange Rate Board

Optionally, a counter can be connected to a bureau de change rates board to display current exchange rates to customers.

The application will support the different models of rates boards as defined in *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

WITN11070108

4.2 Software

4.2.1 Operating System

The target operating system for the counter is Windows NT 4.0 Workstation.

Following completion of the HNG-X Release 1 migration, a subsequent operating system upgrade to another operating system may be undertaken.

4.2.2 Other Third-Party Software

The main other third-party software component used by the business application on the counter will be the Java Run Time Environment.

The Java Run Time Environment is provided and supported by Fujitsu and is delivered as part of the Interstage (version 8) Application Client. No additional libraries or components that are supplied by the Interstage Application Client are used by the application.

Use of further third-party software (including open source software) on the counter will be avoided. If any cases are identified where there is a potential commercial benefit, they will be subjected to a business case and risk analysis. This will include evaluation of the contract by Fujitsu Legal dept, and will identify how such software would be supported and effect operation processes. Also approval must be sort from the CISO and be documented in the project risk register and passed and accepted into live as part of the acceptance process.

Any cases must by explicitly approved before being incorporated into the solution.

Other Software (e.g. system management, network, etc) is covered in more detail within the *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

4.2.3 Tuning & Configuration Requirements

The operating system will be configured to minimise the memory footprint in line with the intended usage of the counter purely as a point of sale terminal. All non essential services will be disabled.

The application will be set up within Java to give priority to the foreground processes and in particular, the user interface components.

The priority and frequency of background processes will also be tuned in line with these objectives.

4.3 Interfaces

The *HNG-X Integration Architecture (ARC/APP/ARC/0002)* provides a description of the major interfaces between platforms and software components.

4.3.1 Local interfaces

There are local interfaces used by the application to communicate with the System and Estate Management Applications that reside on the counter.

In particular, SYSMAN is used by the application for the delivery of the common Reference Data – i.e. the main reference data elements that are shared by all branches

WITN11070108

4.1.2 Remote Interfaces

Remote interfaces are encapsulated within the services layer within the business application.

Through these services the application has interfaces with the following components of the HNG-X Solution:

- Interfaces into the Branch Database: these are used for
  - User session management.
  - Obtaining branch specific reference data and fast changing reference data such as bureau spot rates.
  - Storage of all transaction data.
  - Storage for access and update of branch administration data (stock units and allocation, users, passwords and role, etc.)
  - Retrieval and status update of data distributed to branches (e.g. messages, pouch contents)
- Interfaces for Online Services such as
  - Banking
  - Credit / debit cards
  - Mobile phone E-Top Ups
  - DVLA online
  - APOP services, such as postal orders
  - PAF lookup
- Interfaces for training (CTO's only)
  - For transaction storage
  - For emulation of online services

The communications protocols for these interfaces are described within the network section. The detailed description of the data exchange protocols will be covered (when available) in separate interface documents.

WITN11070108

5 Networks

The application relies on a permanently available network connection to the HNG-X data centre. This connection is provided transparently by the branch router. The branch router is responsible for maintaining a reliable connection and use of alternative network paths as appropriate, including use of mobile network when the primary connection fails.

The counter components that interface with the branch router are descried in the *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

The application is responsible for establishing, managing and using an SSL session with the data centre. All application communication with the data centre will be through this SSL connection. The SSL capability is provided by the Java Runtime Environment on the counter.

The SSL connection is terminated at the network layer within the Data Centre; however this is transparent to the counter application.

The application only interfaces directly with the Branch Access Layer within the Data Centre. All messages, even those directed to Online Services, go via the Branch Access Layer.

All interactions from the application with the Branch Access Layer will require a new socket to be opened. The HTTP “Keep-Alive” setting within the application is irrelevant since the Branch Access Layer will force disconnection of the socket once a response has been returned to the application. Each new message can be handled by any server within the Branch Access Layer; session state will not be maintained by the branch access layer.

5.1 Communication Layer Transport & Protocol

The transport for all application messages is TCP/IP.

The protocol employed is XML over HTTPS.

The decision was made not to use a full SOAP implementation due to the performance overhead in both the Counter and Branch Access Layer servers. Note that the Branch Access Layer components see this as XML over HTTP, since the SSL connection is terminated at the Data Centre network layer.

The target response time is within 1 second for the majority of messages from the application. This includes online services, although the response time for these is dependent on performance of the 3rd party systems.

The timeout value for all messages will be in the order of 30 seconds. The values will be soft configurable, and the precise values will be defined during the design phase, and potentially further tuned in the light of pilot and live operation.

The application relies on the resilience provided within the TCP/IP protocol. Above that the application is responsible for handling any failure modes. See the section on branch exception handling within the Recovery and Resilience chapter.

5.2 Branch Session Management

The application is responsible for establishing a User Session with the Branch Access Layer. This establishes a secure token that will be used to sign all messages within the User Session. This User Session is at the application layer within the protocol, and is independent of the network and SSL connections. See the Security section for further details.

WITN11070108

5.3 Polled Activities

The application polls the Branch Database via the Branch Access Layer for the delivery of reference data, both the branch specific reference data and fast changing data such as bureau spot rates.

The application also polls for status changes on messages broadcast to branch staff.

This polling activity occurs outside of an authenticated User Session.

The polling interval will be set to avoid 30,000 counters polling simultaneously.

5.3.1 Branch-Specific Reference Data

This activity takes place overnight when each branch refreshes the branch specific reference data. The amount of data is anticipated to be a small number of kilobytes.

This activity will be scheduled so that the branches are staggered across a one hour period (usually between 02:00 and 03:00).

This activity will potentially be synchronised with the counter housekeeping activity defined in *HNG-X Counter Architecture* (ARC/APP/ARC/0003).

When a counter has been switched off overnight, then this data will be requested during the application start-up sequence when the counter is next switched on.

5.3.2 Bureau Spot Rates

The updating of Bureau Spot Rates takes place during the day. The amount of data is anticipated to be a few hundred bytes.

Although the new spot rates are normally delivered around 08:00 each day, the rates can be changed during the day and hence will be polled for throughout the day. The polling occurs outside a user session, as some counters have rates boards attached which need to be updated upon the new bureau rates becoming active.

This activity will be synchronised so that all counters within the same branch will activate a change at the same time. This will be achieved through counters polling individually for changes, downloading the changed data and then using effectively dates/times on the spot rates that will cause the Reference Data Manager to activate the new spot rates when they become effective, on each counter at the same time.

The poll interval will be set at approximately 20 minutes. The polling of individual branches is staggered across the 20 minute period to spread the network activity and load on the data centre.

Modifications have been made in Release 4 of HNG-X to Bureau Spot rates as defined in (DES/APP/HLD/0858).

5.3.3 Message Broadcast

This activity takes place during the day. The amount of data is anticipated to be a few hundred bytes.

The data is only needed within a user session.

The data requested is specific to the current logged on user.

The poll interval will be set at approximately 20 minutes. The polling of individual branches staggered across the 20 minute period.

Although the data is only needed whilst a user is logged on, the characteristics of polling is sufficiently similar to bureau spot rates to combine the polling mechanism in order to limit the impact on the network and Branch Access Layer components.

WITN11070108

5.1.4 Emergency Reference Data changes

HNG-X Reference Data Architecture (ARC/APP/ARC/0001) also identifies the need to support emergency updates to reference data during the day. This will be implemented by a polling mechanism, and will be combined with the delivery of Bureau Sport Rates / Message Broadcast.

WITN11070108

6 Manageability

The application can be managed remotely. The interface allowing support staff to remotely connect to the counters is described in the *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

There are specific requirements in the Customer Services requirements for the counter application's manageability aspects.

Various specific manageability features have also been described in other sections of this document.

6.1 Problem Alerting

The application adopts a consistent mechanism for handling application detected error conditions.

Wherever appropriate, errors attributable to user actions should be handled within the application and reported to the user. User error messages must use appropriate terms and should wherever possible identify the underlying cause of the problem.

The application must be fully resilient to peripheral failures and provide the user with meaningful descriptions of the problem. Replacement peripherals should not require a software restart to establish a connection with the replacement peripheral unless absolutely necessary.

Exceptions within the application must be trapped and handled appropriately. The application should be highly resilient to all exceptions that may arise.

Problem alerting by the application will be based on standard log levels and will use a common logging framework. These will be agreed and documented during the development phase and will be described in the appropriate support guides. The SYSMAN component will be responsible for forwarding alerts to operational support staff.

The SYSMAN component will apply a filter to the event log and only critical events will be forwarded to the data centre.

To avoid duplicate issues around the same problem, some filtering logic will be applied to the logging framework.

The application must qualify the data associated with alerts raised so that the same alert from different branches can be easily grouped together, but different problems must be clearly separated.

Additional diagnostics for alerts will be placed in standard locations for ease of access by support staff using standard diagnostics retrieval tools.

Some problems (e.g. due to user error) may will only be reported locally so that support staff can view the records if the user logs a call with the help desk, but otherwise they will not be reported centrally.

6.2 Diagnostics & Trace

The application will use a common diagnostics and trace package developed for use by all subsystems of the application.

The trace levels will be configurable and can be dynamically altered by support staff using the remote management interface. Minimum levels of diagnostics will be set and will be always available to support staff. These will be agreed and documented during the development phase and will be described in the appropriate support guides.

Diagnostic logs are held with specific areas and are managed regularly by housekeeping tasks, so that the disk usage is managed within an overall budget.

The application will obliterate any sensitive data within the diagnostic logs as dictated by security requirements.

WITN11070108

6.3 Statistics & Performance Reporting

Statistics and performance reporting will be done in one of two ways:

- Some statistics will be logged as events that are logged to the Branch Database at the same time as the transaction data is being written. Some of this data will be used overnight to produce SLT reports on performance of the system, e.g. performance report on the time taken for the Settlement Transaction.
- Other statistics will be logged locally to the counter disk. These statistics logs can then be retrieved by support staff as needed, e.g. following problems reported by a user to the Help Desk.

There are some specific requirements on logging of transaction times, so the sequences through transaction steps can be recorded. This data will be logged locally and will always be switched on.

These will be made available to the appropriate management and monitoring tools.

The exact metrics to measure and points of measure will be decided on during design phase.

Further details can be found in SVM/SDM/PRO/0017 - Transaction Time Benchmarking Process

6.4 Starting / Stopping Individual Services

The application must provide a management interface by which the application can be started and stopped.

The application will normally be stopped overnight by the housekeeping process, and it will also need to be stopped to enable software updates by SYSMAN.

The interface will provide options to stop only if no user is logged on, and also provide an override of the current session and force a logout with / without contacting the data centre.

6.5 Updating of Help Pages

Help pages will be delivered to the counter as reference data, and will be updated by the Reference Data Manager.

For more information, refer to the *HNG-X Counter Help System High Level Design (DES/APP/HLD/0102)*.

6.6 Reference Data

The application is controlled by reference data. This dictates the products and services that can be transacted at each branch at a specific date, the price / price range of the products, the tokens (barcodes etc.) that map to the products / services and all other business parameters.

In addition, many of the parameters used to control the application are held with Fujitsu controlled reference data.

The SYSMAN component will be used to deliver the common reference data package. This is a single package that is delivered to all branches. Branch specific reference data is obtained by the application from the Branch Database.

Common reference data is normally delivered as deltas from the previous distributed version. However, the application may request a complete refresh of the common reference data package.

WITN11070108

Full details of reference data distribution are provided in the *HNG-X Reference Data Architecture (ARC/APP/ARC/0001)* and specific assumptions and constraints are contained within the CCD *HNG-X Counter Reference Data Delivery - Agreed Assumptions & Constraints (REQ/CUS/STG/0003)*.

The reference data is distributed as temporal data. Each data record has an effective date (either a start or end date). At the start of each day, the application will parse the reference data to identify the revised set of data to be used for that day[1]. The reference data packages are refreshed each day, and each package version has an associated expiry date – i.e. the date after which the specific package must not be used.

The application will always use a complete reference data package. It will use the branch specific reference data that goes with the latest common reference data package, even though more up to date branch specific data may be available.

When the first business user logs on to the system each day[2], the business application will decide whether it has a valid reference data configuration, and if not valid, will inform the user, request the data via SYSMAN and wait for the data to become available.

If a new version of reference data becomes available during a logged on session, the application will wait until an appropriate opportunity such as between customer sessions to put the new version into use.

The detailed format and interfaces between the reference data distribution system and the business application are described in *HNG-X Reference Data Architecture (ARC/APP/ARC/0001)*.

6.7 Counter Training Offices

The CTO branches will be connected to the live network so that they can be kept up to date in terms of software releases and reference data.

Each CTO branch will be marked as such in reference data and within the branch database, so that the data from these branches cannot get mixed up with live data.

Each CTO will operate as any other branch with respect to reference data; consequently Post Office RDS defines which products are supported at each training branch.

The reports and receipts produced at CTO branches will utilise the CTO reference data field to mark the output as VOID an agreed location on the printed output.

Any statistics logged to the data centre from training counters will be held in the training area within the branch database, and will not be used in any SLT calculations.

The application separates training transactions from live transactions, and also separates each counter training session from training sessions at other counters within the same CTO branch.

The application provides a facility for a trainer to reset the training session for an individual counter back to a pre-set position.

Apart from the above, CTO counters will act as normal counters in relation to diagnostics and trace logs, and will be supportable remotely in the same way as a normal branch counter.

6.8 Incident Reporting by Branch Staff

A facility will be provided within the application to enable branch staff to report an incident without needing to make a phone call to the Help Desk.

[1] The reference data distributed to counters will be defined to become effective at a specific date. Times , (not times during the day).

[2] This check is only needed at the start of the business day since the reference data does not change during the day.

WITN11070108

The incident report will be routed via the Branch Access Layer to the Help Desk system, and a response including reference number will be returned to the counter. There is no interface to provide any subsequent feedback or dynamic access to incident progress.

WITN11070108

7 Security

The *HNG-X Security Architecture (ARC/SEC/ARC/0003)* contains general policy statements and some specific security requirements for the counter.

Appendix 1 has a table which shows the types of data held on the counter which have security implications. It also states the location of the data.

The responsibility for providing security is distributed throughout the application rather than being within a separate subsystem.

This section describes the specific responsibilities of the application.

Other aspects of counter security are covered in *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

Note that Java development guidelines and designer guidelines cover specific security requirements when coding in Java. See DEV/APP/WKI/0001 - *HNG-X DEVELOPER GUIDE FOR JAVA DEVELOPMENTS*, and designer guidelines in DES/GEN/MAN/0001 - *HLD Designer Guidelines* for more details. VB coding standards are present in the NB/STD/001 document. C++ or other languages will require their own security guidance.

7.1 Secure Counter Build

The overall counter architecture, including the secure counter build is described in *HNG-X Counter Architecture (ARC/APP/ARC/0003)*.

Other aspects of counter security are covered in the *HNG-X Security Architecture (ARC/SEC/ARC/0003)*.

The application must take responsibility for not providing any loopholes that enable access to the underlying operating system, including any indirect ways of obtaining command line or access to Windows Explorer.

The application will run with the minimum privileges necessary to perform its allotted tasks. It will not run with full administrator privileges.

The application design must identify and block any capabilities that are provided in 3rd party or standard Java components that could be used to gain access to such capabilities.

The application will be started automatically as part of the counter boot up, and will present a User Logon screen sequence to enable branch users to log on to the system.

Ctrl/Alt/Del sequences must be handled by the counter and not be passed to standard Windows GINA. This will be accomplished by proving a customised GINA DLL.

7.1.1 Java Security

The use of Java to provide the counter business application does introduce some potential security vulnerabilities.

7.1.1.1 P-Code

Java is compiled into p-code (commonly referred to in Java as bytecode) as opposed to native machine code. One consequence of this is that it is quite easy to decompile the p-code back into source code.

In the event that the file system of a counter is compromised, it would be extremely simple using open source tools to decompile the counter into source code.

WITN11070108

To mitigate this risk, the counter does not have a Java compiler installed; however, there is still the possibility of code being compiled on another machine and being injected in the event of a counter being compromised.

7.1.1.2 Code Injection

Java applications are traditionally deployed using JAR files. A JAR file is basically a ZIP archive, containing the compiled Java class files, and a manifest.

JAR files can be secured by digitally signing and sealing of their contents. However, this approach offers a false impression of security.

The purpose of signing JAR files is to offer users who download JAR files in an intranet/internet environment to ensure that the downloaded JAR files have not been compromised, and optionally grant additional security privileges.

However, in the event that the JAR files are compromised, it is a trivial task to remove the signing/sealing.

Whilst the application code could also check to ensure that the JARs are signed / sealed, one must assume that in the event of the JARs being compromised, that any verification code would also be removed.

It is possible to override the default Java policy file, and specify additional rules that insist of code being signed. However, one has to assume that if the JAR files can be compromised, that the policy file can also be comprised, making this defence ineffectual.

Currently, no mitigation has been put in place to address this risk.

7.1.1.3 Memory Usage

The Java runtime environment manages memory automatically using a series of different memory spaces and will copy objects between the various spaces as needed.

Whilst this reduces the complexities usually associated with garbage collection, this does result in potentially multiple occurrences of sensitive data existing in memory simultaneously. Overwriting the data in the application will not necessarily cause all occurrences in memory to be updated.

In the event that a memory dump can be triggered, or if memory can be read by a rogue process, this would potentially allow sensitive data to be captured.

Currently, no mitigation has been put in place to address this risk.

7.2 Branch Session Management

The application is responsible for establishing a User Session with the Branch Access Layer. This establishes a secure token that will be used to sign all messages within the User Session. This User Session is at the application layer within the protocol, and is independent of network and SSL connections.

User credentials are checked against the Branch Database. No use is made of underlying OS user management capabilities. The User Session token is established using a protocol that does not pass the User's password directly using the SRP protocol to authenticate the user's credentials.

Refer to the *HNG-X Architecture - Branch Access Layer (ARC/APP/ARC/0004)* for further details of the logon sequence and details of the protocol.

The only communication with the Branch Access Layer outside of an authenticated User Session is for the delivery of reference data.

WITN11070108

The application will log off from the Branch Access Layer at the end of the User Session. The User will be presented with another User Log sequence to re-establish access to the applications.

If the counter system fails during a User Session, or the counter cannot contact the Branch Access Layer to close down the session tidily, then the application will be informed of the earlier failure at next logon at the counter, and will manage the user through a recovery process. See the Branch Exception Handling section of the Resilience & Recovery chapter for further information.

7.3 User Management

The application provides user management capabilities.

Passwords must not be passed in clear across any network. As SSL is terminated at the network layer in Data Centre, it is not sufficient to protect passwords and a secure authentication protocol is required. The SRP protocol has been selected, which allows authentication without passing a password over the network.

Refer to the *HNG-X Architecture - Branch Access Layer (ARC/APP/ARC/0004)* for further details of the logon sequence and details of the protocol.

There are standard password quality controls and expiry rules that must be enforced.

The password rules are described within the business rules associated with the Use Cases.

There are specific requirements on last login time display, legal messages etc that must be displayed. There are also specific requirements on locking of user accounts after failed logon attempts, and interfaces to unlock accounts / reset passwords.

In addition to normal branch-based users, there are some users / roles that are global across more than one branch, e.g. Engineer, Emergency Manager. From the perspective of the counter, during normal operation of the counter, these users are authenticated via the Branch Access Layer in exactly the same way as branch users.

In the event of the network being unavailable, and hence no possibility of authorising a user, the counter will allow an engineer to access network diagnostics provided by CNIM2.

7.4 Screen Locking & Session Inactivity Timers

The application must provide the capability for a user to lock the current session.

After locking the session, the user must re-present the password to unlock the session.

A full audit trail must be maintained. It is also required within the Post Office business Use Case that another user can logon and force the original user's session to be logged out.

The application must provide session inactivity timers and provide a screen locking capability when there has been no activity for a configurable period.

In addition, after a further configurable period, the system must provide a forced log off capability supporting Post Office defined business rules on what happens to any existing customer session data.

7.5 Data Protection

7.5.1 Use of SSL

All application interactions with Branch Access Layer will be protected over the WAN and branch LAN using SSL.

WITN11070108

SSL will be terminated at network layer components within data centre, and hence does not protect messages as far as the Branch Access Layer.

The SSL server certificates are not secret. Separate certificates are used within the live and test environments to provide additional enforcement of the live/test split (particularly in relation to the disaster recover site).

7.5.2 Use of Encryption

Application level encryption of sensitive data will be used where necessary for business reasons.

Encryption keys used by the counter will be automatically generated by the application and will be distributed using secure routes between the counter and the data centre that are established through the log on process. This avoids the need for a central Key Management System that creates and distributes unique encryption keys for each branch.

For release 5 of HNG-X encryption of data gathered at the counter was introduced. This included a 3rd party FIPS2 conformant Java library on the counter (The FIPS JCE provider used in R5 is RSA Bsafe Crypto-J, version 4.0). See DES/APP/HLD/0069 for more details.

7.5.3 Use of Digital Signatures

The main HNG-X assumption is that there will be no centrally managed counter signing keys.

The server certificate for SSL is global across all counters and not unique to each branch.

Digital signatures will be used for distribution of data and software from the Data Centre to the counters.

Digital signatures will be used to sign all messages within a User Session. The key for each User Session will be created dynamically at the start of the User session. The public key will be sent to the Branch Access Layer as part of the logon protocol, the private key will be held in memory on the counter.

7.5.4 Key Management

There will be no key management system to automatically handle the creation and distribution of centrally managed keys to counters / branches.

Any cryptography implemented will need to identify a secure way of distributing private key material without a counter (or branch) specific key distribution key.

7.5.5 Diagnostic Logs & Trace Files

The application must conform to any specific requirements which constrain the data that can be stored to the local hard disc in diagnostic logs and trace files.

For example PCI imposes some specific constraints on the counter application in this area.

7.5.6 PCI

The Payment Card Industry (PCI) standard, which covers credit and debit card transactions, defines two categories of data (Sensitive Authentication Data and Cardholder data) which must be handled appropriately.

- Neither can be written in clear to the local hard disk. This applies to diagnostic logs and trace files as well as storage of transaction data.

WITN11070108

- Both forms of data must be protected when transmitted over the WAN, which is achieved through the use of SSL.
- Sensitive Authentication Data it cannot be stored at all, even if encrypted. This data will be passed in authorisation request messages for the data centre (protected under SSL over the WAN). The data will be in clear except for PIN blocks which will be encrypted by the PIN Pad. It will be held locally in the counter memory in cases where it may be needed for reversals. It will not be transmitted in the Settlement records.
- Cardholder data will be encrypted or rendered unreadable dependent on usage. The data will be passed in clear in authorisation request messages for the data centre (protected under SSL over the WAN). The data centre will return an encrypted form of the data which will be included within settlement records. In addition, a hash of the PAN will be performed for use in Settlement records and local diagnostic logs. In some cases, e.g. receipts the PAN will be masked with asterisks as appropriate.

Similar protection will be applied to both banking and credit / debit cards. The *Security Architecture (ARC/SEC/ARC/0003)* contains an overview of the end to end data flows relating to PCI.

7.6 Audit

The application plays a significant part in providing a solution that meets the audit requirements. The application is the original source of the majority of audit records related to branch activity. It interacts with the Branch Access Layer to write audit records. Whilst some of the following features are policed by the Branch Access Layer, the application must ensure that the audit records that it produces are compliant.

There must be a clear audit trail of actions by branch staff.

The audit trail must show the identity of the User, Branch and terminal.

Audit records must contain data integrity checks.

The audit trail should be tamper proof to the extent that it can be relied on in court for litigation purposes.

The audit data must be implemented so that there are no gaps in the audit trail. The counter can retry to write an audit record when an earlier attempt fails, but must re-write identical data. Failure to write an audit record will be regarded as fatal and the user will be logged off. See the Branch Exception Handling section of the Resilience & Recovery section for further information.

It is acceptable that some ‘trivial’ audit records are held within the application memory and only posted to the data centre at the end of a customer session.

There will be a mechanism to force certain types of audit record to be written in real time to the central systems independently of the settlement transaction. Overall volumes of such audit events must be within an overall agreed value for online transactions

WITN11070108

8 Recovery & Resilience

This section covers the responsibility of the application with respect to resilience and recovery.

Each counter operates independently of other counters in the same branch.

The application relies on a resilient network connection provided by the branch router.

A counter cannot continue to trade if there is no network connection to the data centre.

The resilience of the network connection is covered within other architecture documents.

8.1 Counter Business Application Software

The application needs to be robust and reliable.

All component design and implementation must consider the error conditions that can arise and the way that the business application can handle the error. The scale of the system is such that errors that are “only 1 in a million” will occur several times per week.

The software must be resilient to hardware errors in attached peripherals.

Any exceptions in lower level components need to be trapped and handled within the calling software. Errors need to be logged so that support staff can analyse the problems encountered, however error reporting must consider the potential load on support systems if all counters report errors at the same rate. Some filtering logic will be applied to avoid excessive error logging.

The overall development guideline is that the application does not break and if it does it does so gracefully.

8.2 Branch Peripheral Failures

The application provides alternative means of data entry for input of data where a counter peripheral has failed; e.g. Bar code reader, magnetic card reader, scales.

Note that for some transaction types, the availability of an alternative means of data entry (e.g. fallback to PKE) is controlled by reference data.

The application provides a print preview mechanism to display the intended print layout on screen. The clerk must manually transcribe the data to produce a manual receipt / report in the event of the counter printer failing.

There is no fallback mechanism for the counter printer when printing vouchers (such as postal orders).

There is only one back office printer per site. It is connected to a single (fixed) PC. The application provides a print preview facility to enable user to complete essential business processes in the event that the back office printer is not available.

In the case of touch screen and keyboard failures, the general design target is that everything that can be done via the touch screen can also be performed via the keyboard and vice versa. There are however, occasional exceptions to this rule, for example not all characters can be input via the keyboard.

In the case of the exchange rates board, the fallback mechanism is to print a copy of the current rates via the back office printer.

For PIN Pad failures, there is a potential to fallback to swiped transactions from Chip & PIN. However, this only applies to card payment (and then only as long as permitted by EMV). Failure of the PIN Pad effectively removes the capability to perform banking transactions at the counter. If no other counter is

WITN11070108

available with a working PIN Pad (e.g. Single Counter Office), then banking transactions will not be available at that branch.

Other cases of peripheral failure are effectively the same as failure of the counter base unit, e.g. VDU.

8.3 Branch Exception Handling

This section relates to exceptions that can occur within the branch due to failures in hardware or other software components outside the business application layer.

The main scenarios described are around the failure to communicate successfully between the counter and the data centre systems – in particular, failure to write transaction data to the Branch Database.

The connections to the Branch Access Layer do not use sticky sessions. There is significant resilience within that layer and there is a high expectation that a retry of a message will succeed.

The *Branch Exception Handling Assumptions and Constraints (REQ/CUS/STG/0002)* CCD provides additional detail. The business process surrounding such exceptions is summarised below.

8.3.1 Online Transactions

The counter will connect to the Data Centre for online transactions during customer sessions.

Where the transaction is recoverable, a recovery record will be written to the Branch Database prior to the connection to the authorisation service. Not all online transactions are recoverable – e.g. DVLA (which is a read only interface and conveys no state) is not a recoverable transaction.

There are further forms of online transactions which simply lodge a recovery record and / or an audit event, but require no further processing within the Data Centre; for example, prior to label printing within postal services transactions.

The counter will manage the connections to the Data Centre, timing out the communications based on configured timeout values, and following business processes agreed for the individual transactions.

8.3.2 Settlement

A counter needs to be able to connect to the Data Centre systems at the end of each customer session.

Recovery records will be updated at settlement time to indicate that they are now complete and do not require further recovery.

The counter will manage the connections to the Data Centre, timing out the communications based on configured timeout values.

In the case of settlement transactions, the counter will automatically attempt to resend the transaction data automatically to the Data Centre a configurable number of times. It will also prompt the clerk to decide if further retries are needed. Failure to successfully execute the transaction will result in a forced log out.

8.3.3 Reporting & Other Data Access

The application uses the Branch Database to store persistent business data in addition to the transaction data written during the settlement transaction. Examples are administration functions such as user administration, stock unit administration, pouch details, bar codes allocated to pouches and track & trace.

The reporting capabilities retrieve the report's data from the Branch Database, and also optionally write "cut off" markers to delineate parts of the reporting data that have already been processed.

WITN11070108

The application will query the Branch Database via the Branch Access Layer for data. Where data needs is to be written back to the branch database, the application will do this in a similar way to settlement transactions. The application will use recovery records as appropriate where the resilience characteristics dictate that the results of a specific counter operation must not be lost.

8.3.4 Recovery & Recovery Records

The format of the data written in the recovery records is transaction specific, but there will be some required fields such as user session identifiers, branch & terminal identifiers and transaction identifiers.

The application must construct the recovery records according to the business rules, for example banking and DCS recovery records need to be PCI compliant.

Recovery records will be updated at settlement time to indicate that they are now complete and do not require further recovery.

- It is anticipated that there will be some form of local session level checking to ensure that all recoverable transactions started within the customer session have been completed – and the recovery process started if there is any not completed. This will check against errors at application level.

The recovery process will be performed only on the original counter after a failed session.

The recovery process will be driven by the counter application. The application will obtain from the Data Centre the list of recovery records outstanding, together with the associated recovery record details.

For some online services (banking, credit / debit cards and E-Top Ups) the counter recovery process may enquire from the authorisation agent the outcome of the original authorisation request.

8.3.5 Log On After Failed Sessions

The recovery process will be performed after the first logon at the original counter that failed.

The system will detect a user logging on after a failed session, either any user on the original terminal or the original user on a different terminal – however, recovery will remain outstanding until a user logs on to the original terminal (or its replacement).

The *Branch Exception Handling Assumptions and Constraints (REQ/CUS/STG/0002)* provides additional detail, particularly on the new concept of the “Recovery Receipt”.

8.3.6 Log On When No Network Connection

The system should detect when there is no network connection and make the user aware prior to presenting the log on screen.

The precise form of the user dialogue will be agreed during the design stage.

8.3.7 Audit Records

One of the major audit requirements is that there are no gaps in the audit trail. The solution adopted is that when the write of the audit record has not been confirmed, the application will re-attempt to write the audit records a number of times (under user control). The record contents must not change between each attempt to write the data.

However, if the user confirms that the attempt must be abandoned, then the user session will be terminated by a forced log off.

WITN11070108

This behaviour must be considered within the design of individual business transactions, and appropriate use of recovery records used if the application could reasonably carry on in certain failure scenarios, or where the application needs to record specific state changes that must not be lost.

WITN11070108

9 Performance

The application needs to operate in conjunction with other software at the counter.

Each counter position is dedicated to a single user.

Counters are required to be left powered on overnight to enable software and reference data updates, however in general there is minimal user activity at counter positions overnight. If they are not left on, reboots after power-on are likely to incur delays as updates are performed.

9.1 Parallel Processing

Fujitsu Services have assessed the opportunities for parallel operation to minimise session duration within the context of the HNG-X Architecture and the transaction flows specified within the POL Use Cases. Fujitsu Services do not believe that any optimisations are feasible at this stage but this will be re-assessed in light of operational experience.

The counter is designed in such a way to allow certain operations to be executed in parallel. In some cases the need for parallel processing has been alleviated by other measures. For instance, the implemented faster printer drivers and faster slip printers have improved the speed at which receipts are printed.

9.2 Volumes of Activity

The volume of activity on a counter varies according to the branch type.

A significant percentage of the estate has only a small amount of daily activity. Other branches are very busy. The main peaks are as follows:

- Benefit payments: These peak in the morning, particularly Monday, Tuesday and Thursday
- Postal Services: These peak in the late afternoon throughout the week. Some counter positions are tied up for significant periods processing parcels for eBay distributors. This will typically increase the size of the basket, but not the volume of transactions since many postal items are all added to the same basket.
- Seasonal (Christmas): There is a significant increase in post office business in the run up to Christmas.
- Reports: Branch counter staff produce daily (cut off) reports. There are two main phases – the “Last Post” reports around 16:30 and the “End of Day” reports after 17:00. There is a further peak on Wednesdays around 17:30 for the weekly reports.

9.2.1 Basket Size

The typical basket size is just under 2 transactions per customer session, together with a further settlement transaction.

The data logged to the Branch Database for transactions is higher than this due to some transactions (e.g. postal services or bureau transactions) having more than one detailed transaction record.

In the extreme case, customer sessions have been recorded comprising over 1,000 transaction records.

9.2.2 Data Volumes

The XML records written at the end of a customer session will typically be in the order of kilobytes in size, but in exception cases may be megabytes in size.

WITN11070108

The detailed design of the record structure is ongoing and will be decided within the design phase.

Online messages, both authorisation messages the response messages, vary with each client interface, but all are within a few hundred bytes.

The reference data volumes are as follows:

- Common reference data: Overall size 30 Mbytes; typical delta size 100 Kbytes; max delta size 1 Mbyte.
- Branch specific reference data: Refreshed daily, 4 Kbytes.
- Bureau reference data: daily spot rates 1 Kbyte; infrequent margins data delta 1 Kbyte, max 20 Kbytes.

The reference data comprises the following:

- Up to 5,000 individual products (with associated product modes)
- Up to 2,500 separate token definitions (including dummy tokens used in ADC)
- Up to 100 banking schemes (including Banking, credit / debit cards and E-Top Ups)
- Up to 500 ADC templates / unique scripts (Post Office controlled application scripts)
- Up to 5,000 individual ADC script steps.
- The Postal Services reference data is currently being analysed and will be included in a future version of this document.

Note that the volume of different ADC scripts and associated script steps is growing quite significantly as Post Office are rolling out new ADC based products, and the individual ADC scripts are becoming more complex with grater number of steps.

9.3 Script Engine Characteristics

The PDL processor is the component within the solution that will enable the Post Office and Fujitsu to script the counter business processes.

The PDL processor is only one of the components that will reside on the counter. As a consequence, the actual resources available to it are only a fraction of the total resources.

Accuracy and Speed are paramount.

Accuracy: Under no circumstances are discrepancies allowed between the execution semantics defined by a PDL Script and the actual behaviour/operation.

Speed: Beyond the dependencies on the response time of external applications, the execution of individual PDL Scripts should be as rapid as possible. Operations such as the initiation of new PDL Scripts, the execution of individual steps, and state transitions should be optimised so that performance does not degrade even when multiple PDL Scripts are concurrently active.

At any given time, there will be several hundreds PDL Scripts deployed (at least one for each business use case plus 500+ for AP-ADC). The set of PDL Scripts will be logically structured based on type hierarchy relations.

At any given time, there may be tens of PDL Scripts in execution (at least one main process for the current transaction, plus a small number of sub- processes and other background processes).

WITN11070108

9.4 User Interface Characteristics

The User Interface is one of the major components of the counter that can impact the acceptability of the application. The Java Swing framework has been identified as the technology choice for this component.

This framework has a proven comprehensive GUI, capable of supporting a reactive 'Rich Client', with fully configurable components designed using MVC principals, and supports a programmable "look & feel".

The GUI runs within the JRE, and is therefore platform independent; performance, although good, will be constrained by the Post Office counter hardware (which has a low performance processor by current standards and only 256Mbytes of memory).

The Java Swing framework has been tested with prototype benchmarks to assess the capabilities on the current counter hardware. The overall assessment of the counter performance with Swing applications was good and from this assessment, the following design constraints can be derived:

- The application does not necessarily need to be constrained to very simple Java Swing components.
- Rendering performance is adequate enough to provide a good looking alternative to the Java Swing default Look & Feel implementations.
- Multi-threaded thick clients are performant on the counter hardware, but care should be taken to keep the number-crunching aspects (including many calculations on different rendering techniques available within Java Swing such as alpha channel transparency, anti-aliasing, shadow calculations etc.) to a minimum.
- As it is expected to be loading some (if not all) reference data on start-up, care will be needed to ensure the data loading threads do not reduce the usability of the system (making the user interface slow to respond to operator actions). However, this can be managed and monitored when the full extent of the application is clearer, and a live client can be profiled for performance bottlenecks.

The performance of the user interface layer will be monitored during the design and implementation phase. Optimisation of the user interface will be performed to ensure that it is responsive to the user and not impacted by background processing. The latter point above relates to the impact of reference data loading, whilst this will normally take place overnight, it can also occur at start-up of the application during the day, and will be optimised to reduce the impact on the user interface.

9.5 Operating System characteristics

The application software must operate within an agreed budget for the memory and processor available within the restricted hardware environment. The different operating systems (NT and XP) and different 3rd party components (e.g. Tivoli) will have different memory and processor usage themselves which need to be taken into account.

The design of the application must recognise that it is not the only application running on the counter.

9.6 Specific Performance Improvements for HNG-X

The overall target is that HNG should be at least as efficient as Horizon, and the expectation is that there are some significant improvements.[3]

A range of scenarios have been defined that will be used as the basis for measurement of transaction performance.

[3] Note that this section is left in for historical now that HNG-X is several releases in live.

WITN11070108

9.6.1 Counter Boot-Up Time

This must be considerably faster than Horizon. No specific target has been set, but it must be significantly faster than the 20 minutes it can take to get a Horizon counter ready to accept a log on request.

Under Horizon, it was necessary to synchronise the message store of a counter when it was started. Under HNG-X this is no longer the case, which should yield an appreciable gain in performance.

9.6.2 Token Lookup

There is an expectation that the lookup process will be significantly faster than on Horizon. On Horizon, there is a delay on banking transaction performance related to the number of AP tokens that are declared.

However, since banking is now mostly Chip & PIN the impact of this lookup should be less than half a second.

There is a similar expectation that AP token lookup will be faster. For example, bar code based tokens will be separated from magnetic card tokens; index based token lookup will be used to avoid long list searching.

9.6.3 No AP Branch Copy Receipt

There application will no longer print an AP Branch Copy Receipt.

This eliminates one complete receipt print and the associated Tear Off user prompt.

9.6.4 Session Receipt Header

The assumption is that the header of the session receipt will be printed whilst the settlement transaction is being committed to the data centre.

9.6.5 Faster Printer Driver

The new Epson printer will be driven in native mode rather than emulating the former Ithica slip printer. A faster printer connection speed will be used.

The improved performance of the Epson printer combined with the faster connection speed should reduce print times.

9.6.6 Faster Menu Navigation

Traversing the Horizon menu is slow – the assumption is that HNG-X will be faster, but no precise targets have been set.

9.6.7 Reports

The counter daily reports will be batched together under the headings of “Last Post” and “End of Day” into “Group Reports”.

The revised business process associated with the execution of these reports should be conducted in a shorter time with a reduced number of steps.

WITN11070108

In addition, the data required for these reports will be aggregated to reduce the number of individual requests made to the Branch Database.

It is also expected that the Branch Database queries for the reports will be faster than using Riposte, despite being centralised with resources shared across the whole estate.

9.6.8 UI Usability

The UI will be subjected to usability trials to ensure that it is at least as efficient as Horizon.

9.6.9 Basket Mixes

The overall counter performance benchmark will be measured across a set of typical baskets. There will be some wins and some losses. The main loss is from the settlement interaction with the Data Centre rather than writing the transaction data to the local disk.

9.7 Transaction Benchmarking

The counter will be implemented with a built-in “Transaction Benchmarking” capability.

This needs to be designed to be efficient enough so that it can be left on permanently.

The logging of the performance data will be to local disk as part of the trace mechanism rather than in event messages sent to the Data Centre.

The instrumentation needs to identify where user input is awaited as well as Data Centre response times.

9.8 Settlement: Online Transaction Times

There will be an SLT on Settlement transaction times.

The data will need to be recorded and sent with the next auditable message sent to the Data Centre.

WITN11070108

10 Migration

Details of the overall migration process are documented in the *HNG-X Migration Strategy (ARC/MIG/STG/0001)* and the *HNG-X Migration Strategy - Agreed Assumptions & Constraints (REQ/CUS/STG/0001)* CCD.

The migration of the business application software on the counter is controlled by the Systems Management component (SYSMAN). See *HNG-X System & Estate Management Architecture (ARC/SYM/ARC/0001)*.

10.1 Horizon Counter Migration to HNG-X

This is a branch wide migration. All counters in a branch are migrated at the same time.

Certain pre-migration functions are provided by the Horizon counter application in conjunction with special data centre components and process to support branch migration. This includes pre-population of the branch users, stock unit and pouch details, as well as creation of historical transaction data for the branch in the Branch Database. The *HNG-X Migration Strategy – Agreed Assumptions & Constraints (REQ/CUS/STG/0001)* CCD contains further details.

10.1.1 Horizon Counter Responsibilities

The final details of the business steps that will be supported by the Horizon counter will be confirmed during the design stage. They will be based on the overview in The *HNG-X Migration Strategy – Agreed Assumptions & Constraints (REQ/CUS/STG/0001)*, together with any specific additional requirements documented within the business and systems requirements.

In summary, the following facilities will be provided by the Horizon business applications:

- A new EOD process will produce accumulated totals that will be passed through TPS to the Branch Database. These totals will indicate the precise position of each stock unit each night.
    - This new process will deduce a start position based on the last trading period roll over.
    - Each night the position will be moved forwards for the days transactions (aligned with the EOD marker).
    - On each new trading period roll over, the position derived from the trading period roll over will be compared with the ongoing daily position and alerts raised if different.
    - These EOD processes run for a number of days prior to the actual migration to HNG-X.
- A Branch Pre-Migration process will be produced to prepare the branch for the migration to HNG-X. This will perform a number of steps:
    - Check that a number of pre-conditions have been met; e.g. no other users logged on, no outstanding recovery processes, sufficient time left before EOD, etc.
    - Produce the “Pre-Migration Office Summary Report”, which provides a consolidated position of the branch accounts. This will be based on the office balance snapshot report, enhanced for some specific areas not directly covered. The report is both printed for the branch manager and an electronic copy supplied to the Branch Database for use by the HNG-X Counter Business Application the following day.
    - Produce a migration request package to be sent to the data centre. This includes a list of users and the current mapping to stock units, the set of currently active T&T and already used pouch bar codes, plus other data that needs to be migrated.
    - Ensure that the above is complete before the EOD process runs.

WITN11070108

- Prevent any users from logging on until the whole branch has migrated to HNG-X or has regressed back to Horizon. This will not be before the following morning.

- Put the branch into a position ready to run the SYSMAN migration package.

10.1.2 HNG-X Counter Responsibilities

The overall objective of the migration process is that the branch can carry on trading the day following migration with minimal impact on branch staff other than the new user interface to the application.

The reference data for the branch for the application will be derived from the Branch Database and the common reference data. This must be managed to produce consistent data between the Horizon and HNG-X branches.

The software capability for HNG-X will be controlled by SYSMAN. This needs to be co-ordinated so that no business capability is lost over the migration point.

The application has some specific capabilities that are provided to support the migration process.

One specific capability is the production of the “Post-Migration Office Summary Report”. This is a special report purely for branch migration to HNG-X. It provides a summary of the accounting position of every stock unit and the overall branch balance position. The application also compares the report against the “Pre-Migration” report and highlights and discrepancies. The results of the comparison are reported to the branch manager and to support staff.

Provided no trading has started on the branch, it is also possible for Help Desk support staff to initiate regression of the branch back to Horizon.

At some time post migration the Horizon data that was previously held on the counter will be deleted. The precise timing for this step will be decided during the design phase – it needs to take account of post migration support enquiries as well as the potential regression capability described above.

10.2 New Counter Roll-Out

In addition to the migration of existing counters from Horizon to HNG-X, new branches and counters can be rolled out directly onto HNG-X.

Certain business application capability will be required to support this process.

See HNG-X System & Estate Management Architecture (ARC/SYM/ARC/0001) for further information on the spares strategy.

WITN11070108

11 Testing & Validation

The application must be designed to be testable.

It may be necessary to provide additional interfaces and capabilities specifically so that testing can be automated.

In some cases it is necessary to bypass security measures to allow performance tests to be scaled up. Any such measures must contain protection against accidental (or deliberate) usage in the live environment.

In some cases emulation facilities may be needed to reduce the scope of the test environment.

Individual application components must be developed to independent interfaces that can be validated without needing end to end test rigs.

Tests should be automated so that as much regression testing as is practical can be completed.

The development needs to address any specific testing requirements within the Post Office and Fujitsu requirements sets.

The overall approach to testing, constructing test environments and test governance is documented in the *HNG-X Testing Strategy (TST/GEN/STG/0001)*.

11.1 Testing During the Development Process

Testing must be incorporated into each stage of the development.

11.1.1 Component Testing

Component testing (Unit testing) of programming artefacts should be performed by developers during development (not after) using a framework such as JUnit.

A useful way of monitoring that this is actually being done is to have tools in the build system that periodically report the code coverage as a percentage of code that has associated unit tests.

Unit tests are an essential tool to achieve the following benefits:

- **A suite of automated regression tests**: Unit tests allow developers to change and add to application code, and immediately verify whether they have broken anything previously working by running the suite of unit tests.
- **Protection against staff turnover**: Unit tests protect against dependence on an individual developer that may leave the company or go on vacation. Because code can be quickly regression tested, individual “ownership” of code can be better avoided, and other developers will be able to find the courage to fix, change or refactor code another developer has written.
- **Simplifies integration testing**: Integration testing of components and the application itself becomes easier, as most lower-level details will already have been tested to some degree.
- **“Living documentation” of a system**: The unit tests in themselves define the expected behaviour of the software, and thus becomes a living, always up-to-date document of requirements (if unit tests do not reflect requirements, they either will fail, or have to be changed).
- **Promotes decoupling and good development practices**: since unit tests should be run on a single class in isolation (potentially with mocked up dependencies), unit testing promotes good development practices, such as low coupling, few dependencies, defined object roles and development by interface rather than implementation.

WITN11070108

Furthermore, when new bugs are found, a unit test should be written that tests for that explicit bug. Doing this will in effect give the project an effective regression test suite that protects against the recurrence of bugs that have already been fixed.

11.1.2 Build System & Continuous Integration

Many aspects of testing and validation can be automated through the build system. For instance unit tests and scripted integration, performance and acceptance tests can be added to the build process, meaning that you get an automated regression test suite run on each build of the software.

For further control and enforcement, continuous integration can be used with tools like *CruiseControl*: in effect this can be configured to run an automatic build with tests every 30-60 minutes with the latest code, meaning that any failures or broken builds will be caught early on, rather than at a later date when the problem can be harder to track down.

Furthermore, continuous integration allows us to plug in different tools that enforce policies such as unit testing, for instance tools that report on broken builds, unit test code coverage etc can be plugged in.

It is the recommendation from the architecture team that a rigorous build system using continuous integration be implemented for the software development of the project.

11.1.3 Service Interface Validation Testing

The interfaces of the Branch Access Layer services will be defined as XML Schema Definitions. This means that inputs and outputs from a particular service can be tested. In the case of the application, its outputs need to be validated against the schemas of the services provided by the Branch Access Layer.

The responses provided to the application should be validated against the respective schemas of the services. This can start with validation of test data and subsequently, integration testing will use Branch Access Layer components whose outputs have already been validated.

It is important to get a wide number of possible response permutations out of the system, in order to test the different variations of possible XML responses. Ideally diagnostic capability should be built in so that validation of the inputs and outputs can be performed whilst running workloads through the system (recognising that there will be a performance impact from such diagnostics).

11.1.4 Component Integration Testing

Component Integration Testing is the phase where individual components and services are placed end to end to perform some aspects of the business capability. In the context of the application, this means integration of the application with initially stubbed interfaces for the connections to the Branch Access Layer.

Subsequently, this will include integration between the application and the Branch Access Layer, but may, for example not include the full set of security components or all online interfaces.

Integration testing can be automated and aided to some degree with the help of scripted tests and associated testing tools. The precise set of tools to be used will be decided in conjunction with the testing team.

Integration testing should, just as all types of testing, also test potential failure scenarios.

11.1.5 Performance Testing

Performance testing will test the performance characteristics of the application. It may do this in isolation, since many of the application capabilities may operate locally without connection to the Data Centre, or may use stubbed interfaces for interactions with the Branch Access Layer.

WITN11070108

Automated test scripts will be executed to load the system beyond the capability of human operators, and to monitor system resource usage to check for items such as memory leaks or high CPU utilisation.

It is envisaged that automated scripts created for regression purposes can be adapted for use in performance testing.

Performance testing will take place throughout the development cycle, with checks in particular for memory and CPU usage since these are scarce resources on the target hardware.

11.2 Other Testing Considerations

In addition to the mentioned testing areas and considerations, the system will also have to support various testing modes. This means that certain functions, such as branch session management may need to be disabled or stubbed out, or additional interfaces provided to enable automated testing.

11.2.1 Peripheral Simulation

Special peripheral interfaces will be need, for example of the PIN pad to enable automated testing to proceed.

Other peripheral devices which are pure input or output (e.g. bar code readers, smart card readers and counter printers) can potentially operate using additional comms ports and additional cables, but ideally all such peripheral interfaces would have a diagnostic mode that allows a test automation package to provide the input or capture the output of the device. This also means providing the ability to simulate errors on the peripheral interface.

11.2.2 GUI Test Automation

Consideration should be given to including a test interface capability within the user interface logic. For example, the UI construct level may be a good point at which to intercept input and output that is independent of the GUI representation.

The testing of the UI rendering can then be automated with a significantly reduced automated test suite that is dedicated to the UI testing only.

11.2.3 Branch Database Emulation

It is assumed that there will be a test version of the counter services that access the Branch Access Layer, so that development and early integration testing can be performed on stand alone counter systems.

For example,

- An implementation of the Settlement Service that merely records the transaction data in a local log file and returns configurable responses.
- An implementation of the Reporting Service that merely returns pre-defined report data streams.
- An implementation of the Online Services that enable specific responses to be generated.
    - This is potentially a complex area due to the variety of response message formats – however, there may be some synergy with the Training simulation of Online Services.

The precise set of capabilities to be provided will be defined in the high level designs for the business application components, following discussion with development and test teams.

WITN11070108

11.1.4 Security

The target environment for the application is a fully “locked down” build that prevents access to the underlying operating system. The application is responsible for several aspects of the counter security as described in section 7.

For some development and testing environments it may be necessary to by-pass some of the security features, for example to apply reference data changes locally in a development environment (including RDT). However, testing of the full counter build must ensure that the security requirements are addressed, and that any such by-pass mechanisms are disabled within the live build. This will include a check of finished counter and sign of by PO and POA security.

WITN11070108

12 Requirements Traceability

Traceability of Business, Customer Service and System Requirements is detailed in a separate Traceability Matrix designated *ARC/APP/RTM/0001*.

WITN11070108

13 Appendices

Appendix 1 – Security Sensitive Data Held on the Counter

Table 2 – Security Sensitive Data Held on the Counter

#	Data Type	Policy Classification	Reason for Classification	Time Limited?	Managed ?	Location	Notes
1	Reference Data	Company Restricted	Contains commercially sensitive information	Yes	Yes	Counter Hard-Drive	
2	Diagnostic Tracing/Logging	Company Restricted	Potentially contains sensitive information depending on what is being logged and the level of logging applied	No	Yes	Counter Hard-Drive	Tracing data is only created and retrieved when required. It is not an ongoing and automated task for the Counter. Any files created are managed manually or by an automated SYSMAN process.
3	Certificates	Company Restricted	Root CA and Sub-CA certificates. These are Public Key Certificates and are, by definition, non-sensitive. Access to the Data Centre is further controlled by the VPN and other network and platform access controls	Yes	Yes	Java Key Store	
4	Certificates	Company Restricted	ACE Blade SSL Certificate. These are Public Key Certificates and are, by definition, non-sensitive. Access to the Data Centre is further controlled by the VPN and other network and platform access controls	Yes	Yes	Counter Memory	
5	Logon Key Pair	Company Secret	Created per Clerk session. These are sensitive for the duration of the session, then they are discarded and are useless outside the context of the session in which they were created	Yes	No	Counter Memory	
6	PAN Hash Seed	Company Secret	Sensitive. Disclosure of the PHS together with access to the PAN Hash Algorithm would enable disclosure of clear text PANs. The value of clear text PANs is however, minimal.	Yes	Yes	Counter Memory	

WITN11070108

7	VPN Keys	Company Restricted	VPN keys themselves are distributed through Dimensions to the Counter, as they are protected by a PIN value.	Yes	Yes	Counter Hard-Drive	
8	VPN Key PIN	Company Secret	The VPN Key PIN value unlocks the VPN Keys for use. This would allow an unauthorised system to connect to the VPN if known. The PIN value per Counter is changed to a random value following its initial use. Therefore, disclosure of a PIN for one Counter will not unlock the VPN key pair on another. Each new VPN key has a unique PIN value.	Yes	Yes	Counter Registry / Memory	
9	SSH Public Keys	Company Restricted	These will only work with the secret keys of each support user	No	Yes	Counter Hard-Drive	
10	SSH Host Keys	Company Restricted	These keys are automatically created by the SSH software on install and first use. They are unique to each system and effectively act as a 'fingerprint' to identify the Counter. These could be used to spoof the identity of a Counter if accessed.	No	Yes	Counter Hard-Drive	
11	Event Log Data	Company Restricted	The event log data is predominantly non-sensitive though there are always data that are more sensitive than the rest. System event logs are cycled on a regular basis and 'events of interest' are forwarded by the Tivoli agent. Any trace logs are sanitised (from a PCI perspective) prior to them being written.	Yes	Yes	Counter Hard-Drive	Managed means log files are rotated and events collected by Tivoli
12	CNIM2 Data (inc. Branch Router Data)	Company Restricted	Contains diagnostic data	No	Yes	Counter Hard-Drive	
13	Distributed Software Packages	Company Restricted	Defined by Policy	No	Yes	Counter Hard-Drive	Managed by SYSMAN
14	PIN Pad updates	Company Restricted	These files are transitory in nature on the Counter hard drive.	Yes	Yes	Counter Hard-Drive	Managed as part of the installation
15	PIN Pad keys	Company Secret	Keys are encrypted under a Pin Pad specific key. These files are transitory in nature on the Counter hard drive.	Yes	Yes	Counter Hard-Drive	Managed as part of the installation
16	User Password Encryption Key	Company Secret	Protects the user's new password	Yes	Yes	Counter Memory	Only in memory during logon for a password change.
17	User Password	Company Secret	Protects the user's password in memory	Yes	Yes	Counter M	

WITN11070108

	Hash						
18	Journal Sequence Number	Company Restricted	Last number used is saved on the Counter for Recovery purposes. This is a non-sensitive number, not encrypted and does not matter if it's absent (if absent an interaction has to take place with the clerk).	Yes	Yes	Counter Hard-Drive	