

# SOUTHWEST RESEARCH INSTITUTE™

6220 CULEBRA RD. 78238-5166 • P.O. DRAWER 28510 78228-0510 • SAN ANTONIO, TEXAS, USA • (210) 684-5111 • WWW.SWRI.ORG

AUTOMATION AND DATA SYSTEMS DIVISION  
FAX: (210) 522-5499

March 9, 2004

[REDACTED]  
Program Manager  
VISN 16/17 Scheduling Replacement Project  
One Technology Center  
[REDACTED]

San Antonio, Texas 78229

Subject: Southwest Research Institute® (SwRI®) Scheduling Replacement Project  
(10.04674), Software Design Document, Version 1.0

Reference: CASU Contract No. V674P-3209

Dear Mr. [REDACTED]

Enclosed you will find Version 1.0 of the Software Design Document (SDD) for the Veterans Health Administration (VHA) Scheduling Replacement Project.

It should be noted that some elements of this version of the SDD reflect the interim solutions provided in SwRI Proposal No. 10-34199H for Veterans Health Administration Replacement Scheduling Application (RSA) Engineering Change Requests (ECRs).

This document is provided on CD-ROM because of its size and construction. Dynamic links in the SDD point to the interactive Design Model as well as database design documents.

If you have any questions, please call me at [REDACTED] For contractual questions, please call [REDACTED]

Sincerely,

[REDACTED]  
Project Manager

Approved:

[REDACTED]  
Manager  
Medical Information Technology  
Medical Systems Department

TAM/md  
Enclosure

c: [REDACTED]



DETROIT, MICHIGAN (248) 353-2550 • HOUSTON, TEXAS (713) 977-1377 • WASHINGTON, DC (301) 881-0226


# **REPLACEMENT SCHEDULING APPLICATION**

## **SOFTWARE DESIGN DOCUMENT**

### **VETERANS HEALTH ADMINISTRATION (VHA) OUTPATIENT APPOINTMENT SCHEDULING REPLACEMENT PROJECT**

Prepared for:

Veterans Integrated Services Networks 16 and 17 Consortium  
One Technology Center

  
San Antonio, Texas 78229

Prepared by:

**SOUTHWEST RESEARCH INSTITUTE®**

Automation and Data Systems Division

P. O. Drawer 28510

San Antonio, Texas 78228-0510

SwRI® Project No.10.04674

March 2004

# REPLACEMENT SCHEDULING APPLICATION

## SOFTWARE DESIGN DOCUMENT

### VETERANS HEALTH ADMINISTRATION (VHA) OUTPATIENT APPOINTMENT SCHEDULING REPLACEMENT PROJECT

Prepared for:

VETERANS INTEGRATED SERVICES NETWORKS 16 AND 17 CONSORTIUM

One Technology Center  
7411 John Smith, Suite 110  
San Antonio, Texas 78229

Prepared by:

**SOUTHWEST RESEARCH INSTITUTE®**

Automation and Data Systems Division  
P. O. Drawer 28510  
San Antonio, Texas 78228-0510

SwRI® Project No.10.04674  
March 2004

8 Mar. 2004

Date

3/8/04

Date

# TABLE OF CONTENTS

	<u>Page</u>
<b>1. SCOPE .....</b>	<b>1</b>
1.1 PROJECT IDENTIFICATION .....	1
1.2 SYSTEM OVERVIEW.....	1
1.2.1 <i>Background</i> .....	1
1.2.2 <i>Project Description</i> .....	2
1.3 DOCUMENT OVERVIEW .....	3
1.4 RELATED DOCUMENTS .....	4
<b>2. SYSTEM COMPONENTS .....</b>	<b>6</b>
2.1 SYSTEM REQUIREMENTS.....	6
2.2 SYSTEM DESIGN DECISIONS .....	6
2.3 SYSTEM COMPONENTS.....	7
2.3.1 <i>Common Components (cmn)</i> .....	10
2.3.2 <i>Client-Tier Components</i> .....	10
2.3.2.1 Core Components (core).....	13
2.3.2.1.1 Administrative Entity Tree (aetree) .....	13
2.3.2.1.2 RSA Context Manager (context) .....	13
2.3.2.1.3 RSA Frame (frame) .....	13
2.3.2.1.4 User Preferences (userpref) .....	14
2.3.2.1.5 User Manager (usermanager) .....	14
2.3.2.2 Data Managers (data).....	14
2.3.2.3 Notification Components (notify) .....	14
2.3.2.4 Daily Operations Components (ops) .....	15
2.3.2.4.1 Appointment Operations (appt) .....	16
2.3.2.4.2 Appointment Request Operations (apptreq) .....	16
2.3.2.4.3 Patient Operations (patient) .....	16
2.3.2.4.4 Scheduling Operations (sched).....	16
2.3.2.5 Reporting Component (report) .....	16
2.3.2.6 RSA Configuration Components (rsaconfig) .....	17
2.3.2.6.1 Administrative Entity Configuration (ae).....	17
2.3.2.6.2 Change Configuration (change).....	18
2.3.2.6.3 Resource Configuration (resource).....	18
2.3.2.6.4 Standard Configuration (standard) .....	18
2.3.2.7 Section Configuration Components (sectionconfig) .....	19
2.3.2.7.1 Appointment Purpose Configuration (apptpurpose).....	19
2.3.2.7.2 Resource Configuration (resource).....	19
2.3.2.7.3 Scheduling Configuration (sched) .....	20
2.3.2.7.4 Section Configuration (section).....	20
2.3.2.8 Client-Tier Utility Components (util).....	21
2.3.2.8.1 Calendar Control.....	21
2.3.2.8.2 Print Manager .....	21
2.3.3 <i>Middle-Tier Components</i> .....	21
2.3.3.1 Web Services .....	24
2.3.3.1.1 Administrative Entity Web Service (adminentity) .....	25

2.3.3.1.2	Schedule Management Web Service (cmn.opsfw)	25
2.3.3.1.3	Appointment Purpose Web Service (apptpurpose)	26
2.3.3.1.4	Attribute Management Web Service (attribute)	26
2.3.3.1.5	User Authorization Web Service (authorization)	26
2.3.3.1.6	Configuration Change Web Service (configchange)	26
2.3.3.1.7	Patient Web Service (cmn.opsfw)	26
2.3.3.1.8	Resource Management Web Service (resource)	27
2.3.3.1.9	Template Management Web Service (template)	27
2.3.3.2	Middle-Tier Common Components (cmn)	27
2.3.3.3	Middle-Tier Utilities (util)	27
2.3.3.3.1	RSABusinessUtilities	27
2.3.3.3.2	UserManager	27
2.3.3.4	External Interface Components	28
2.3.3.4.1	Event Servlets (eventservlets)	28
2.3.3.4.1.1	Patient Maintenance Event Servlet (patient)	28
2.3.3.4.1.2	Person Maintenance Event Servlet (person)	28
2.3.3.4.2	Proxy Adaptors (proxies)	30
2.3.3.4.2.1	Appointment Event Proxy (apptevent)	30
2.3.3.4.2.2	Compensation and Pension Proxy (candp)	30
2.3.3.4.2.3	Check In Point Proxy (checkinpoint)	30
2.3.3.4.2.4	Local Purpose Proxy (localpurpose)	30
2.3.3.4.2.5	Patient Proxy (patient)	31
2.3.3.4.2.6	Person Proxy (person)	31
2.3.3.4.3	Facade Adaptors (facadeadp)	31
2.3.3.4.3.1	Administrative Entity Tree Adaptor (aetreeadaptor)	31
2.3.3.4.3.2	Appointment Adaptor (apptadaptor)	31
2.3.3.4.3.3	Appointment Query Adaptor (apptqueryadaptor)	32
2.3.3.4.3.4	Encounter Adaptor (encounteradaptor)	32
2.3.3.4.3.5	Patient Adaptor (patientadaptor)	32
2.3.3.4.3.6	Person Adaptor (personadaptor)	32
2.3.3.4.4	VistALink Interface (vistalinkintf)	33
2.3.3.4.5	Vitria Interface (vitria)	33
2.3.3.5	Data Layer Components	33
2.3.3.5.1	Transaction Managers (tm)	34
2.3.3.5.1.1	Administrative Entity Transaction Manager (adminentity)	34
2.3.3.5.1.2	Resource Transaction Manager (resource)	34
2.3.3.5.1.3	Template Transaction Manager (template)	34
2.3.3.5.1.4	Appointment Purpose Transaction Manager (apptpurpose)	34
2.3.3.5.1.5	Configuration Change Package Transaction Manager (configchange)	34
2.3.3.5.1.6	Schedule Transaction Manager (sched)	36
2.3.3.5.1.7	Retrieval Transaction Manager (retrieve)	36
2.3.3.5.1.8	Patient Transaction Manager (patient)	36
2.3.3.5.1.9	Utility Transaction Manager (utility)	36
2.3.3.5.2	BC4J Data Tier Components (dtc)	37
2.3.3.5.2.1	Administrative Entity DTC (adminentity)	37
2.3.3.5.2.2	Appointment DTC (appt)	37
2.3.3.5.2.3	Lookup DTC (lookup)	37
2.3.3.5.2.4	Request DTC (request)	37
2.3.3.5.2.5	Maintenance DTC (opsmaint)	37
2.3.3.5.2.6	Search DTC (search)	39

2.3.3.5.2.7	Patient DTC (patient).....	39
2.3.3.5.2.8	Appointment Purpose DTC (apptpurpose) .....	39
2.3.3.5.2.9	Configuration Change DTC (configchange) .....	39
2.3.3.5.2.10	Patient Group DTC (group) .....	39
2.3.3.5.2.11	Resource DTC (resource).....	39
2.3.3.5.2.12	Template DTC (template).....	40
2.3.3.5.2.13	Utility DTC (util) .....	40
2.3.4	<i>RSA Database – VADB</i> .....	40
2.3.5	<i>RSA Utilities</i> .....	40
2.3.6	<i>Other Components</i> .....	40
2.3.6.1	Administrative Entity Tree Service .....	41
2.3.6.2	Oracle Internet Directory (OID) .....	42
2.4	SYSTEM RESOURCES .....	42
2.5	CONCEPT OF EXECUTION .....	42
2.5.1	<i>Configuration</i> .....	43
2.5.1.1	State Machine for CCP Related Operations .....	43
2.5.1.2	State Machine for CCP Entities .....	48
2.5.1.3	Normal Base Data Item State Machine .....	48
2.5.1.4	Build RSA Configuration .....	50
2.5.1.5	Create CIP .....	50
2.5.1.6	Create Resource.....	50
2.5.1.7	Allocate Resource.....	54
2.5.1.8	Remove Resource Allocation .....	54
2.5.1.9	Create Resource Set.....	54
2.5.1.10	Create Resource Carve-outs.....	57
2.5.1.11	Create Local Appointment Purpose .....	59
2.5.1.12	Create Appointment Purpose Set .....	59
2.5.1.13	Deactivate Schedulable Admin Entity .....	59
2.5.2	<i>Daily Operations</i> .....	63
2.5.2.1	Make Appointment Request .....	63
2.5.2.2	Search for Appointment Opportunities.....	66
2.5.2.3	Make a Single Appointment.....	66
2.5.2.4	Enter Walk-In Appointment .....	69
2.5.2.5	View Appointments.....	69
2.5.2.6	View Resource Schedule .....	69
2.5.2.7	Appointment Maintenance .....	72
2.5.2.8	Appointment Request Maintenance.....	72
2.5.2.9	Cancel Appointment .....	77
2.5.2.10	Reschedule Appointment .....	77
2.5.2.11	Update Appointment Status .....	77
2.5.2.12	Manage Linked Appointments.....	78
2.5.2.13	Handle Patient Maintenance Events .....	79
2.5.3	<i>External Interfaces</i> .....	79
2.5.3.1	Create New CIP External .....	79
2.5.3.2	Modify CIP External .....	82
2.5.3.3	Appointment Events .....	83
2.5.3.4	Retrieve Provider Information.....	84
2.5.3.5	Retrieve Patient Information .....	85
2.5.3.6	Retrieve Resource.....	86
2.5.3.7	Lookup Patient.....	87

2.5.3.8	Lookup C&P.....	88
2.5.3.9	Patient Maintenance Event .....	89
2.5.3.10	Person Maintenance Event.....	90
2.5.3.11	External Appointment Query .....	91
2.5.3.12	External Appointment Request .....	92
2.5.3.13	External Appointment Book .....	93
2.5.3.14	Create Encounter Event .....	97
2.5.3.15	Patient Context Change.....	98
2.5.4	<i>Exception and Error Handling</i> .....	99
<b>3.</b>	<b>SYSTEM DESIGN.....</b>	<b>101</b>
3.1	INTERFACE DESIGN .....	101
3.2	SOFTWARE MODULE DETAILED DESIGN .....	101
<b>4.</b>	<b>DATA DICTIONARY .....</b>	<b>107</b>
<b>5.</b>	<b>REQUIREMENTS TRACEABILITY .....</b>	<b>108</b>
<b>APPENDIX A -</b>	<b>GLOSSARY, TERMS, AND ACRONYMS .....</b>	<b>A-1</b>
<b>APPENDIX B –</b>	<b>RSA DATABASE .....</b>	<b>B-1</b>

# LIST OF FIGURES

	<u>Page</u>
Figure 1. RSA's Three-Tiered Architecture .....	8
Figure 2. RSA High-Level Architecture .....	9
Figure 3. Client-Tier Packages .....	11
Figure 4. Presentation Layer Architecture .....	12
Figure 5. RSA DataManagers and Web Service Stubs .....	12
Figure 6. Rendering and Printing a Notification Letter .....	15
Figure 7. Reporting Component .....	17
Figure 8. PrintManager Uses XSL to Format the Print Job .....	22
Figure 9. RSA Middle Tier .....	23
Figure 10. Web Service and Consumer Logical Architecture .....	24
Figure 11. External Interface Components .....	29
Figure 12. TransactionManagers Maintain Transactional Control .....	35
Figure 13. Structure of a Data Tier Component (DTC) .....	38
Figure 14. Essential Data Structure .....	41
Figure 15. RSA Configuration Components .....	44
Figure 16. State Machine for Entities During Add Operations .....	45
Figure 17. State Machine for Entities During Inactivation Operations .....	46
Figure 18. State Machine for CCP Related Operations .....	47
Figure 19. State Machine for Maintaining CCP Operations .....	49
Figure 20. Normal Base Data Item State Machine .....	49
Figure 21. Building Configurations Within RSA .....	51
Figure 22. Create Check In Point .....	52
Figure 23. Create Resources .....	53
Figure 24. Allocate Resource .....	55
Figure 25. Remove Resource Allocation .....	56
Figure 26. Creating a Resource Set .....	57
Figure 27. Create Resource Carve-out .....	58
Figure 28. Creating a Local Appointment Purpose .....	60
Figure 29. Create Appointment Purpose Set .....	61
Figure 30. Deactivate Schedulable Administrative Entity .....	62
Figure 31. Daily Operations Activities .....	64
Figure 32. Make Appointment Request .....	65
Figure 33. Search for Appointment Opportunities .....	67
Figure 34. Make a Single Appointment .....	68
Figure 35. View Appointments .....	70
Figure 36. View Resource Schedule .....	71
Figure 37. Patient Appointment State Machine .....	74
Figure 38. Appointment State Machine .....	75
Figure 39. Appointment Request State Machine .....	76
Figure 40. Linking Unlinked Appointments .....	78
Figure 41. Updating With the Link Set ID .....	78
Figure 42. Handle Patient Maintenance Events .....	80
Figure 43. CIP Creation Event .....	81
Figure 44. CIP Modification Event .....	82
Figure 45. Appointment Events .....	83



Figure 46. Create New Provider Event .....	84
Figure 47. Retrieve Patient Information .....	85
Figure 48. Retrieve Resource.....	86
Figure 49. Lookup Patient .....	87
Figure 50. Lookup C&P.....	88
Figure 51. Patient Maintenance Event .....	89
Figure 52. Person Maintenance Event .....	90
Figure 53. External Appointment Query .....	91
Figure 54. Retrieve Request Information.....	92
Figure 55. Save Appointment Request .....	93
Figure 56. Getting Appointment Search Data.....	94
Figure 57. Make Appointment Search .....	95
Figure 58. Book Appointment Event .....	96
Figure 59. Create Encounter Event.....	97
Figure 60. Patient Context Change .....	98
Figure 61. Exception Handling and Logging in RSA.....	100
Figure 62. Class Specification in Web Published RSA Design Model.....	103
Figure 63. Class Diagram Displayed in the Web Published RSA Design Model.....	104
Figure 64. Classes May Be Stereotyped and Assigned to a Component .....	106

## LIST OF TABLES

	<u>Page</u>
Table 1. Project Identification.....	1
Table 2. Request List Criteria .....	73

# 1. SCOPE

## 1.1 Project Identification

This task is being conducted by the Veterans Integrated Services Networks (VISN) 16 and 17 Consortium under Engineering Research and Development Contract (ERDC) No. P674P-3209.

**Table 1. Project Identification**

Project Title:	VHA Replacement Scheduling Application
Project Number:	10-04674
Abbreviation:	RSA
Version Number:	1.0
Release Number:	1.00

## 1.2 System Overview

The Replacement Scheduling Application (RSA) is being developed as a part of the VHA Outpatient Appointment Scheduling Replacement Project. The following sections provide background information on the Scheduling Replacement Project and provide an overview of RSA.

### 1.2.1 Background

The appointment scheduling software currently being used by the Veterans Health Administration (VHA) is over two decades old. Over the last twenty years, VHA appointment scheduling processes have evolved, often in directions contrary to existing software capabilities. In the year 2000, a VHA study concluded that the process/technology gap could not be bridged through continued modification of the existing Veterans Health Information Systems and Technology Architecture (*VistA*) scheduling software. A scheduling replacement project was therefore initiated in response to a request for proposal to (1) re-engineer existing VHA appointment scheduling processes and (2) replace existing appointment scheduling software with a more responsive, flexible software solution.

Business Process Re-engineering (BPR) of the VHA outpatient appointment scheduling process was accomplished during 2001. The results and conclusions of the BPR effort are documented in a current model [1], a future model [2], and a process improvement framework [4]. The results of the BPR have also been translated into system [3] and software [5] requirements. Terms used to describe the scheduling process can be found in the Replacement Scheduling Application (RSA) Glossary [8].

Specific objectives of the scheduling replacement project from the Scheduling BPR Request for Proposal (RFP) as adapted for the anticipated Future Business Process Model (FBPM) environment include:

- Prevent scheduling of patients at different facilities on the same day
- Collectively schedule resources (staff, rooms, equipment)
- Schedule (and reschedule) collections of appointments that have sequential relationships (laboratory appointment comes before provider visit which is followed by pharmacist visit)
- Reserve Appointment Opportunities for special types of patients (e.g., patients with service-connected eligibilities)
- Block out time periods
- Prevent new patient types on a specified schedule
- Schedule group appointments
- Provide flexible appointment structure (appointment duration, arrangement of appointments)
- Provide a graphical user interface that supports the clerical staff's handling of patients
- Maintain communications with other **VistA** packages
- Maintain communication with the Computerized Patient Record System (CPRS)
- Support patient appointment viewing and scheduling via MyHealtheVet

The Scheduling Replacement Project team has been assigned the task of producing an enterprise-level outpatient appointment scheduling application that incorporates the results of VHA outpatient appointment management BPR. The system described in this document is to be a replacement and enhancement for the outpatient appointment scheduling functionality currently present in **VistA**. Upon completion of development and integration, the replacement scheduling application will become part of the HealtheVet **VistA** suite of capabilities.

### **1.2.2 Project Description**

RSA is not a **VistA** patch; it will replace the existing VHA scheduling system. RSA is a significant overhaul of the outpatient scheduling business process and its associated information management system. The VHA's patient appointment scheduling process has been re-engineered, and RSA will be developed from the ground up to satisfy desired future appointment scheduling needs. RSA will provide a centralized, enterprise-wide database for scheduling data which will provide much needed insight into the scheduling operation by reporting metrics on scheduling activities. Specifically, RSA will provide for the administration of the schedulable entities in the enterprise, assist the scheduler in making appointments, and provide reports on scheduling activities.

RSA will provide the scheduling administrators the ability to define the parameters that control the configuration of the various schedulable entities within the enterprise. The administrators will assign resources to the schedulable entities in the form of providers, facilities, and equipment (i.e., people, places, and things). RSA will provide administrators the ability to

establish operating parameters for the schedulable entities, including establishment of hours of operation and appointment purpose “carve-outs,” to set aside resource time for specific appointment purposes.

RSA will provide the scheduler the ability to schedule appointments for individuals and groups into schedulable entities (sections) according to assigned privileges. RSA will also allow schedulers to request appointments in sections for which they do not have scheduling authority.

RSA will permit the definition of appointment sets and resource sets. An appointment set is a series of appointments necessary to accomplish an associated appointment purpose. RSA will search for the series of appointments necessary to satisfy an appointment set and display the possible selections to the scheduler. RSA will allow the creation of resource sets made up of a group of resources that the system will use when searching for available appointments that meet the criteria of the entire resource set. RSA will also allow users to take into account patient day/time preferences when searching for appointments to try and meet the specific scheduling needs of each patient. Once the scheduler selects an appointment, RSA will book the appointment reserving the resources necessary for the appointment. Provisions are made in RSA to allow overbooking of resources within limits set by schedule administrators.

RSA will provide reports/metrics that will give management insight into the efficiency and effectiveness of scheduling activities.

### **1.3 Document Overview**

This document is a companion document to the Baseline System Architecture Document (BSAD); together the BSAD and the Software Design Document (SDD) describe both the architecture and design of RSA. The Baseline System Architecture presents different architectural views to represent both functional and non-functional components of RSA. It covers the software, hardware, and related infrastructure issues. This document, the SDD, provides a description of the high-level design of RSA as well as the detailed software module design. To provide clarity and context, some of the same information is provided in both the BSAD and the SDD.

Section 2 of this document presents the high-level design of RSA. It describes the software components that will comprise the new system and discusses the interactions that will take place between the various software components. Section 3, System Design, includes the detailed design of the individual software components identified in Section 2.

Section 3 will be generated from the Rational Rose model that is being used to design the system. The output of this tool will be multiple Hypertext Markup Language (HTML) files that can be viewed within a browser and fully describes all Java classes, and all methods contained within those classes.

Section 4, Data Dictionary, will also be completed during detailed design describing all the enterprise-level data elements used by this system. Thus, this dictionary is a living document, updated at various stages of the software development lifecycle, and the final version delivered upon completion.

Section 5 is the Requirements Traceability Matrix, where each software module is matched to identified requirements as outlined in the Software Requirements Specification (SRS).

## **1.4 Related Documents**

1. Current Business Process Model, Version 1.02, Veterans Health Administration Outpatient Appointment Scheduling Replacement Project, Southwest Research Institute, 25 September 2001.
2. Future Business Process Model, Version 1.02, Veterans Health Administration Outpatient Appointment Scheduling Replacement Project, Southwest Research Institute, 14 November 2001.
3. System Specification, Version 1.02, Veterans Health Administration Outpatient Appointment Scheduling Replacement Project, Southwest Research Institute, 05 February 2002.
4. Process Improvement Framework (Draft), Version 1.00, Veterans Health Administration Outpatient Appointment Scheduling Replacement Project, Southwest Research Institute, 07 Jun 2002.
5. Software Requirements Specification Volumes I and II, Version 2.00, for the VHA Replacement Scheduling System, Draft Version, 12 Nov 2003.
6. Interface Control Document, Version 2.03, Veterans Health Administration Outpatient Appointment Scheduling Replacement Project, Southwest Research Institute, 20 Jan 2004 (Working Version).
7. Replacement Scheduling Application Baseline System Architecture Document, Version 2.0, Veterans Health Administration Outpatient Appointment Scheduling Replacement Project, Southwest Research Institute, 25 February 2004.
8. RSA Glossary for the VHA Replacement Scheduling System, February 2004.
9. Enterprise Architecture, Department of Veterans Affairs, Veteran Health Administration, 01 January 2001.
10. Cross-Application Integration Protocol (CAIP) Interim Project Guidance, Version 1.0, Department of Veteran Affairs, 3 December 2003.
11. GUI Patient Lookup, System Administrator/Developer Manual, Health Systems Design and Development, 24 December 2003.
12. GUI Patient Lookup, Software Requirements Specification, Health Systems Design and Development, October 2003.

13. VistALink VERSION 1.5 (Pre-Alpha Build: 1.4.5) Resource Adapter Deployment Guide, HSD&D, 14 January 2004.
14. Java™ 2 Platform Enterprise Edition Specification, v1.4 Sun Microsystems, 19 August 2002.
15. Person Services, Prototype Users Guide, Version 1.0, HSD&D, 10 November 2003.
16. Patient Services, Prototype Installation Guide, Release 1, HSD&D, 1 October 2003.
17. Installation Procedures for Cache and DSM, VistALink for Java Basic (VLJ-B) (v.1.0.1), Department of Veterans Affairs, 9 December, 2002.
18. BusinessWare Installation and Configuration Cookbook, Delivered with Vitria BusinessWare, No Date or Attribution.
19. Instructions for Installing End-To-End TestBed, Delivered with Vitria Business Ware, No Date or Attribution.
20. J2EE Patterns Catalog. (Sun's BluePrints) <http://java.sun.com/blueprints/patterns/catalog.html>.

## 2. SYSTEM COMPONENTS

### 2.1 System Requirements

For a complete list of system requirements refer to the document entitled “Software Requirements Specification for the Replacement Scheduling Application” dated 12 November 2003.

### 2.2 System Design Decisions

This section provides an outline of design decisions that are due to architectural decisions or constraints that have impact on the RSA design and the environment in which it operates. Many of these decisions were driven by the VHA’s overall architectural decisions. Some of those VHA decisions have been revised or rescinded for future VHA systems but remain in effect for RSA, as agreed upon by the VHA’s Information Technology (IT) Project Manager, in order to meet project cost and schedule constraints.

The outline below simply provides a categorized list of these decisions. For a complete description of each decision along with the rationale and impact for each decision, refer to the “Replacement Scheduling Application Baseline System Architecture Document,” Version 2.0, dated 25 February 2004 [7].

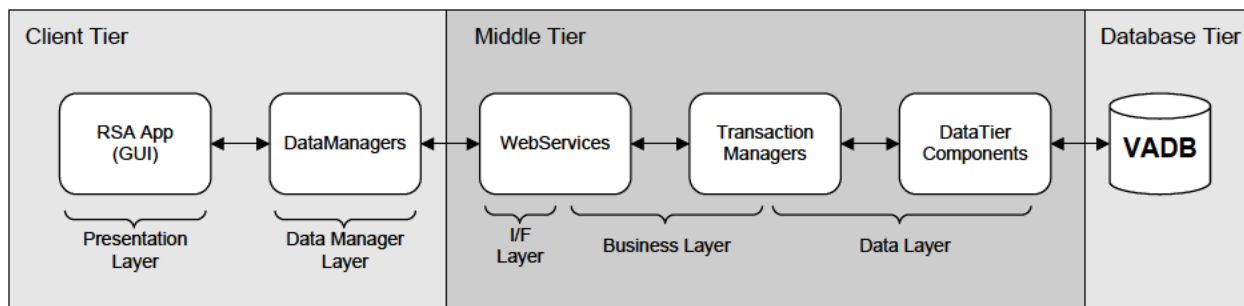
- VHA Architecture-Driven Decisions
  - Enterprise Architecture Decisions
    - Three-Tier Architecture
    - Oracle 9i Database
    - Oracle 9i Application Server (AS)
    - Windows for Client Nodes
    - Linux for Application Server
    - Linux for Database Server
    - Single Logical Database
    - Application Server Per *VistA* Site
    - Structured Query Language (SQL) and PL/SQL Use
    - Two Clusters at Primary Site and One Offsite Cluster for Failover
    - Implement Business Rule in Code
    - Oracle Internet Directory for User Preference
  - Interface Decisions
    - VistALink Interface to Encapsulation Application Program Interface
    - Java Bean Based Interface to Services
    - Health Level 7/Vitria to Push Interface
  - Security Decisions
    - Clinical Context Object Workgroup (CCOW) – Single Sign On (SSO)
    - Security Plug-in
    - Client to Middle Tier Security
    - Maintain Audit Info



- Oracle Internet Directory for User Authorization Info
- Functionality Decisions
  - Basic Functionality Decisions
    - No Purge
    - Data Download to Application Servers for Resiliency
  - Responsibility Allocation Decisions
    - Synchronization between Replacement Scheduling Application and **VistA**
    - Patient Lookup Responsible for Access Requirements
    - Letter Notification Only
    - Integrated Notification
    - Interim Administrative Entity Tree Service
    - No changes to Patient Demographics Data
    - Correlation of Person Resources with **VistA**
    - Appointment to Encounter Correlation
    - **VistA** Hospital Location File (HLF) Entries Correspond to Check In Points
    - HLF Entry to Local Purpose Correlation
    - Maintain Patient Admission Status
    - Keep 2507 Form to Appointment Request Correlation
    - Provide Decision Support System (DSS) Identification (ID) Sets in Appointments, Not HLF Entries
    - Service for MyHealtheVet
- Internal Design Decisions
  - Use of Business Components for Java (BC4J)
  - Use Stateless Web Services from Client Tier to Middle Tier
  - Rich Java Application
  - Client Application Not in HealtheVet View
  - Oracle Real Application Cluster (RAC)
  - Oracle Clustered File System
  - EMC Storage Area Network
  - Non-Clustered Infrastructure Server
  - User Information Serialized Locally
  - Static Data in Memory in Middle Tier

## 2.3 System Components

RSA is a three-tiered (Figure 1) Java 2 Enterprise Edition (J2EE) application developed in Oracle 9i JDeveloper (version 9.0.3) and designed to run integrated with the Oracle 9i Application Server. RSA utilizes web services in the middle tier and Oracle's Business Components for Java (BC4J) for database access in a modified version of the J2EE design pattern known as "Fast-Lane Reader" [20]. Fast-Lane Reader is part of the J2EE Patterns Catalog.



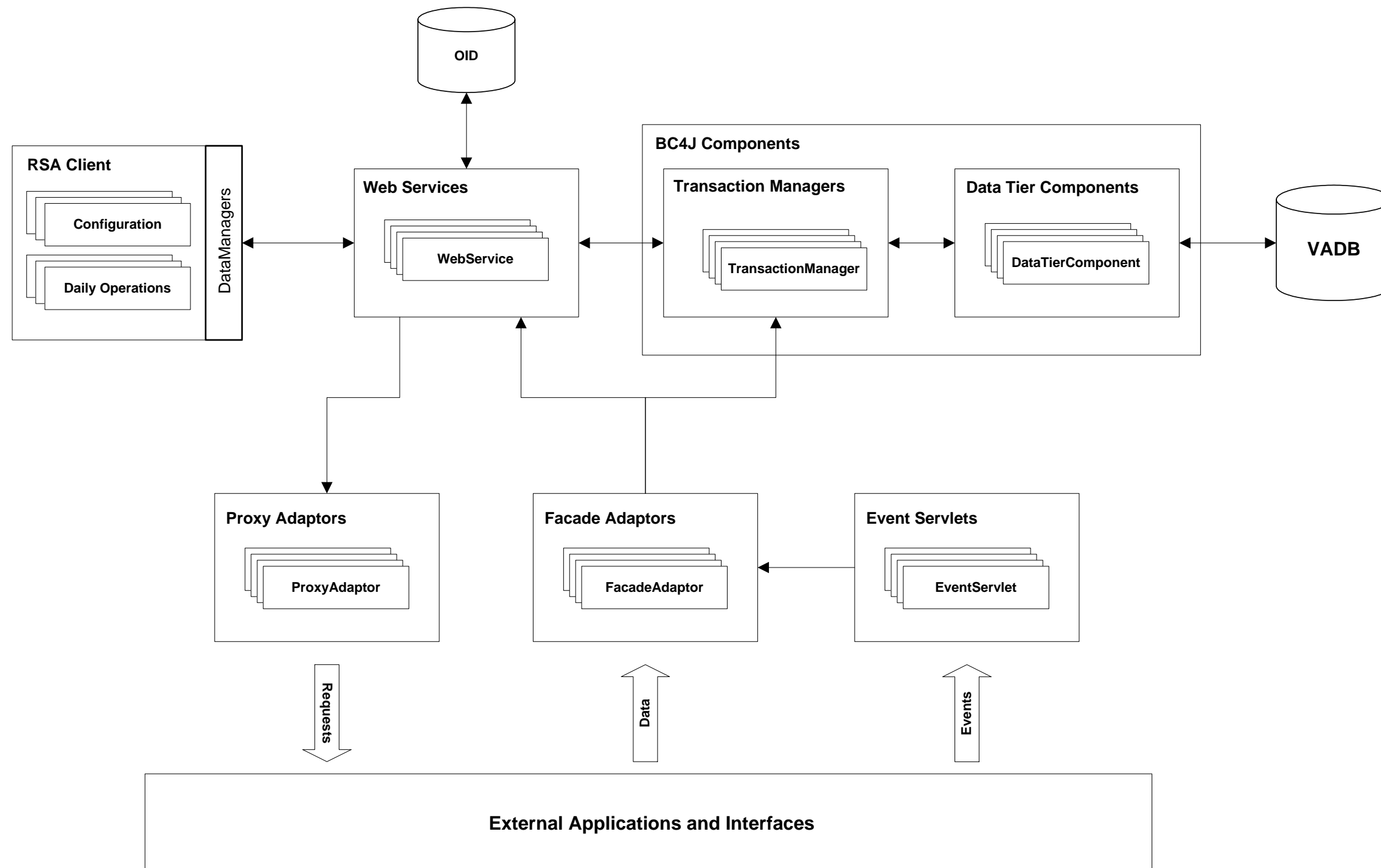
**Figure 1. RSA's Three-Tiered Architecture**

A high-level view of RSA is shown in Figure 2. RSA's client tier is a rich Java application functionally separated into two distinct areas: Configuration and Daily Operations. Low-level tasks associated with web service connections, Web Method calls, etc., are abstracted from the Graphical User Interface (GUI) by a layer of DataManagers.

RSA implements the Health Level 7 (HL7) standard known as Clinical Context Object Workgroup (CCOW) and, in fact, performs no user authentication and will always expect to receive the authenticated user in context. However, RSA does maintain its own user authorization information using an Oracle Internet Directory (OID). CCOW is not shown in Figure 2.

RSA contains multiple interfaces to external applications. Applications wishing to pass data to RSA do so through RSA's Facade Adaptors which expose an Enterprise Java Bean (EJB) interface. Additionally, RSA receives and processes HL7 messages via Event Servlets. RSA implements Proxy Adaptors for sending data to other applications.

The following paragraphs are an enumeration of all RSA components. The component's project unique identifier is given along with the Java package where the code resides in parenthesis as well as a brief description of the component.



**Figure 2. RSA High-Level Architecture**

### 2.3.1 Common Components (cmn)

**Package:** gov.va.med.rsa.ops.cmn

RSA's "Value Objects" are contained in the cmn package. Value Objects are immutable Java classes used solely for transferring data throughout the system (i.e., they contain no business logic, only getters and setters). In this document they may be referred to as either "Value Object" or "Data Object." The two terms may be used interchangeably.

RSA's Data Objects are bundled into a single common package as there are specific rules governing their behavior. The names of these types of objects are always preceded with "Transferable." This is to indicate that the Data Object is intended to be Simple Object Access Protocol (SOAP) serialized and passed to/from a web service. These objects must be derived from a single base class, RSAObject. Additionally, they must implement the Java interface, java.lang.Serializable.

### 2.3.2 Client-Tier Components

**Package:** gov.va.med.rsa.ops.gui

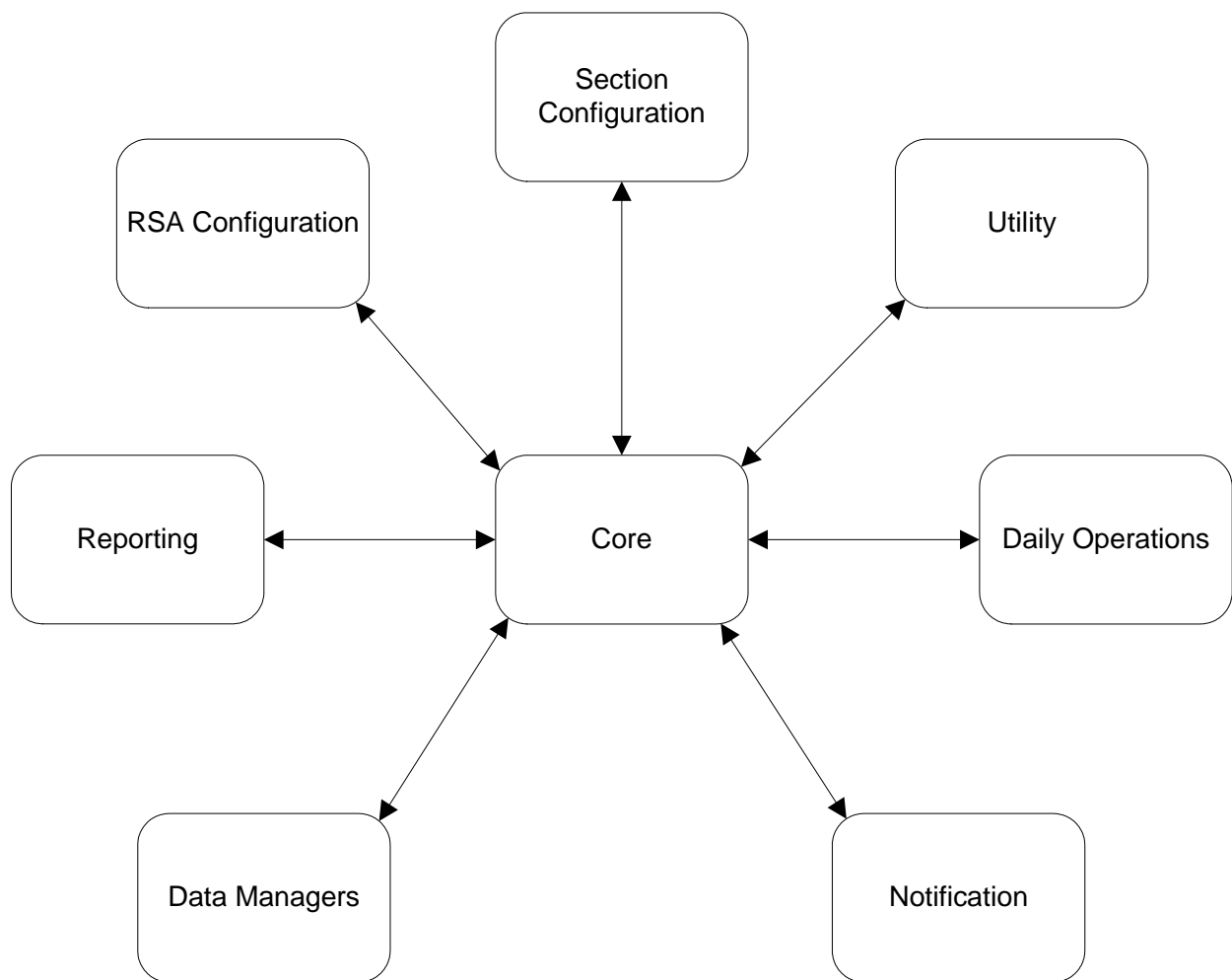
GUI components in RSA are further broken down and packaged based on the functionality they provide. These packages, shown in Figure 3, each contain one or more components necessary to provide this functionality.

On instantiation, RSAMain retrieves a list of tasks from the Tasks.cfg configuration file (Figure 4). The configuration file specifies the text identifier to be used on the menu, and the class name of the RSAGuiPanel extension specific to those tasks. It uses this information to create the Tasks menu on the main RSA window, allowing the user to select the operations they wish to perform.

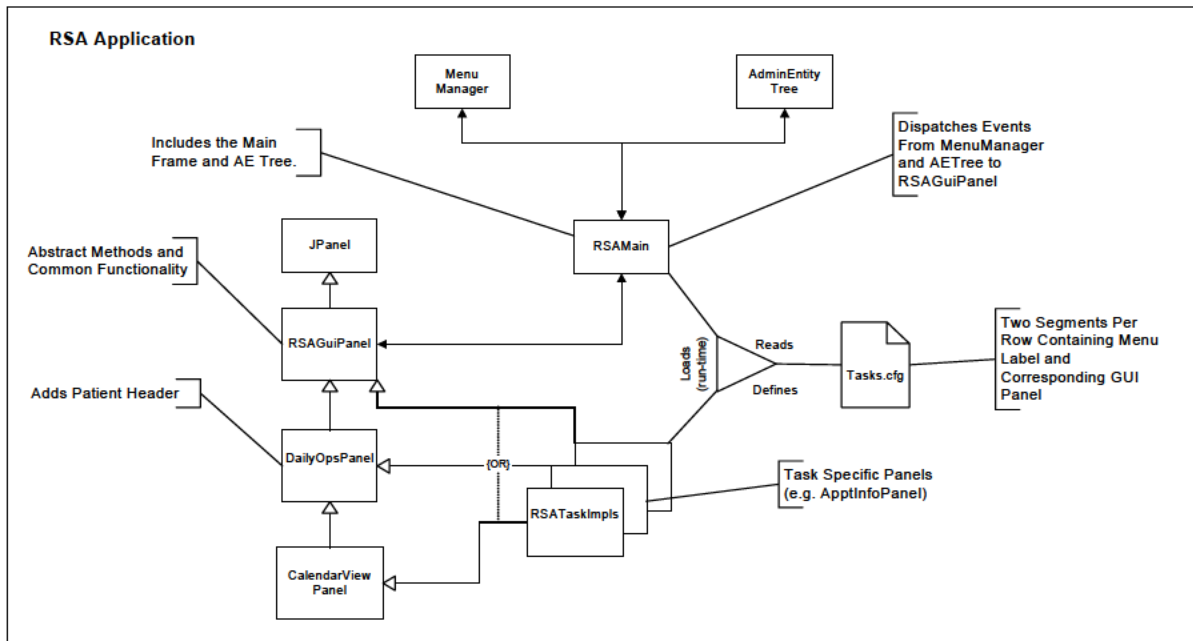
When the user selects a particular task, either through the menu or toolbar, RSAMain uses reflection to create an instance of the specified class, and places it in the operational window. This is done only once; later selection of a specific task will return the instance to the front of the window, rather than creating another instance.

Once a task panel is active, RSAMain acts as an intermediary, passing information from the panel to the MenuManager and AdminEntityTree, and vice versa.

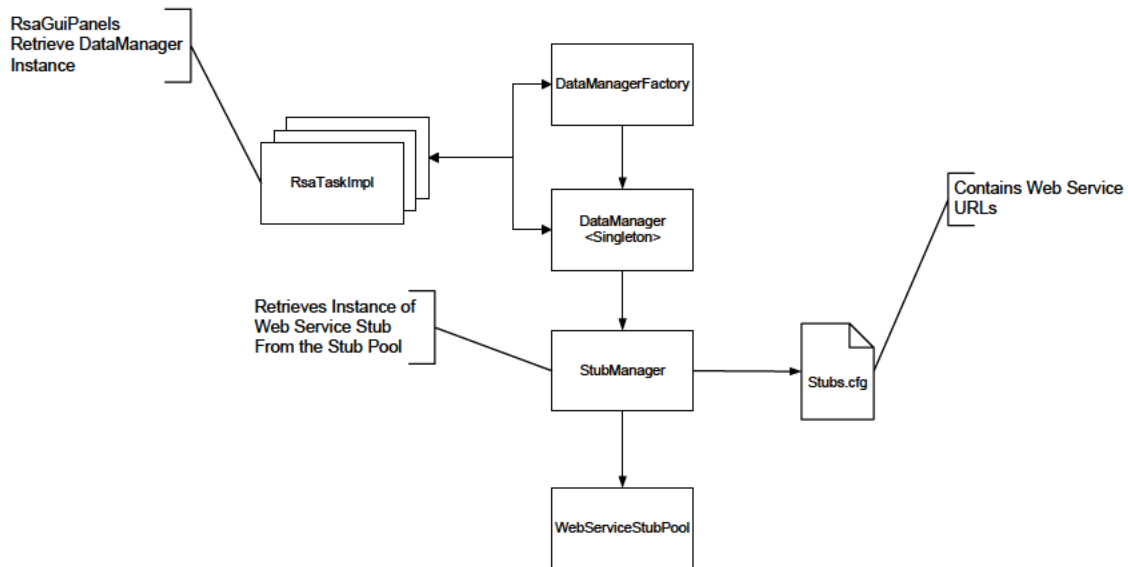
The primary DataManager implementations retrieve data from the business layer through the use of web services and have appropriate web service stub classes (Figure 5). Rather than maintaining the stubs themselves, the DataManager retrieves an appropriate stub from the StubManager, which in turn retrieves information on the stub's target endpoint from the Stubs.cfg configuration file.



**Figure 3. Client-Tier Packages**



**Figure 4. Presentation Layer Architecture**



**Figure 5. RSA DataManagers and Web Service Stubs**

This allows multiple server targets to be established and easily configured for the RSA GUI. For each data request which requires a call to a web service, the DataManager retrieves a stub from the pool by requesting it from the StubManager. This stub is then used for the single call and returned to the pool. This avoids multithreading issues, as Oracle web service stubs are not inherently thread-safe.

### **2.3.2.1 Core Components (core)**

**Package:** gov.va.med.rsa.ops.gui.core

The core package contains components which are fundamental to the operation of the RSA Client application but do not relate to any specific task. This includes the base classes for the RSA task panels, Administrative Entity (AE) Tree, RSA Context Manager, User Preferences, and User Management.

#### **2.3.2.1.1 Administrative Entity Tree (aetree)**

The Administrative Entity Tree supports the tree model for managing and maintaining the VHA's administrative entity organizational hierarchy. It maintains and presents a tree of AE nodes, potentially with Resource objects as scattered leaf nodes. The AE Tree will dispatch event messages to all registered listeners. This will primarily be the RSAMain panel, which will then distribute notification of the events to the various RSA GUI panels.

#### **2.3.2.1.2 RSA Context Manager (context)**

The RSA Context Manager is designed to encapsulate the behavior of the Context Management interfaces. This component is charged with implementing the interfaces necessary to accept a user context change and to initiate and accept patient context changes.

#### **2.3.2.1.3 RSA Frame (frame)**

The top-level GUI component of the RSA Application is embodied in the RSAMain class. Containing the menu bar, toolbar, administrative entity tree, and task control area, RSAMain contains all the primary components of the RSA GUI. It also manages all necessary communications between them, allowing each primary component to be only loosely coupled to the others.

As an extension of JPanel, RSAMain may be instantiated and placed in any GUI framework desired. The primary initialization point for the RSA client is through the RSAApplication class, which simply creates an RSAMain, wraps it in a window (JFrame), and displays it. This component-style development allows the RSA GUI to be run from any Java environment, be it application or applet, assuming that security issues such as the applet sandbox are resolved.

#### **2.3.2.1.4 User Preferences (userpref)**

User preferences fall into two categories: state information and specifically set user preference information. The state information preferences are those that are automatically recorded as a user navigates through RSA. This includes things such as tab locations, current screens, etc. When the user shuts down, the current settings will be saved so that the current state is the same when the user logs back in. This includes things such as tab locations, which panel is active, etc. The specifically set preferences include what columns should be displayed in tables, what order the columns should be displayed in, the date format to be utilized, etc. All of these preferences are stored when the user logs off of RSA and are retrieved when the user logs on to RSA.

#### **2.3.2.1.5 User Manager (usermanager)**

The user manager component is designed to encapsulate the interaction between authorization and preference usage by the RSA components and the interaction with the OID. The user manager consists of a GUI part and a middle-tier part. The GUI part provides methods to get and set personalization strings for user preferences and also to check user authorizations at the GUI level. The middle-tier part consists of encapsulated behavior that will interact with the OID and save copies of user data locally so that the other middle-tier components can use these local objects and not need to go to OID on every method call.

#### **2.3.2.2 Data Managers (data)**

**Package:** gov.va.med.rsa.ops.gui.data.

In order to retrieve data from the business layer, the RSA GUI makes use of an abstract data retrieval interface named DataManager. A number of unique DataManager implementations exist, typically one for each major category of data, retrieved via the DataManagerFactory. The DataManagers provide data to the GUI in an implementation-independent manner, allowing the GUI to isolate itself from the specifics of the data source. There is typically a one-to-one mapping of a DataManager to an RSA web service.

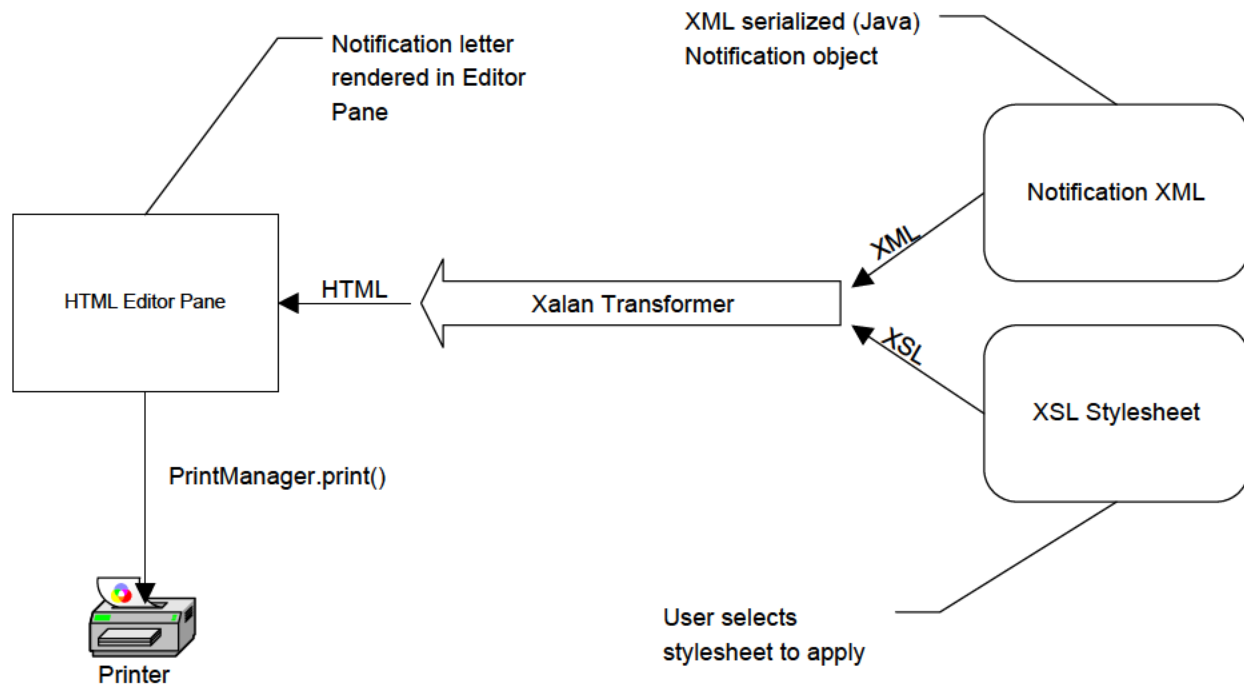
#### **2.3.2.3 Notification Components (notify)**

**Package:** gov.va.med.rsa.ops.gui.notify.

RSA prints letters for mailing to notify patients of certain significant events. The types of notifications include pre-appointment, no-show, cancellation by facility, cancellation by patient, pending list, long-term appointment request pending list, ready to schedule, and generic correspondence. Configurable “templates” must be provided so that the notification letter can be reformatted if so desired. This is accomplished through the use of Extensible Stylesheet Language (XSL) style sheets.



Notification information is contained in the Notification data object. This (Java) object is first Extensible Markup Language (XML) serialized, and then an XSL stylesheet is applied using the Xalan XSL transformer (Apache). The resulting Hypertext Markup Language (HTML) is displayed in a Java HTML Editor Pane (Figure 6). The letter may be printed to a user-selectable printer using the PrintManager utility.



**Figure 6. Rendering and Printing a Notification Letter**

#### 2.3.2.4 Daily Operations Components (ops)

**Package:** gov.va.med.rsa.ops.gui.ops

The Daily Operation package contains the GUI components that allow the user to submit an appointment request, view possible appointment opportunities, schedule an appointment, and view appointments that have been scheduled. This package also provides components that display pending appointment requests for an entire section and appointment history for an individual patient. Components in the Daily Operations package are further explained in the following sections.

#### **2.3.2.4.1 Appointment Operations (appt)**

The Appointment Operations component contains the GUI classes to display appointments for a given patient, allows the appointment details to be viewed, and supports the saving of ancillary information for those appointments. This component also contains appointment management functionality, such as linking appointments or modifying appointment statuses (e.g., canceling or rescheduling appointments).

#### **2.3.2.4.2 Appointment Request Operations (apptreq)**

The Appointment Request Operations component contains the classes that handle all appointment request tasks, including making appointment requests, viewing appointment opportunities received from the search, managing request statuses, and supporting viewing of the requests.

#### **2.3.2.4.3 Patient Operations (patient)**

The Patient Operations component contains the classes that handle the retrieval of patient header information using the Swing Application invocation of the Patient Lookup Service. Using the Patient Lookup Service requires setting user parameter information, calling the Patient Lookup dialog, and then calling the method to retrieve a portion of the patient's demographic information. This component provides the capability to manage patients within named groups. This capability provides the user with a way to add new patients to a named group, activate/inactivate patients in a named group, and maintain patient appointment statuses for a group appointment.

#### **2.3.2.4.4 Scheduling Operations (sched)**

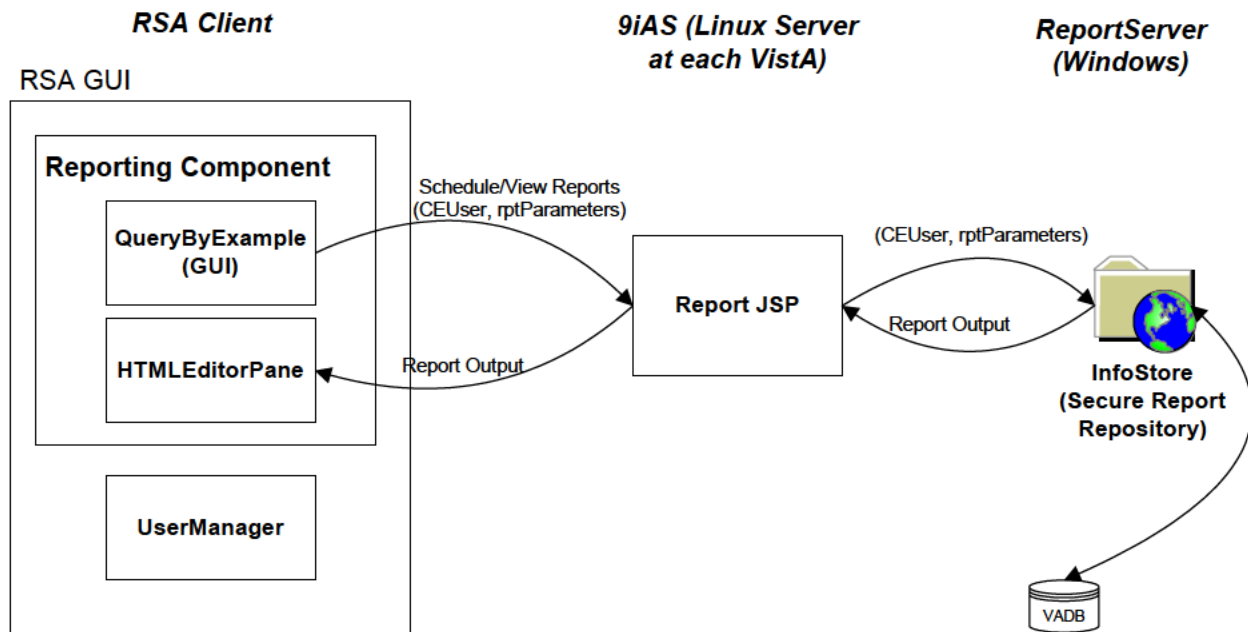
The Scheduling Operations component contains the classes that handle the management of appointment views for a section or resource. Various views (i.e., compact list and calendar view) are provided to support various user viewing needs and Section 508 requirements. The Resource Daily Calendar View displays the resource's daily schedule in a calendar style based on the user-selected administrative entity and the user-selected day of the month. The Section Daily Calendar View displays the section's resources daily schedule in a calendar style based on the user-selected section and the user selected day of the month. The compact list view provides the same information from the calendar views but in a table format.

#### **2.3.2.5 Reporting Component (report)**

**Package:** gov.va.med.rsa.ops.gui.report.

“Crystal Enterprise” will be used for generating reports within RSA. This will require a Windows server to function as the report server. A Java Server Pages (JSP) application will reside on the local 9iAS (Linux) server. “Managed” reports are stored in Crystal Enterprise's report repository called the “InfoStore.” User access to reports is controlled through a set of

operations and permissions set in the UserManager component as well as through Crystal Enterprise. Reports are then displayed in an HTMLReporterPane as shown in Figure 7. Batch reports are scheduled through Crystal Enterprise.



**Figure 7. Reporting Component**

### 2.3.2.6 RSA Configuration Components (rsaconfig)

**Package:** gov.va.med.rsa.ops.gui.rsaconfig.

The RSA Configuration package provides the GUIs that are used by a System Administrator to configure and maintain the RSA to support scheduling and schedule administration. RSA Configuration GUIs provide for the input of information and setting to be used VHA-wide.

This package allows the user to manage the RSA configuration. It is further divided into the components described in the following sections.

#### 2.3.2.6.1 Administrative Entity Configuration (ae)

The Administrative Entity Configuration component contains a set of screens where authorized users can view AE attributes that are configured in the AE Tree Service (Section 2.3.6.1) as well as configure AE Parameters. AE parameters can be configured at any administrative entity to set defaults or impose values to schedulable entities. Some examples of AE attributes are AE name, AE abbreviation, and AE full path. Some examples of AE parameters are No-show Count, Number of Call Backs, Overbooks Minutes per 4-Hour Period, and Planning Horizon.

#### **2.3.2.6.2 Change Configuration (change)**

This component provides GUIs to maintain Configuration Change Package (CCP) information. The CCP allows the user to group certain types of configuration changes. All changes in a CCP will be activated at the same time and will have the same effective dates. The CCP structure also helps users find conflicts with changes that they are making and resolve them by making further changes, which are also placed into the CCP. These screens provide functionality such as viewing, modifying, validating, and activating change packages as well as adding change items to a CCP and resolving conflicts from a change.

#### **2.3.2.6.3 Resource Configuration (resource)**

This component provides GUIs to maintain RSA resource and resource allocation information. These screens provide functionality for assigning a resource to an AE, as well as the creation, modification, and inactivation of resources and resource allocations.

#### **2.3.2.6.4 Standard Configuration (standard)**

The Standard Configuration component contains a set of screens where authorized users can enter national, VHA-wide information. Such information includes National Appointment Purposes, Holidays, Patient Special Needs, and Notification Templates.

**National Appointment Purpose:** There is a National Appointment Purpose screen where an authorized user can create and edit National Appointment Purposes. A National Appointment Purpose is comprised of a name, a description, and an associated DSS-ID that a user will enter.

**Holiday Schedule:** The Holiday Schedule screen allows an authorized user to create and edit a holiday name and its date for the holidays that the Veterans Affairs (VA) observes.

**Patient Special Needs:** The Patient Special Needs screen allows an authorized user to create and edit a special needs option to the list of special needs that can be selected for a patient with a special need. The screen includes a name and description of the special need.

**Notification Template:** A Notification Template is a set of parameterized and combinable pieces of text that are used to create messages that can be delivered through standard notification delivery mechanisms (such as letters) and that are associated with a specific type of notification list. The types of notifications include pre-appointment, no-show, cancellation by facility, cancellation by patient, pending list, long-term appointment request pending list, ready to schedule, and generic correspondence.

### **2.3.2.7 Section Configuration Components (sectionconfig)**

**Package:** gov.va.med.rsa.ops.gui.sectionconfig.

The Section Configuration Package provides the GUI components that are used by a schedule administrator to configure and maintain an organization that schedules appointments. Section Configuration GUIs support the management of scheduling parameters such as hours of operation, overbooking limitations, individual resource schedule constraints, and patient appointment notifications.

Components in this package allow a user to manage section configurations. They are described in the following sections.

#### **2.3.2.7.1 Appointment Purpose Configuration (apptpurpose)**

The Appointment Purpose Configuration component contains screens to allow an authorized user to create and configure local appointment purposes and appointment purpose sets. A local appointment purpose is used for scheduling appointments; and during creation, it is associated with a national appointment purpose and the national appointment purpose's associated DSS-ID. An appointment purpose set is a set of multiple local appointment purposes that will be scheduled together.

**Local Appointment Purpose:** The Local Appointment Purpose screen allows a user to enter a name, abbreviation, description, and select a resource set to use with that purpose. The appointment purpose is also assigned to a schedulable entity. A local appointment purpose also has preparation instructions for the purpose and indicates whether or not the purpose is for an individual or a group, and indicates whether or not medical records or x-rays will be needed for an appointment.

**Appointment Purpose Set:** The Appointment Purpose (AP) Set screen allows a user to enter a name, abbreviation, and description of the set, and also to select appointment purposes from different sections to add to the set. For each entry in the set, minimum and maximum time intervals will be added to indicate the amount of time that should occur before the next appointment.

#### **2.3.2.7.2 Resource Configuration (resource)**

This component provides GUIs to maintain resource allocation configuration and resource set information. These GUIs will be used by a schedule administrator, rather than a system administrator, to maintain resources and resource allocations. The screens will provide functionality for the modification of a resource allocation's configuration and creation and the modification and inactivation of resource sets.

#### 2.3.2.7.3 Scheduling Configuration (sched)

This component provides GUIs to maintain schedule template information. The screens will provide functionality such as creation, modification, and deactivation of resource and section carve-out blocks. Carve-out blocks can disallow appointments in a portion of the schedule, restrict a portion of the schedule to only certain types of appointments, and change the planning horizon for a portion of the schedule (open access).

#### 2.3.2.7.4 Section Configuration (section)

The Section Configuration component is a set of screens that allow authorized users to configure attributes that are applicable only to schedulable entities. These attributes include selecting and creating Check In Points (CIPs), specifying notification templates to use for each kind of notification, setting up hours of operation, configuring operation permissions, and creating named groups.

**Check In Points:** The Check In Points screen will display the administrative location of the selected Section and allow authorized users to select an existing CIP in the facility or enter a new CIP for the section to use.

**Standard Notification:** The Standard Notification screen allows a user to select whether or not a notification of that type will be sent out to patients. If the section chooses to send that notification, a notification template must be selected. The pre-appointment notification must also have the notification days specified to indicate the number of days before the scheduled appointment that a notification will be sent. The no-show notification must also have a consecutive no-show value to indicate when a notification should be sent.

**Hours of Operation:** The Hours of Operation screen allows a user to specify when the section is open. A block of time can be chosen to add to the hours of operation or to remove from the hours of operation. Hours of operation can be viewed weekly or monthly and can be set for a day of the week, a day of the month, a specific date, or every nth day of the week.

**Operations:** The Operations screen lists the operations of a section. The section can indicate for each operation if anyone can perform this operation, only specific roles or explicit users can perform the operation, or if no one should perform this operation.

**Patient Group:** The Patient Group screen allows a user to create a new named group. A name, description, and group size should be specified for each group that is created. A user can also modify attributes of an existing group on this screen. The list of patients assigned to the group will be visible on the screen, but the user cannot modify the list of patients from this screen (that is done from the Patient Group Configuration package).

### 2.3.2.8 Client-Tier Utility Components (util)

**Package:** gov.va.med.rsa.ops.gui.util.

This package contains components that provide utilities used by multiple components within the client tier. This consists mainly of GUI utility classes, as well as the calendar control and printing utility.

#### 2.3.2.8.1 Calendar Control

This component provides calendar widgets that will be used in other RSA screens. These widgets include a calendar which displays an entire day in vertical columns, a small panel which displays a monthly calendar and allows the user to choose a date, and a widget similar to a combo box which, when opened, displays a small monthly calendar and allows the user to choose a date.

#### 2.3.2.8.2 Print Manager

The Print Manager is a Java class that wraps the Java Application Programming Interface (API) for printing to either a local printer or a network printer. This class is used for relatively simple print jobs that do not warrant creating a report.

XSL stylesheets are used to format the item to be printed for rendering and display. The data to be printed is contained within an RSA Value object. The object is first XML serialized and the stylesheet is applied. Using the Xalan transformer (Apache) as shown in Figure 8, the transform takes place and the document is displayed as an HTML page in a Java HTML Editor Pane. The document is displayed exactly as it will be printed. Utility methods within PrintManager allow the user to select a printer and print the document.

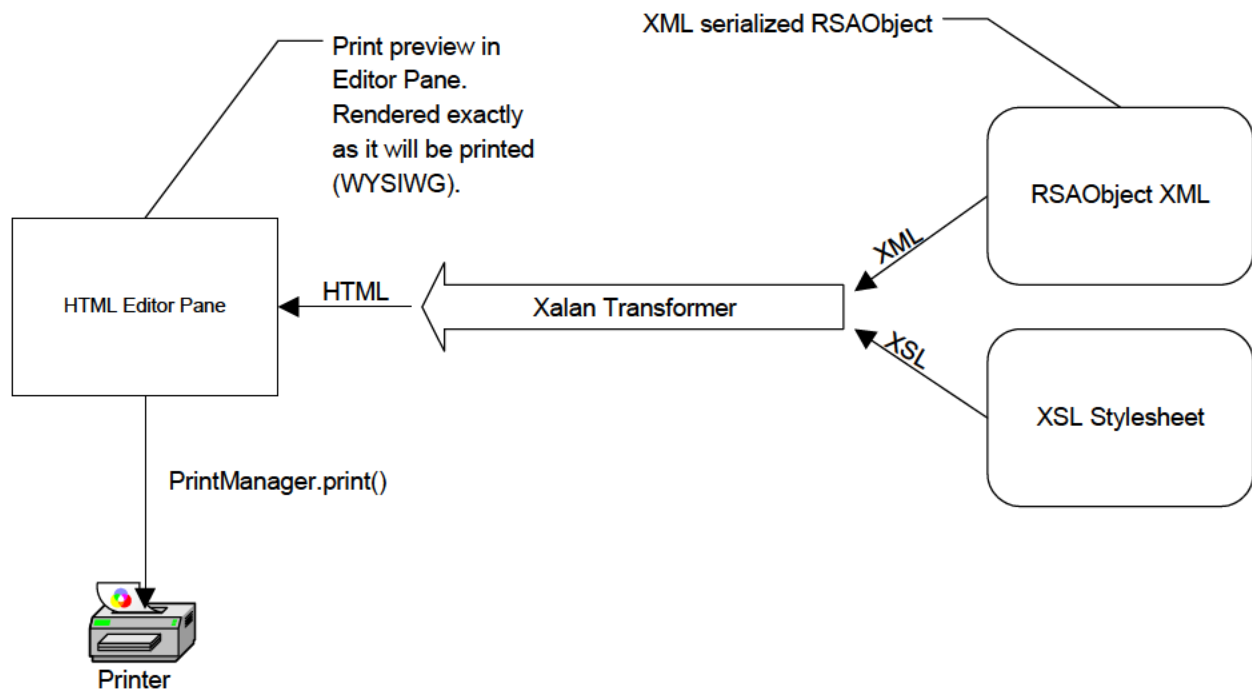
The Print Manager is used when the print job is fairly simple and all data to be printed is already present in the GUI. An example of this would be printing an appointment slip immediately after booking a new appointment.

### 2.3.3 Middle-Tier Components

**Package:** gov.va.med.rsa.ops.biz.

RSA's middle tier consists of multiple web services, Transaction Managers (TM), and Data Tier Components (DTC). These components form three layers within the middle tier as shown in Figure 9:

- **Interface/Integration Layer:** These are web service Java classes. Web Service Description Language (WSDL) is generated from these classes that provide the interface for the client application.



**Figure 8. PrintManager Uses XSL to Format the Print Job**



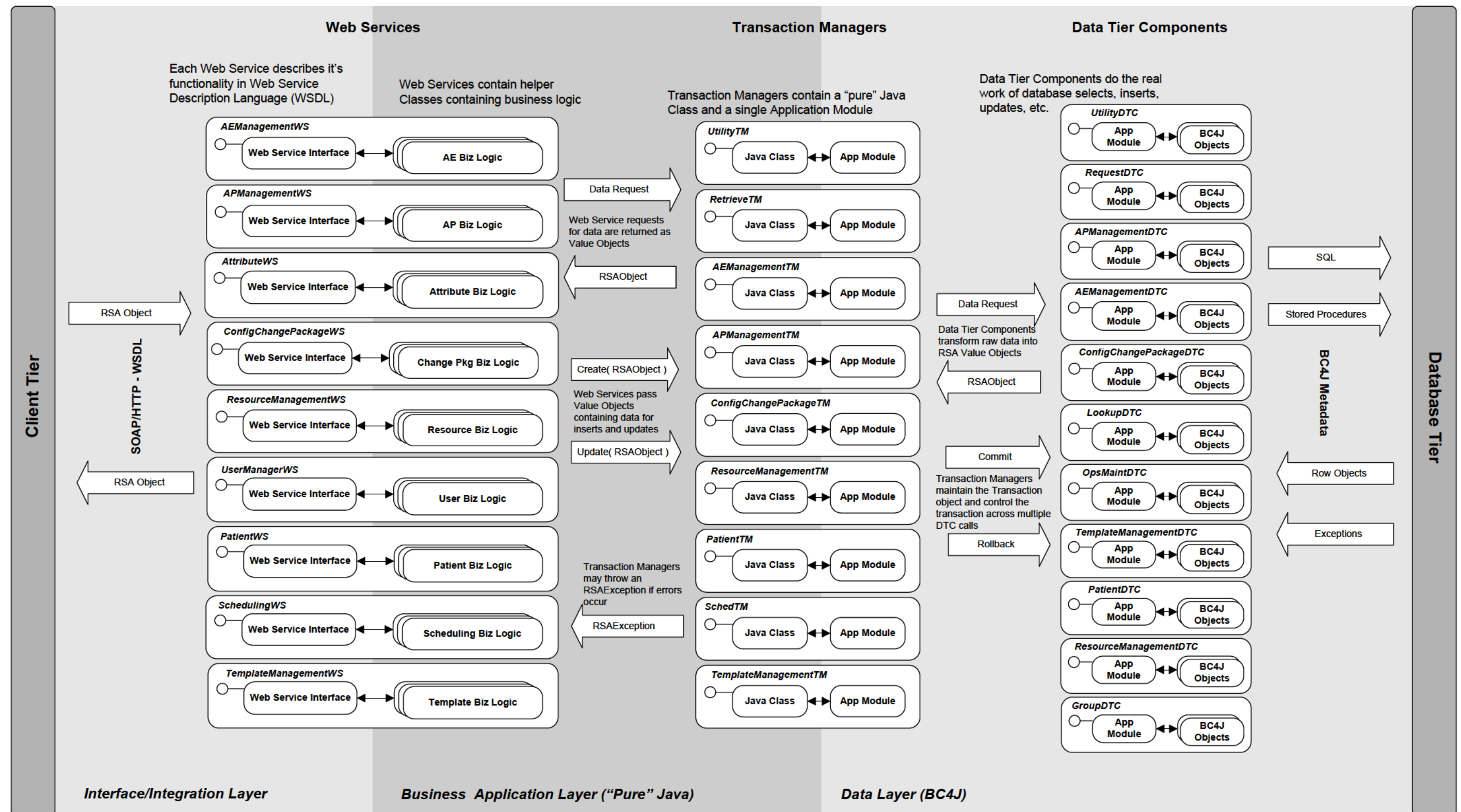


Figure 9. RSA Middle Tier

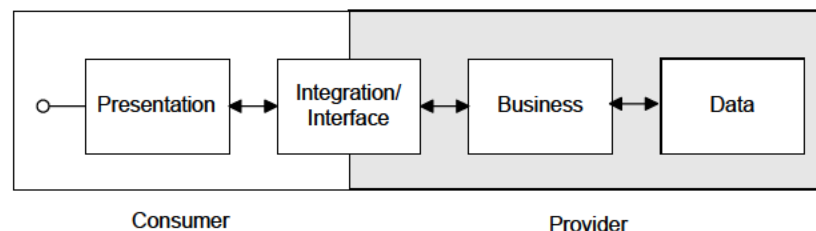
- **Business Application Layer:** This layer is contained within web service helper classes that perform data validation, logging, etc., as well as “pure” Java classes within the TM components that allow these components to be deployed outside the web service’s Oracle Containers for Java (OC4J) container. Pure Java classes contain code that is not derived from, or dependent on, BC4J libraries.
- **Data Layer:** Components in the Data Layer contain the BC4J classes that perform all data access. They each contain an Application Module, ViewObjects, EntityObjects, and other classes and code to perform database selects, inserts, updates, etc.

### 2.3.3.1 Web Services

RSA consists of multiple web services. The logical architecture of web service enterprise applications includes the three traditional layers of presentation, business, and data but also includes an integration/interface layer as shown in Figure 10. This layer exposes the service’s functionality to the consumer.

The RSA integration/interface layer provides a set of standardized methods for interacting with other RSA software components within the architecture. RSA’s middle and client tiers represent a logical grouping of RSA web service providers and web service consumers.

Web services apply protocols such as XML and SOAP to provide business services while also utilizing these technologies to achieve platform and programming language independence. The RSA uses Remote Procedure Call (RPC) Style web services for its means of communication. RPC-Style web services package a method call and parameter list into a SOAP document; the application server receiving the request is responsible for translating that method call into the appropriate Java objects, executing the right method calls, and finally repackaging the results back into a SOAP message to be sent to the client. This means of communication is used throughout RSA, creating a model that supports loosely coupled components and an efficient process of change.



**Figure 10. Web Service and Consumer Logical Architecture**

All of the RSA’s web services fully describe themselves in WSDL. Using WSDL, a web service can describe everything about what it does, how it does it, and how consumers of that web service can go about using it.

#### **2.3.3.1.1 Administrative Entity Web Service (adminentity)**

This Administrative Entity Web Service contains the business logic and rules that are related to AE management within RSA. It provides business logic functionality and the interface methods for the AE and standard table GUIs. This package provides the following functionality:

- View AE tree
- Maintain AE
  - Retrieve AE general information
  - Retrieve/update AE parameters
- Maintain schedulable AE
  - Maintain CIPs
  - Select notification templates
  - Maintain CIP operating hours
  - Maintain patient groups
  - Maintain operation permissions
- Maintain RSA standard table
  - Maintain holidays
  - Maintain special needs
  - Maintain facility level patient notification templates

To retrieve data from the database, the Administrative Entity Web Service will send requests to the data layer. The Administrative Entity Web Service will mainly request and update information in the AE Management DTC but will also request information from Resource Management, Template Management, and Appointment Purpose Management DTC.

#### **2.3.3.1.2 Schedule Management Web Service (cmn.opsfw)**

The Schedule Management Web Service exposes interfaces that are used to coordinate the activities needed to schedule an appointment.

The Request component contains the business logic and rules that provide the client application with the capability of validating and saving an appointment request. It also provides interfaces to maintain request statuses. Finally, it provides lookup services that return request information to the client application. The Appointment Request GUI components invoke the Request component.

The Search component contains the business logic and rules that match search criteria into available appointment opportunities. Appointment opportunities can be requested by resource or section. The Search package also supports the following permission based searches: override, overbook, or override/overbook. The Appointment Request GUI components invoke the Search component.

The Appointment component provides services for validating and saving appointments as well as saving ancillary test information. Along with scheduling appointments, the Appointment

component contains classes that support linking and unlinking related appointments, updating appointment information, looking up appointment information to return to the client, and maintaining appointment statuses. The Appointment Request GUI components invoke the Appointment component.

#### **2.3.3.1.3 Appointment Purpose Web Service (apptpurpose)**

This web service component contains the set of classes that implement the business logic and the business rules that are related to appointment purpose management. It also provides web service interface functions for the remote Appointment Purpose GUI components. The web service interface functions are published from the main business process Java class. These interface functions provide the client with the capability to create, modify, remove, and retrieve national appointment purposes, local appointment purposes, and appointment purpose set information.

#### **2.3.3.1.4 Attribute Management Web Service (attribute)**

The Attribute Management Web Service is the interface for the GUI for retrieving attribute management type static data. This web service will receive calls from the attribute data manager on the GUI and will interact with the Utility Transaction Manager to get the requested data.

#### **2.3.3.1.5 User Authorization Web Service (authorization)**

The User Authorization Web Service consists of the classes necessary for the GUI to get user authorization information. This service controls the process of retrieving the user from the OID, storing the user object locally on the application server, and sending the user object back to the RSA client application.

#### **2.3.3.1.6 Configuration Change Web Service (configchange)**

The Configuration Change Web Service contains the set of classes that implement the business logic and the business rules that are related to Configuration Change Package (CCP) management. It also provides web service interface functions that are published from the main business process class for the remote configuration GUIs. These interface functions provide the client application with the capability to create, modify, and inactivate section operating hours, local appointment purposes, resources, resource allocations, and section and resource carve-outs.

#### **2.3.3.1.7 Patient Web Service (cmn.opsfw)**

The Patient Web Service contains the set of classes that implement the business logic and the business rules that are related to maintaining patient named groups and retrieving patient information. The Patient Web Service supports adding and removing patients from named groups, updating statuses for named group members, and named group membership retrievals. Additionally, the Patient Web Service retrieves patient information from Patient Services. Patient Group Configuration and Patient Operation GUIs invoke Patient Web Services.

#### **2.3.3.1.8 Resource Management Web Service (resource)**

The Resource Management Web Service contains the set of classes that implement the business logic and business rules that are related to resource management. It also provides web service interface functions that are published from the main business process class for the resource configuration and scheduling appointment GUIs. These interface functions provide the client application with the capability to create, modify, remove, and retrieve resources, resource types, resource sets, and resource allocation information.

#### **2.3.3.1.9 Template Management Web Service (template)**

The Template Management Web Service contains the set of classes that implement the business logic and the business rules that are related to template management. It also provides web service interface functions that are published from the main business process class for the template GUIs. These interface functions provide the client application with the capability to maintain resource and section carve-out blocks. Carve-out blocks can disallow appointments in a portion of the schedule, restrict a portion of the schedule to only certain types of appointments, and change the planning horizon for a portion of the schedule (open access).

#### **2.3.3.2 Middle-Tier Common Components (cmn)**

This package contains Java classes that are used by various middle-tier components and are common among these components. It contains classes for data validation and other functionality that is used throughout the middle tier.

#### **2.3.3.3 Middle-Tier Utilities (util)**

The biz.util package contains utility classes used throughout the middle tier such as the RSABusinessUtilities and UserManager classes described in the next two sections.

##### **2.3.3.3.1 RSABusinessUtilities**

RSABusinessUtilities is a class containing several static methods responsible for converting one commonly used type within RSA to another, for instance, converting a java.util Date to an Oracle.JBO.Domain Date.

##### **2.3.3.3.2 UserManager**

This class is the interface used by all web services and transaction managers in the RSA to perform user authorization checks. The UserManager class encapsulates the necessary algorithms to determine if a user may perform a specific process at a specific administrative entity level. This determination is based upon roles, explicit permissions, and section configuration information.

#### **2.3.3.4 External Interface Components**

External interface components are modeled around Cross Application Integration Protocol/Project/Prototype/Specification (CAIP) compliant proxy adaptors and facade adaptors. The interface components implement various communications technologies depending upon the type of application that is being interfaced. This list includes the interface engine (Vitria), VistALink, and Java-to-Java (EJB, web service) applications as shown in Figure 11. RSA implements several independent adaptors to correspond with the chosen technology of the other applications that RSA must interact with. These implementations include Hypertext Transfer Protocol (HTTP) Connectors (Servlets), EJB connections, web service connections, and VistALink interfaces.

##### **2.3.3.4.1 Event Servlets (eventservlets)**

**Package:** gov.va.med.rsa.ops.eventservlets.

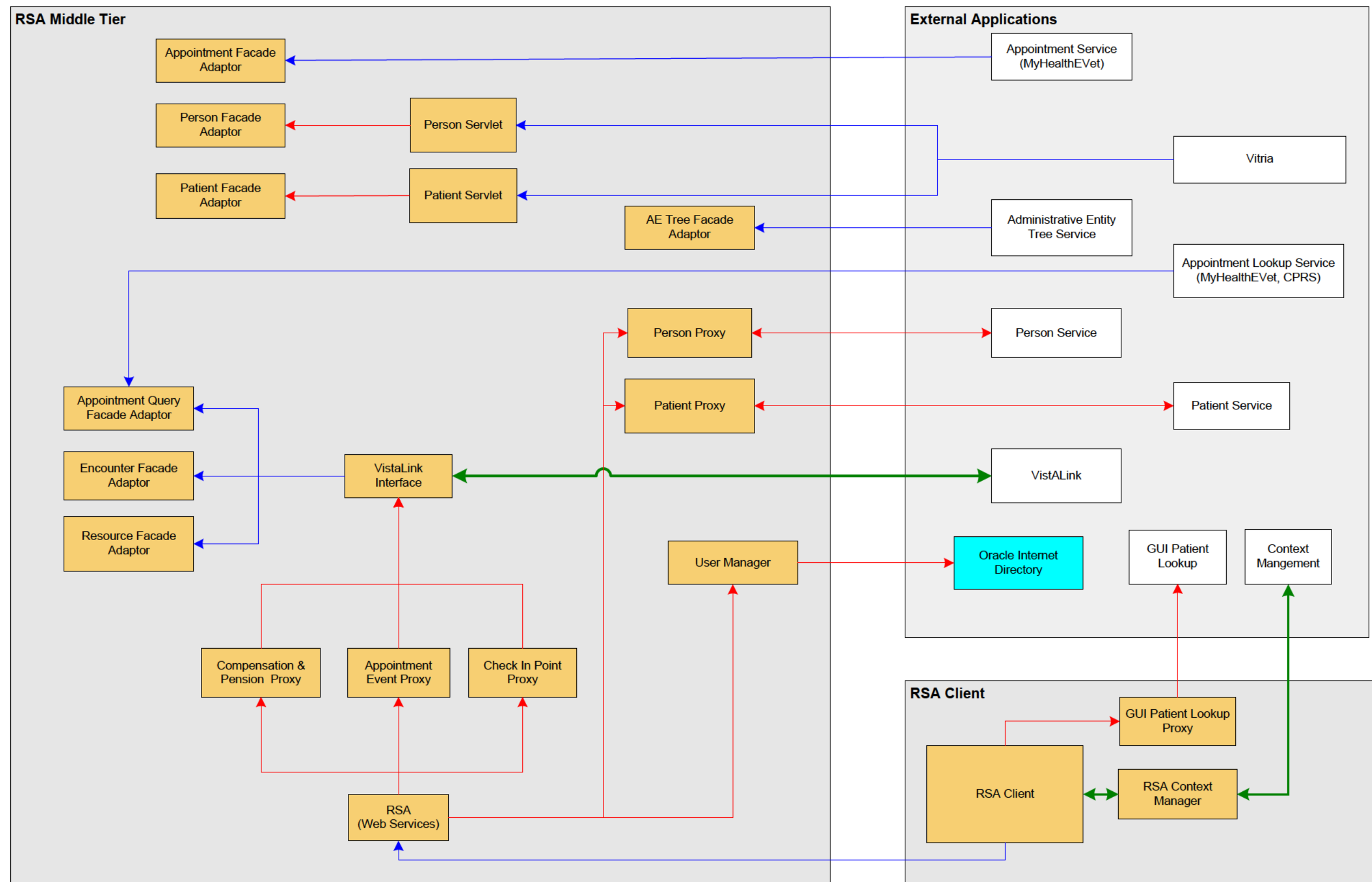
HTTP servlets are the communications mechanisms that outside services use to send asynchronous events to RSA. These outside services will send an HL7 message to the Vitria Interface Engine. The Vitria Interface Engine will then do a lookup in an internal configuration file to determine what connectors are registered to receive the HL7 message. RSA will register three HTTP Connectors with Vitria for a variety of HL7 messages. The URL's for these HTTP Connectors will connect to two event servlets residing on the RSA web server. Vitria will call the doPost() method on the servlet passing in the HL7 message as a parameter.

###### **2.3.3.4.1.1 Patient Maintenance Event Servlet (patient)**

The Patient Maintenance Event Servlet component is deployed on the 9i web server and contains a servlet class that acts as an HTTP connector for the Vitria interface engine. This PatientMaintenanceEventServlet class implements the Java Interface httpconnector and has a single doPost() method that receives as a parameter an HL7 message. The PatientFacadeAdaptorBusinessDelegate class implements the code necessary to connect to the Patient Facade Adaptor EJB that is running in the 9i Application Server. This delegate is the same delegate that outside services will use to connect to the Patient Facade Adaptor EJB. This implementation was chosen to enhance maintenance efforts.

###### **2.3.3.4.1.2 Person Maintenance Event Servlet (person)**

The Person Maintenance Event Servlet component is deployed on the 9i web server and contains a servlet class that acts as an HTTP connector for the Vitria interface engine. This PersonMaintenanceEventServlet class implements the Java Interface httpconnector and has a single doPost() method that receives as a parameter an HL7 message. The PersonFacadeAdaptorBusinessDelegate class implements the code necessary to connect to the Person Facade Adaptor EJB that is running in the 9i Application Server. This delegate is the same delegate that outside services will use to connect to the Person Facade Adaptor EJB. This implementation was chosen to enhance maintenance efforts.



#### **2.3.3.4.2 Proxy Adaptors (proxies)**

**Package:** gov.va.med.rsa.ops.biz.proxies.

A Proxy Adaptor is a Java class that implements the business delegates needed to connect to outside services. Proxy adaptors allow the RSA to be technology independent and encapsulate the necessary behavior to communicate with the outside world. As technologies used by other services change, then the only part of RSA that needs to change would be the proxy adaptor.

##### ***2.3.3.4.2.1 Appointment Event Proxy (apptevent)***

The Appointment Event Proxy encapsulates five appointment events: the booking of a new appointment, a patient not showing up for an appointment, canceling of an appointment, a patient checking in to a section, and a patient checking out of a section. The Appointment event proxy is called by several of the RSA transaction managers when one of these events occurs. The Appointment Event Proxy sends each of the five events to Scheduling Encapsulation.

##### ***2.3.3.4.2.2 Compensation and Pension Proxy (candp)***

The Compensation and Pension Proxy encapsulates a single event, the retrieval of Compensation and Pension (C&P) data from Scheduling Encapsulation. The Compensation and Pension Proxy will be called by the Appointment Transaction Manager and will pass the request on to Scheduling Encapsulation. Once Scheduling Encapsulation returns, then this data is sent back to the Schedule Transaction Manager.

##### ***2.3.3.4.2.3 Check In Point Proxy (checkinpoint)***

The Check In Point Proxy encapsulates three events: CIP creation, CIP inactivation, and CIP activation. The Check In Point Proxy will receive these events from the Administrative Entity Transaction manager and the Configuration Change Package Transaction Manager. The Check In Point Proxy will then pass the request to Scheduling Encapsulation utilizing the VistALink interface. The Scheduling Encapsulation returns an Internal Entry Number (IEN) on the creation event and that IEN is passed back to the Administrative Entity Transaction Manager.

##### ***2.3.3.4.2.4 Local Purpose Proxy (localpurpose)***

The Local Purpose Proxy encapsulates three events: local purpose creation, local purpose inactivation, and local purpose activation. The Local Purpose proxy will receive these events from the Appointment Purpose Transaction manager and the Configuration Change Package Transaction Manager. The Local Purpose proxy will then pass the request to Scheduling Encapsulation utilizing the VistALink interface.



#### **2.3.3.4.2.5 Patient Proxy (patient)**

The Patient Proxy is a java class with a single public method. This method is called by the Patient Web Service to initiate the process of retrieving a set of patient demographics information from Patient Services. The Patient Proxy will create an instance of the Patient Services business delegate and use it to retrieve the set of patient demographics data. The search criteria for Patient Services is the patient's Integration Control Number (ICN). The set of patient demographic data returned will then be turned into an internal RSA patient object and returned to the Patient Web Service.

#### **2.3.3.4.2.6 Person Proxy (person)**

The Person Proxy is a java class with a single public method. This method is called by the Resource Transaction Manager to initiate the process of retrieving a set of provider information from Person Services. The Person Proxy will create an instance of the Person Services business delegate and use it to retrieve a set of provider data. There are several search criteria mechanisms [VHA Person Identifier (VPID), last name, partial name, last four digits of Social Security Number] that may result in the return of multiple sets of provider data. This data will then be turned into internal RSA person objects and returned to the Resource Transaction Manager.

#### **2.3.3.4.3 Facade Adaptors (facadeadp)**

**Package:** gov.va.med.rsa.ops.biz.facadeadp.

A Facade Adaptor is an entity that encapsulates the RSA's interface to the outside world. Facade Adaptors are implemented in various technologies including EJB, VistALink, and Vitria.

##### **2.3.3.4.3.1 Administrative Entity Tree Adaptor (aetreeadaptor)**

The Administrative Entity Tree Adaptor encapsulates the behavior of three events: creation of a new AE tree node, modification of an AE tree node, and the deactivation of an AE tree node. The events will be encapsulated in an enterprise Java bean interface. The AE Tree Service will initiate all events to this adaptor. The Administrative Entity Tree Adaptor will receive the events and then pass the information to either the Administrative Entity or the Configuration Change Package Transaction Manager to handle the changes. There will not be a return notification to the AE Tree Service when the update is complete.

##### **2.3.3.4.3.2 Appointment Adaptor (apptadaptor)**

The Appointment Adaptor is an interface that allows external applications the ability to make appointment and appointment requests without going through the RSA Client. This adaptor publishes five methods. Two of the methods are for appointment request creation and three are for appointment creation. To accomplish create appointment request or create appointment, the application must first retrieve a set of information from the RSA that can be used to actually

make the request or make the appointment. This process is shown in Sections 2.5.3.12, and 2.5.3.13. These events will be encapsulated in an enterprise Java bean interface. They will be initiated from an outside service, most likely MyHealthVet. These events will be received, the data verified, and then passed on to the Schedule Transaction Manager for processing. The return data will then be sent back to the requesting service.

#### ***2.3.3.4.3.3 Appointment Query Adaptor (apptqueryadaptor)***

The Appointment Query Adaptor encapsulates the behavior of a single appointment query event, the retrieval of appointment information. This event has a variable parameter list so the calling method can choose the query parameters and also select the information about the appointment that should be returned. The request can come in via VistALink or via an EJB interface. The request will come to the adaptor which will verify the request parameters and then pass the request on to the Appointment Lookup Transaction Manager for processing. The return data will be sent back to the calling service.

#### ***2.3.3.4.3.4 Encounter Adaptor (encounteradaptor)***

The Encounter adaptor encapsulates a single event, the association of the encounter identifier to the appointment identifier (ID). This method can be called via VistALink or via an EJB interface. The input parameters include the appointment ID, the encounter ID, and the encounter type. The encounter type should include enough information to ensure that the proper appointment state can be achieved.

#### ***2.3.3.4.3.5 Patient Adaptor (patientadaptor)***

The Patient Adaptor encapsulates a single event, the processing of a patient maintenance event. The event will be initiated by the Patient Maintenance Servlet when the servlet receives an event from Vitria. There are several different HL7 message types that can be sent as a parameter in the patient maintenance event including an Admission Discharge Transfer (ADT) message. The Patient Adaptor will receive the HL7 message and will then parse the message to find the exact type of HL7 and determine the type of maintenance event. The events that RSA processes (such as patient merge, name change, etc.) will then be sent on to the Patient Transaction Manager for processing and storage in the RSA database.

#### ***2.3.3.4.3.6 Person Adaptor (personadaptor)***

The Person Adaptor encapsulates a single event, the processing of a person maintenance event. The event will be received from the Person Maintenance Servlet. The Person adaptor will receive an HL7 message and will parse the message to find the exact type of maintenance event. Those events that RSA processes (such as VIP merge, provider class change, etc.) will then be sent on to the Resource Transaction Manager for processing.

#### 2.3.3.4.4 VistALink Interface (vistalinkintf)

**Package:** gov.va.med.rsa.ops.ext.vistalinkintf.

VistALink is a transport layer that allows Java to communicate with Massachusetts General Hospital Utility Multi-Programming System (MUMPS, or M) remote procedures. VistALink is based on standard technologies, both on the Java and M sides. VistALink 2.0 is to be a transportation layer that also allows an M procedure to make a J2EE compliant EJB call allowing a Java application to communicate to an M procedure. The VistALink interface is utilized for all communications between the RSA and the Scheduling Encapsulation M procedures inside the **VistA** application.

#### 2.3.3.4.5 Vitria Interface (vitria)

**Package:** gov.va.med.rsa.ops.biz.ext.vitria.

Vitria BusinessWare is a third party communications mechanism designated by the VA to be utilized for asynchronous data events. It encapsulates all the knowledge of reliable and guaranteed data delivery. Vitria BusinessWare is deployed on a Windows 2000 server and may not be co-located with the RSA Application Servers. Vitria utilizes a configuration file located on the Vitria server where applications can subscribe (by manually editing this file) to certain events sent by other applications. This configuration file holds the name of the event and connection information to those applications subscribing to this event. RSA will deploy three HTTP connectors that will subscribe to various events. One connector will subscribe to patient maintenance events, one will subscribe to person maintenance events, and one will subscribe to AE Tree Service events.

#### 2.3.3.5 Data Layer Components

**Package:** gov.va.med.rsa.ops.biz

The data layer in RSA consists of multiple Java components that implement BC4J. To decouple the data layer from the business application layer, the following types of components have been defined:

- **Transaction Managers:** These components manage a single Transaction object across multiple calls to the database through multiple DTCs. Transactional control is maintained in each TransactionManager
- **Data Tier Components:** These components contain ApplicationModules, ViewObjects, EntityObjects, and other BC4J classes that perform all Data Manipulation Logic (DML) operations such as queries, inserts, and updates. Additionally, they perform data transformations from raw database data to RSA Data Objects.

#### **2.3.3.5.1 Transaction Managers (tm)**

Transaction Managers (TMs) provide the capability to manage transactions that span across one or more DTCs as shown in Figure 12. TMs contain a “pure” (no BC4J dependencies ) Java class that is instantiated by the caller (e.g., web service or adaptor). By providing this object-based interface, the abstract database operates from business logic, allowing not only small-scale masking of database changes, but also a means to restructure the data layer independently of any business logic changes. TMs also contain an ApplicationModule derived class that contains the Transaction object that will be used for multiple DTC calls.

##### ***2.3.3.5.1.1 Administrative Entity Transaction Manager (adminentity)***

The Administrative Entity Transaction Manager provides transaction control for AE activities such as modifying AE parameters, modifying a CIP, adding and modifying a section group, and changing section permissions. It also serves as the interface to the database tier for any other AE related activities.

##### ***2.3.3.5.1.2 Resource Transaction Manager (resource)***

The Resource Transaction Manager provides the interface for the resource management activities and related calls to the AE DTC.

##### ***2.3.3.5.1.3 Template Transaction Manager (template)***

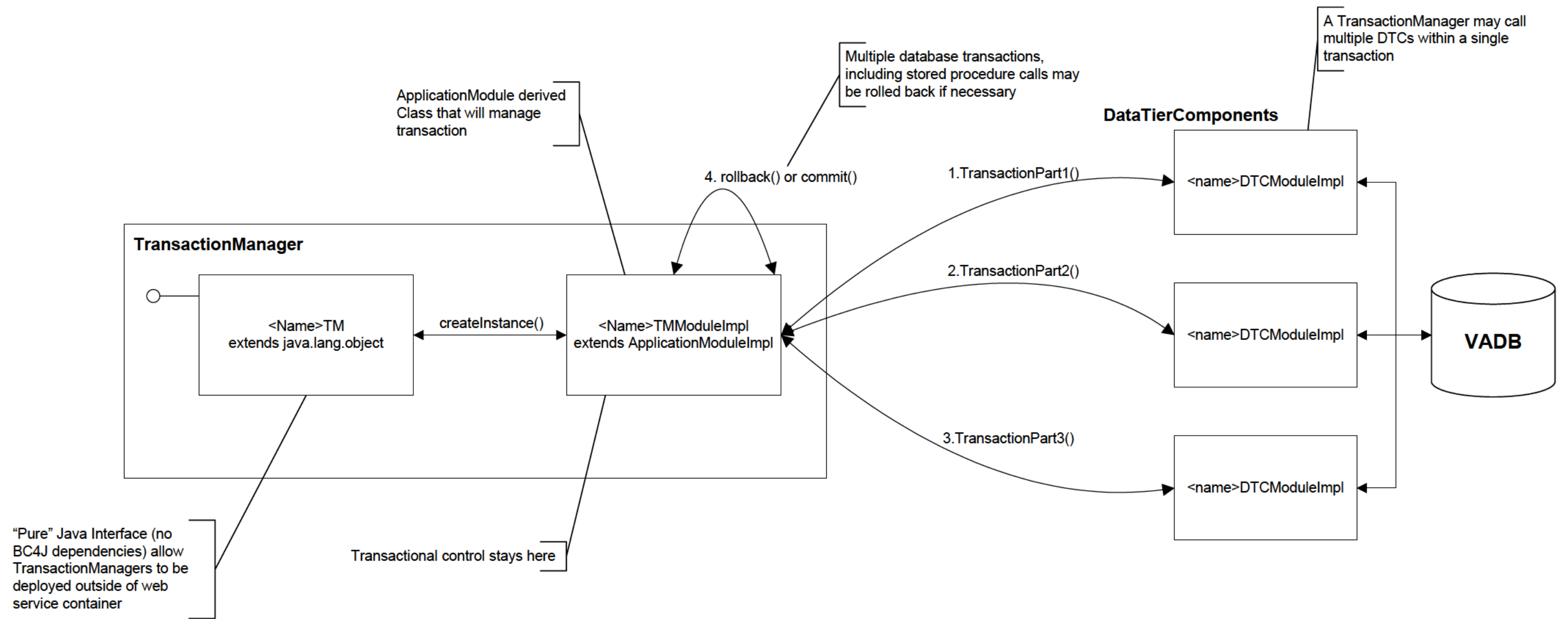
The Template Transaction Manager provides the interface for schedule template activities and related calls to other configuration DTC.

##### ***2.3.3.5.1.4 Appointment Purpose Transaction Manager (apptpurpose)***

The Appointment Purpose Transaction Manager provides the interfaces for configuring appointment purpose activities and related calls to other configuration DTCs.

##### ***2.3.3.5.1.5 Configuration Change Package Transaction Manager (configchange)***

The Configuration Change Package Transaction Manager provides transaction management for configuration change package activities such as creating, modifying, and removing change packages and items; validating change packages; and activating change packages, as well as the interface for other change package related activities.



**Figure 12. TransactionManagers Maintain Transactional Control**

#### ***2.3.3.5.1.6 Schedule Transaction Manager (sched)***

The Schedule Transaction Manager provides transaction control for the following three functional areas: requests, searches, and appointments. Specifically saving an appointment requires the appointment data to be saved as well as updating the appointment's associated request status.

#### ***2.3.3.5.1.7 Retrieval Transaction Manager (retrieve)***

The Retrieval Transaction Manager provides the interfaces for both search activities and appointment and request data retrievals.

#### ***2.3.3.5.1.8 Patient Transaction Manager (patient)***

The Patient Transaction Manager provides transaction control for the various types of patient events received from external sources. These activities include the following:

- Patient Merge
- Patient Death
- Patient Revival
- Patient Name Change
- Ward Location Change
- Admission Status Change

In the event a patient notification is received for either a patient merge, patient death, or patient revival, the Patient Transaction Manager will update the patient information and then perform the appropriate appointment, request, and named group maintenance activities.

#### ***2.3.3.5.1.9 Utility Transaction Manager (utility)***

The Utility Transaction Manager will cache local copies of semi-static data for use by other middle-tier components. In the event that a local copy does not exist or the cache is determined to be stale, this component will go through the Utility DTC to go to the database and retrieve the database version of the data. Examples of this include error messages, business rules objects, state machines, etc.).

#### **2.3.3.5.2 BC4J Data Tier Components (dtc)**

**Package:** gov.va.med.rsa.ops.biz.dtc.

Data Tier Components (DTC) are the set of classes and interfaces (Figure 13) that give business logic within RSA a means of communication with the database. A DTC is used by creating an instance of its ApplicationModule derived class. This class creates ViewObjects that contain ViewRow Objects that are the result sets returned from queries. ViewObjects execute structured query language (SQL) on the database directly. Results are cached in EntityObjects for enhanced performance. In many cases, callable stored procedures are invoked directly from the ApplicationModule to further increase performance as well as return the sequence value on inserts.

##### **2.3.3.5.2.1 Administrative Entity DTC (*adminentity*)**

This component provides the link to the database for data related to AE management within RSA. It provides business logic functionality and the interface methods for other business layer components that need AE information. This package provides update and retrieve functionality for AE tree data, AE general information, schedulable AE data, and RSA standard table data.

##### **2.3.3.5.2.2 Appointment DTC (*appt*)**

The Appointment DTC contains classes that allow appointment business logic a means to save both appointments and ancillary tests. It effectively abstracts and masks the communications with the database that are required to save single patient appointments, ad-hoc group appointments, and named group appointments.

##### **2.3.3.5.2.3 Lookup DTC (*lookup*)**

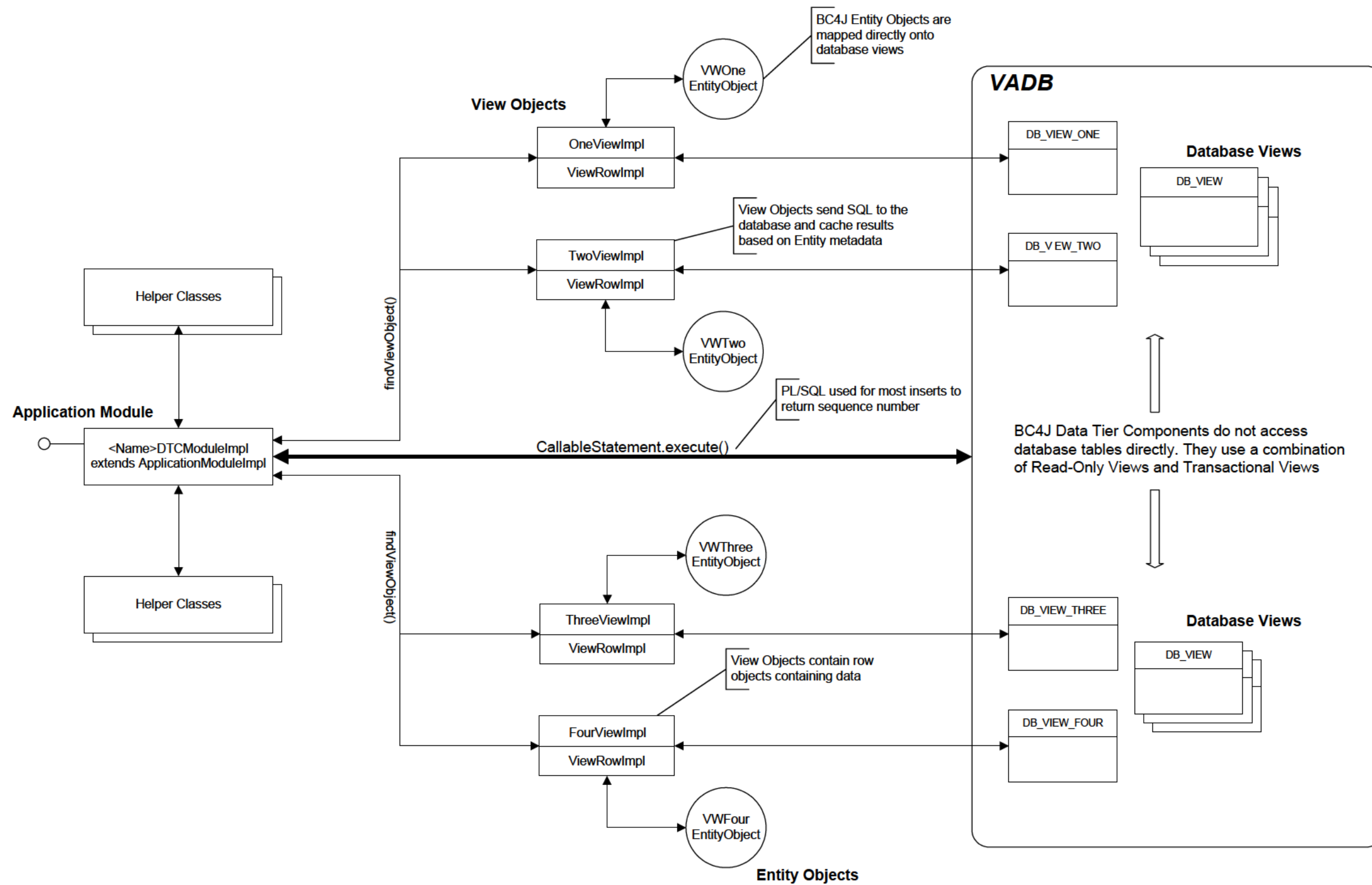
The Lookup DTC contains the classes that support the retrieval of appointment and request information. It provides functions to return request list information, appointment query event information, and appointment and request data to support the RSA GUI.

##### **2.3.3.5.2.4 Request DTC (*request*)**

The Request DTC is responsible for saving an appointment request to the RSA database. It provides a layer of abstraction above the RSA database which gives other components the ability to save an appointment request without knowing the details of the underlying database.

##### **2.3.3.5.2.5 Maintenance DTC (*opsmaint*)**

The Maintenance DTC contains the classes that provide appointment and request maintenance functions. These functions include updating appointment and request base data, linking and unlinking appointments, and changing appointment and request statuses.



**Figure 13. Structure of a Data Tier Component (DTC)**



#### **2.3.3.5.2.6 *Search DTC (search)***

The Search DTC contains the classes necessary to allow search business logic a means to access search constructs in the database. It provides, for the purpose of searching, a read-only view of this data structured in such a way that allows efficient and timely retrieval of critical search data.

#### **2.3.3.5.2.7 *Patient DTC (patient)***

The Patient DTC supports the patient update functions needed to keep the RSA patient information synchronized with the Patient Services information.

#### **2.3.3.5.2.8 *Appointment Purpose DTC (apptpurpose)***

The Appointment Purpose DTC contains the data access methods for appointment purpose management. It also provides the internal data access methods for other business layer packages that have a need for appointment purpose information. These methods provide direct access to and from database views that allow creating, modifying, removing, and retrieving data to and from database tables. Database tables for appointment purpose management contain information for national appointment purposes, local appointment purposes, and appointment purpose sets.

#### **2.3.3.5.2.9 *Configuration Change DTC (configchange)***

The Configuration Change DTC contains the data access methods for configuration change package management. These methods include retrieving, modifying, removing, and creating configuration change package information to and from the database. The Configuration Change Package Transaction Manager invokes the Configuration Change DTC.

#### **2.3.3.5.2.10 *Patient Group DTC (group)***

The Patient Group DTC contains classes that support named group functions. These functions include adding and removing patients from named groups, updating named group membership statuses, and providing group membership retrieval methods.

#### **2.3.3.5.2.11 *Resource DTC (resource)***

This package contains the data access methods for resource management. It also provides the internal methods for other business layer packages that have a need for resource management information and the business logic for such methods. The methods include retrieving, updating, and inserting Resources, Resource Types, Resource Sets, and Resource Allocation information from/to the database.

#### **2.3.3.5.2.12 Template DTC (template)**

This component contains the data access methods for template management. It also provides the internal methods for other business layer packages that have a need for template management information. Templates consist of resource and section carve-out blocks. Carve-out blocks can disallow appointments in a portion of the schedule, restrict a portion of the schedule to only certain types of appointments, and change the planning horizon for a portion of the schedule (open access).

#### **2.3.3.5.2.13 Utility DTC (util)**

The Utility DTC is called by the Utility Transaction Manager to refresh its cache of semi-static and rarely changing data. This DTC retrieves data from the RSA database such as appointment types, appointment categories, and medical priorities.

### **2.3.4 RSA Database – VADB**

The database for RSA is a data repository designed to support a number of functional needs of the production system. Figure 14 illustrates the essential design structure of that database as it is to operate in the production RSA environment.

As Figure 14 shows, the RSA database includes sets of database tables and views to support:

1. Daily operations of the systems especially with respect to the scheduling and management of appointments for patients.
2. Configuration data management allowing the various administrative entities to configure the data they need to perform scheduling operations.
3. Reporting functionality.
4. Internal operations of the RSA application itself.

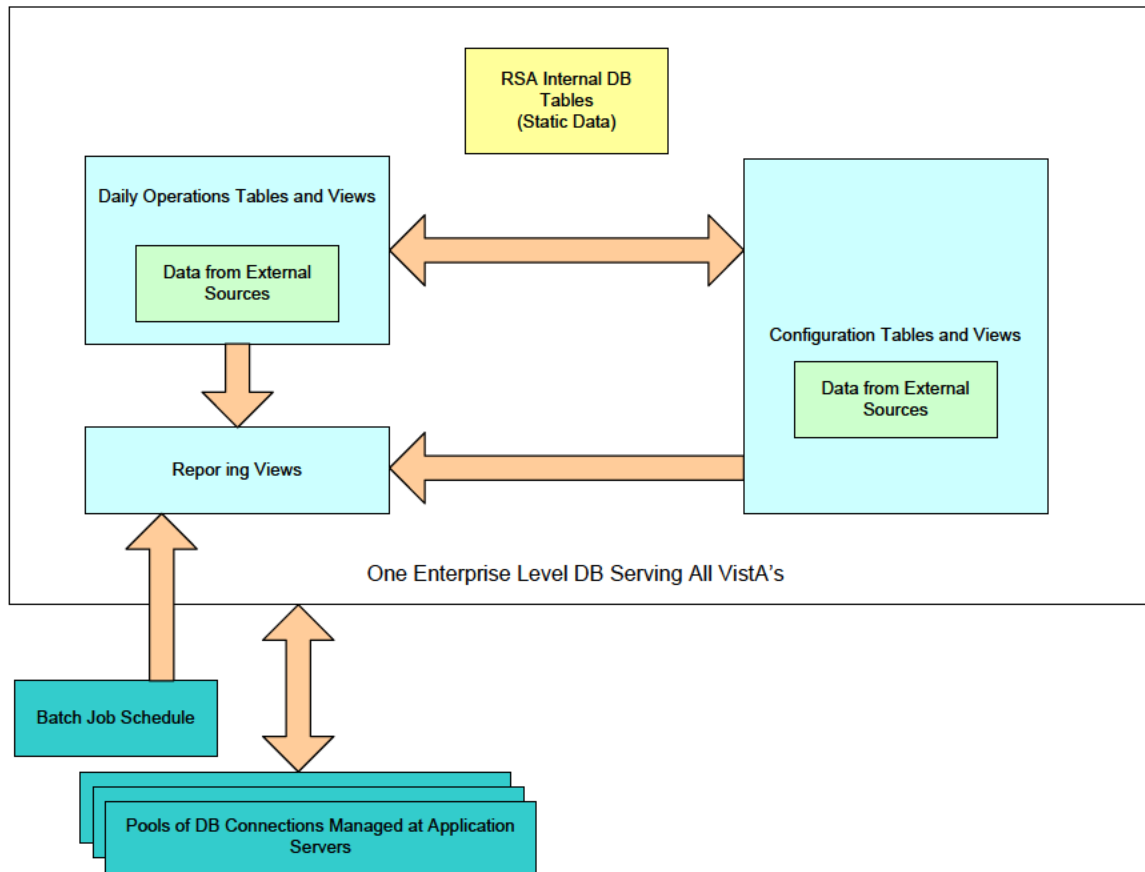
### **2.3.5 RSA Utilities**

**Package:** gov.va.med.rsa.ops.util.

This package contains multiple Java classes for providing utility functionality useful throughout RSA. These utility classes perform various tasks such as XML serialization/deserialization, data type conversions, data validation utilities, logging, and exception handling.

### **2.3.6 Other Components**

Components that do not fall into one of the previous categories of components are described here.



**Figure 14. Essential Data Structure**

### 2.3.6.1 Administrative Entity Tree Service

The AE Tree Service (ATS) will provide the VHA Administrative Entity (AE) hierarchy information for RSA. Implementation of ATS will supply the necessary information about the AE hierarchy such that RSA can configure scheduling rules and schedule the appointments.

ATS basic functionalities:

- 1) ATS will maintain the entire VHA AE information as a tree data structure from the VHA (as the root) to the Schedulable Entity level (as the leaf). ATS will store the AE Category (AE categories include VistA Instance, Facility, Schedulable AE, and other) associated with the appropriate AE.
- 2) ATS will maintain AE types in order to provide roll-up levels for RSA Reporting features.
- 3) ATS will provide a mechanism to publish the messages to RSA for the existing tree information, the tree structure changes, and the AE basic information changes.

- 4) The ATS will be part of the RSA software, and its interface will not be exposed to applications outside of RSA.

Please note that the ATS deployment is proposed as an interim solution because of the lack of VHA service. The proposal to fund ATS has not been approved at the publication of this document.

### **2.3.6.2 Oracle Internet Directory (OID)**

OID is a Lightweight Directory Access Protocol (LDAP) provider. OID is a standards-based directory service based on the Oracle 9i Database. It can be acquired with both Oracle 9i Application Server and Oracle 9i Database. Directory services are designed to hold a wide variety of information about people, network devices, resources, and other objects. This information follows an assumption that the ratio of data updates to data access will be relatively low.

The RSA will use OID as a repository for all user authorization and personalization information. This information will have high data access frequency but will maintain low data update frequency, following the assumptions of directory service information. Authorization data includes users roles and privileges for an identified administrative entity and will be used to determine if a user has the privileges to perform a given operation in the RSA. Personalization data consists of display formats set by the user along with other elements that help to bring about a better user experience when working in the RSA. This information will be stored as a directory entry in OID and Java Naming and Directory Interface (JNDI) will be used to store and retrieve authorization and personalization details.

## **2.4 System Resources**

Refer to Section 6, Infrastructure View of the Baseline System Architecture Document (BSAD), for a discussion on issues related to system resources within RSA.

## **2.5 Concept of Execution**

The following paragraphs describe the concept of execution among RSA's software modules. In the "4+1" architectural views, this section represents the process view. This section includes state diagrams, sequence diagrams, and various other diagrams that describe the dynamic relationship among the software modules during system operation.

## **2.5.1 Configuration**

Figure 15 shows the RSA configuration components and the flows of execution of the configuration activities between the client and middle-tier components. The general activities in the client tier are viewing, updating, and creating RSA configuration information using various GUIs and sending GUI requests through the Data Managers to the appropriate web service components in the middle tier. Activities in the middle tier consist of:

- Receiving the request, verifying user authorization, and performing the business logic using the web service components.
- Managing the transaction control for the cross component's functionalities using the Transaction Manager (TM) components.
- Accessing the database for retrieving/saving data using the DTC.

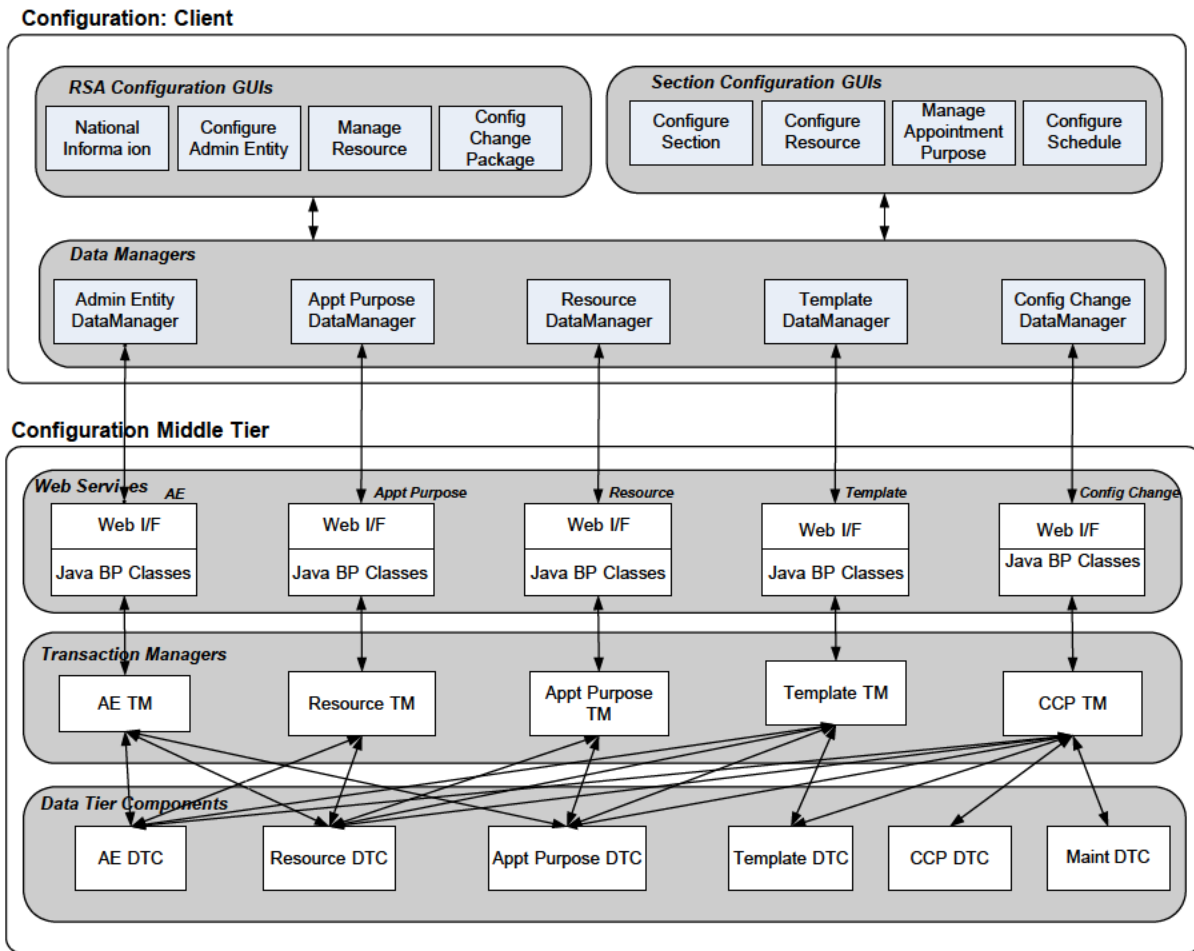
In the following paragraphs, state refers to the state of a change to an RSA configuration item referred to as an operation. Configuration within RSA contains the following three state machines:

- State machine for CCP related operations
- State machine for managing CCP operation
- State machine for normal operations that are not part of a CCP

### **2.5.1.1 State Machine for CCP Related Operations**

The first state machine is for entities associated with operations that run as part of CCP. RSA allows users to generate and manage configuration changes with multiple user actions, such as inputting changes, validating changes, resolving conflicts, and activating changes. Therefore, this state machine shows the state of an entity at any given time, from creation of the operation to the cancellation of the operation or the effective date of the change package. The operations that use this state machine are:

- 1) Adding, modifying, and inactivating CIP normal hours of operation;
- 2) Decreasing resource quantity;
- 3) Inactivating a resource;
- 4) Adding, modifying, and inactivating a resource allocation;
- 5) Inactivating a local appointment purpose; and
- 6) Adding, modifying, and inactivating carve-outs.

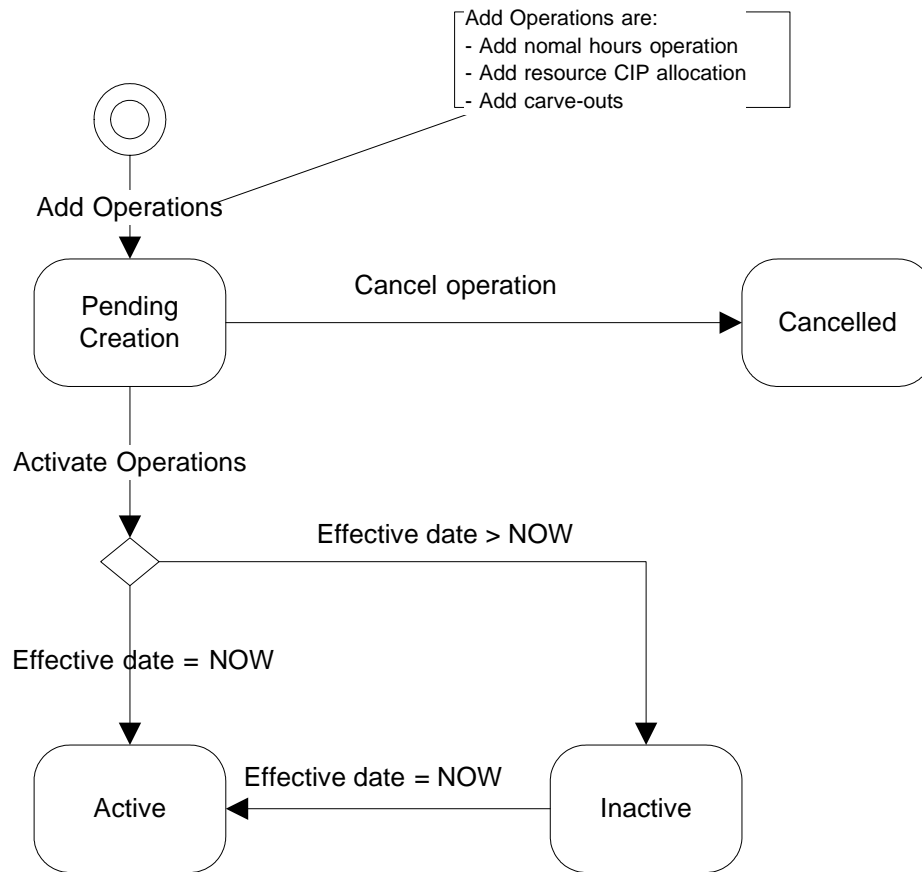


**Figure 15. RSA Configuration Components**

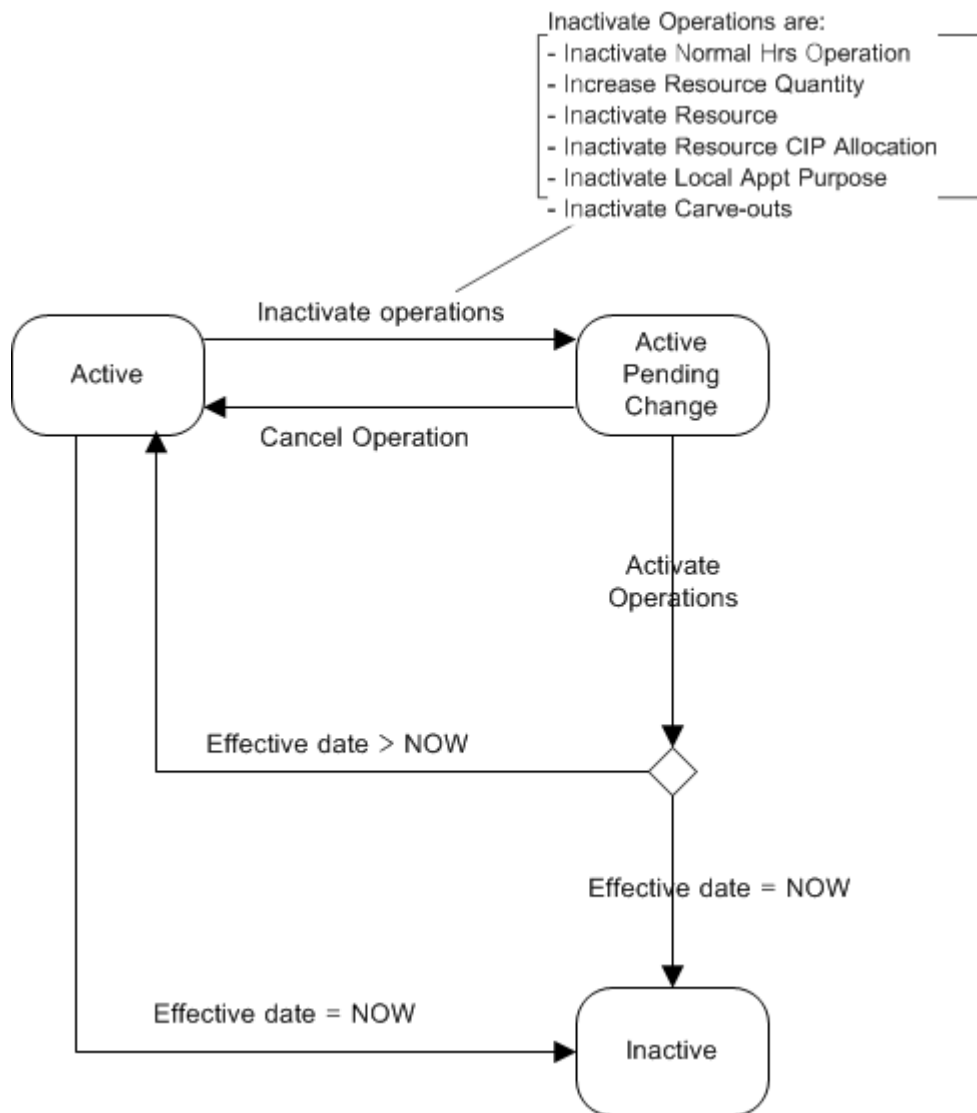
Because of the complexity that is involved with CCP-related operations, two separate state diagrams are used to describe the state of entities during these operations. The first diagram (Figure 16) shows states during operations that add entities. The second diagram (Figure 17) depicts the state of entities during operations that inactivate a CCP item. When an entity is modified, the old version of the entity is deactivated and the new one is activated so both diagrams apply. The third diagram (Figure 18) combines the first two diagrams to describe the whole entity state diagram.

Entity state during add operations is shown in Figure 16. These operations consist of:

- Adding CIP normal hours of operation.
- Adding CIP resource allocation
- Adding Carve-outs

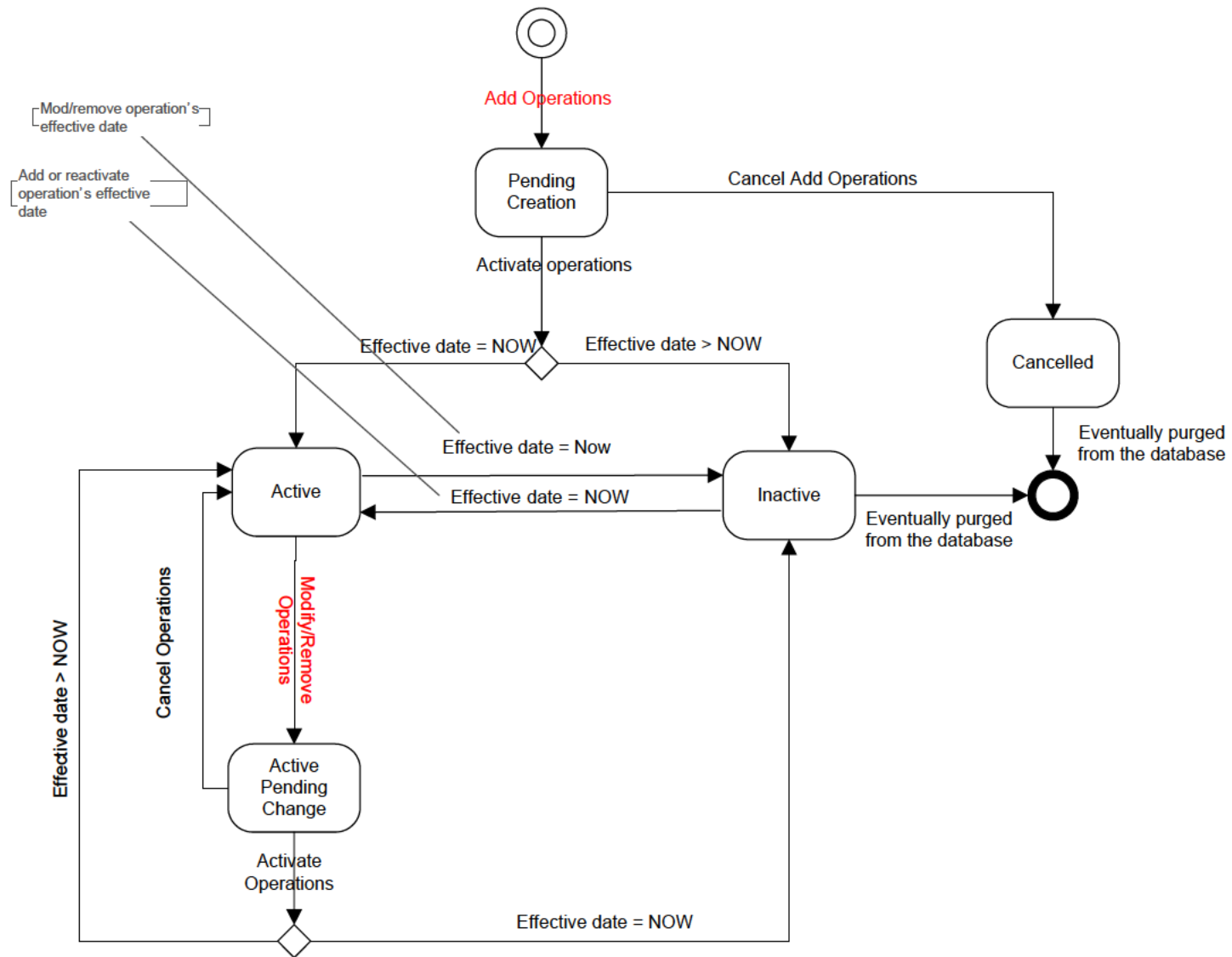


**Figure 16. State Machine for Entities During Add Operations**



**Figure 17. State Machine for Entities During Inactivation Operations**





**Figure 18. State Machine for CCP Related Operations**

When an add operation is created for a new entity, the entity is placed in a state of “Pending Creation.” In this state, the entity is saved in the database but it is only visible to the current CCP and does not affect daily operations. If the operation add an entity is cancelled or removed from the change package, that entity’s state becomes "Cancelled." If the CCP is activated, the entity’s state depends on the effective date of the CCP. The entity is activated as of the effective date of the CCP.

An entity state during operations that inactivate an operation is shown in Figure 17. These operations consist of:

- Inactivating normal hours operation.
- Decreasing resource quantity.
- Deactivating a resource.
- Inactivating a resource CIP allocation.
- Deactivating a local appt purpose.
- Inactivating carve-outs.

An entity that is inactivated via a CCP begins in the "Active" state; it is an active part of the system. When an deactivate operation related to an entity is added to the CCP, the entity is placed into the "Active Pending Change" state, which means that it is still an active part of the system, but it cannot be added to any other CCPs. If the operation is cancelled or removed from the CCP, the entity is returned to the "Active" state. If the CCP is activated, the state of the entity depends on the effective date of the CCP. The entity is deactivated as of the effective date of the CCP.

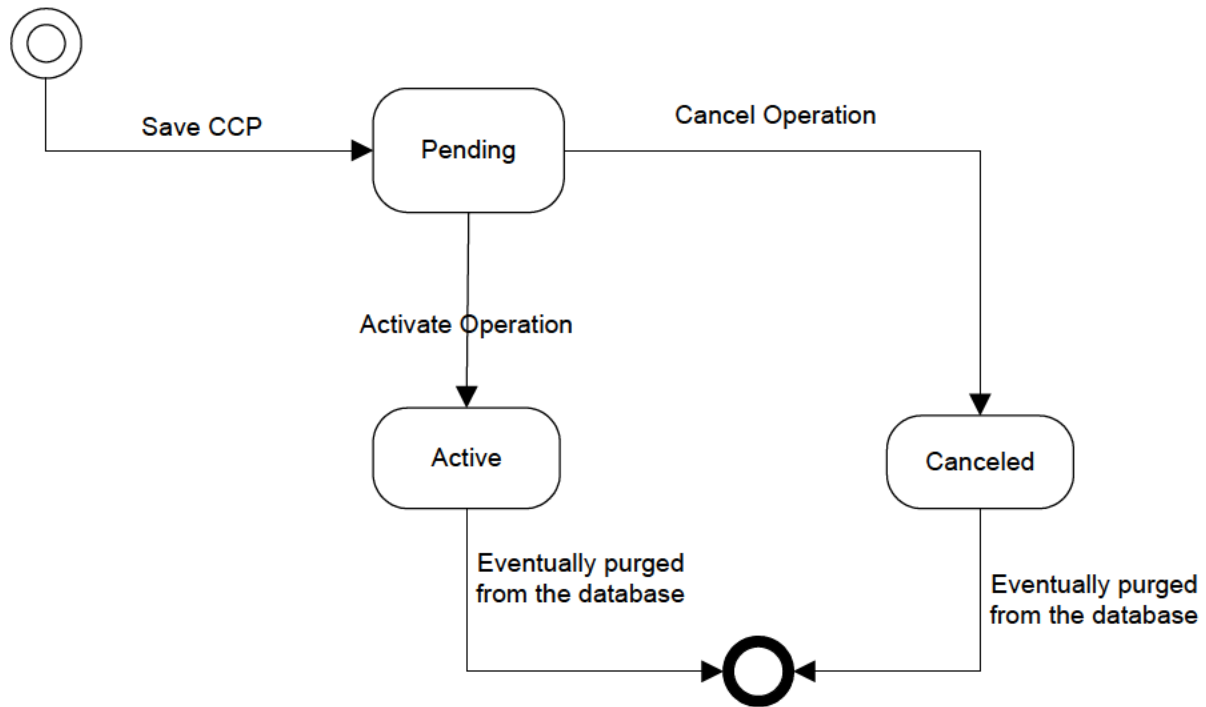
Overall CCP operations are shown in Figure 18. It describes entity state for both adding and inactivating operations.

### **2.5.1.2 State Machine for CCP Entities**

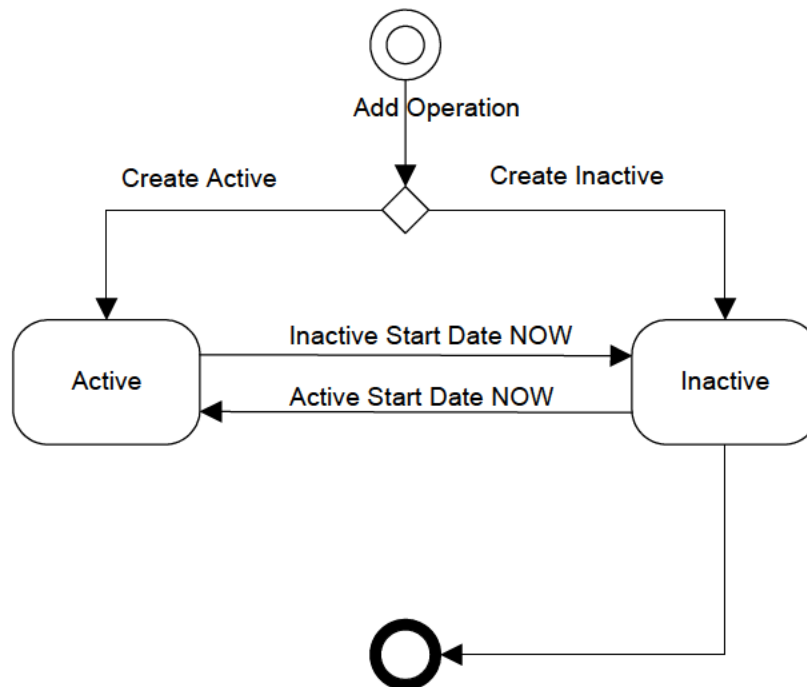
Figure 19 depicts the entity state for those entities that make up the CCP. When an initial CCP operation is created, a CCP entity containing CCP name, operation type, and a pointer to the operation is created and put into a state of “Pending.” When activating the CCP, the CCP entity becomes “Active.” When the CCP is cancelled, the CCP entity becomes “Canceled.” CCPs in a state of “Active” or “Canceled” can eventually be purged from the database.

### **2.5.1.3 Normal Base Data Item State Machine**

A base data item is any item that can be manipulated directly by middle-tier components. Figure 20 depicts the base data item state for operations that are not part of a CCP; for instance, creating a resource, modifying AE parameter values, creating a national appointment purpose, or deactivating a resource type. These operations are effective as of the date specified as the effective date of the operation. Items in the “Inactive” state may eventually be purged from database.



**Figure 19. State Machine for Maintaining CCP Operations**



**Figure 20. Normal Base Data Item State Machine**

#### **2.5.1.4 Build RSA Configuration**

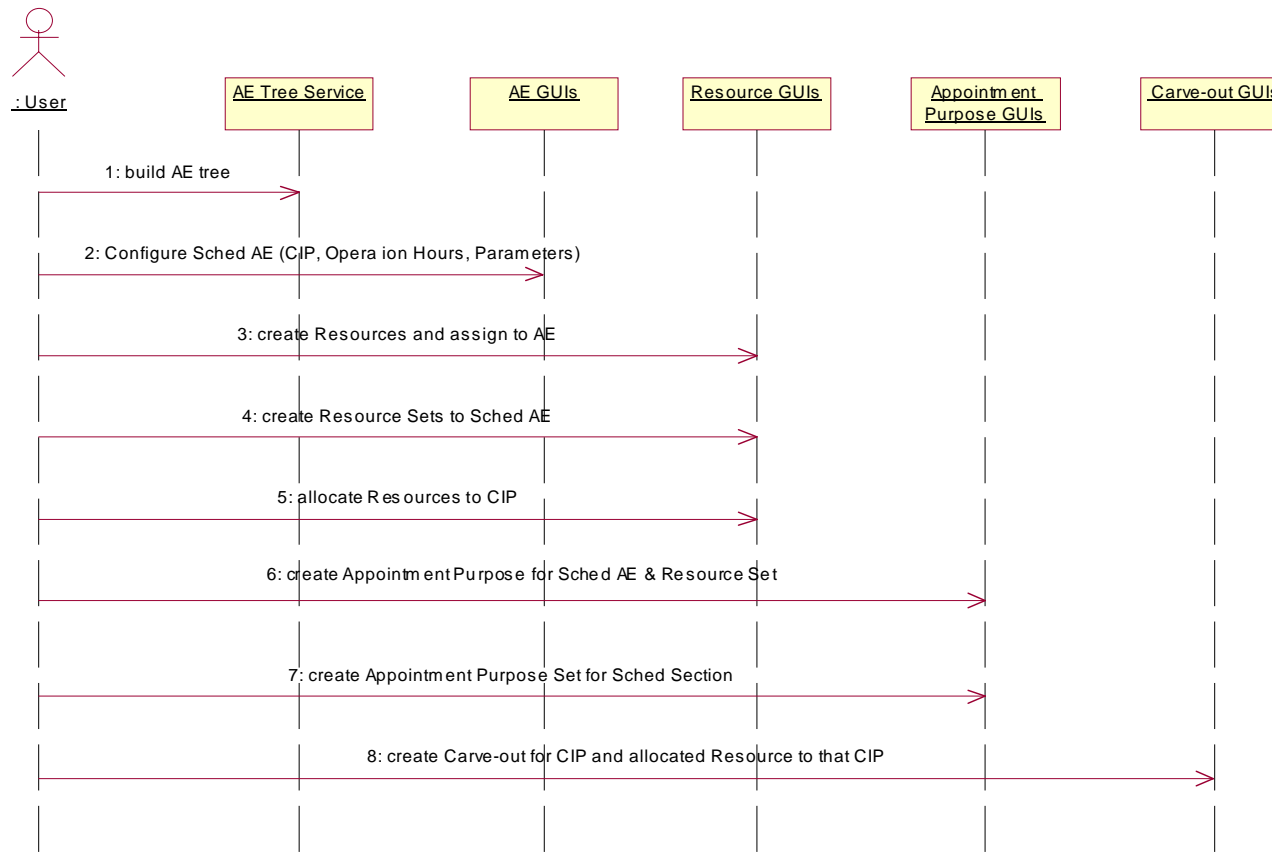
The main goal of the configuration component is to build an environment that can provide flexible scheduling information for daily operations, such as making appointments. Figure 21 shows the sequence of events for setting up the initial configuration environment to be used for scheduling. RSA configuration starts with building the AE Tree. Users enter the AE information using ATS. ATS allows users to create and maintain AEs in a tree hierarchy and send AE tree information to RSA. Once the AE Tree has been configured, users create CIPs and define the hours of operation and various scheduling parameters on the schedulable AE. Users are allowed to create and assign resources to the AE, allocate Resources to a CIP, and create resource sets for a CIP. To make resources available for scheduling, the resources have to be a part of a resource set. Next, users need to build the appointment purposes on the schedulable AE and build the relationship to the Resource Set. Finally, users can create carve-outs on the CIP or on the resource allocation using the Appointment Purposes and Patient Eligibility.

#### **2.5.1.5 Create CIP**

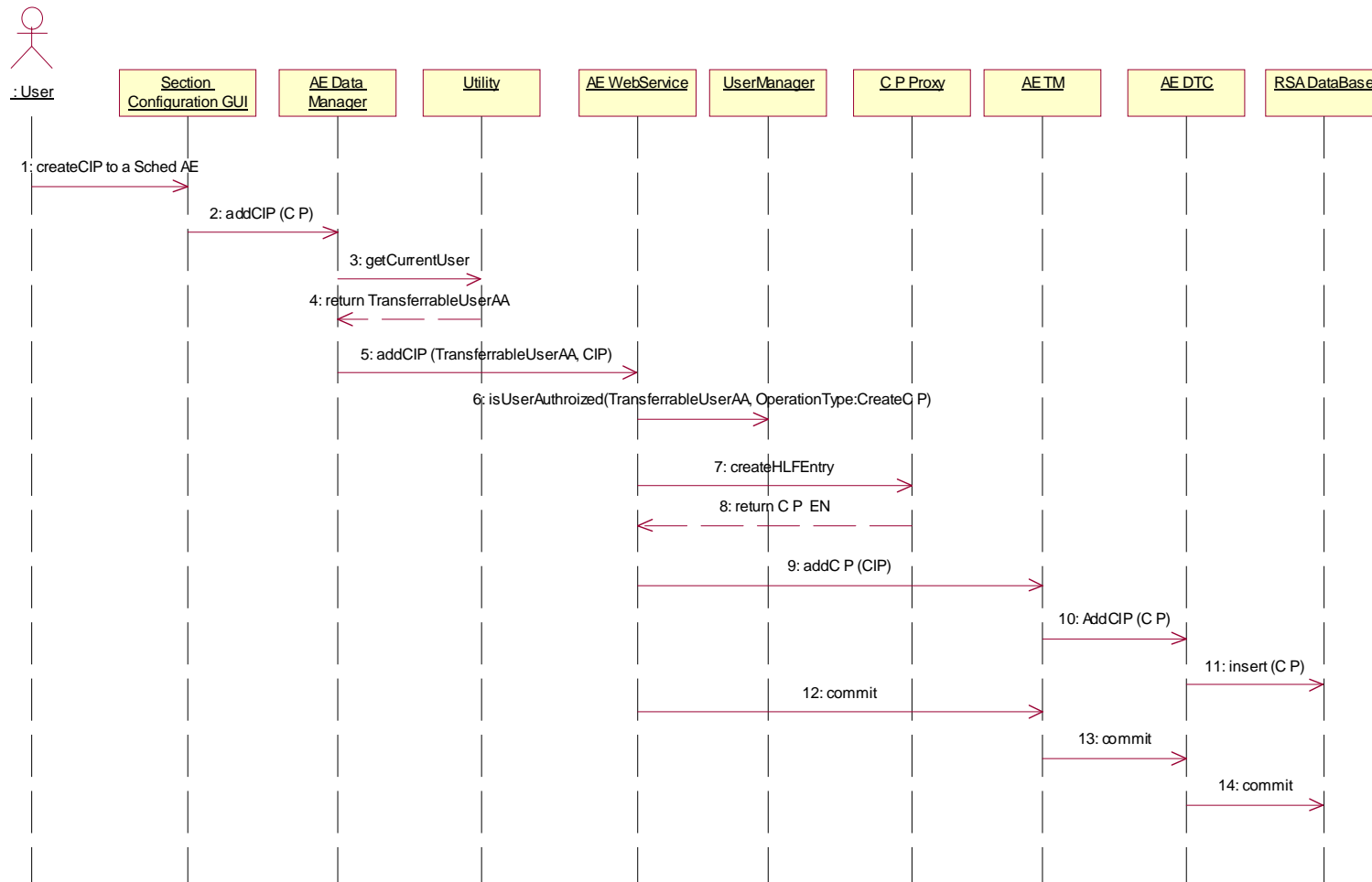
Figure 22 shows the flow of creating a CIP for a Schedulable AE. Users select a schedulable AE from the AE Tree and enter new CIP information on the configuration GUI. The GUI passes the CIP information to the middle tier's AE Web Service using the AE Data Manager. The AE Data Manager is responsible for creating the TransferrableUserAA object (user value object) that identifies the user making the connection to the server and sending the request along with CIP and user information to the AE WebService. When the AE WebService receives the request, it validates the user and the operation type using the UserManager utility and then sends CIP information to the VA using the CIP Proxy to create a Hospital Location File (HLF) entry and to request the IEN for the new CIP. When the AE WebService receives the IEN from the CIP Proxy, it passes the CIP information, including IEN, to the AE Transaction Manager. Finally, the AE Transaction Manager inserts new CIP information into the database using the AE DTC. Since the sequence of operations for steps 3, 4, and 6 are common throughout the RSA system, they will not be included in the rest of the configuration sequence diagrams.

#### **2.5.1.6 Create Resource**

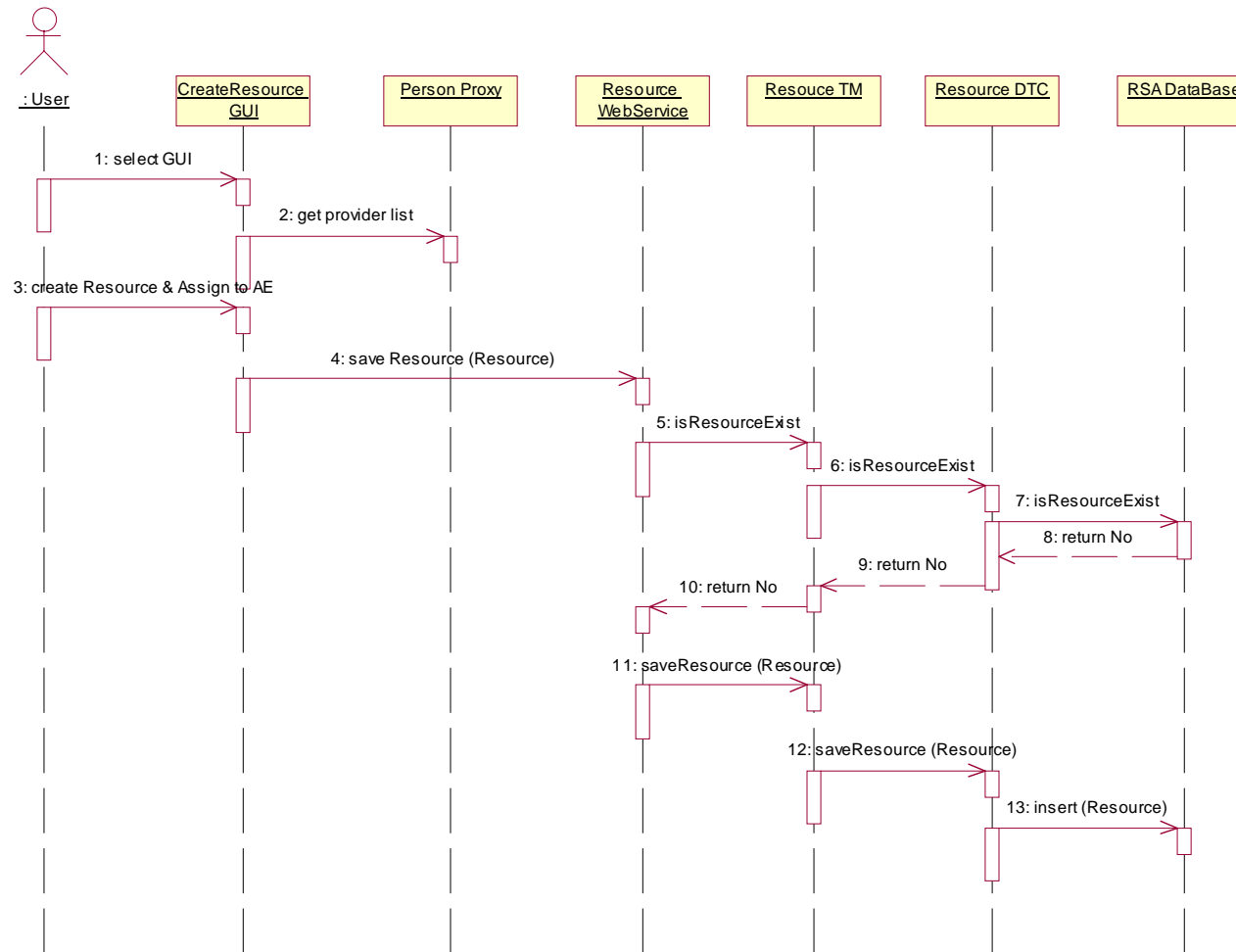
To load the create provider screen, the GUI will use the Resource Proxy to get the resource (provider) list. Users will select an AE from the AE Tree that the resource will be assigned to and then enter the resource information on the Resource screen. The GUI passes the resource information to the middle-tier resource component using the ResourceDataManager. The ResourceDataManager is responsible for: creating the TransferrableUserAA object that identifies the client node, making the connection to the server, and sending the request along with the resource and user information to the ResourceWebService. When the ResourceWebService receives the request, it validates the user and operation type through the User Management utility. It will also validate whether the resource already exists in RSA or not. If not, the resource information is passed on to the ResourceTM. Finally, ResourceTM inserts the resource into the database (DB) using the ResourceDTC and returns the full Resource back to the GUI. (See Figure 23)



**Figure 21. Building Configurations Within RSA**



**Figure 22. Create Check In Point**



**Figure 23. Create Resources**

### **2.5.1.7 Allocate Resource**

Figure 24 shows the sequence of events for allocating a resource to a schedulable AE. Users select a schedulable AE from the AE Tree and enter the resource allocation information (i.e., start/end date time and weekly/monthly/specific date allocation) on the Resource Allocation screen. Since this operation runs as part of a CCP, the system leads the user to the CCP screen. Users are required to enter the name and the effective and activate dates. The CCP GUI passes the CCP and allocation information to the middle-tier CCP component using the CCPDataManager. The CCPDataManager is responsible for creating the TransferrableUserAA object that identifies the client node, making the connection to the server, and sending the request along with the CCP, resource allocation, and user information to the CCP WebService. When the CCP WebService receives the request, it validates the user and operation type using the User Management utility and passes the CCP and resource allocation information to the CCP TM. The CCP TM will pass the CCP information to the CCP DTC and the resource allocation information to the ResourceTM to save it to the database. This resource allocation operation will be saved in the “PendingCreation” state and will not be effective to the system until the user activates the CCP. To activate the resource allocation, the user uses the CCP GUI to load the CCP that holds the resource allocation and activates the CCP.

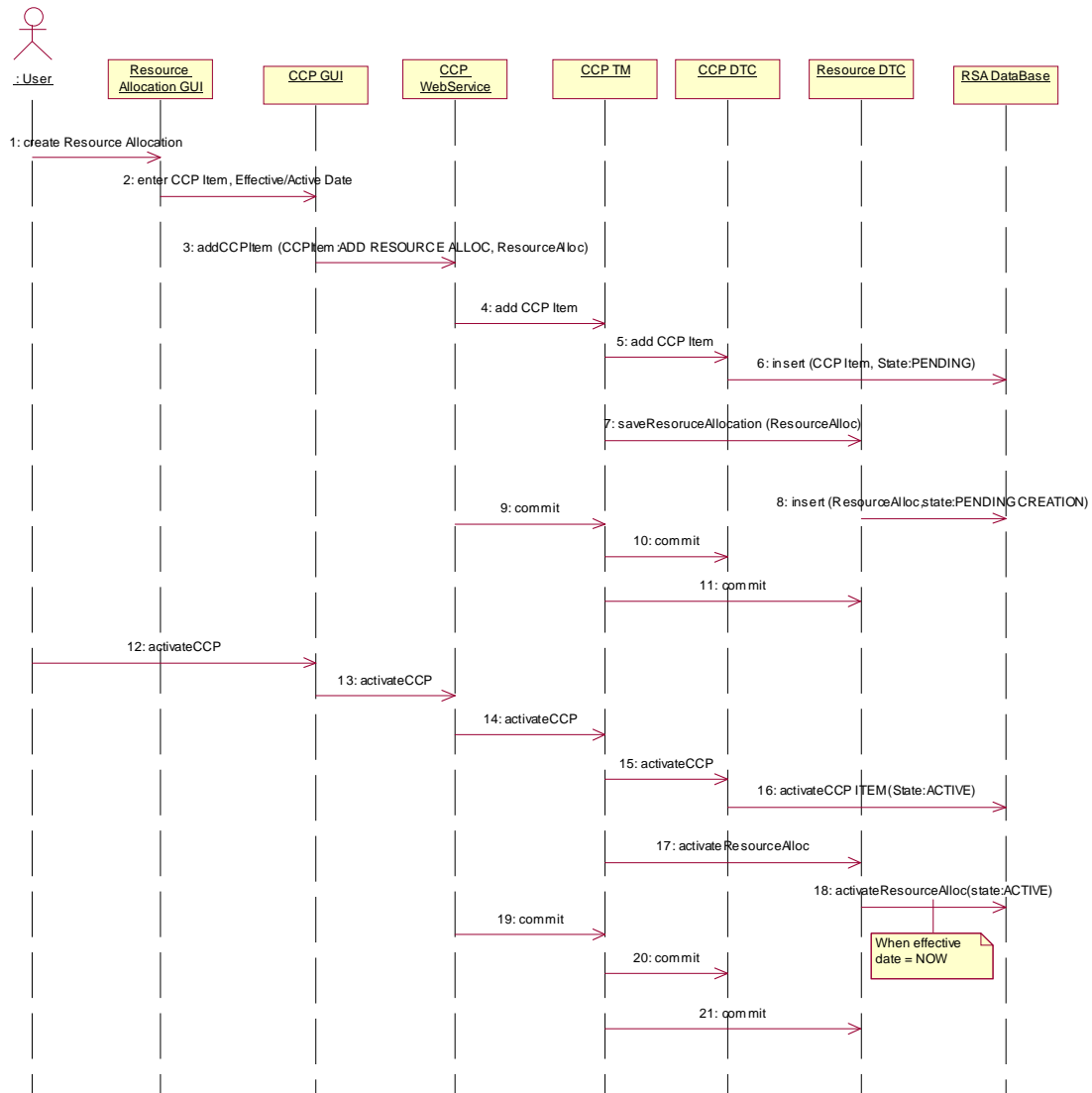
### **2.5.1.8 Remove Resource Allocation**

Figure 25 shows the sequence of events for removing a resource allocation. Since this operation may impact some existing carve-outs and scheduled appointments, it is processed as a part of a CCP. In the CCP, users are able to view the conflicts that will be caused by the change and resolve them before finalizing the changes. To process removing a resource allocation, users select the desired resource allocation on the Resource Allocation screen and then select to remove it. The system then leads users to the Configuration Change screen. Users are required to enter the name, as well as effective and activate dates, for this change. In the following screen, users can select to view the conflicts. If there are conflicts, the screen will display the carve-outs that are associated with the resource allocation and the scheduled appointments that are associated to the carve-outs. Users have to resolve these conflicts in the Configuration Change screen first and then finalize (activate) the operation, thus removing the resource allocation.

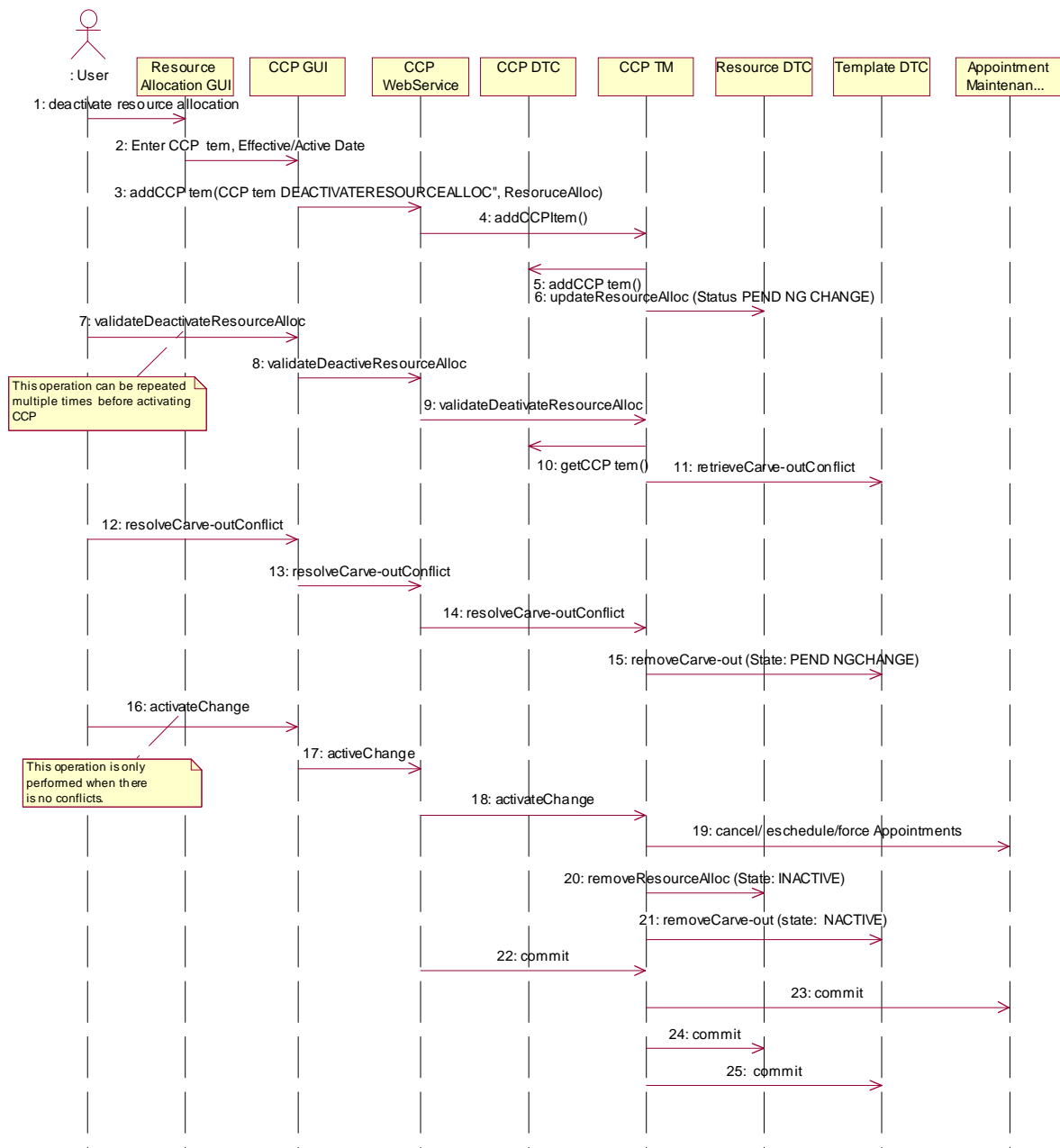
### **2.5.1.9 Create Resource Set**

Figure 26 shows the sequence of events for creating a resource set and assigning it to an AE. Users select a schedulable AE from the AE Tree and enter the resource set information on the Resource Set screen. The GUI passes the resource set information to the middle-tier resource component using the ResourceDataManager. The ResourceDataManager is responsible for creating the TransferrableUserAA object that identifies the client node, making the connection to the server, and sending the request along with the resource set and user information to the ResourceWebService. When the ResourceWebService receives the request, it validates the user and operation type using the UserManager utility and passes the resource set information to the ResourceTM. Finally, the ResourceTM inserts the resource to DB using the ResourceDTC.

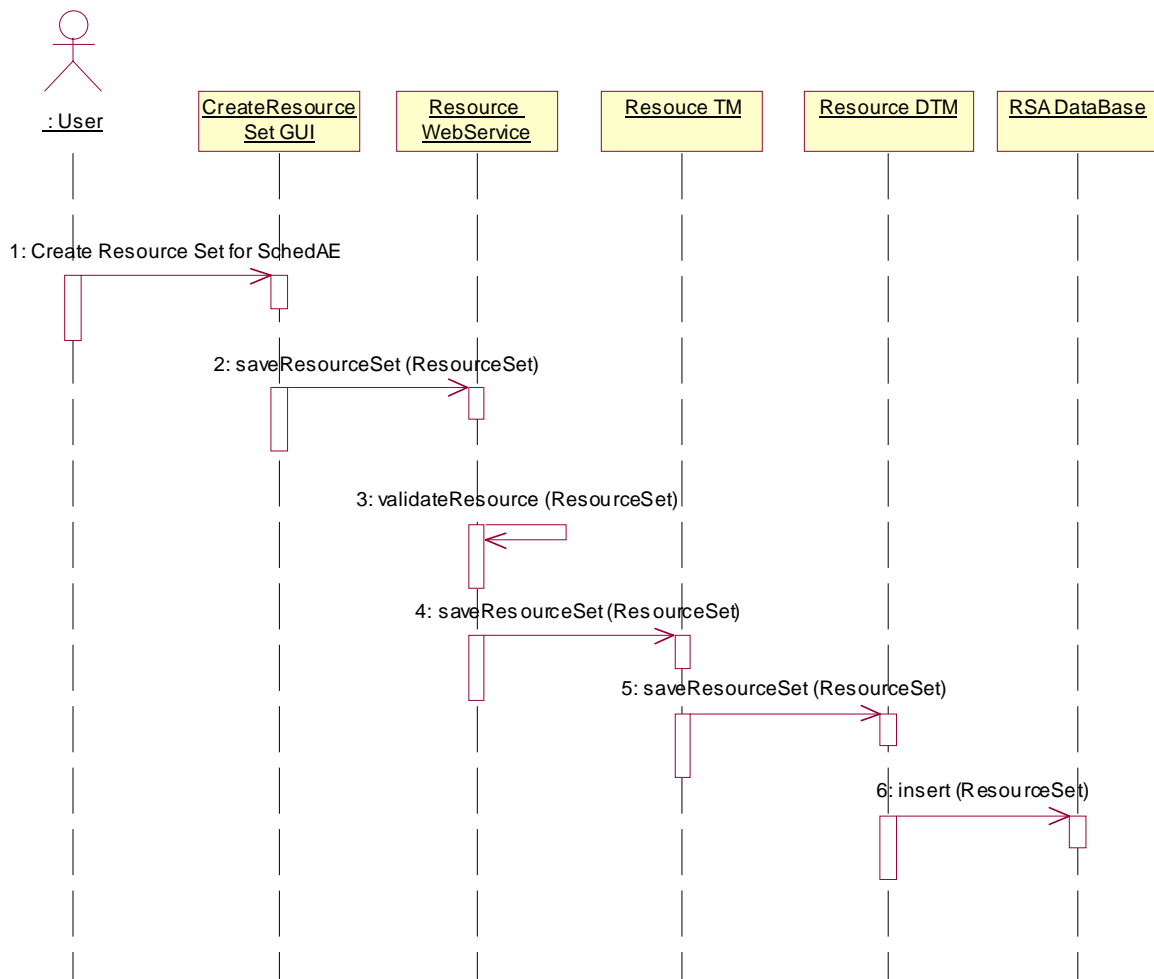




**Figure 24. Allocate Resource**



**Figure 25. Remove Resource Allocation**

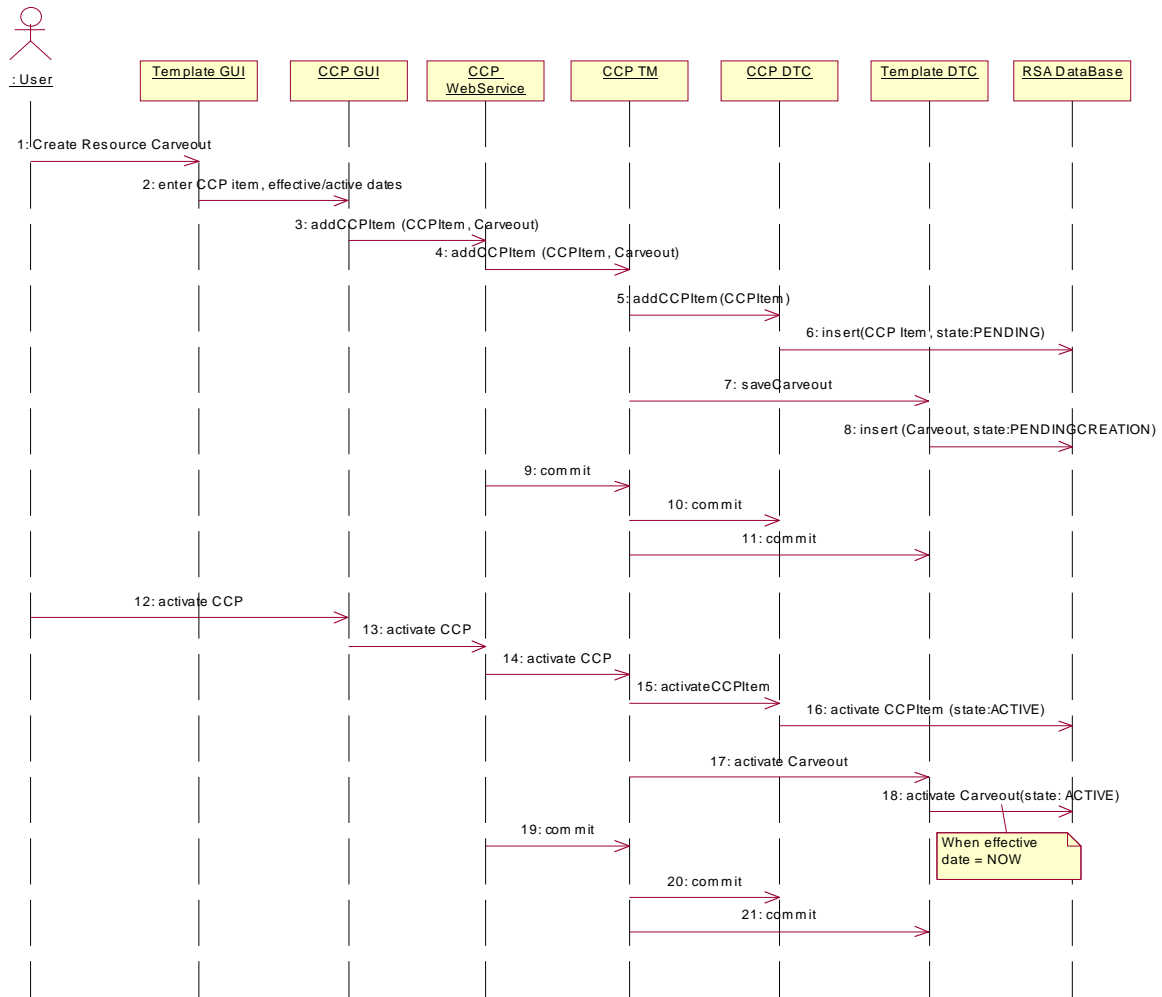


**Figure 26. Creating a Resource Set**

#### 2.5.1.10 Create Resource Carve-outs

Figure 27 shows the sequence of events for creating a carve-out to a resource allocation. Users select a resource allocation from the AE Tree and enter the carve-out information, such as start and end date time, appointment purposes, and patient eligibilities on the Template GUI. Since this operation runs as part of a CCP, the system leads the user to the CCP screen. Users are required to enter the name, as well as the effective and activate dates. The CCP GUI passes the CCP and carve-out information to the middle tier, CCP Component, using the CCPDataManager. The CCPDataManager is responsible for creating the TransferrableUserAA object that identifies the client node, making the connection to the server, and sending the request along with the CCP, carve-out, and user information to the CCP Web Service. When the CCP Web Service receives

the request, it validates the user and operation type using the User Manager utility and passes the CCP and carve-out information to the CCP TM. The CCP TM will pass the CCP information to the CCP to save them to the database. This carve-out operation will be saved in the “PendingCreation” state and will not be effective until user activates the CCP. To activate the carve-out, the user uses the CCP GUI to load the CCP that holds the carve-out and activate the CCP.



**Figure 27. Create Resource Carve-out**

### **2.5.1.11 Create Local Appointment Purpose**

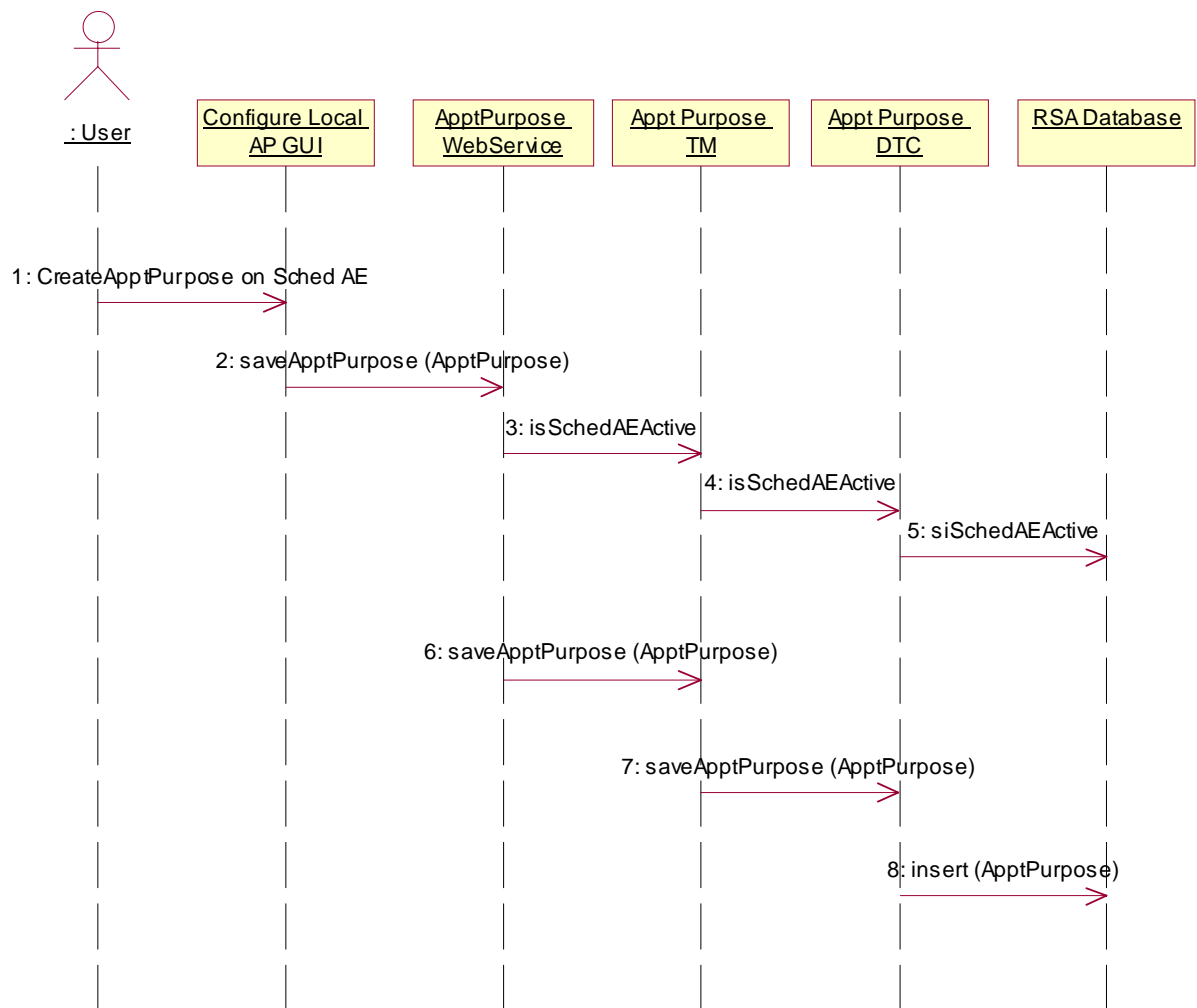
Figure 28 shows the sequence of events for creating a local appointment purpose to a schedulable AE. Users select a schedulable AE from the AE Tree and enter the appointment purpose information on the Configure Local AP screen. The appointment purpose information includes the resource set name. The relationship between the appointment purpose and the resource set is made on this screen. The GUI passes the appointment purpose information to the middle-tier, Appointment Purpose Component, using the ApptPurposeDataManager. The ApptPurposeDataManager is responsible for creating the TransferrableUserAA object that identifies the client node, making the connection to the server, and sending the request along with the appointment purpose and user information to the ApptPurposeWebService. When the ApptPurposeWebService receives the request, it validates the user and operation type using the UserManager utility and then passes the appointment purpose information to the ApptPurposeTM. Finally, the ApptPurposeTM inserts the resource into the database using the ApptPurposeDTC.

### **2.5.1.12 Create Appointment Purpose Set**

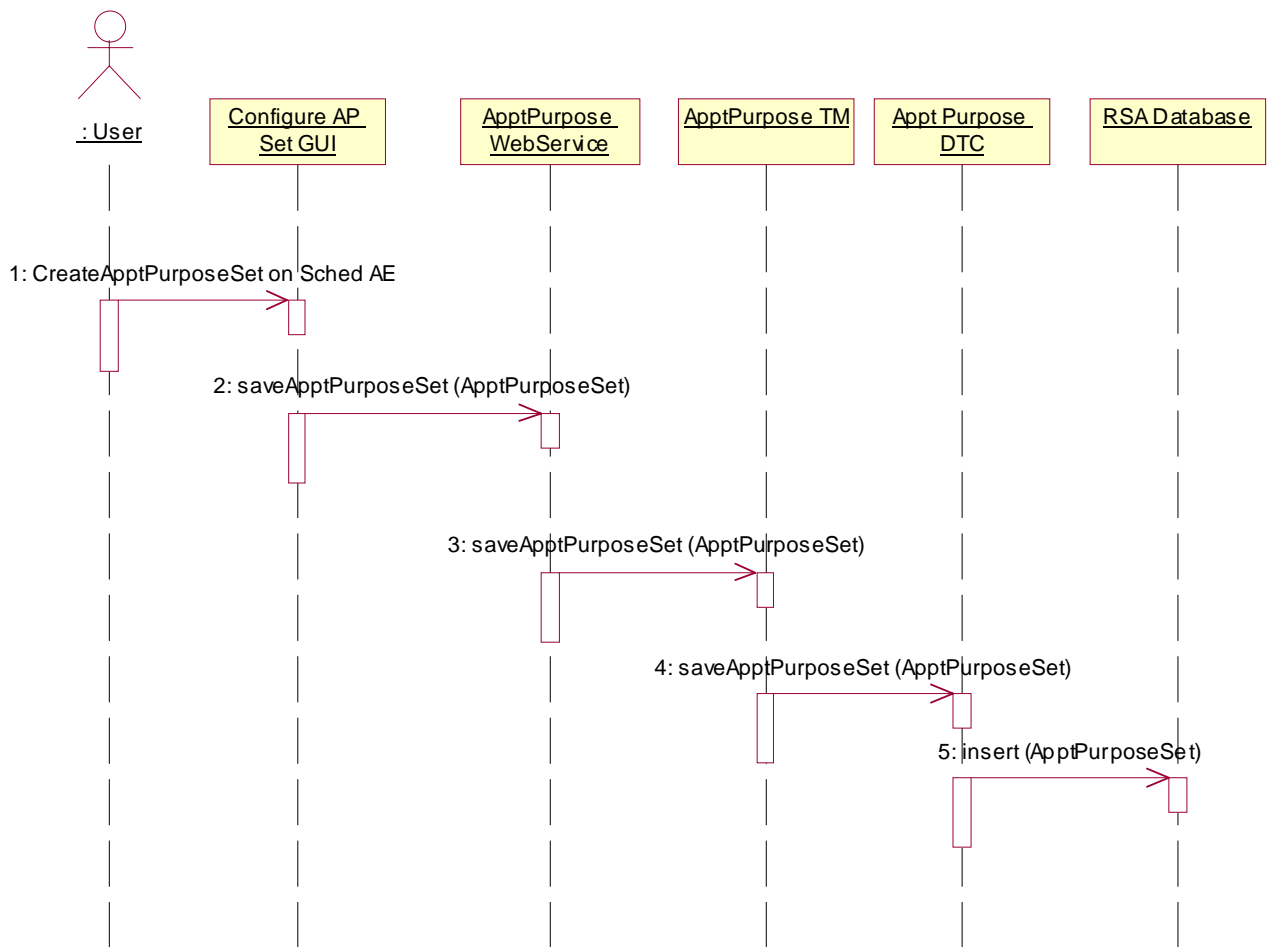
Figure 29 shows the sequence of events for creating an appointment purpose set to a schedulable AE. By setting the appointment purpose set, users are allowed to search and schedule a series of appointments. Users select a schedulable AE from the AE Tree and enter the appointment purpose set information on the Configure Local AP screen. The GUI passes the appointment purpose set information to the middle tier, Appointment Purpose Component, using the ApptPurposeDataManager. The ApptPurposeDataManager is responsible for creating the TransferrableUserAA object that identifies the client node, making the connection to the server, and sending the request along with the appointment purpose and user information to the ApptPurposeWebService. When the ApptPurposeWebService receives the request, it validates the user and operation type using the UserManager and passes the appointment purpose information to the ApptPurposeTM. Finally, the ApptPurposeTM inserts the resource into the database using the ApptPurposeDTC.

### **2.5.1.13 Deactivate Schedulable Admin Entity**

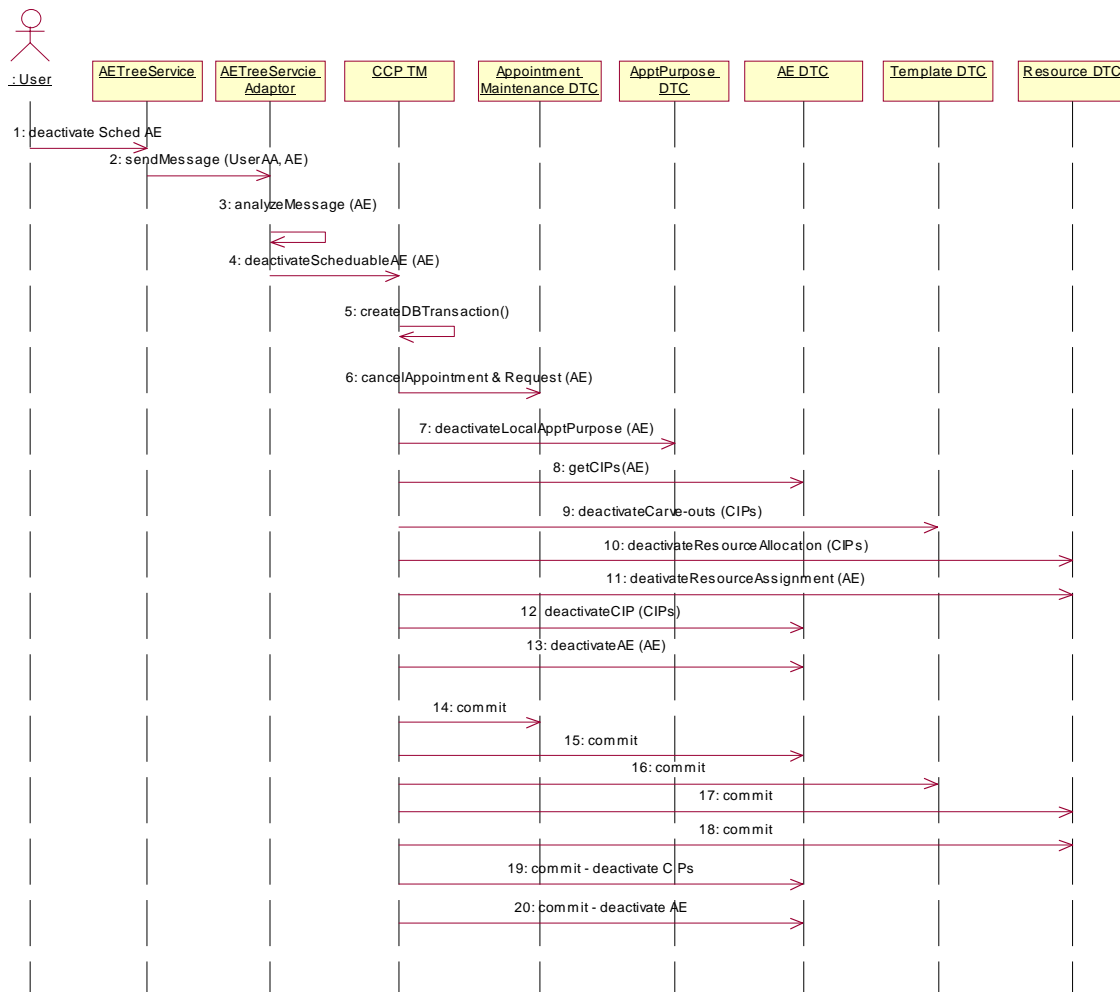
Figure 30 shows the sequence of events for deactivating a schedulable AE. Users remove a schedulable AE from the AE tree using ATS. ATS sends a message that contains the AE information that needs to be removed from the AE tree. The AE Tree Service Adaptor in RSA receives the message and passes the information to the CCP Management component. The CCP Management component then checks all of the conflicts that are caused by removing the schedulable AE. If there are any, the CCP will resolve all of them by updating the status to cancel or inactivate before inactivating the schedulable AE. The conflicts are the appointments made after the inactivation date of the AE, the appointment request in the pending lists, and all of the configuration data that was attached to that schedulable AE, such as CIPs, allocated resources, resource sets, carve-outs, local appointment purposes, and appointment purpose sets. Resources assigned to the AE will be moved to the parent AE after the inactivation date. This operation is processed without any user intervention based on the message received from the ATS.



**Figure 28. Creating a Local Appointment Purpose**



**Figure 29. Create Appointment Purpose Set**



**Figure 30. Deactivate Schedulable Administrative Entity**



## **2.5.2 Daily Operations**

The high-level RSA daily operations components and the execution flows of the daily operations activities between the client and the middle-tier components are illustrated in Figure 31. The data managers will act as liaisons between the client application and middle-tier web services. The data managers will also be responsible for creating the user authorization object, making the connection to the server, and sending the request along with the user information to the appropriate web service. (Note: For brevity, the user authorization calls will not be shown in the following Daily Operations sequence diagrams.)

Most daily operations client application requests are handled by either the patient or scheduling web services. Each web service will delegate the client application request to the appropriate business logic. The business logic is responsible for gathering all appropriate data and taking the action required to handle the request. Database activity is handled through the transaction managers, which entrust actual database calls to the DTCs.

The Patient Web Service presents methods in two main categories: patient maintenance and group. The patient maintenance business logic will handle activities such as adding patients to RSA and handling patient maintenance events. The group business logic will handle adding patients and removing patients from named groups and maintaining their group statuses.

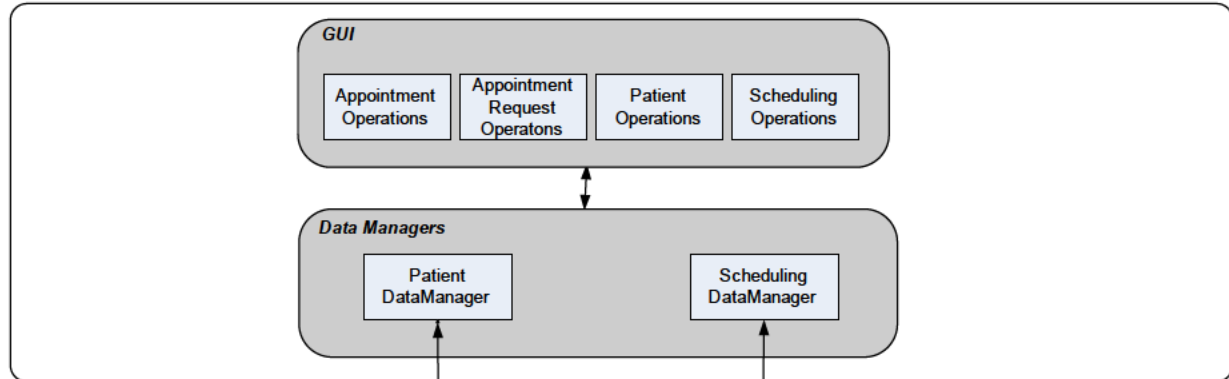
The Scheduling Web Service presents methods in three categories: appointments, requests, and search. Both appointments and requests categories handle activities such as saving, look-up, and maintenance of their respective responsibilities. Search provides the ability to search for appointment opportunities within the RSA database.

Together these web services, their respective business logic, transaction managers, and DTCs work together to provide the following core Daily Operations RSA activities: patients, groups, searching, and the scheduling, maintenance, and look-up of appointments and requests.

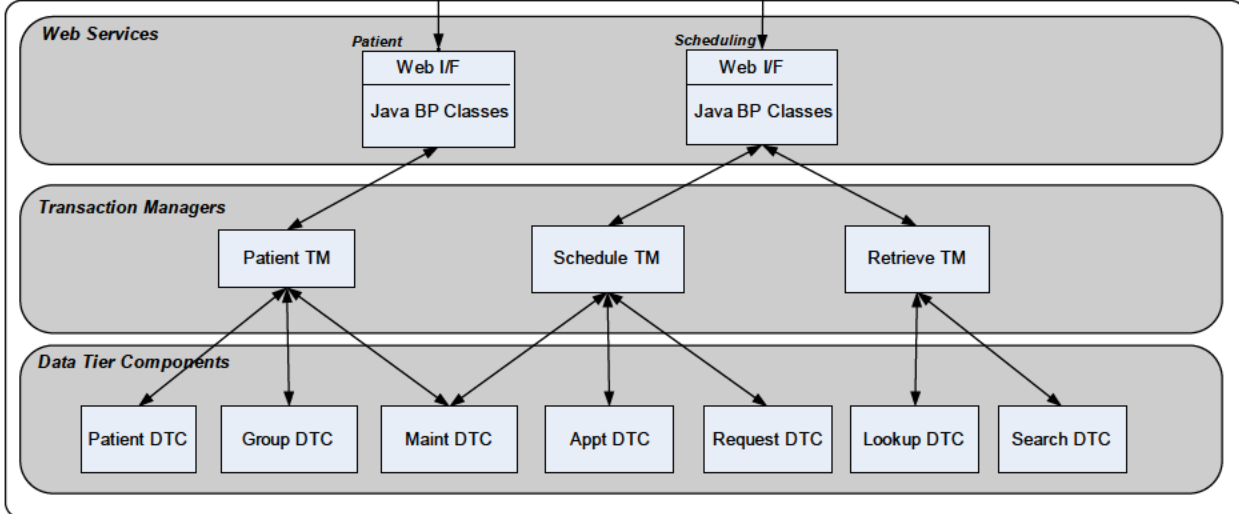
### **2.5.2.1 Make Appointment Request**

To make a new appointment request (Figure 32), the user will initiate a make new request event. The appointment request operations GUI will load the request information and then present it to the user. After selecting the appropriate request information, the user will submit the request. In the event the user selects a C&P request, the C&P Proxy will be called to retrieve the patient's list of open 2507s. The user will then be required to select one of the 2507s. The GUI will then pass the request criteria to the middle-tier web service via the Scheduling Data Manager. The Scheduling Data Manager calls the server-side Scheduling Web Service requesting that the middle tier save an appointment request. Prior to saving the request, the Utility Transaction Manager is used to retrieve data to validate the request criteria. The Scheduling Web Service calls the Schedule Transaction Manager to save the appointment request, which in turn calls the Request DTC. If saving the request information fails, the request business logic will return the appropriate error message.

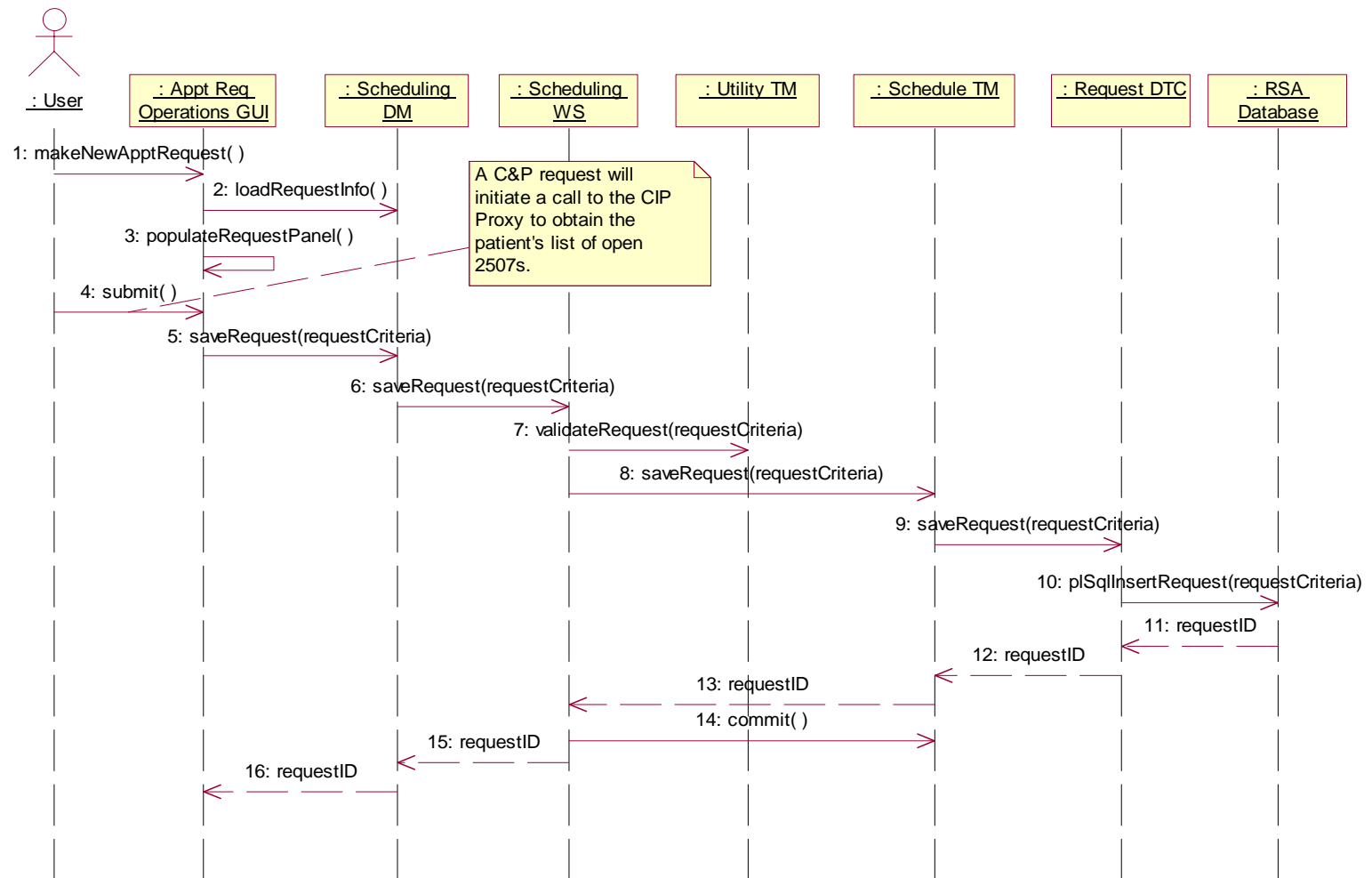
### Daily Operations : Client



### Daily Operations : Middle Tier



**Figure 31. Daily Operations Activities**



**Figure 32. Make Appointment Request**

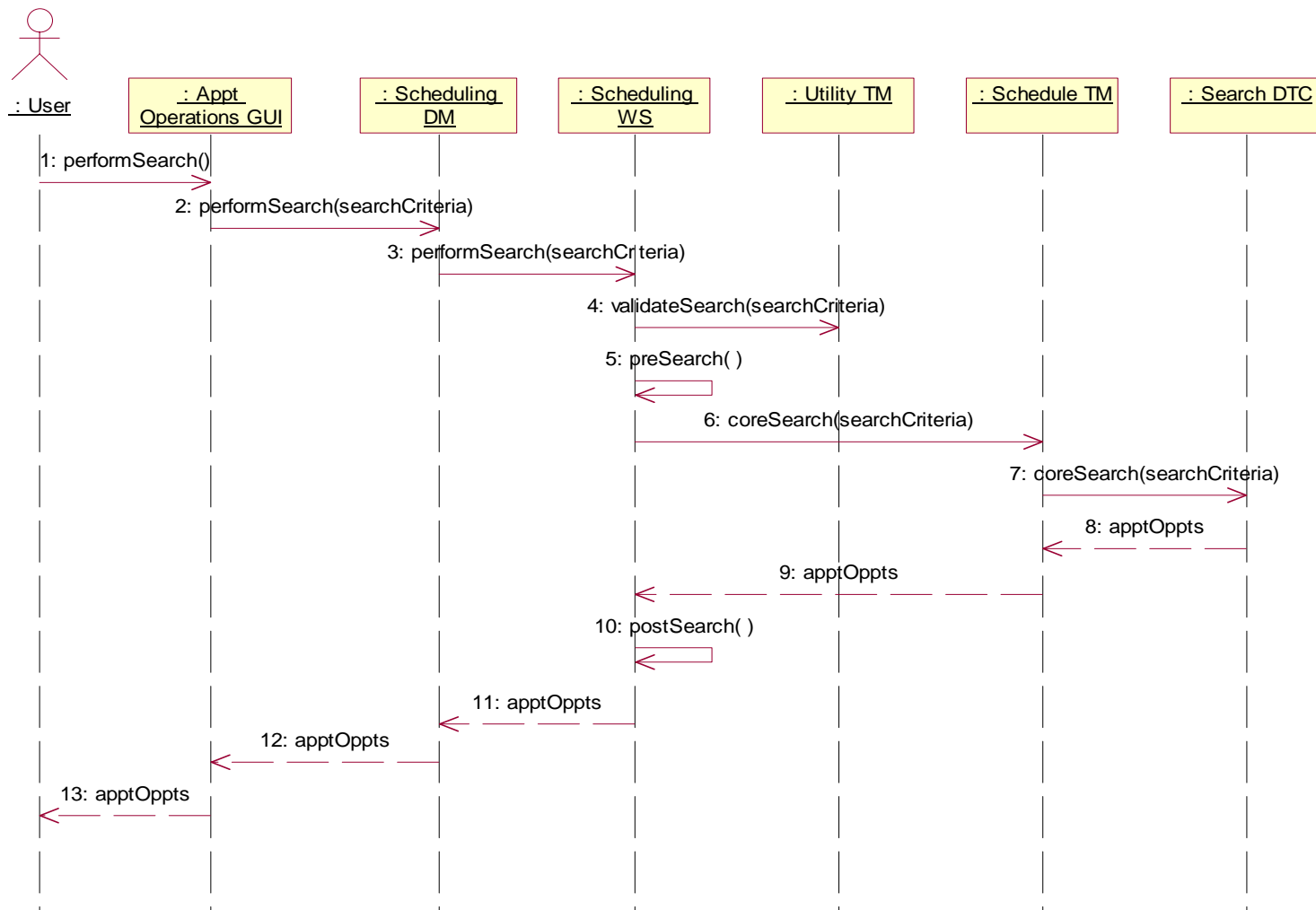
The Schedule Transaction Manager provides a single interface to all DTCs required to save an appointment request, and it also provides coordination with the database which allows the database transaction to be committed or rolled back as a whole. Upon success, all necessary appointment request confirmation data is repackaged in the business logic and sent back to the client application.

#### **2.5.2.2 Search for Appointment Opportunities**

As shown in Figure 33, the sequence begins with the user initiating a search request. The Appointment Operations GUI will send the search criteria to the Scheduling Web Service, via the Scheduling Data Manager. The Scheduling Web Service will validate the search criteria using preloaded data in the Utility Transaction Manager. The pre-search is executed which performs the following activities: breaking up larger requests into fundamental pieces (recurring and appointment purpose set searches), conducting mapping exercises (administrative entities to section, national appointment purposes to local appointment purposes), and executing database queries to obtain search-related data (resource sets, resources involved in the search). After successfully completing the pre-search, the core search is executed by the Search DTC to retrieve a resource's availability times. The post-search activities are then executed to filter the resource availabilities according to their appointment start times and durations. Finally, the results are repackaged and returned to the client application.

#### **2.5.2.3 Make a Single Appointment**

As shown in Figure 34, the sequence begins with the user initiating a search request. After the search completes, the appointment opportunities will be returned back to the GUI for display to the user. The user will then select the desired appointment opportunity and, with the proper permissions, will be allowed to book the appointment. The GUI will pass the search criteria to the Scheduling Web Service via the Scheduling Data Manager. The booking information will be validated with data from the RSA Database. After validation, the Schedule Transaction Manager will save the appointment by calling the Appointment DTC, which in turn calls the Procedure Language/Structured Query Language (PL/SQL) callable stored procedure for the actual appointment insert into the RSA database. The callable stored procedure will also be responsible for updating the appropriate search auxiliary table. Upon successful insertion into the database, the appointment ID will be propagated back to the Scheduling Web Service. A separate database call, within the same transaction, is made to the Request DTC to update the associated request status. After a successful request update, the Scheduling Web Service will issue the commit for the entire transaction (i.e., saving the appointment and the request update), using the transaction manager, and then will push the appointment information data to the Appointment Event Proxy. Upon success, all necessary appointment data is repackaged in the business logic and sent back to the client application.



**Figure 33. Search for Appointment Opportunities**

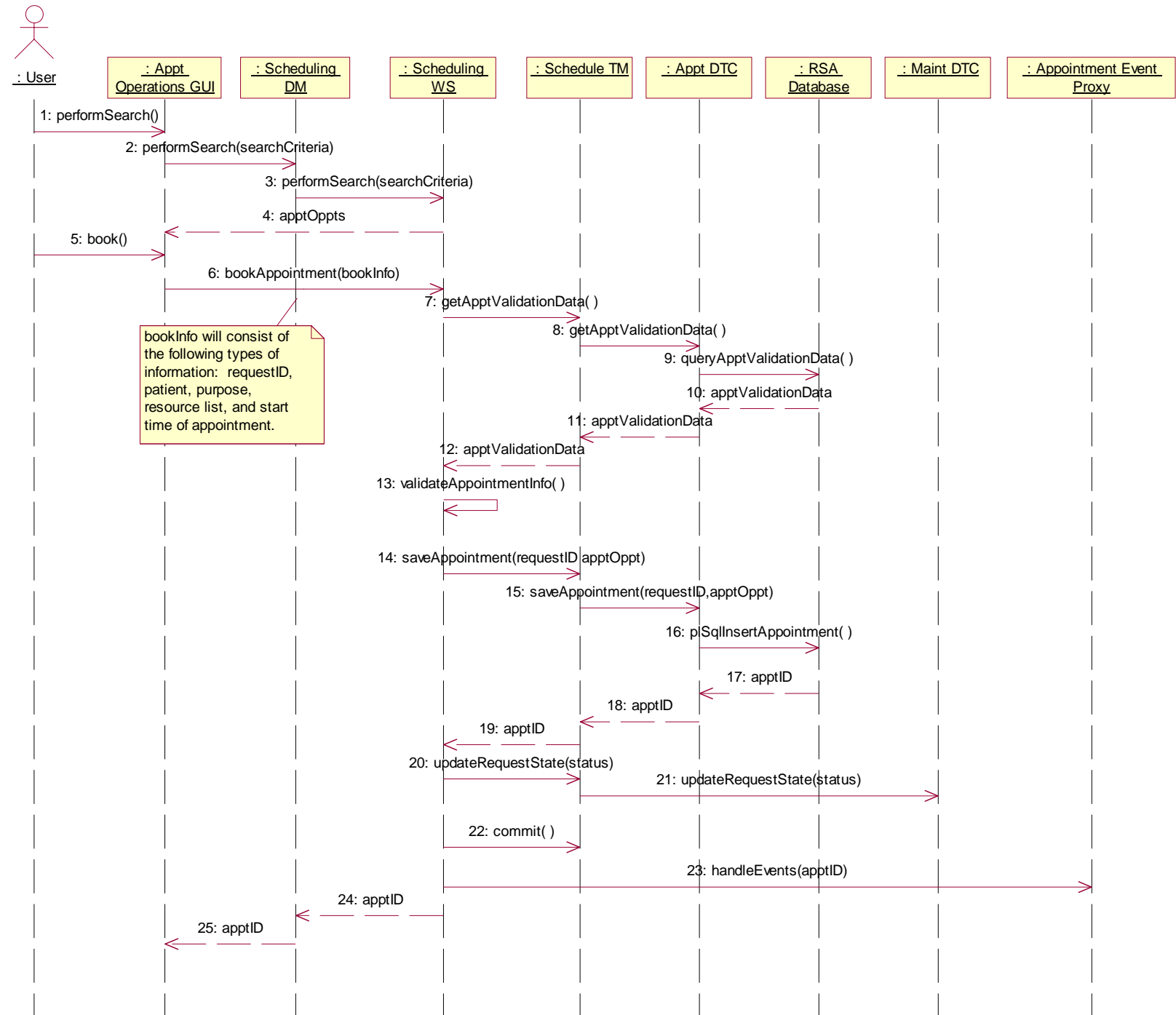


Figure 34. Make a Single Appointment

#### **2.5.2.4 Enter Walk-In Appointment**

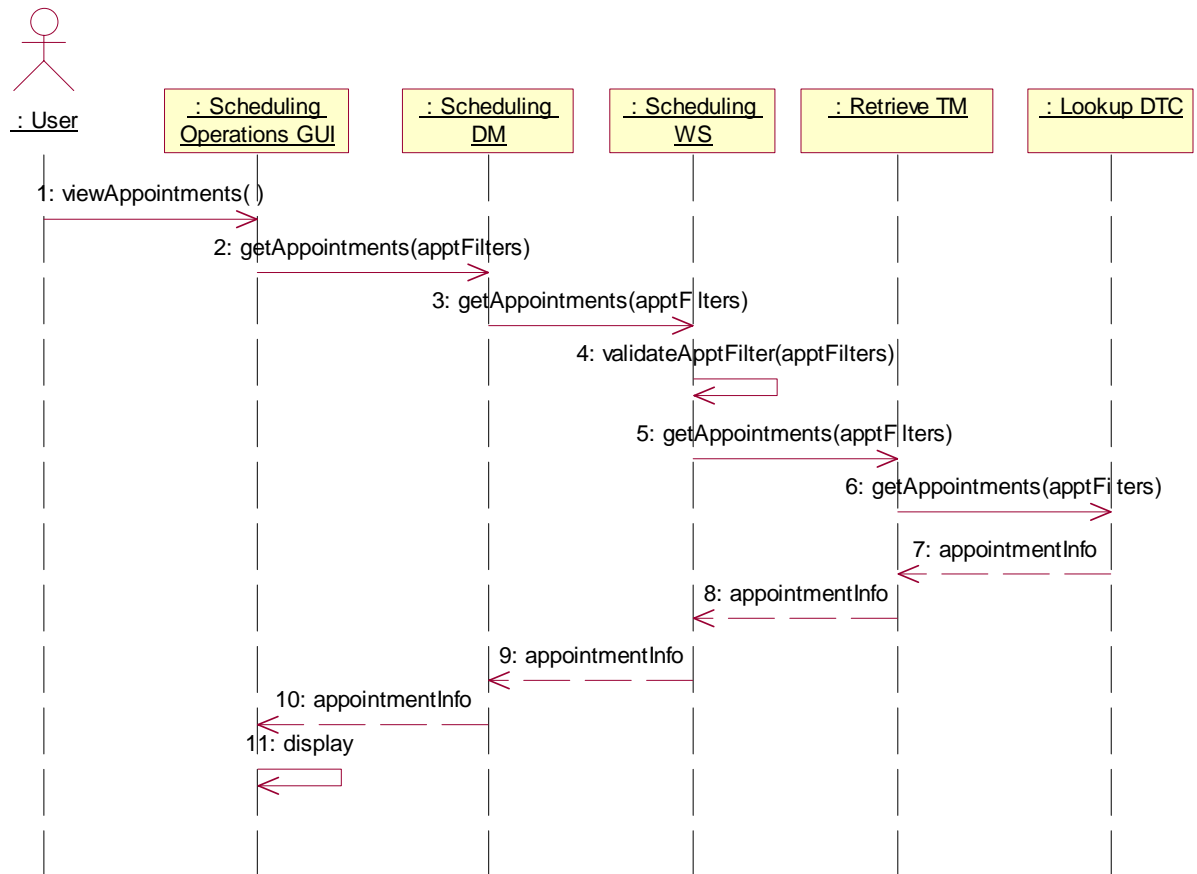
In order to process a walk-in Appointment, the user will proceed through the standard process of making a single appointment (Figure 34) selecting a target date of today. After executing the search, the user will be presented with the standard list of possible appointment opportunities that match the criteria. In the event that there are no opportunities, the user will then cancel the appointment selection and will be presented with a set of choices to allow them to book the unscheduled appointment.

#### **2.5.2.5 View Appointments**

To view appointments (Figure 35), the user will initiate a view appointment event to the scheduling operations GUI. The GUI will then pass the appointment filter criteria to the middle-tier web service via the Scheduling Data Manager. The Scheduling Data Manager calls the Scheduling Web Service requesting that the middle tier retrieve the appointments that satisfy the appointment filter information. Prior to retrieving the appointment information, the Utility Transaction Manager is used to retrieve data to help validate the appointment filter criteria. The Scheduling Web Service calls the Retrieval Transaction Manager to get the appointment information, which in turn calls the Lookup DTC. Upon success, the appointment information is propagated back up to appropriate scheduling operations GUI.

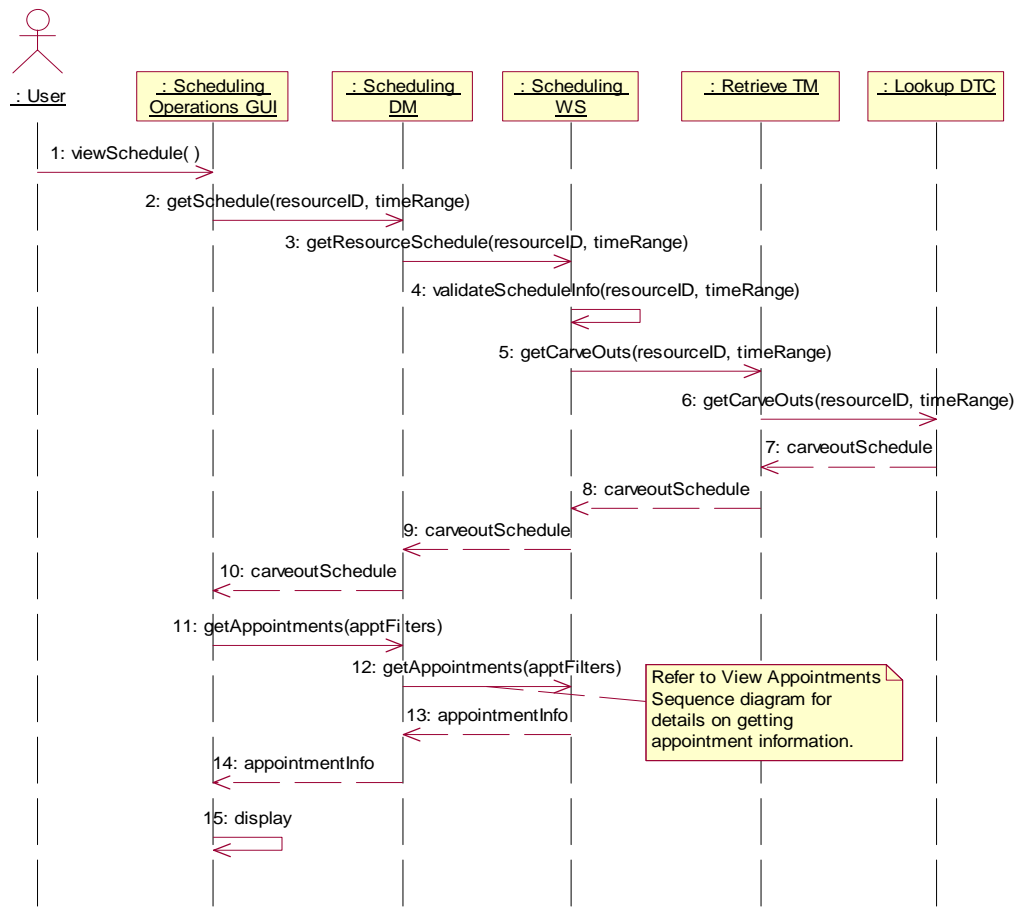
#### **2.5.2.6 View Resource Schedule**

To view a resource's schedule (Figure 36), the user will initiate a view schedule event to the scheduling operations GUI. The GUI will then pass the resource ID and the time-range criteria to the middle-tier web service via the Scheduling Data Manager. The Scheduling Data Manager calls the Scheduling Web Service requesting that the middle tier retrieve the resource's schedule that satisfies the filter criteria. Creating a resource's schedule requires two categories of data: a resource's carve-out schedule and a resource's appointments. Prior to retrieving both types of data, validation checks are performed on the filter criteria (e.g., verifying a valid time-range). A resource's carve-out information will be retrieved from the same auxiliary structures, via the Lookup DTC, that are used for the search. Using the pre-computed carve-out table will significantly reduce database trips and boost overall performance. Upon success, the carve-out information will be propagated back to the GUI. The GUI will then initiate another web service call to retrieve the resource's appointment information (refer to Figure 35 for details steps for retrieving a resource's appointment schedule). Once both retrievals are performed, the GUI will assimilate the information for display in either a table or calendar view.



**Figure 35. View Appointments**





**Figure 36. View Resource Schedule**

### 2.5.2.7 Appointment Maintenance

The patient appointment state machine diagram (Figure 37) shows the states that a single RSA patient appointment may be in at any given time. These states make it possible for the RSA to generate appointment notifications and gather reporting information, including appointment wait time metrics.

The patient appointment state machine is related to the appointment state machine (Figure 38) and the appointment request state machine (Figure 39) in that the state changes made in one state machine may trigger a state change in another state machine. For example, when a patient appointment is created and placed in the “Scheduled” state, the associated request is placed in the “Booked” state. This tells the RSA that the request has been satisfied and that it should no longer exist on any of the request lists. In the event an error is made transitioning the patient appointment status, RSA will support the capability of undoing state changes (refer to Section 2.5.2.11).

The patient appointment state machine may also trigger a state change in the appointment state machine. For example, if at least one patient appointment is placed in the “Kept” state, the associated group appointment is also placed in the “Kept” state. Conversely, if a group appointment is placed in the “Kept” state, all of the patient appointments that are not in a “Cancelled,” “No-Show,” or Left Without Being Seen (LWOBS) state are placed in a “Kept” state.

The appointment state machine diagram (Figure 38) shows the states that an RSA appointment may be in at any given time. This state machine is a simplified version of the patient appointment state machine. The states in this simplified state machine map back to the states in the patient appointment state machine. Having the simplified appointment state machine serves the following purposes:

1. It allows the RSA to have simplified states for a group appointment, but still have the more complex states of check-in, check-out, no-show, and LWOBS that are typically associated with a single patient in the group.
2. It allows the RSA to easily perform queries for reporting on group appointments.

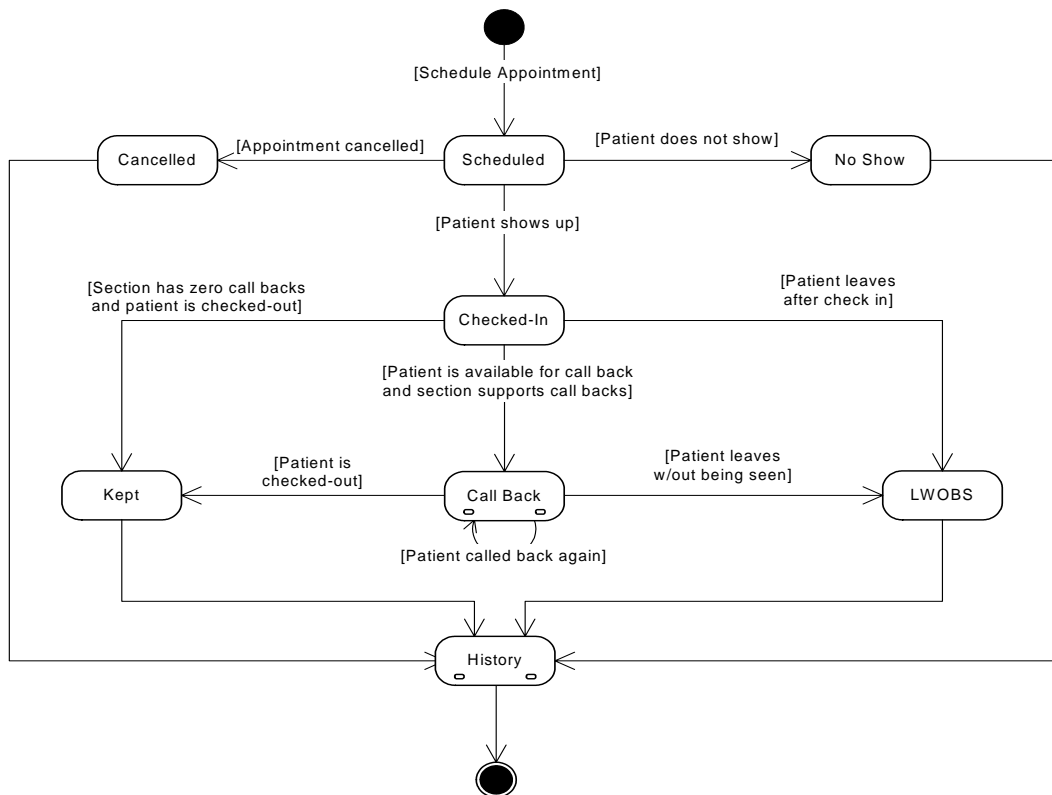
### 2.5.2.8 Appointment Request Maintenance

The appointment request state machine (Figure 39) shows the states that an RSA appointment request may be in at any given time.

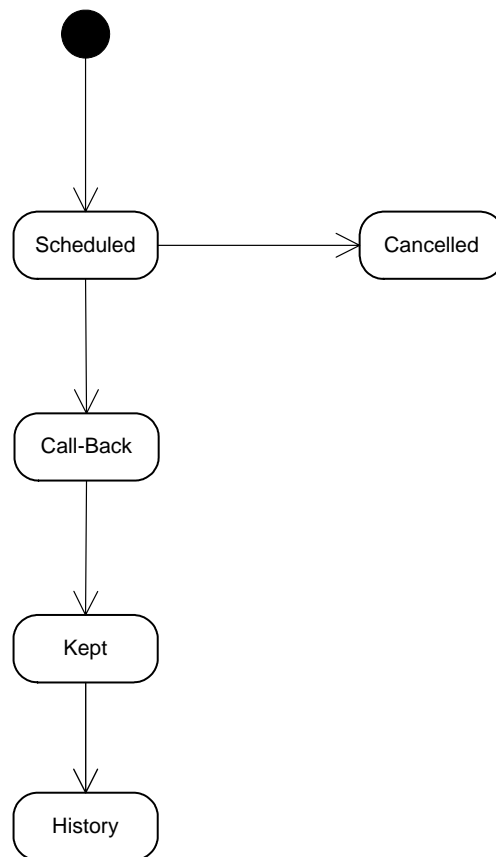
The states above make it possible for the RSA to generate the appointment request lists and to gather reporting information about the effectiveness of these lists. Table 2 shows the criteria used to determine which request list a request should reside on.

**Table 2. Request List Criteria**

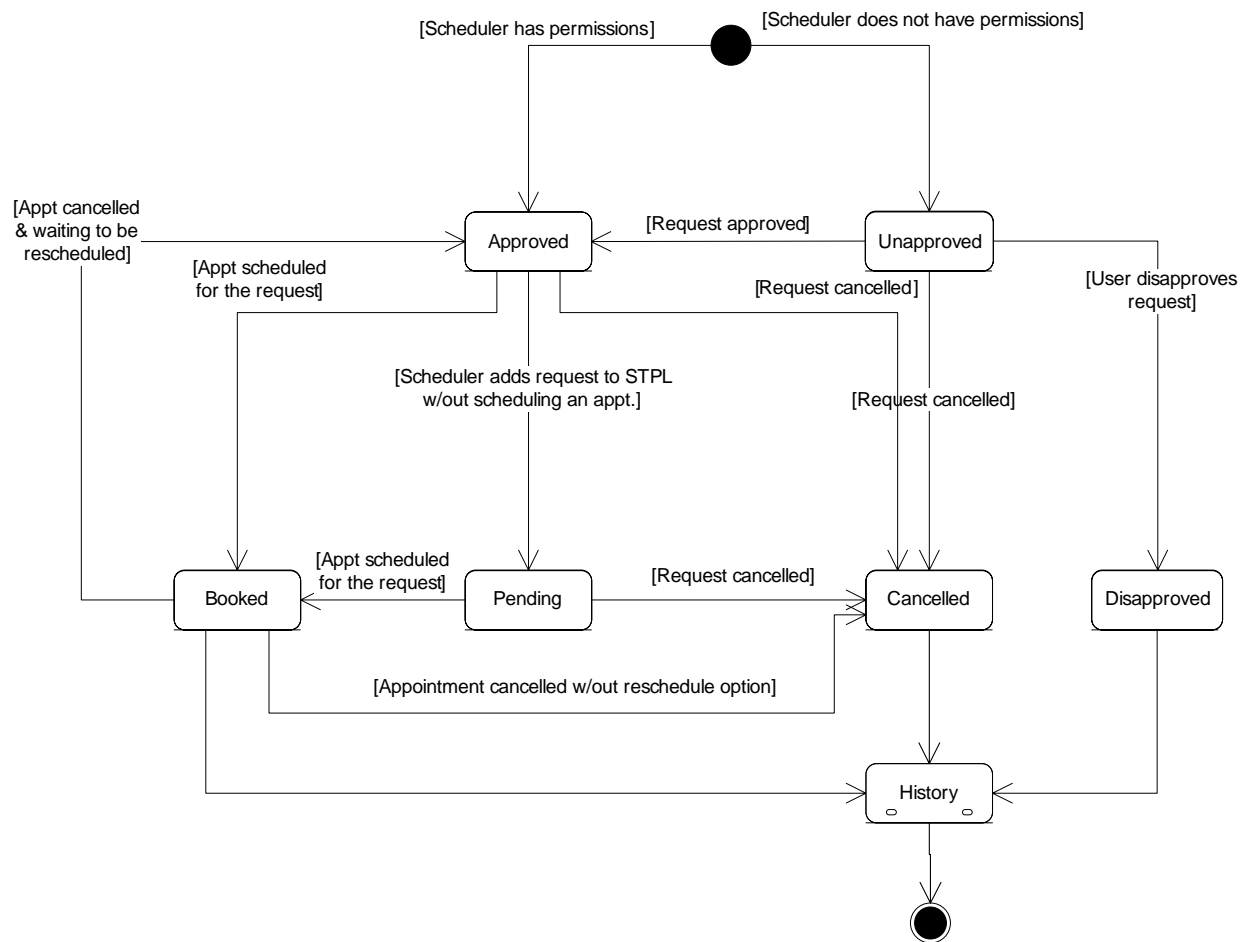
#	Request List	Criterion
1.	Unprocessed Request List	The request state is "Unapproved."
2.	Long-Term Pending List	The request state is "Approved," AND the target date is beyond the section's planning horizon.
3.	Short-Term Pending List	The request state is "Pending." Requests in the "Pending" state will be linked to requests that already have a scheduled appointment. These are requests that have scheduled appointments but the requestor desires an earlier appointment if one becomes available.
4.	Reschedule List	The request state is "Approved," AND the request's previous state was Booked." These are appointments that have been cancelled and need to be rescheduled.
5.	Composite Ready to be Scheduled List	All requests that are on the short-term and reschedule lists and all requests in the "Approved" state that have a target date within the section's planning horizon.



**Figure 37. Patient Appointment State Machine**



**Figure 38. Appointment State Machine**



**Figure 39. Appointment Request State Machine**

Special attention should be given to the Short Term Pending List (STPL). The purpose of the STPL is to allow the user to schedule an appointment for a patient while at the same time giving the user the opportunity to provide the patient with an earlier appointment should one become available. In order to provide this functionality, the RSA creates an appointment for the original request, and then creates a new request that is placed in the “Pending” state. Thus, when a user schedules an appointment and chooses to place the request on the STPL, a new request is created and linked back to the existing satisfied request. This gives the RSA the flexibility of treating the two requests as separate entities, yet still provides a mechanism for tracking the number of appointments created from the STPL, i.e., the effectiveness of the STPL.

#### **2.5.2.9 Cancel Appointment**

The user can cancel one or more appointments in the "Scheduled" state. The user will then be prompted to enter data relating to the reason for the cancellation, including who cancelled the appointment, a selection from a predefined list of reasons for the cancellation, and any other remarks that may be necessary. After the cancellation information has been recorded, the user will then have the option to reschedule the appointment.

#### **2.5.2.10 Reschedule Appointment**

There are several situations where rescheduling an appointment is permissible (e.g., the user or facility cancelled the appointment or a patient is dispositioned as a no-show or LWOB). Rescheduling an appointment in the above instances will simply require the request to change from the "Booked" state to the "Approved" state as was previously shown in the appointment request state machine diagram (Figure 39). The rescheduled list is generated with requests that have a current state of "Approved" and a previous state of "Booked."

#### **2.5.2.11 Update Appointment Status**

The user will have the capability to change appointment statuses for a selected appointment. Refer to Figure 37, Patient Appointment State Machine, for an illustration of the various valid appointment status transitions.

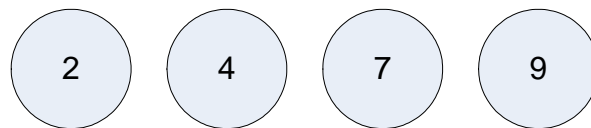
In the appointment request, patient appointment, and group appointment state machines, situations may arise where the user mistakenly updates the state of a request or an appointment. For example, the user may mistakenly disposition a patient's appointment from "Checked-In" to "LWOBS," when in fact the patient momentarily stepped out of the waiting room. To deal with these situations, the RSA will give certain users the ability to undo the most recent state transition, placing the appointment in the state it was in previously. In the example above, the user with the appropriate permissions would undo the "LWOBS" state, which would place the appointment back in the "Checked-In" state. When this situation occurs, the state transition that was made in error will still be preserved in the RSA database. However, the reason for the state transition will indicate the transition was made in error.

It is important to note that certain changes made to an appointment state may also trigger a change to the appointment request state. Consequently, when the user performs an undo on one of the appointment states, this may also impact the state of the appointment request. During the process of performing a state undo on the appointment, the RSA will also perform any necessary state transitions on the appointment request.

### 2.5.2.12 Manage Linked Appointments

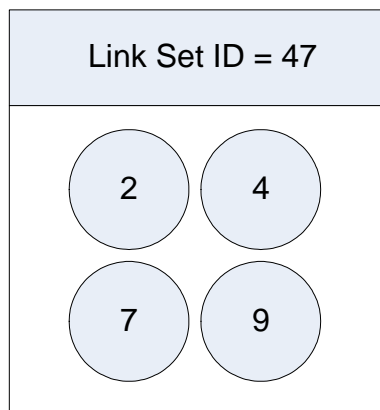
In the RSA, individual appointments can be linked together into a single linked appointment set. Appointments will be linked together using a Link Set ID. The Link Set ID is a unique identifier shared by all the appointments that are linked together. The example below shows how individual appointments are linked together in the RSA:

Suppose the user chooses to link appointments 2, 4, 7, and 9, which are currently unlinked (Figure 40).



**Figure 40. Linking Unlinked Appointments**

- 1) First, the RSA ensures that none of the appointments are already part of a linked appointment set.
- 2) After the RSA has found that none of the appointments are part of a linked appointment set, it requests the next available link set ID from the RSA database.
- 3) Finally, the RSA updates appointments 2, 4, 7, and 9 with the next link set ID retrieved from the RSA database (Figure 41).



**Figure 41. Updating With the Link Set ID**



### 2.5.2.13 Handle Patient Maintenance Events

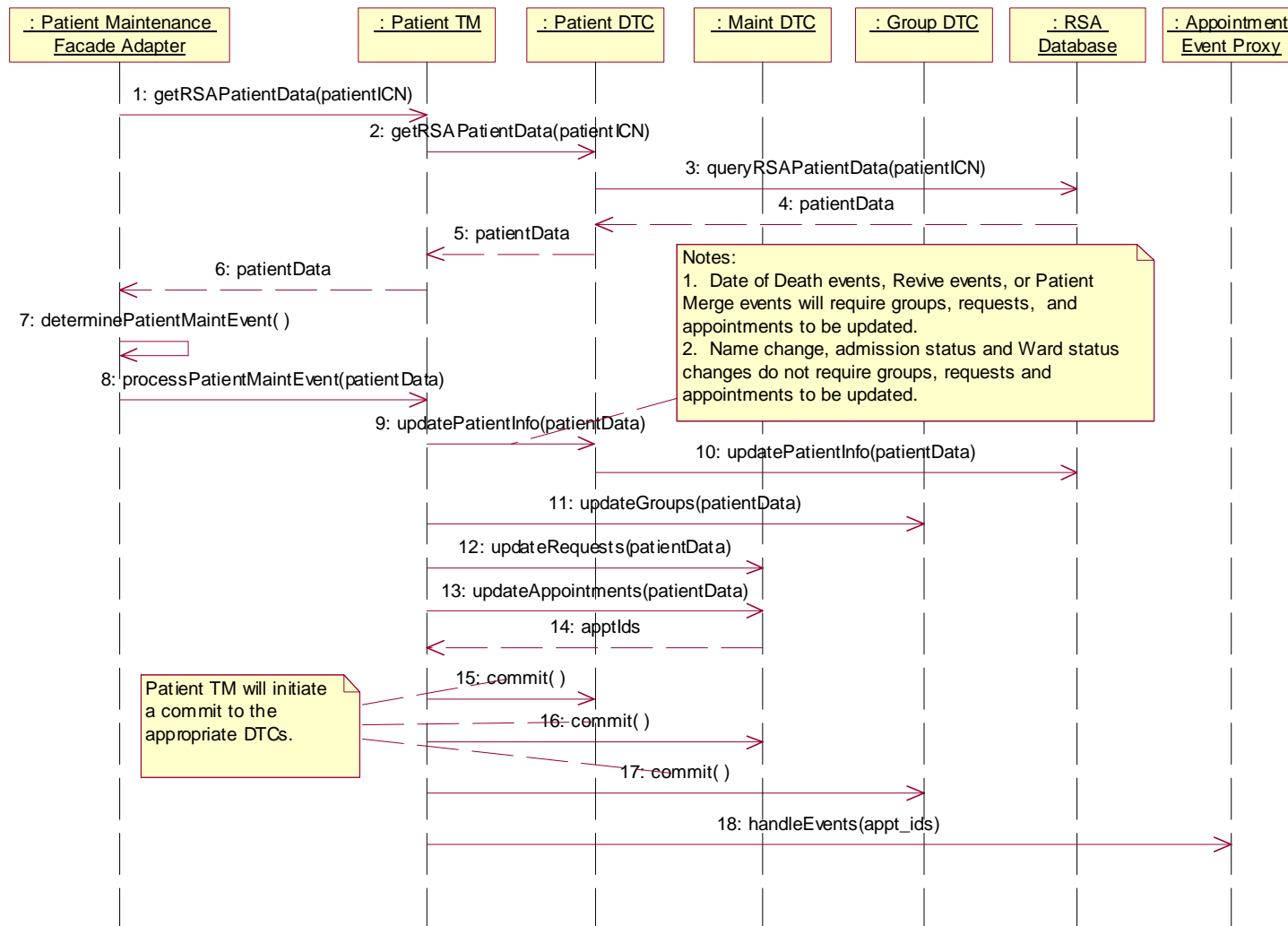
As shown in Figure 42, patient maintenance events originate from an external component, Patient Services, when the patient data changes (refer to Section 2.5.3.9, Patient Maintenance Event). The Patient Maintenance Facade Adaptor will accept these patient maintenance events and determine the correct course of action by comparing the incoming patient information with that patient's RSA information (Step 8). The Patient Transaction Manager then calls the Patient DTC to update the patient's information. Date of death events, patient revival events, or patient merge events require updates to requests, appointments, and named group entries (Steps 11-13). The list of appointment IDs will be returned to the Patient TM. The Patient TM will commit the entire transaction and then push the appointment information data to the Appointment Event Proxy.

## 2.5.3 External Interfaces

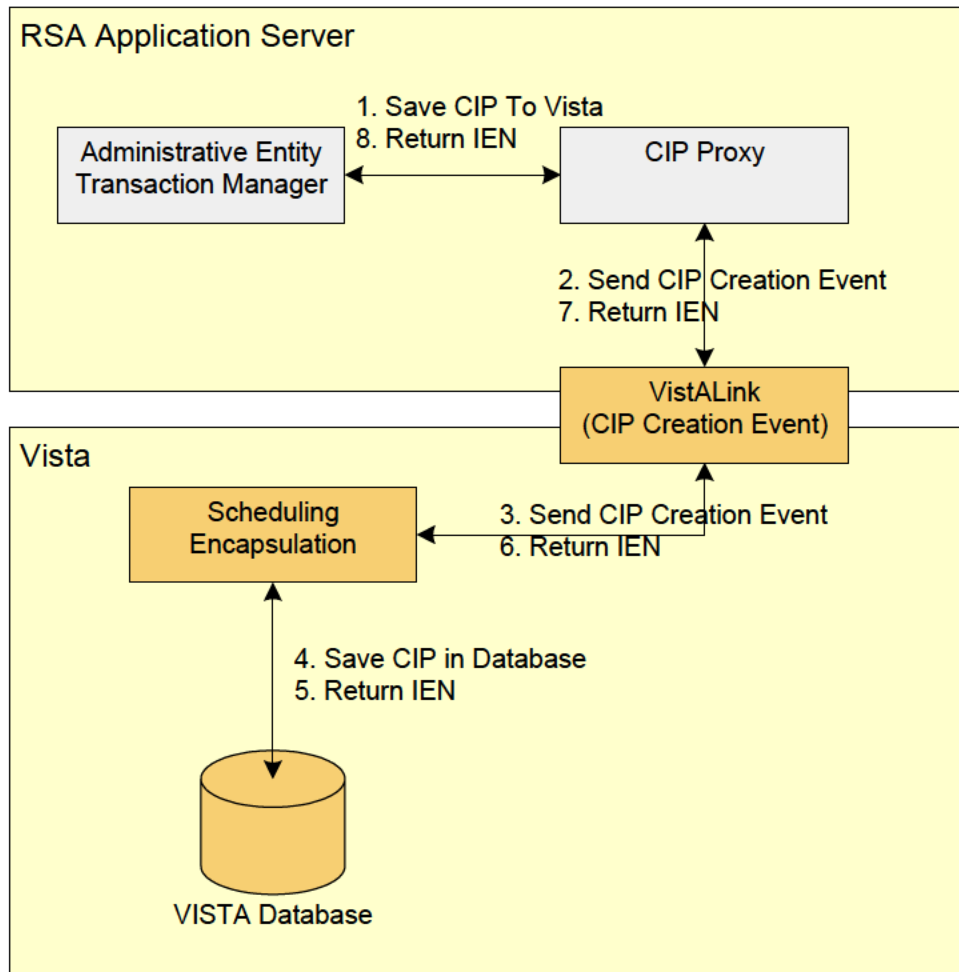
RSA External interfaces are modeled around the design patterns of facade adaptors for externally generated communications and proxy adaptors for internally generated communications. These are the same design pattern modeled in the VA CAIP compliance guidelines; however, since these guidelines are not yet formalized, RSA interfaces were designed utilizing the best currently defined practices. The following sections model both events that are initiated by external services and events initiated by internal RSA processes. As previously shown in Figure 11, External Interface Components, all communications with these outside applications flow through the adaptors. This allows process control and also helps ease the process of maintenance by separating the technologies used by the external applications from the processes that utilize the services of those applications.

### 2.5.3.1 Create New CIP External

The creation of a new CIP requires that some information be saved in the RSA database and that some information be saved in the appropriate local **VistA** database. The external interfaces part of this event is an integral part of the creation of a new CIP. As shown in Figure 43, this event begins when the Administrative Entity Transaction Manager makes a call to the CIP Proxy with the appropriate data that will be sent to the local **VistA** database. VistALink is utilized as the communications channel with all calls to the local **VistA** database. The data is sent via VistALink to Scheduling Encapsulation where a record will be added to the Hospital Location File containing the new CIP information. The IEN assigned to this newly created record is then returned to RSA and saved as part of the CIP. This event is part of an overall transaction. If this event succeeds but the overall transaction fails, then it will be necessary to send a CIP modification event to inactivate the new entry.



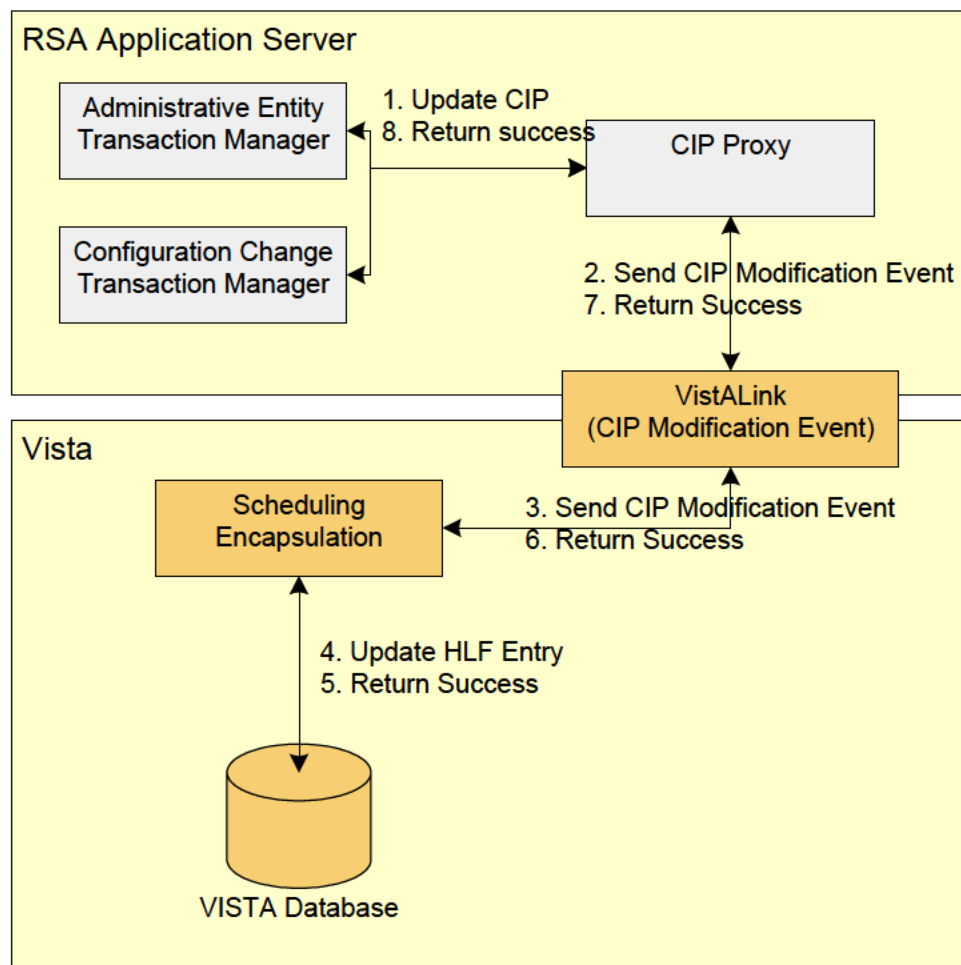
**Figure 42. Handle Patient Maintenance Events**



**Figure 43. CIP Creation Event**

### 2.5.3.2 Modify CIP External

The modification of an existing CIP requires that the local *VistA* be notified of the change. The only CIP modifications that will be sent to the *VistA* are the date and time that a currently active CIP will be inactivated or the date and time when a currently inactive CIP will be activated. This action may happen when a CIP Creation transaction fails or as part of an RSA change package. As shown in Figure 44, this event begins when either the Administrative Entity Transaction Manager or the Configuration Change Transaction Manager makes a call to the CIP Proxy with the appropriate data that will be sent to the local *VistA* database. The data is sent via VistALink to Scheduling Encapsulation where the appropriate HLF entry record will be updated with the new CIP information. *VistA* will return an indication of whether the update was successful or not.



**Figure 44. CIP Modification Event**

### 2.5.3.3 Appointment Events

There are five appointment events that are sent from RSA to Scheduling Encapsulation. They are sent when the status of an appointment is updated. The five statuses tracked with appointment events are new appointment, check-in, check-out, LWOBS and cancel. In all cases, these events are initiated by updates to the appointment status. The new appointment occurs when an appointment is booked into the RSA database (Figure 45). The check-in, check-out, and LWOBS are all events initiated from an RSA GUI appointment status update or may be initiated during an encounter create event. The Cancel event occurs when an appointment has been cancelled and can initiate from the RSA GUI or from a CCP event. In any case, the Appointment Proxy will be called and a series of appointment IDs passed into it. Appointment Event Proxy will then call the Appointment Transaction Manager to get the entire appointment event data set for each ID. The appointment will then be sent via VistALink to Scheduling Encapsulation.

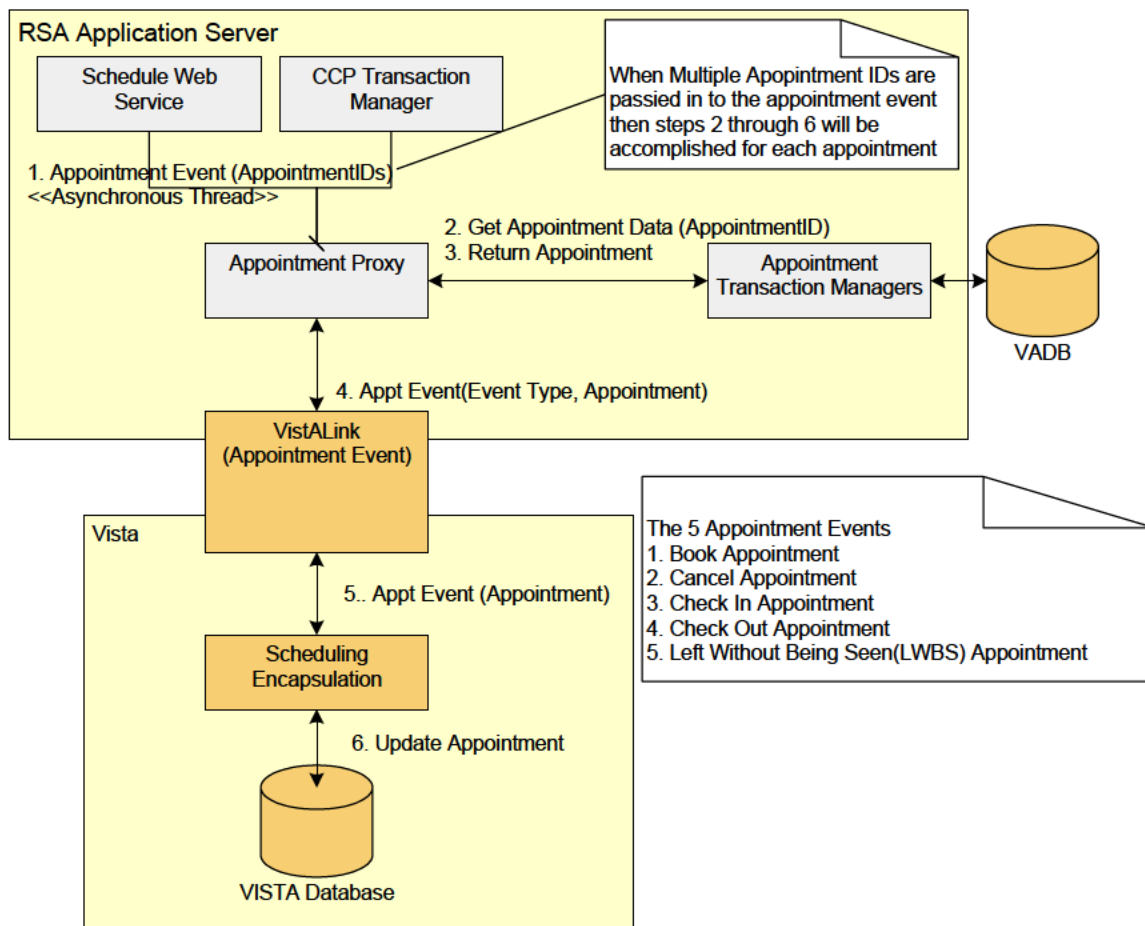
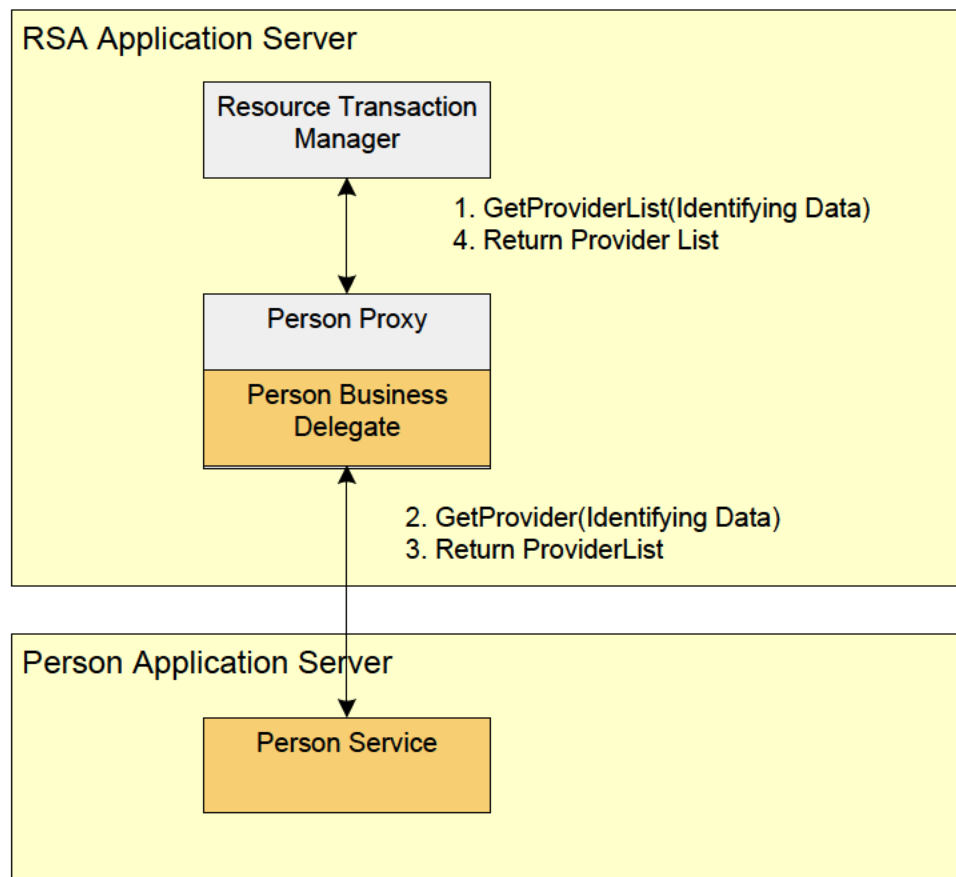


Figure 45. Appointment Events

#### 2.5.3.4 Retrieve Provider Information

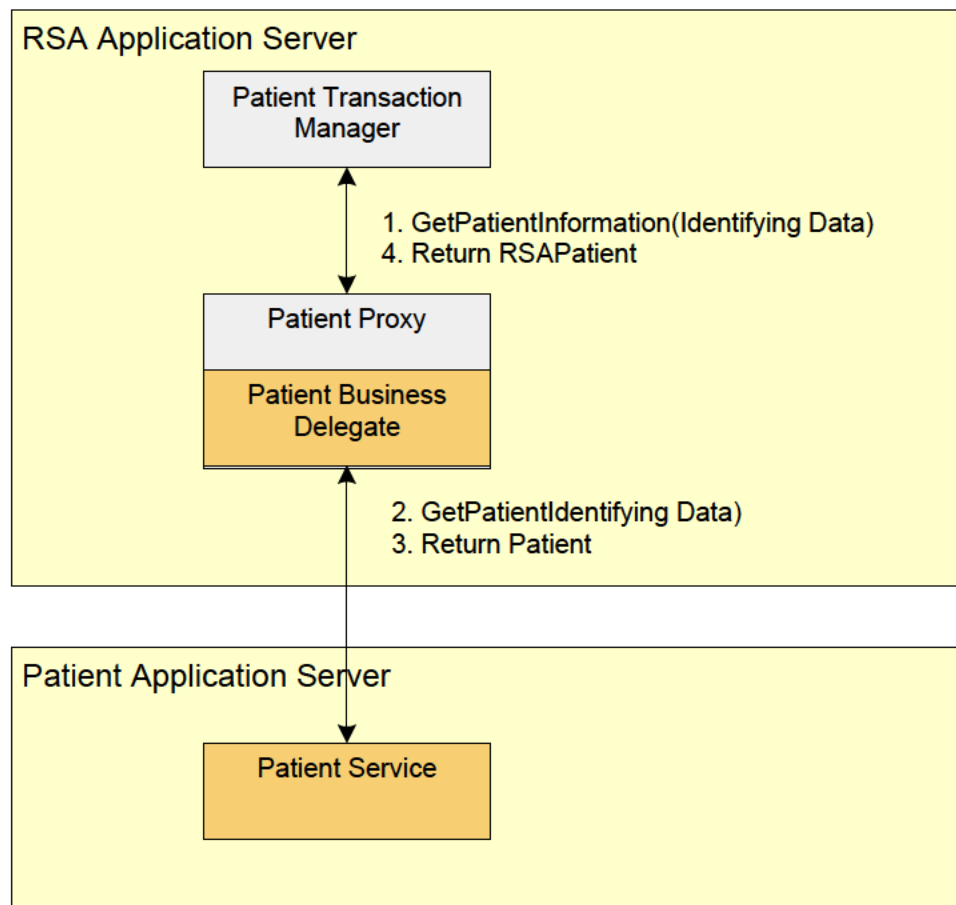
When a new resource or type provider is being added as an RSA resource, some information stored in the Person Service is necessary to complete the process of adding the provider to RSA. This sequence is used to retrieve that information from the person service and pass it back to the Resource Transaction Manager so that the user can add the correct provider information. The RSA Client application will initiate the event to retrieve provider information from the Person Service. As shown in Figure 46, the Resource Transaction Manger will call the provider proxy passing in provider identifying information (first name, SSN, partial name, etc.). The Person Proxy will then call the Person EJB passing this lookup type information. The Person EJB will pass back to the Person Proxy a list of providers with their accompanying data that matches the search criteria. This information will be returned to the Resource Transaction Manager.



**Figure 46. Create New Provider Event**

### 2.5.3.5 Retrieve Patient Information

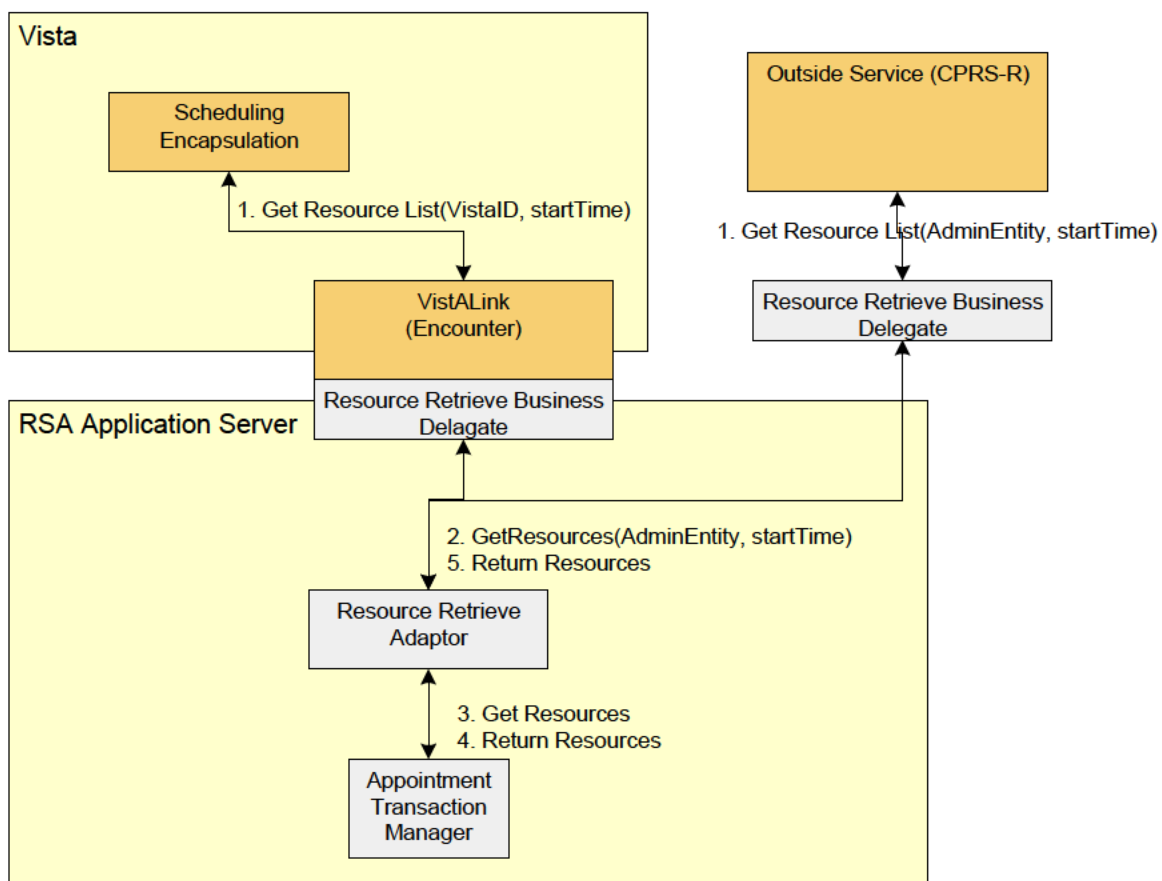
The RSA is not the authoritative source of patient demographics information. This means that the RSA is required, in certain processes, to retrieve a patient's demographic data from the authoritative source. The authoritative source is Patient Services. This event is initiated when RSA needs to retrieve certain pieces of a patient's demographic data for internal processing reasons. It may also be initiated when some other RSA process needs access to some patient information that does not exist in the minimal subset returned from the GUI Patient Lookup calls or stored in the RSA database. As shown in Figure 47, the Patient Proxy will be called by the Patient Transaction Manager. The Patient Proxy will then call the Patient EJB passing the patient ICN. The Patient EJB will pass back to the Patient Proxy the patient demographics data. This data will then be changed to the internal patient object in RSA and returned to the calling application.



**Figure 47. Retrieve Patient Information**

### 2.5.3.6 Retrieve Resource

The Resource Retrieve Event occurs when an application outside of RSA needs to access the list of resources allocated below a certain administrative entity level. As shown in Figure 48, the application will utilize the business delegate provided by RSA to make a call to the Resource Retrieve EJB. Applications can pass the location information by passing a fully qualified AE Tree node, a *VistA* instance identifier, a facility identifier, or a CIP identifier. The CIP Identifier is the IEN of the CIP's corresponding HLF. The Resource Retrieve Facade Adaptor will translate the data into internal RSA format and will then call the Resource TM to return a list of the resources meeting the specified criteria. This list will be returned to the calling application.

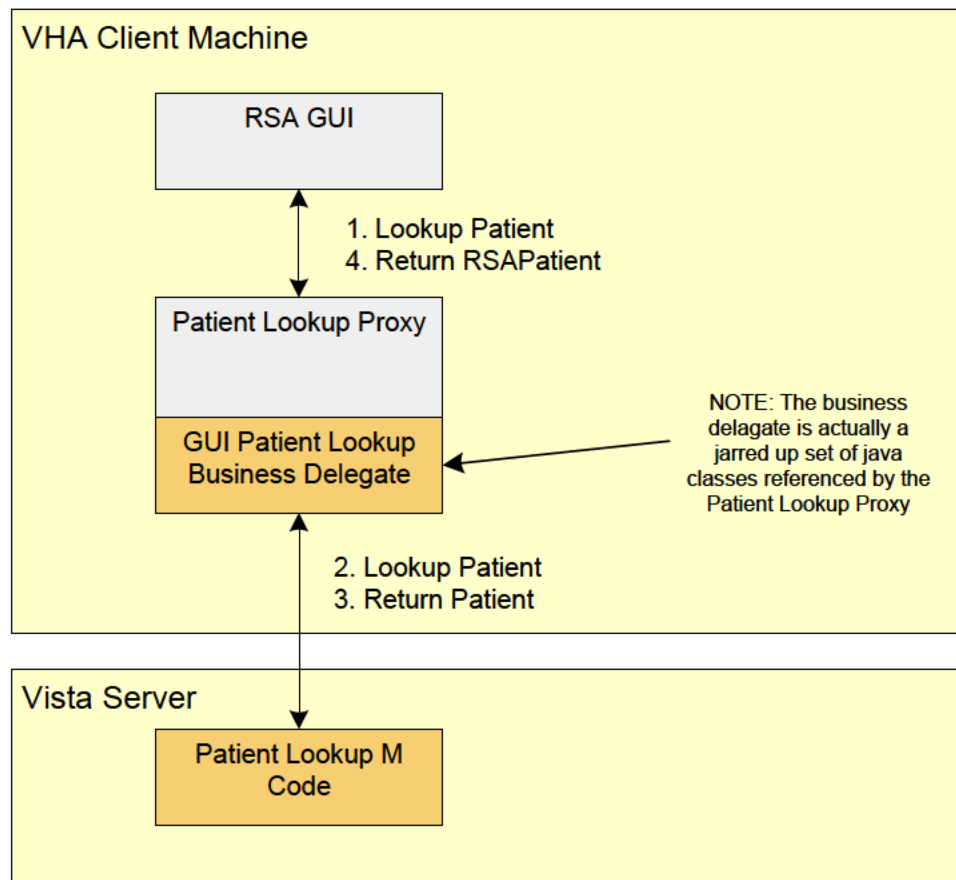


**Figure 48. Retrieve Resource**



### 2.5.3.7 Lookup Patient

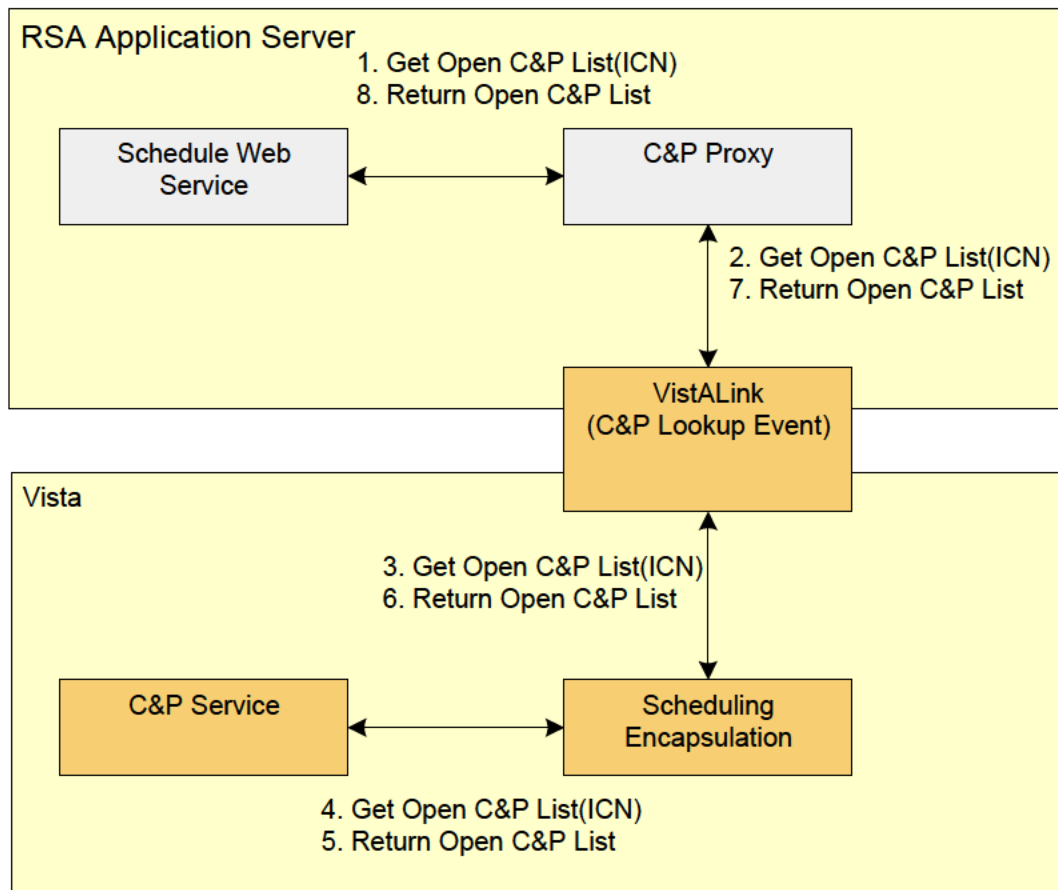
The Patient Lookup component is utilized whenever an internal RSA process needs to retrieve patient information when they do not have a pointer to a patient on the internal RSA screen. The RSA Client application will call the Patient Lookup component's retrieve patient method as shown in Figure 49. This Patient Lookup component will go through the process of looking up a patient and then will return to the RSA GUI a subset of the chosen patient's identifying data and the RSA Client application can then process the data as needed.



**Figure 49. Lookup Patient**

### 2.5.3.8 Lookup C&P

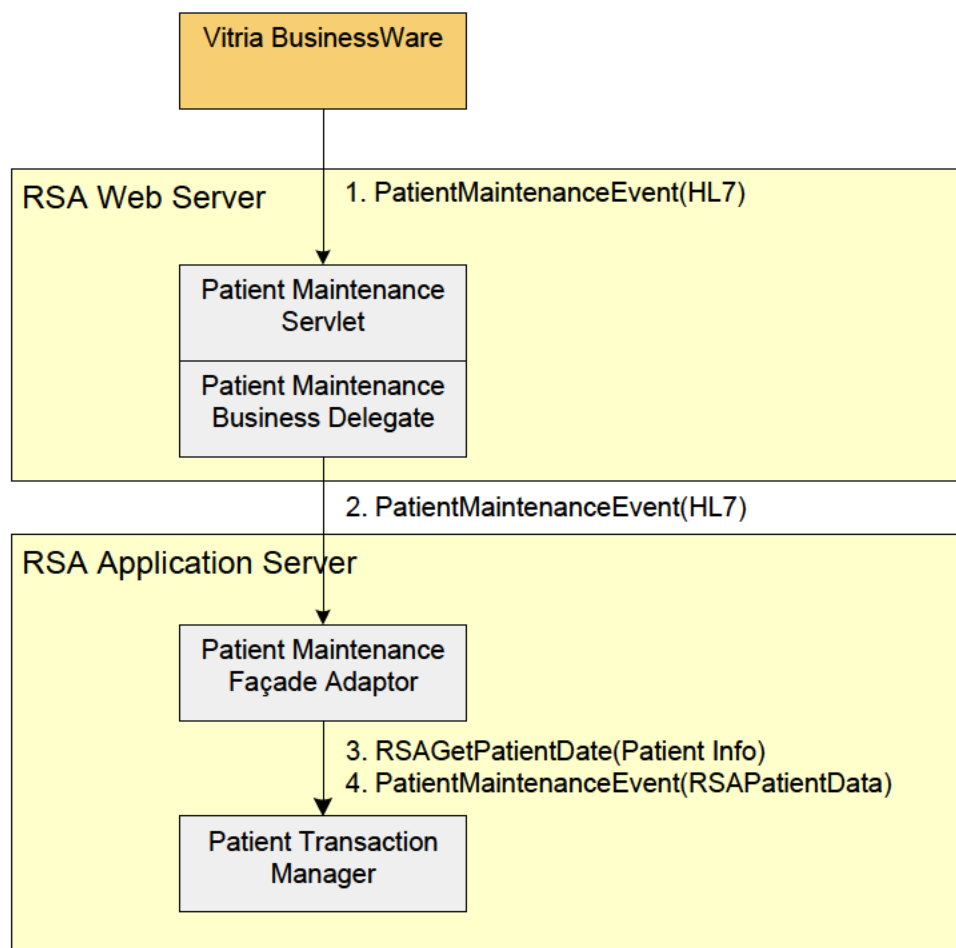
The C&P information is needed by RSA when an appointment request is being made that is for an appointment type of compensation and pension. This type of appointment must be linked to a specific 2507 form and the C&P service is the authoritative source of the 2507 forms. As shown in Figure 50, when this data is needed, the Schedule Web Service will call the C&P Proxy with the patient's ICN. This will be sent to Scheduling Encapsulation who will then communicate with the C&P Service to retrieve the list of active 2507's for this particular patient. This information will be returned to the calling Scheduling Web Service and sent back to the RSA client application so the correct 2507 can be linked to the request.



**Figure 50. Lookup C&P**

### 2.5.3.9 Patient Maintenance Event

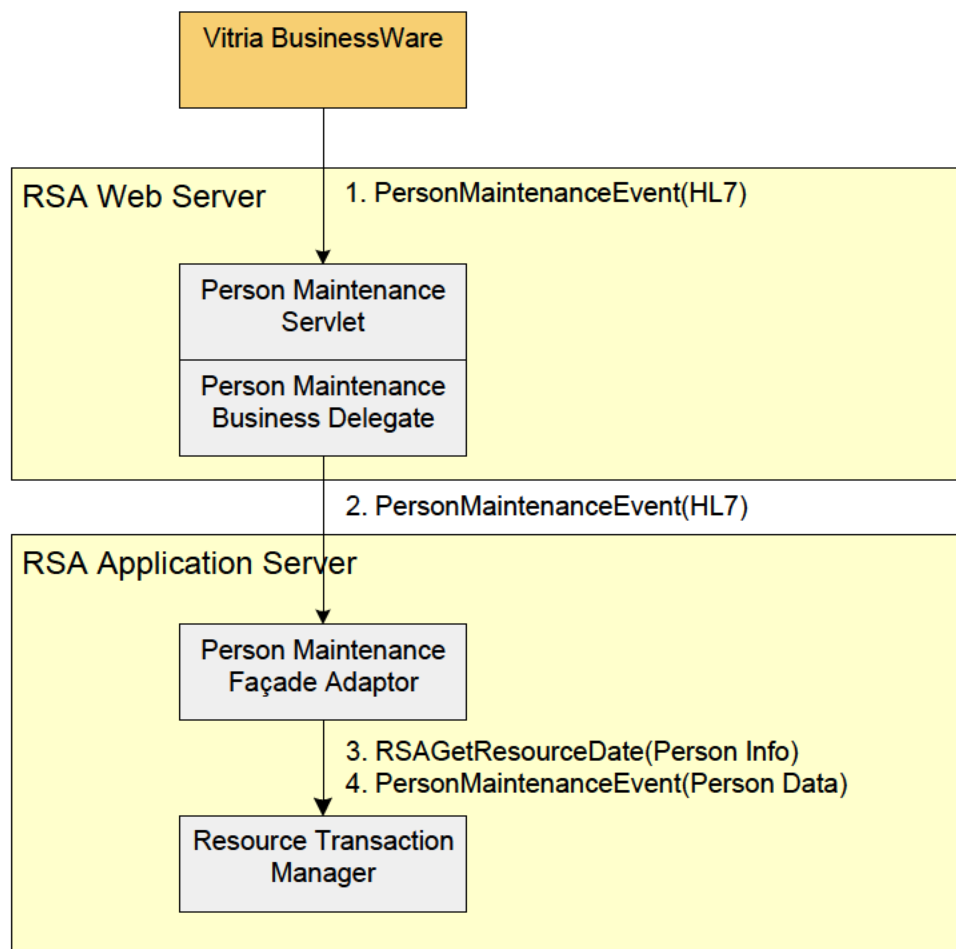
RSA is not the authoritative source of patient information but RSA does store a subset of patient data to enhance scheduling performance. RSA utilizes asynchronous patient update events sent by multiple VHA applications to keep the subset of data synchronized with the authoritative source for that data. The patient maintenance event is sent from Patient Services when the patient data changes as shown in Figure 51. This event will always pass through the Vitria interface engine, which will initiate the call to the Patient Maintenance Servlet. This servlet receives the event and calls the Patient Facade Adaptor EJB. This EJB is responsible for parsing the HL7 to determine what kind of message was received. The EJB will retrieve the current patient data from the RSA database and compare the old data with the newly received data. Any inconsistent data that RSA keeps track of will be updated in the RSA database.



**Figure 51. Patient Maintenance Event**

### 2.5.3.10 Person Maintenance Event

RSA is not the authoritative source of person or provider information, but RSA does store a subset of person data such as the provider class (cardiologist, anesthesiologist, etc.) and the VHA Person Identifier (VPID). RSA utilizes asynchronous person update events sent by multiple VHA applications to keep this subset of data synchronized with the authoritative source for that data. The person maintenance event is sent from these applications when the person data changes as shown in Figure 52. This event will always pass through the Vitria interface engine, which will initiate the call to the Person Maintenance Servlet. This servlet receives the event and calls the Person Facade Adaptor EJB. This EJB is responsible for parsing the HL7 to determine what kind of message was received. The EJB will retrieve the current person data from the RSA database and compare the old data with the newly received data. Any inconsistent data that RSA keeps track of will be updated in the RSA database.



**Figure 52. Person Maintenance Event**

### 2.5.3.11 External Appointment Query

An appointment lookup event is an event sent by an external source requesting appointment data from the RSA as shown in Figure 53. The request can originate from Scheduling Encapsulation and come in via VistALink or originate from a rehosted Java application and come in via the EJB Interface. In either case, the request will come into the Appointment Query Facade Adaptor. The request will consist of two parts. The first part will be the data indicating the parameters for the query (examples are date ranges, specific sections, etc.) and the second part will be parameters indicating what parts of the appointment should be returned (appointment ID, check-in time, appointment type, etc.). The Appointment Lookup Facade Adaptor will take the outside request parameters and turn them into RSA specific queries (i.e., turning a patient ICN into an RSA Patient ID). Once all data transformations have been completed, the Appointment Lookup Facade Adaptor will call the Appointment Transaction Manager passing in the query parameters. The data will be returned from the Transaction Manager. The Appointment Lookup Facade Adaptor will then select the return parameters and send them back to the originating source.

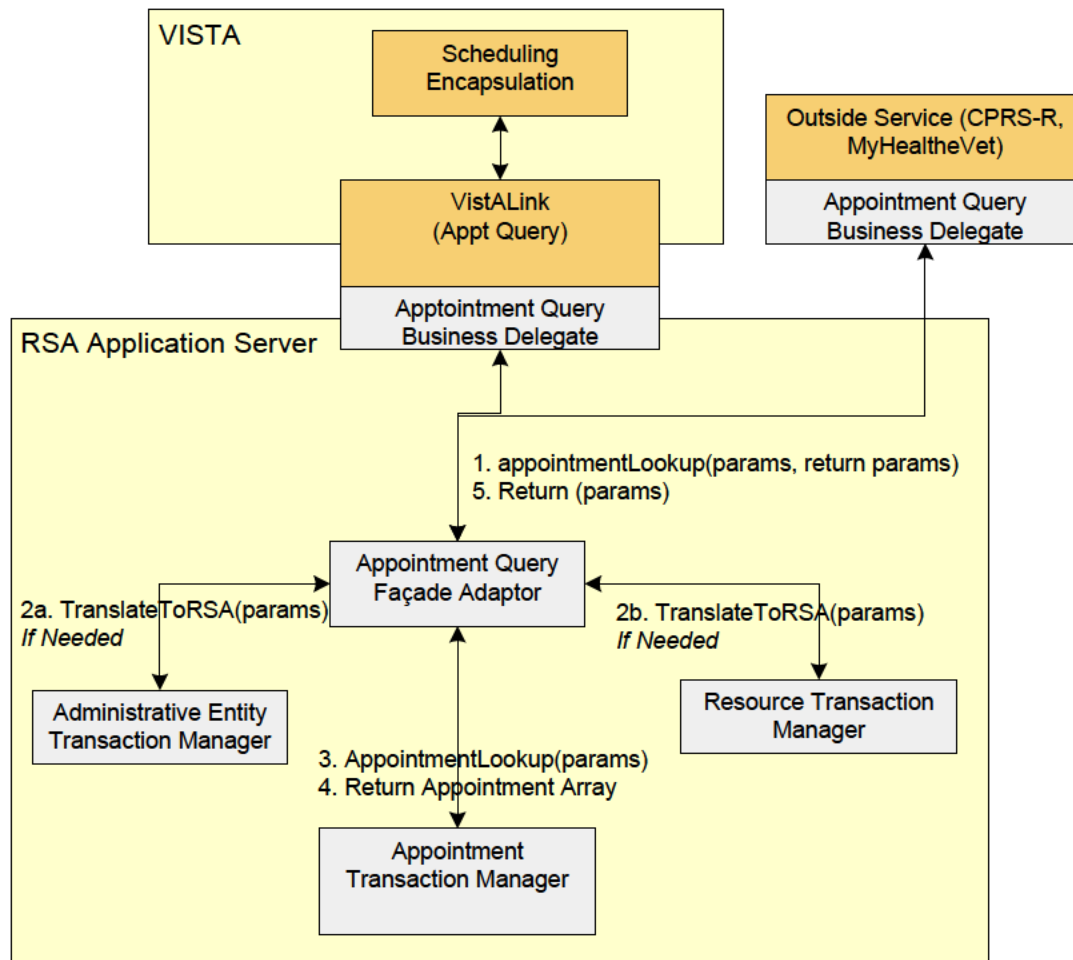
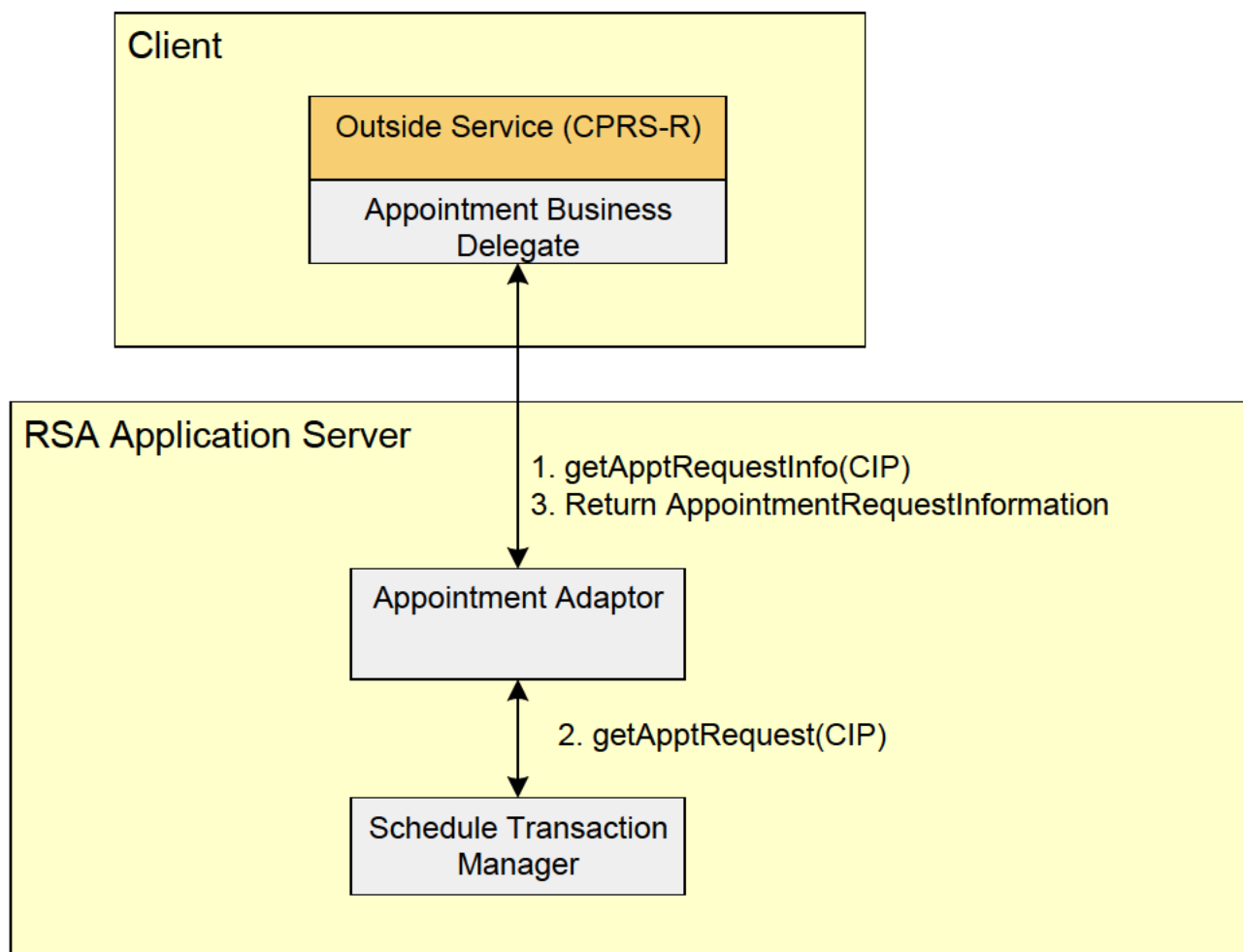


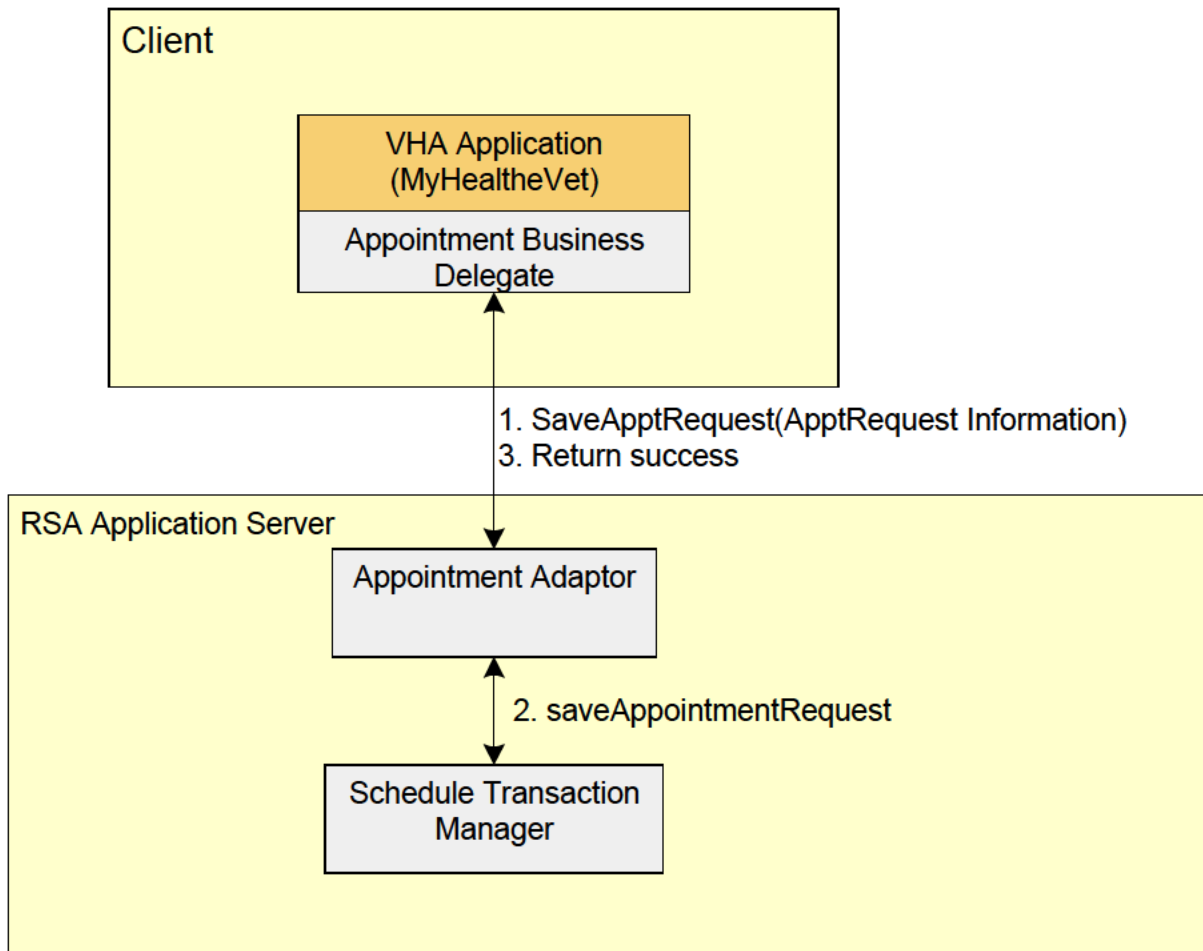
Figure 53. External Appointment Query

### 2.5.3.12 External Appointment Request

External applications, such as MyHealthVet, can make appointment requests in the RSA. This is accomplished by the external application first requesting the information necessary to make a request from the RSA and then using specific pieces of that information to save the request. Appointment requests are specific to a CIP; therefore, the external applications must specify a specific CIP when they request the data to make an appointment request. The data sent back includes local appointment purposes, allocated resources, etc. The external application will display this data and allow the user to select certain fields. This data will then be sent back to RSA to be saved as an appointment request. Figure 54 and Figure 55 show the initial call by the application to get the CIP's information (Figure 54) and the saving of the appointment request (Figure 55).



**Figure 54. Retrieve Request Information**

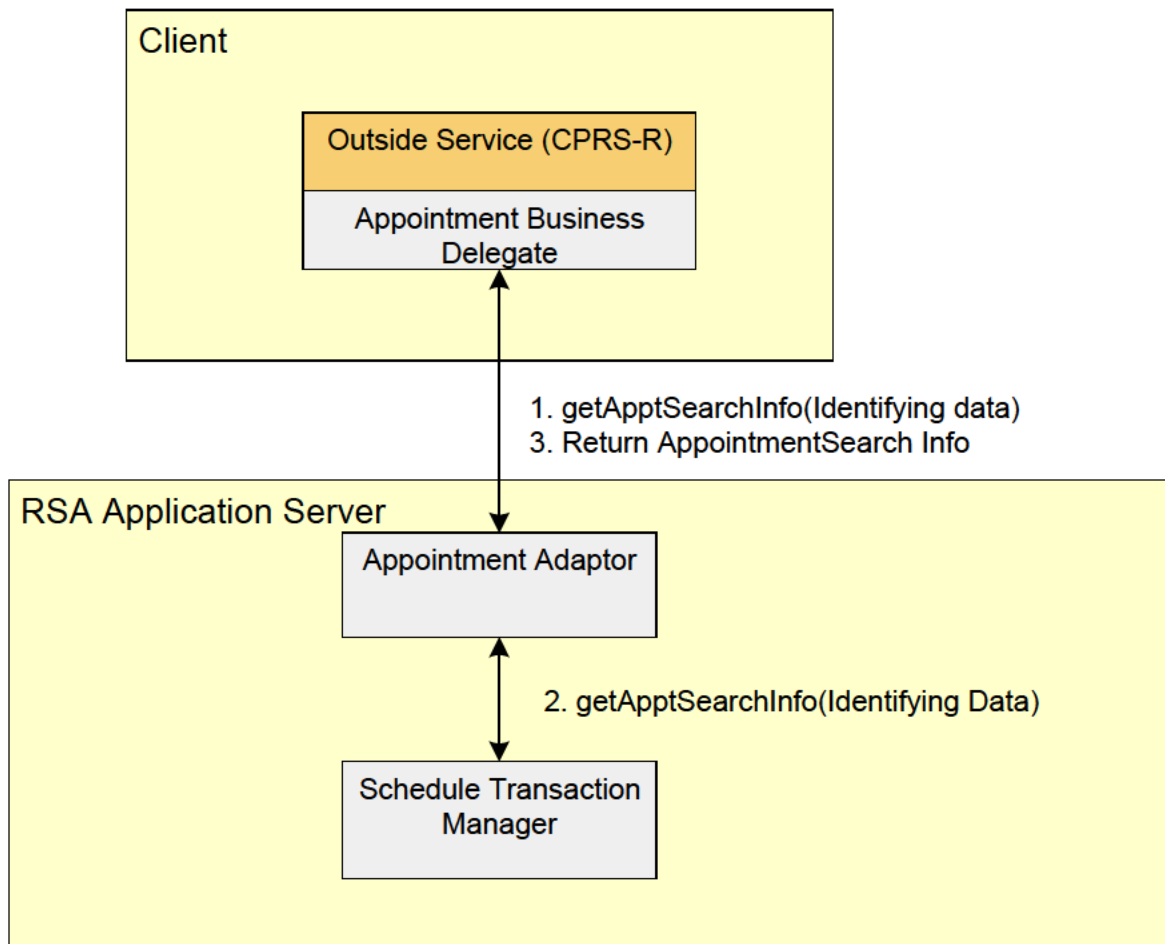


**Figure 55. Save Appointment Request**

### **2.5.3.13 External Appointment Book**

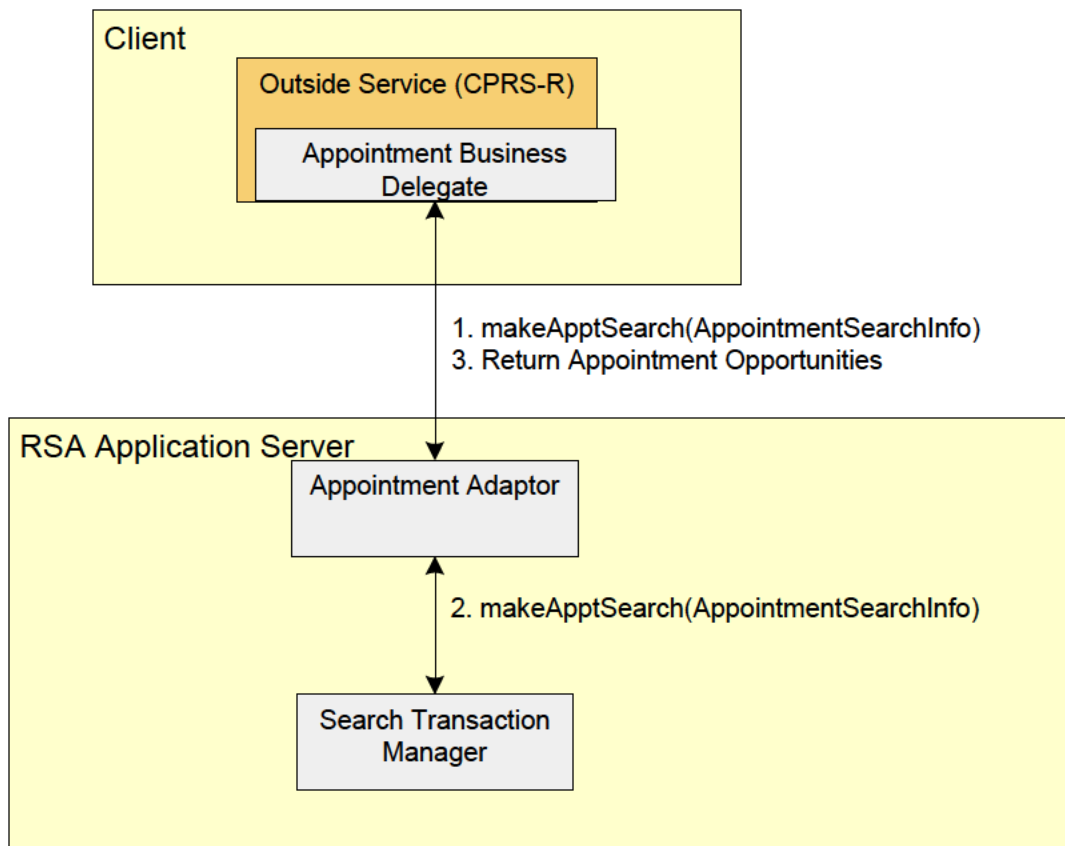
External applications, such as MyHealthVet, can make appointments in some sections depending upon that section's configuration. These external applications must go through three separate and distinct steps to make an appointment:

1. Request the information from the section that will be used to search for an appointment. This includes such things as appointment type, appointment purpose type, specific resources requested, etc., as shown in Figure 56.
2. Once the appropriate subset of data has been chosen from the data returned in Step 1 and after providing other data, the external application will need to request from RSA a list of available appointment opportunities slots that meet the criteria as shown in Figure 57.
3. After choosing one of the appointment opportunities from the list returned in Step 2, the external application will need to return that appointment opportunity slot to be saved to the RSA database (Figure 58).

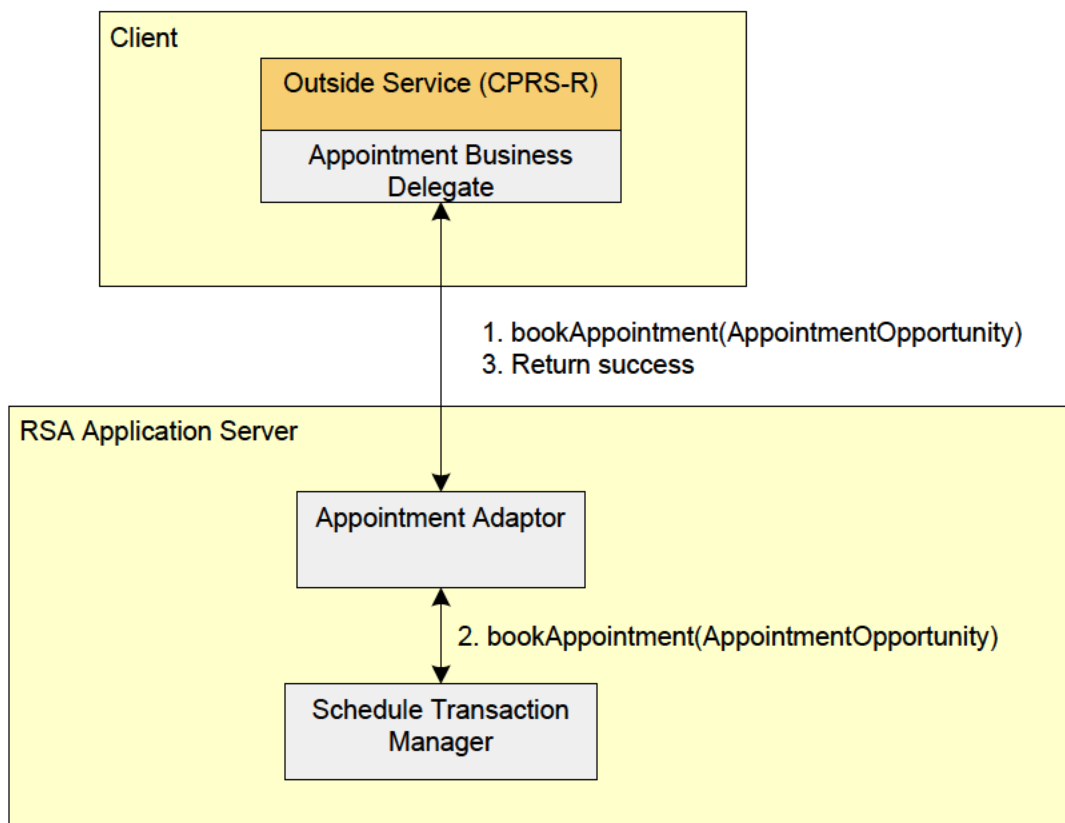


**Figure 56. Getting Appointment Search Data**





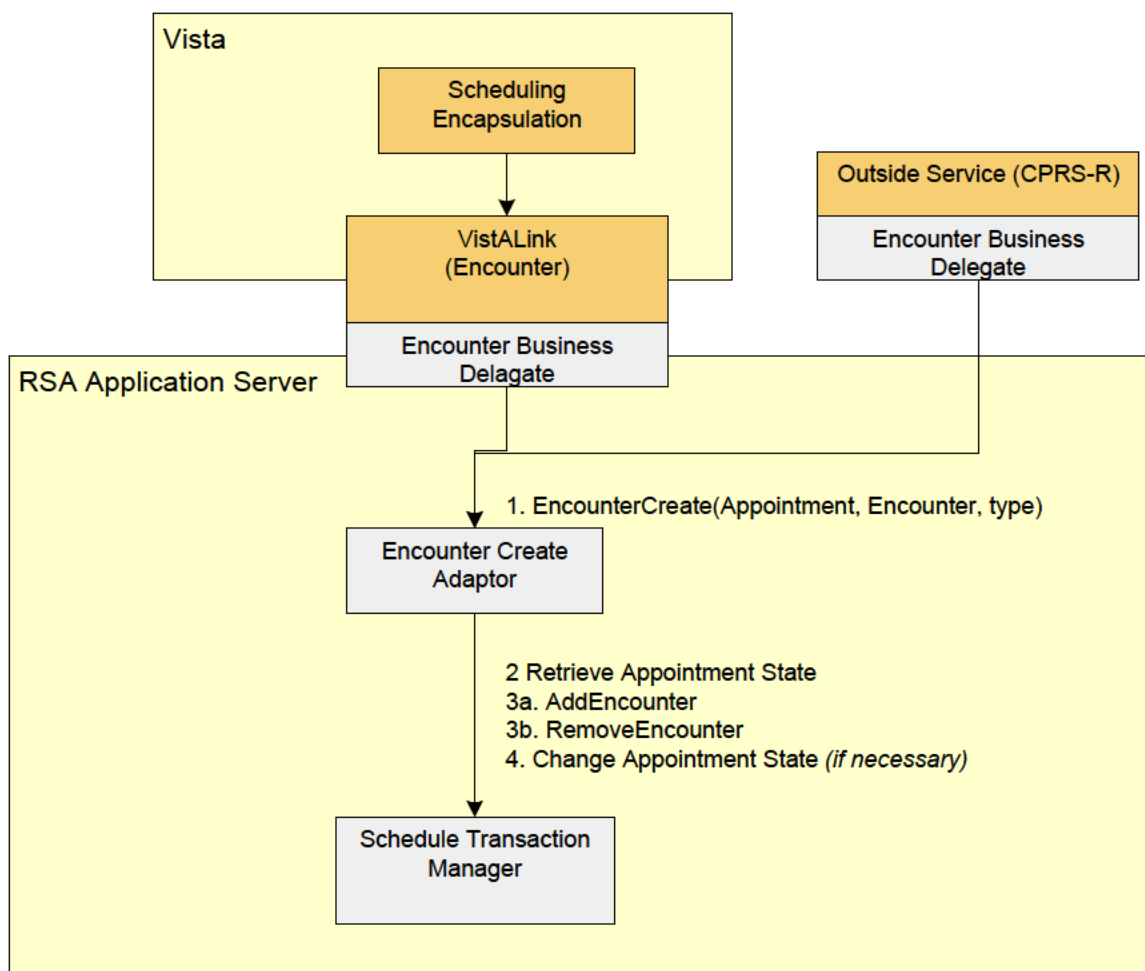
**Figure 57. Make Appointment Search**



**Figure 58. Book Appointment Event**

#### 2.5.3.14 Create Encounter Event

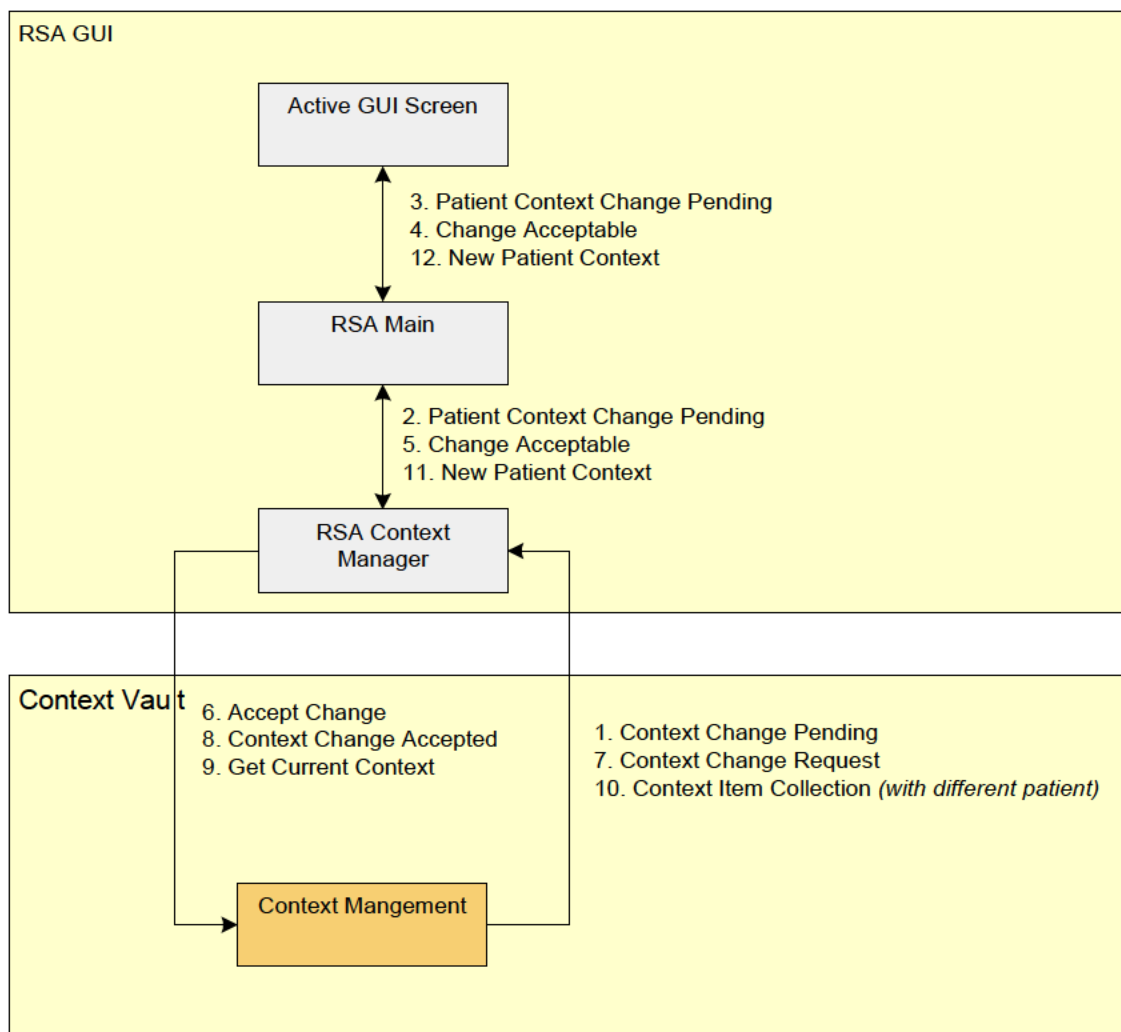
An external application, such as CPRS, may create an encounter for each appointment made by the RSA. When an external application creates an encounter, it will send that encounter to RSA via the Encounter Create Facade Adaptor. The create encounter event will have three parameters: the appointment ID, the encounter ID, and a type parameter indicating if it is necessary to change the appointment state. There are four types of create encounter events: “Encounter Create,” “Encounter Create and State Change,” “Encounter Delete,” and “Encounter Closed.” When the event is received, the appointment state may change based upon the current state and the type of event received. The process of these changes is shown in Figure 59.



**Figure 59. Create Encounter Event**

### 2.5.3.15 Patient Context Change

The RSA Client application is synchronized with all other clinical applications on the user's desktop via the context management software. All clinical applications will be working on the same patient at the same time. This means that the same patient will be in context in every application. When a clinical application wants to place a patient in context, it will call the context management software and that software will send a "context change pending" notification to all other applications on the user's desktop that are participating in context management. As shown in Figure 60, when the RSA Context Manager receives this notification, it will notify the currently active GUI component that a context change is pending. The GUI component may respond with "Accept" or "Conditional Accept." In most cases, the GUI will "Accept" the context change. Once all applications have accepted the change, the RSA will retrieve the context item collection that holds the new patient information. This information will be sent to the currently active GUI component and this will be the new patient shown in context in RSA.



**Figure 60. Patient Context Change**

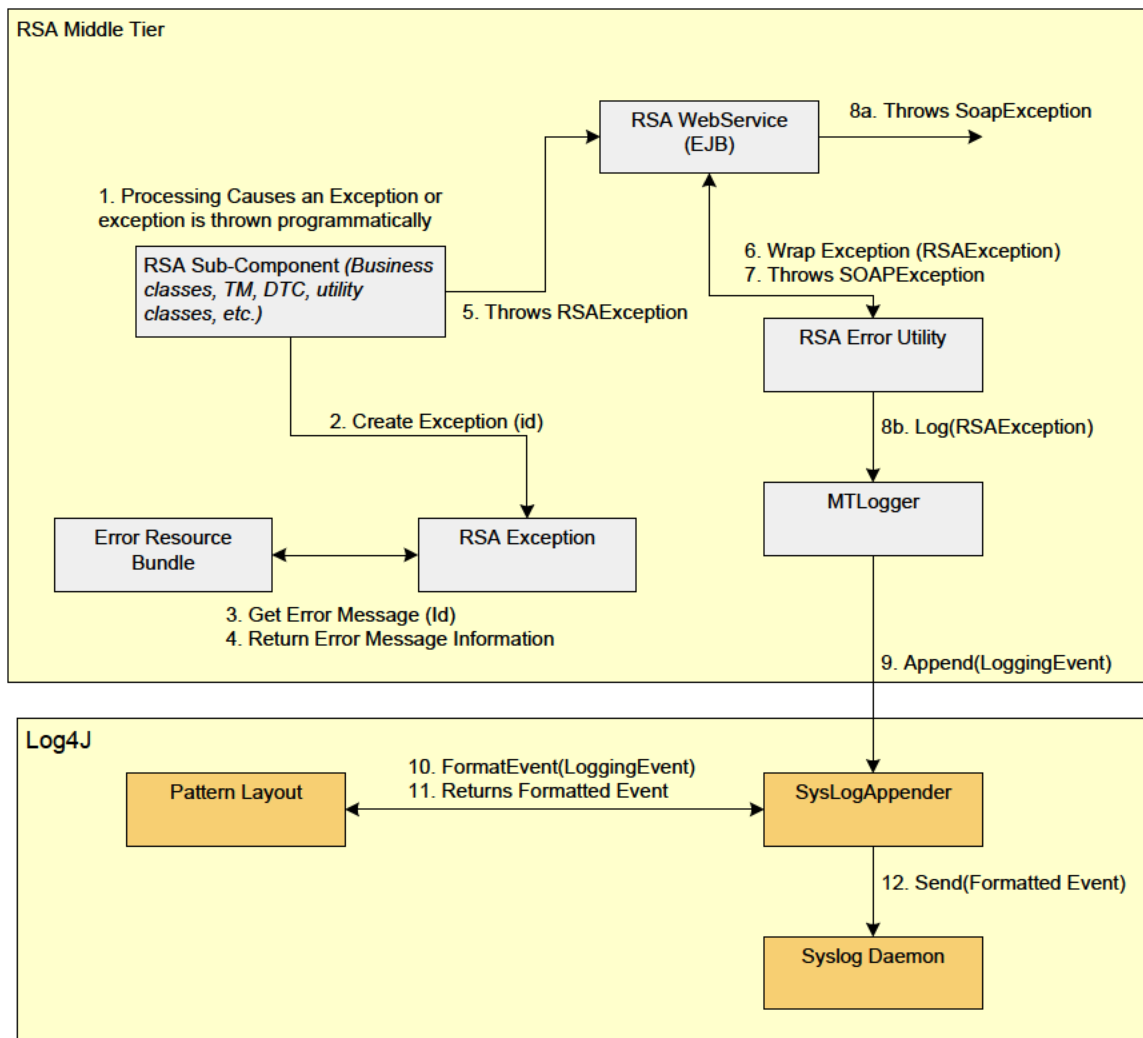
## 2.5.4 Exception and Error Handling

The Java language uses exceptions to provide error-handling capabilities for its programs. An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. RSA uses 'exception chaining' to retain information on the history and original causes of errors. The application also uses checked exceptions to allow exceptions to be handled in the most appropriate location in the application.

RSA groups exceptions into logical categories and uses resource bundles to store error messages. The application sends error/exception information to the RSA client tier from a web service using SOAP exceptions. It also sends error/exception information to interfaces from an EJB using regular checked Java Exceptions that contain the unique RSA error code and a user-readable error message. RSA Components such as DTCs, transaction managers, and business classes will create an RSA exception when an error event occurs and put in a usable readable message. The exception will be thrown and caught at the highest level. This level is either the Web Service class or the EJB class. At that point, the error will be wrapped into a SOAP Exception, logged to the Syslog, and thrown back to the calling application.

Logging for RSA middle-tier components is done on the 9iAS application server through the Linux/Unix Syslog daemon. The logging is done utilizing the log4J utilities. These utilities provide RSA the ability to format the messages and write to the syslog. Logging for client application components is done on the client workstation through the Windows NT Event Viewer. All errors are logged at the web service before sending the error up to the client tier. They are also logged at the client application, just before passing a message to the user. Additional errors and messages may be logged to these loggers by calling the loggers directly from any RSA Component.

Exception handling and logging in RSA is depicted in Figure 61.



**Figure 61. Exception Handling and Logging in RSA**

## 3. SYSTEM DESIGN

### 3.1 Interface Design

RSA is a software application that will replace that portion of the current **VistA M** application that performs outpatient scheduling. To perform its internal functions, the RSA will require data owned and maintained by external applications. Likewise, external applications will require data owned and maintained by the RSA. Thus, interfaces will exist that allow data to be exchanged between the RSA and external applications.

As previously shown in Figure 11, External Interface Components, RSA implements several mechanisms to interface with external applications.

- **Facade Adaptors:** These components allow external applications to pass data to RSA.
- **Proxy Adaptors:** These components enable RSA to pass data to external applications.
- **VistALink Interface:** Interface to the **VistA M** system.
- **Event Servlets:** Receive HL7 events from Vitria Businessware.

For a more comprehensive description of each of these components, refer to Section 2.3.3.4, External Interface Components. For detailed design information of each component, refer to Section 3.2, Software Module Detailed Design. For detailed information and descriptions of the data being passed across these interfaces, refer to the Interface Control Document (ICD), Version 2.03, 20 January 2004.

### 3.2 Software Module Detailed Design

Rational Rose (Version 2002.05.01) from Rational Corporation is the software design tool of choice for the RSA project. Within Rational Rose, a model of RSA is developed and is the primary design artifact for RSA. Rational Rose offers many benefits to support large teams of analysts, architects, and software developers. In particular, Rational Rose:

- Allows parallel development of a model by supporting decomposition of the model into versionable units called controlled units.
- Permits model files and controlled units to be moved or copied among workspaces by using the virtual path map mechanism.
- Is integrated with ClearCase, the configuration management tool used on the RSA project.
- Automatically adds existing frameworks to the model (i.e., J2EE).

Detailed design documentation has been generated from the Rational Rose model and is viewable as multiple HTML files included on the delivery media with this document at: [DesModel\DesModel.htm](#).

To view the Rose model documentation, it is necessary to have a browser installed on your computer. The model appears as a tree view on the left with class specifications, diagrams, etc.,

appearing on the right. The Logical View of the model has been provided as detailed design documentation. A component view has not been included since Section 2 of this document already describes each component of RSA. The Use Case view is not included since these are described in Appendix A of the Software Requirements Specification (SRS). It is assumed that the reader of this documentation has some familiarity with Rational Rose and the Unified Modeling Language (UML). Detailed instruction on Rational Rose and UML is not within the scope of this document.

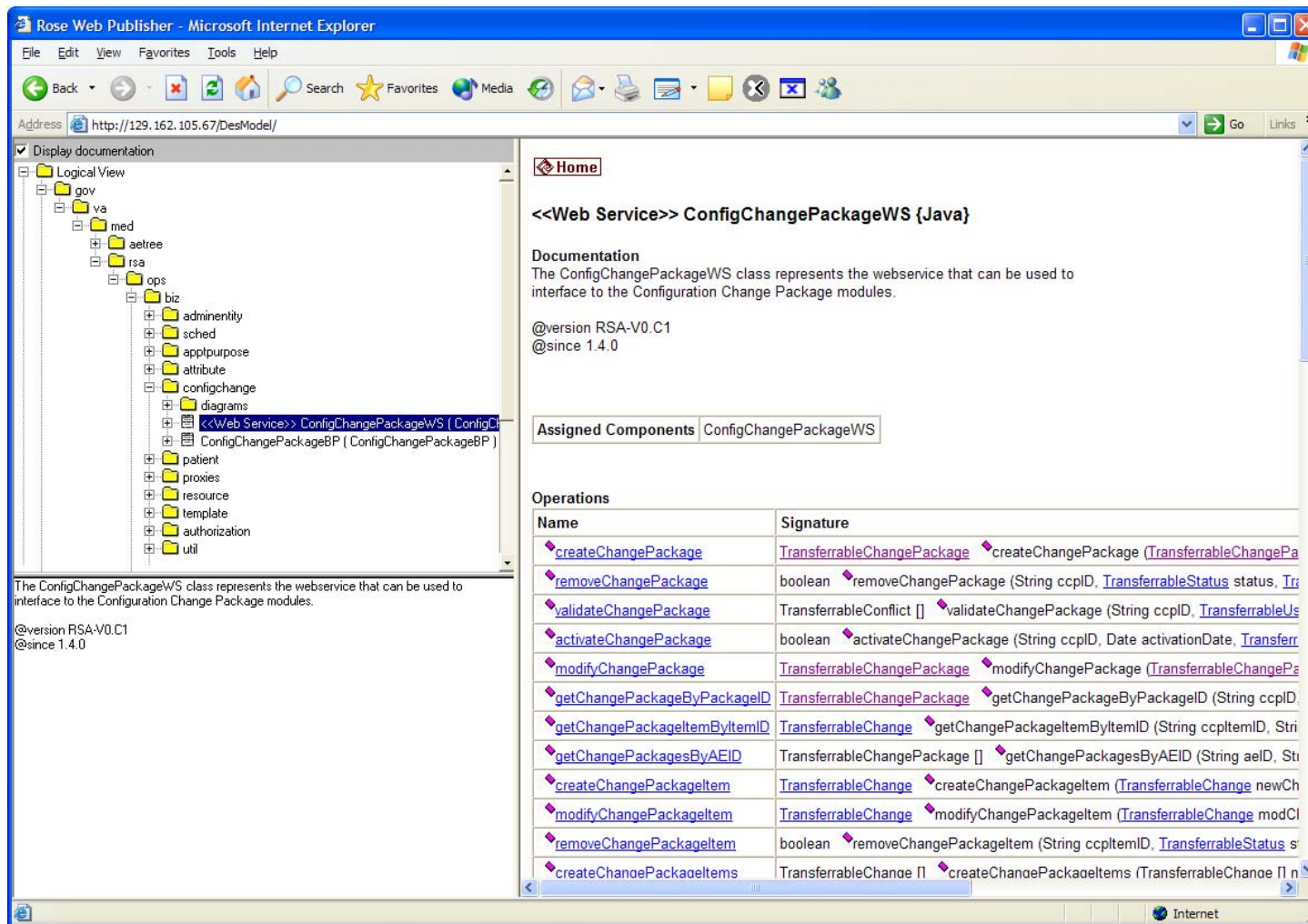
The following is a brief description of the high-level packages in the Rose model's Logical View.

- ***gov.va.med.rsa.ops.biz:*** This package contains middle-tier components such as web services, adaptors, Data Tier Components, and Transaction Managers.
- ***gov.va.med.rsa.ops.cmn:*** This package contains data objects
- ***gov.va.med.rsa.ops.ext:*** This package contains the Vitria and VistALink interfaces.
- ***gov.va.med.rsa.ops.gui:*** This package contains RSA client components such as GUI panels, DataManagers, etc.
- ***gov.va.med.rsa.ops.util:*** This package contains utility classes.
- ***gov.va.med.rsa.ops.eventservlets:*** This package contains event handlers for HL7 events.

To view a class Specification, first navigate down the Logical View in the tree and double click the class of interest. The class Specification will be shown in the right pane as shown in Figure 62.

Diagrams have been placed in “diagram packages” for most components. Detailed design diagrams such as class diagrams and sequence diagrams are displayed by double clicking the diagram in the tree as shown in Figure 63.





**Figure 62. Class Specification in Web Published RSA Design Model**

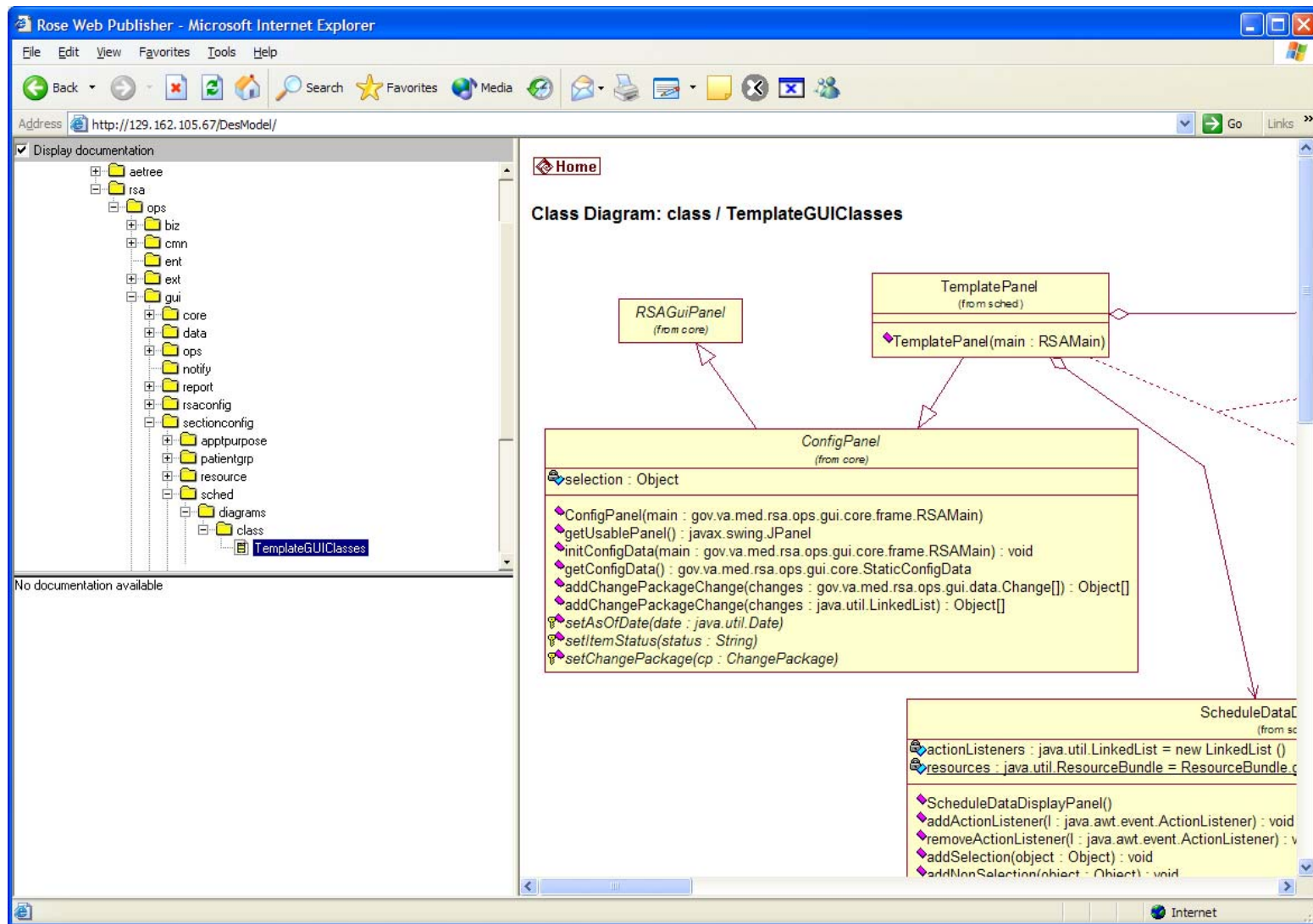


Figure 63. Class Diagram Displayed in the Web Published RSA Design Model.

Java classes in the design model are assigned to a component with the component name appearing in parenthesis after the class name. Rational Rose models only allow one public class to be assigned to a component. Therefore, classes in the RSA design model are typically assigned to a component of the same name. Additionally, stereotypes are frequently used to aid developers and system designers. A stereotype represents the subclassification of a model element. It represents a class within the UML metamodel itself (i.e., a type of modeling element) and they appear in brackets to the left of the class name in the tree view. Custom stereotypes are used to represent the basic building blocks of the system and convey additional information when appropriate. Custom stereotypes used in the RSA design model are:

***ApplicationModuleImpl:*** The class is a BC4J derived Application Module.

***ViewObjectImpl:*** The class is BC4J derived ViewObject.

***EntityObjectImpl:*** The class is a BC4J derived Entity Object.

***ViewRowImpl:*** The class is a BC4J derived View Row.

***Web Service:*** A web service class.

***Interface:*** A Java interface:

***Work In Progress:*** Indicates the design of this class is not yet complete and has not been through peer review.

***ThirdParty:*** Third party libraries used within RSA such as the Sentillion Java libraries used to implement context management.

***Static:*** A static method.

***Abstract:*** An abstract class.

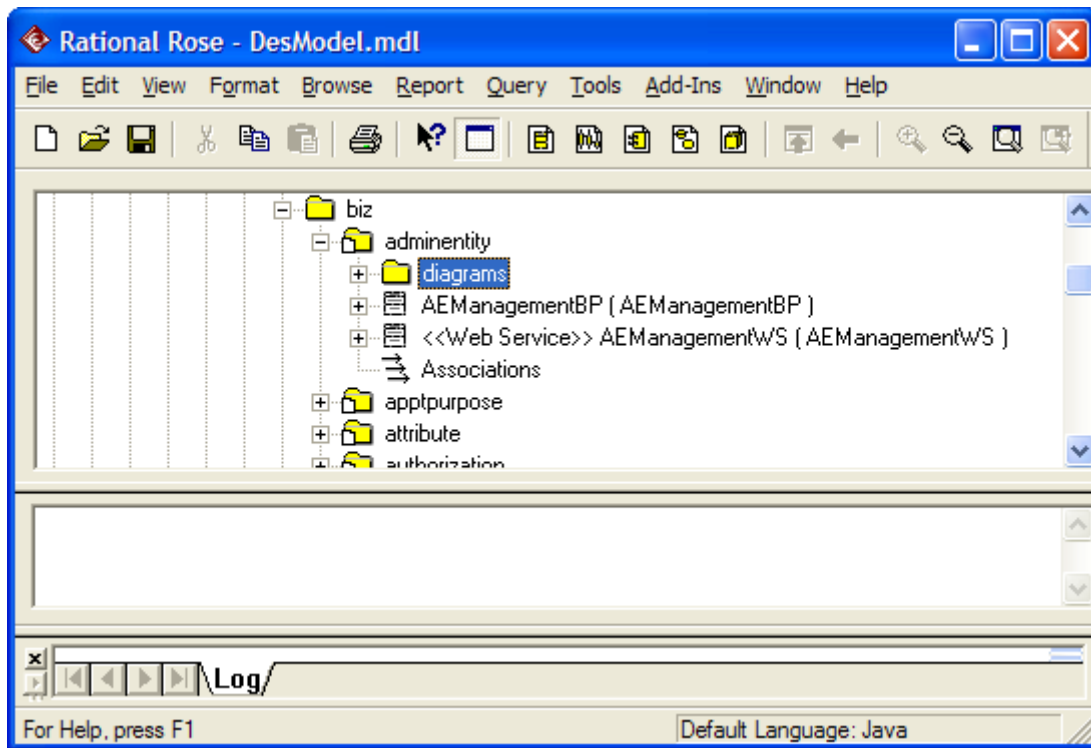
***Data Object:*** A data transfer object. Sometimes referred to as a “value object.”

***EJBRemoteInterface:*** An interface to the main EJB class.

***EJBHomeInterface:*** Finds and creates EJBs.

***EJBRemoteMethod:*** A public EJB method.

An example is shown in Figure 64. This figure shows two classes in the RSA design model named “AEManagementBP” and “AEManagementWS.” AEManagementBP is assigned to a component of the same name, AEManagementBP. The second class, AEManagementWS is assigned to a component named AEManagementWS and is stereotyped as a web service.



**Figure 64. Classes May Be Stereotyped and Assigned to a Component**

## 4. DATA DICTIONARY

Appendix B, RSA Database, contains detailed information on the RSA database (VADB). A complete and comprehensive listing of all data stored in VADB is available in electronic format at [DatabaseDocs\ckvwdef.pdf](#). This report has been generated from Oracle Designer and constitutes the data dictionary for RSA.

## **5. REQUIREMENTS TRACEABILITY**

The requirements traceability has not been incorporated into the Software Design Document (SDD) at this time and will be finalized when the design model has been completed. The traceability of requirements is influenced as the final details of the design model are completed, and as the iterative development cycle of RSA transitions through construction iterations. This parallel maturation of the detailed design and requirements traceability through the RSA construction iterations will necessitate an update to the SDD to reflect the updated design model and its corresponding requirements traceability matrix. It is anticipated that the final “As Built” design model and corresponding requirements traceability matrix will be provided as an update to this SDD sometime in the early phases of Alpha testing.

## **APPENDIX A. GLOSSARY, TERMS, AND ACRONYMS**

## APPENDIX A - Glossary, Terms, and Acronyms

ADT	Admission Discharge Transfer
AE	Administrative Entity
AP	Appointment Purpose
API	Application Programming Interface
AS	Application Server
ATS	Administrative Entity Tree Service
BC4J	Business Components for Java
BPR	Business Process Re-engineering
BSAD	Baseline System Architecture Document
C&P	Compensation and Pension
CAIP	Cross Application Integration
	Protocol/Project/Prototype/Specification
CCOW	Clinical Context Object Workgroup
CCP	Configuration Change Package
CIP	Check In Point
CPRS	Computerized Patient Record System
DB	Database
DML	Data Manipulation Logic
DSS	Decision Support System
DTC	Data Tier Components
EJB	Enterprise Java Bean
ERD	Entity Relationship Diagrams
ERDC	Engineering Research and Development Contract
FBPM	Future Business Process Model
GUI	Graphical User Interface
HL7	Health Level 7
HLF	Hospital Location File
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICD	Interface Control Document
ICN	Integration Control Number
ID	Identification or Identifier
IEN	Internal Entry Number
IT	Information Technology
J2EE	Java 2 Enterprise Edition
JNDI	JAVA Naming and Directory Interface
JSP	Java Server Pages
LDAP	Lightweight Directory Access Protocol
LWOBS	Left Without Being Seen
M	MUMPS – Massachusetts General Hospital Utility
	Multi-Programming System
OC4J	Oracle 9iAS Containers for J2EE
OID	Oracle Internet Directory



OO	Object Oriented
PL/SQL	Procedure Language/Structured Query Language
RAC	Real Application Cluster
RFP	Request for Proposal
RPC	Remote Procedure Call
RSA	Replacement Scheduling Application
SDD	Software Design Document
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRS	Software Requirements Specification
SSN	Social Security Number
SSO	Single Sign On
STPL	Short Term Pending List
SwRI	Southwest Research Institute
TM	Transaction Manager
UML	Unified Modeling Language
VA	Veterans Affairs
VADB	Veterans Administration Database
VHA	Veterans Health Administration
VISN	Veterans Integrated Services Network
<b>VistA</b>	Veterans Health Information System and Technology Architecture
VPID	VA Person Identifier
WSDL	Web Service Description Language
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

## **APPENDIX B. RSA DATABASE**

## **APPENDIX B – RSA Database**

### **1.0 CONCEPTUAL MODEL**

At the conceptual level, the RSA database supports two major categories of functionality:

1. Transactions for scheduling or configuration.
2. Reporting for management.

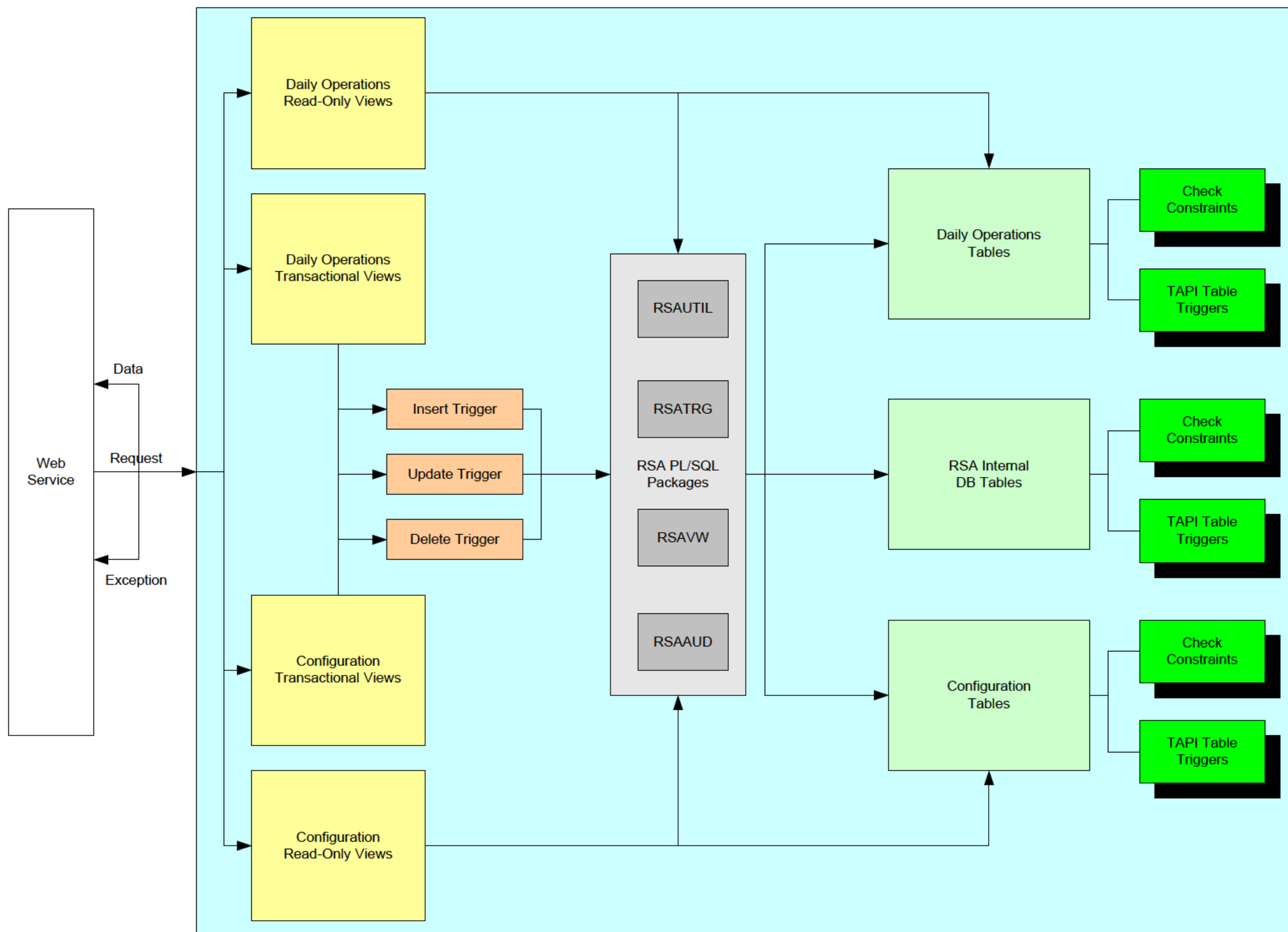
Hence, the RSA database is conceptually both an online transactional processing system (OLTP) as well as an operational data store (ODS). There is a fundamental tension between the design goals of hybrid OLTP and ODS databases. OLTP databases are designed to support efficiencies in transactions (insert, update, and delete operations) as well as data integrity through database normalization. ODS databases are designed to support efficiencies in the retrieval of data; however, they may create difficulties in maintaining data integrity because of the denormalization of table structures. Database design is about picking the right balance between these pressing needs.

#### **1.1 Transactional Conceptual Model**

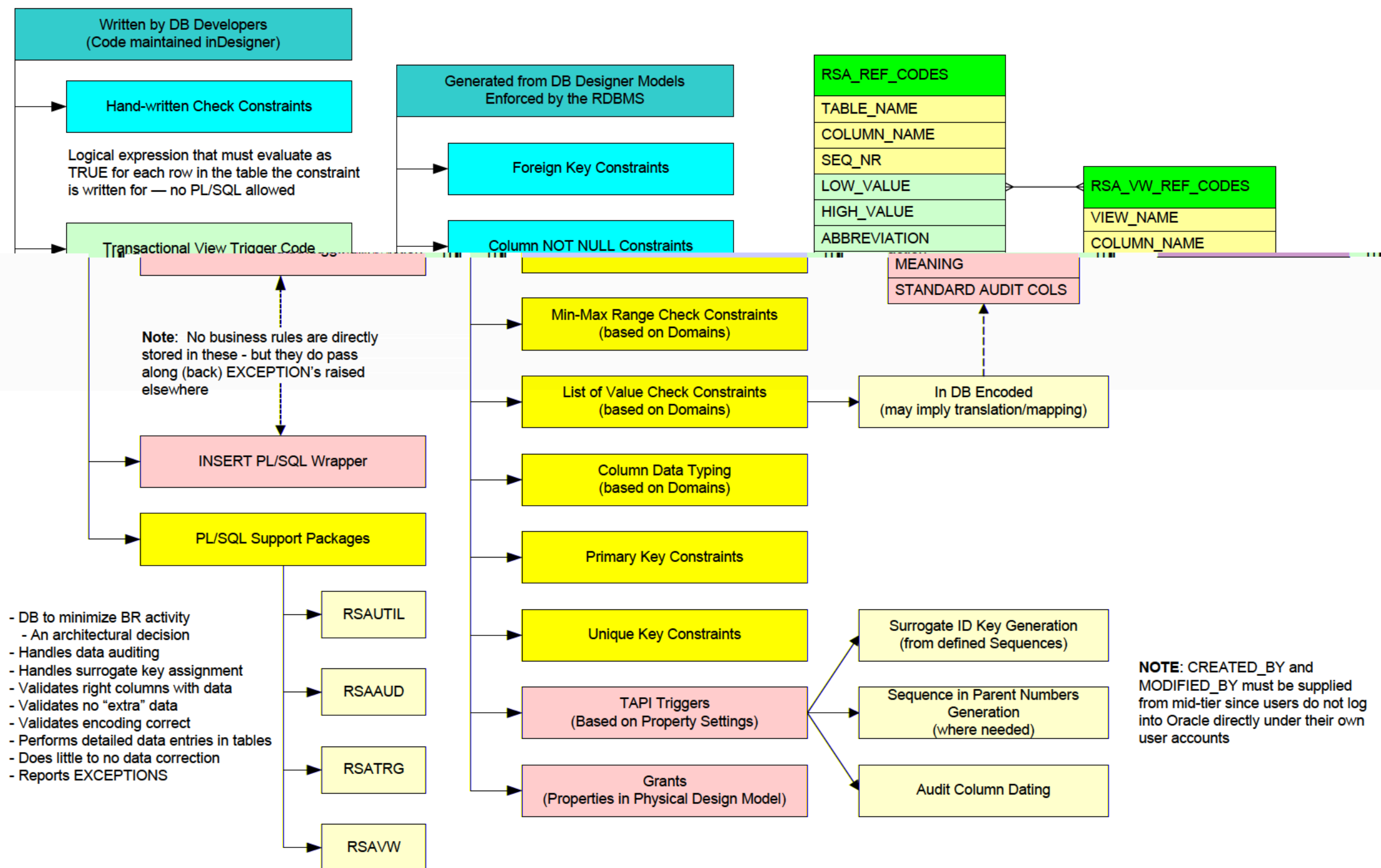
Figure B-1 illustrates the conceptual model for the RSA database as an OLTP database, which the logical and physical models will show to be a mostly normalized database. The RSA middle tier will access the RSA database tables indirectly using a set of read-only and transactional views. Read-only and transactional views will be defined as needed to support the data retrieval and data transactions for RSA daily operations (scheduling and appointment management) and for RSA data configuration and administration of administrative entities and the resources assigned to them.

The key concept in this design is the use of transactional views and Oracle INSTEAD OF triggers. Transactional views in Oracle allow for the simpler interfaces between the middle-tier object-oriented (OO) code and the database tables. A transactional view can be used for data retrieval in the same manner that an ordinary read-only view can be used. Transactional views, however, also allow insert, update, and delete operations to be performed against all the tables that are referenced as part of the view definition. These types of operations are accomplished by use of trigger code that will call Procedure Language/Structured Query Language (PL/SQL) procedures in the database to perform the detailed data manipulation. The middle-tier OO developers, therefore, need to know less about the intricacies of the RSA data tables.

The bulk of the work that each transactional view trigger needs to accomplish happens in the PL/SQL functions and procedures that the trigger directly or indirectly invokes. As shown in Figure B-2, PL/SQL is organized into a set of packages that provide auditing functions (RSAAUD), utility functions (RSAUTIL), and transactional functions (RSATRIG and RSAVW). These packages interact directly with the RSA database tables in all three areas (Daily Operations, Configuration, and Internal).



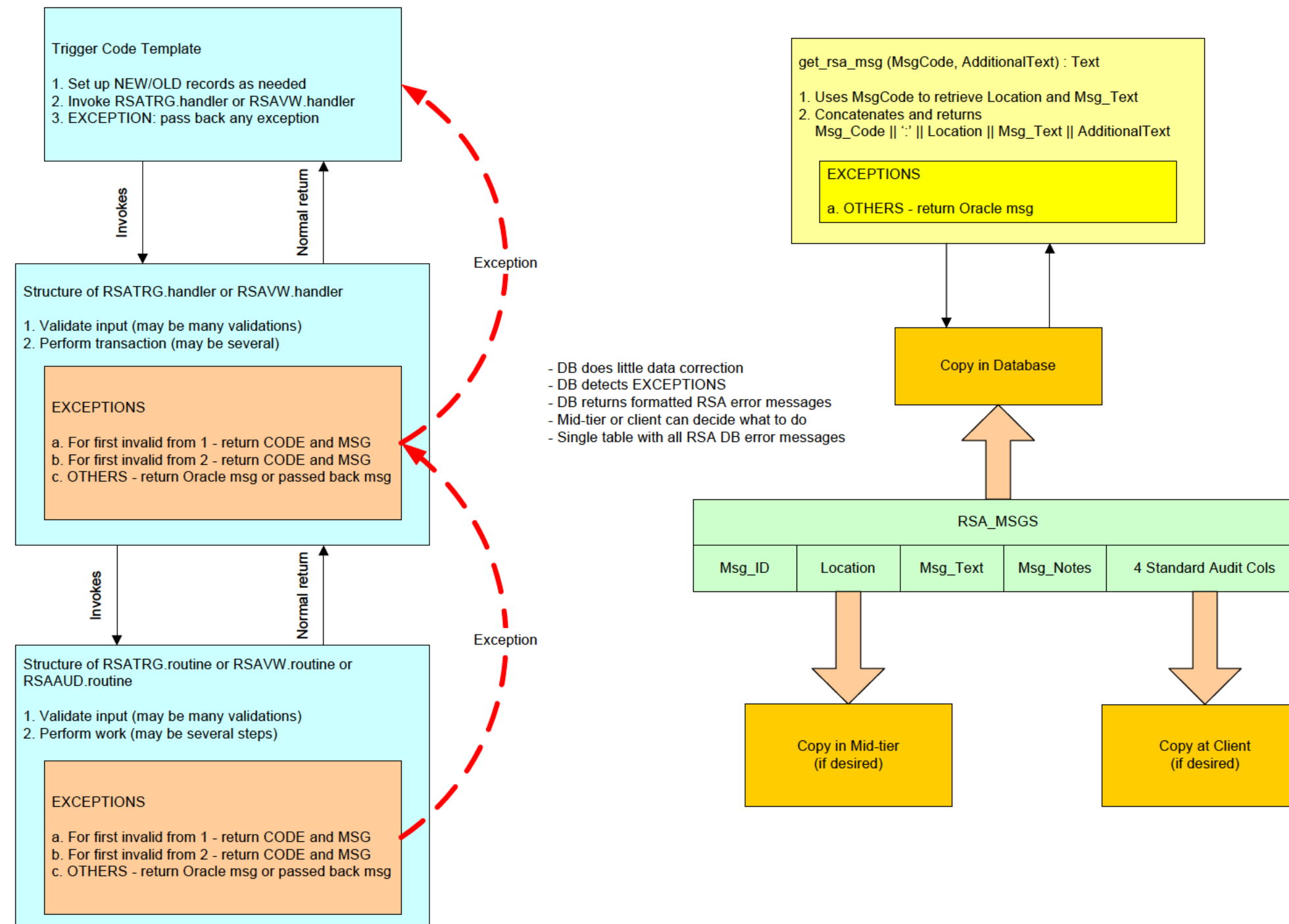
**Figure B-1. OLTP Conceptual Model**



**Figure B-2. Data Integrity in the Database**

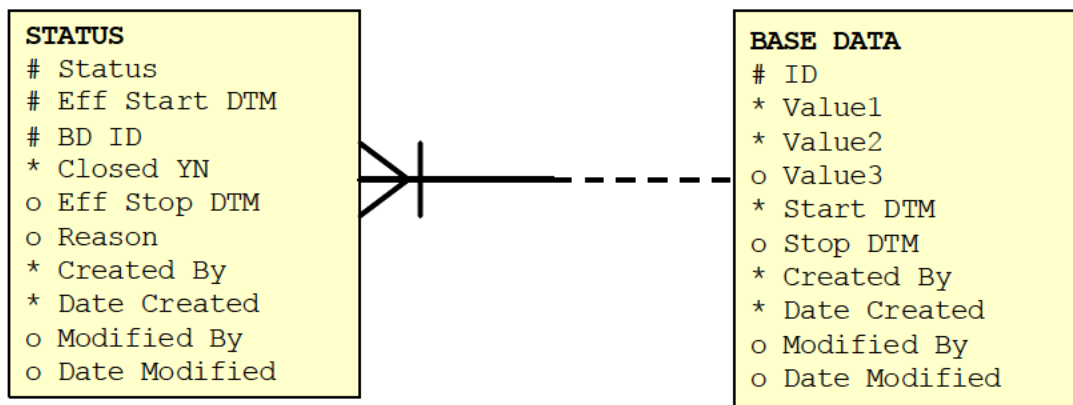
While an overall goal of RSA is to have business rule validation at the middle tier, the database itself also provides for certain levels of data validation. Figure B-2 shows the two major categories of this data validation: validation hand-written by database developers and validation automatically generated by the database models in Oracle Designer. The former includes check constraints and validation done in the trigger code and its invoked PL/SQL; the latter includes ordinary database constraints (such as primary key constraints and foreign-key constraints) arising from the database logical model itself as well as table triggers generated by Oracle's Table API (TAPI) interface. In some cases, these validations are redundant with middle-tier validation; however, they are included in the database design because there are cases now and in the future where data may come into the RSA database from outside the middle tier such as from load scripts.

A final aspect of the conceptual model of the OLTP database concerns exception handling. Figure B-3 illustrates two points in the concept of operation for database exceptions. Firstly, the database does not attempt to take any corrective action when a data exception is detected. Secondly, the database will return either an Oracle exception code or an RSA-specific exception code depending on the nature of the data exception. The former design decision is appropriate since the database is not the final arbitrator when a data exception is discovered; that responsibility rests with the middle. As a result, while the database performs table transactions, it never executes a COMMIT or ROLLBACK. The latter decision does allow the database to provide appropriate feedback to the middle tier about any detected exception in such a way that the middle-tier code may decide what action to take. Note that the RSA-specific database exceptions are maintained in an RSA internal table with a consistent format that will allow the middle tier to determine the nature of the exception.



**Figure B-3. Concept of Operation for Database Exceptions**

Finally, the RSA database includes one ubiquitous database design pattern, illustrated in Figure B-4.



**Figure B-4. Event-Driven Design Construct**

The transactions that occur in RSA are in the form of events. Both the daily operations and configuration functions in the RSA application work against base data items that are rows in tables in the RSA database. As events occur, base data item records are inserted, updated, or logically deleted in the database. A given base data item may go through several states over time. The nature of the allowable states for a given class of base data items is determined by the rules surrounding the class of the base data item. The database design structure is similar to the generic design shown in Figure B-4.

1. The data element is stored in a BASE DATA table and includes:
  - a) A surrogate key ID
  - b) A set of mandatory and optional values
  - c) A pair of date tracking columns
  - d) Standard audit columns
2. The complete status history over time of the BASE DATA item is stored in a set of records in an associated STATUS table that includes:
  - a) Status data
  - b) Effective start and stop date and times of the status
  - c) Foreign-key link to the BASE DATA row
  - d) An open/closed indicator
  - e) An optional reason for the status
  - f) Standard audit columns

The logical and physical models contain numerous instances of this design pattern that is well suited to handling and tracking the many events to which RSA must respond.



## **2.0 REPORTING CONCEPTUAL MODEL**

At the conceptual level, the requirements of the ODS (reporting) aspect of the system will be met by physical model constructs that may include any or all of the following:

1. Views
2. Materialized views
3. Indexes
4. Table Partitioning

The physical design constructs that reporting will require are still in the design process. More details will be available in later iterations when those parts of the system are to be addressed.

## **3.0 LOGICAL MODEL**

The RSA logical model is a set of Oracle Entity-Relationship Diagrams (ERD). These diagrams were created in Oracle Designer and use a notation developed by Richard Barker for Oracle. This discussion assumes familiarity with that notational convention. The diagrams are meant to represent a fairly normalized model of the database, while the entities and relationships are used to generate a first-cut physical model. The ERD pages can be found at:

[DatabaseDocs\RsaLMDB.pdf](#)

There are currently twelve (12) diagrams. Each diagram represents some logically related portion of the RSA logical database design. Some entities appear on more than one diagram, but are in reality the same entity. The label in the box at the upper left-hand corner of each diagram indicates the logical portion of the RSA database being modeled. The models in each diagram are arranged so the general flow follows a recommended standard that allows one to read the model from right to left and from bottom to top.

### **3.1 Logical Model Reports**

The ERD models define a set of entities, attributes, subtypes, and relationships. They also include specifications of mandatory and optional attributes as well as of primary keys. Embedded in the Oracle Designer repository is additional data about the data types of each attribute and the specification of domains for attributes. Hence, the ERD models are read in conjunction with standard reports that are generated from Oracle Designer. For the logical model, there are five (5) such reports, each listed as a link to an Adobe PDF file below. After each link, there is a brief description of the content of the report. Note that the data in the reports overlaps some between the reports; that is, the reports present different views of some of the same data. Which report one needs to examine will depend in part on what information is being sought.

### 3.1.1 [DatabaseDocs\ckentdef.pdf](#)

This report contains details for entities in the RSA database model in the Designer repository. The details in the report include the synonyms, description, attributes, relationships and unique identifiers for each specified entity.

### 3.1.2 [DatabaseDocs\ckattb14.pdf](#)

This report lists all the entities for the RSA database model in the Designer repository, together with details of the attributes describing each entity. The details include:

1. **Entity Name:** The full name(s) of any entities in the specified container.
2. **Attribute Name:** The name(s) of any attributes describing the listed entity.
3. **Sequence:** The sequence number to be used when displaying the values of this attribute. This information may be used by the Generator Products (for example, for determining the order of columns in the first-cut table design produced by the Database Transformer).
4. **Optional:** Indicates whether the attribute is optional.
5. **Format:** The format of the attribute.
6. **Length:** The length of the attribute.
7. **Dec Pl:** The number of decimal places allowed for this attribute.
8. **Attribute Description:** A description about the attribute.
9. **Attribute Notes:** Any notes or comments about the attribute.

### 3.1.3 [DatabaseDocs\ckattdef.pdf](#)

This report contains comprehensive details of all the attributes describing entities in the RSA database model in the Designer repository. This report includes the following types of information if they are entered in the repository:

1. **Attribute:** The name of the attribute.
2. **Of Entity:** The name of the entity that is described by the attribute.
3. **In Domain:** The name of the domain to which the attribute belongs.
4. **Sequence in Entity:** The sequence position in which this attribute should normally be displayed within the entity.
5. **Comments:** A brief description of the attribute.
6. **Optional:** Indicates whether the attribute is optional.
7. **On Condition:** The conditions under which the attribute can be used. This field applies only to optional attributes.
8. **Format:** The format of the attribute.
9. **Length:** The maximum length of the attribute.
10. **Unit of Measure:** The unit of measure for the attribute (e.g., meters, minutes, hours).
11. **Responsibility of:** The identification of the user who owns this attribute.
12. **Authority:** The level of user authority needed for updating this attribute.
13. **Default Value:** The default value of this attribute. A default value is used only for mandatory attributes.

14. **Null Value:** Indicates that the attribute is not present for a particular entity.
15. **Derivation:** The algorithm or expression that describes how the value for a derived attribute is established.
16. **Description:** A description of the attribute.
17. **Attribute Values**
  - **Value:** A valid value for this attribute. If a range of values is defined, this value defines the low value.
  - **High Value:** A high value for the attribute, if a range of values is defined.
  - **Abbreviation:** An abbreviation, or short code, for the value or meaning.
  - **Meaning:** The full meaning for the abbreviation that describes the value or range specified.
18. **Domain Values:**
  - **Value:** A valid value for the domain to which this attribute belongs. If a range of values is defined this value defines the low value.
  - **High Value:** A high value for the domain, if a range of values is defined.
  - **Abbreviation:** An abbreviation, or short code, for the domain value or meaning.
  - **Meaning:** The full meaning for the abbreviation, or code value, which describes the domain value or range specified.
19. **Attribute Validation Rules:** The attribute validation rules used when testing and implementing the system.
20. **Domain Validation Rules:** The domain validation rules used when testing and implementing the system.
21. **Sequence in Sort Key:** The normal sequence of this attribute when sorting entities.
22. **Sorting Order:** Indicates whether the normal sort sequence of the attribute is ascending or descending.

#### 3.1.4 [DatabaseDocs\ckemodrf.pdf](#)

This report contains complete details defined for specified entities in the RSA database model in the Designer repository. The data in this report includes:

1. **Entity Name:** The name of the entity.
2. **Short Name:** The short name of the entity.
3. **Synonyms:** Shows any synonyms defined for this entity.
4. **Summary**
  - Whether the entity is a subtype of another entity
  - Any subtypes that this entity has
5. **Attributes:** Shows a summary of the attributes defined for this entity (also shows the attributes of the sub-type of this entity):
  - \* indicates a mandatory attribute
  - o indicates an optional attribute
6. **Relationships:** This section shows a summary of the relationships defined for this entity.

7. **Details:** The Details section is included only if the “Include Details?” parameter is set to Yes. The section shows:
  - Details of the attributes defined for the entity: optionality, domain, format and default value
  - Details of the attributes defined for subtypes, if any, of this entity
    - Business functions that use this entity
    - Business units that use this entity
  - Volume information for the entity
  - Notes and other text recorded for the entity
8. **Defined Attributes for Entity:** The name of the entity (or one of its sub-types, if any). The attribute details shown for this entity are:
  - **Attribute Name:** The attribute name
  - **Optional:** Whether a value is optional for this attribute
  - **Domain:** The domain, if any, to which the attribute belongs
  - **Format:** The attribute format, that is, the datatype
  - **Default Value:** The default value for the attribute
9. **Used by Business Functions:** For the specified entity (and each of its sub-types, if any), this section shows the short definition and the label of the business functions that use this entity.
10. **Used by Business Units:** For the specified entity (and each of its sub-types, if any), this section shows the name and short name of the business units that use this entity.
11. **Volumetric Information of Entity:** Shows volume information about the specified entity, namely the:
  - **Initial:** The initial number of occurrences
  - **Average:** The Average number of occurrences expected
  - **Maximum:** The maximum number of occurrences expected
  - **Annual Growth (%):** The expected annual growth rate
  - **Note and Other Text:** Any notes about the entity.

### 3.1.5 [DatabaseDocs\cddom2.pdf](#)

This report lists all attributes within domains for the RSA database model in the Designer repository. The data in this report includes:

1. **Domain:** The name of the domain in which the attributes within the specified container occur.
2. **Entity Name:** The name of the entity that the attribute describes.
3. **Attribute Name:** The name of the attribute within the listed domain for the specified container.
4. **Format:** The format of the listed attribute.
5. **Length:** The length of the listed attribute.
6. **Precision:** The number of decimal places allowed for this attribute.

## 4.0 PHYSICAL MODEL

The RSA physical model is a set of Oracle table/view diagrams. These diagrams were created in Oracle Designer and use a variation on the notation developed by Richard Barker for Oracle for ERD modeling. This discussion assumes familiarity with that notational convention. The diagrams represent the physical model of the database. The physical model pages can be found at:

[DatabaseDocs\RsaPMDb.pdf](#)

There are currently nineteen (19) diagrams. The first twelve (12) diagrams are in one-to-one correspondence to the twelve diagrams in the logical model. The remaining seven (7) diagrams illustrate the current set of transactional views in the physical model.

Each diagram in the first set of table diagrams represents some logically related portion of the RSA physical database design. Some tables appear on more than one diagram, but are in reality the same table. The label in the box at the upper left-hand corner of each diagram indicates the portion of the RSA database being modeled. The models in each diagram are arranged so the general flow follows a recommended standard that allows one to read the model from right to left and from bottom to top.

Note that the physical model closely mirrors the logical model since the OLTP system is a mostly normalized database design. The following represent the three major exceptions from that normalized model:

1. **PARTIES:** This is a highly denormalized (i.e., flat) representation of the PARTY entity and its subtypes. PARTIES represent the basic things about which RSA stores configuration data (i.e., patients, groups, resources, and administrative entities). Having this data in a denormalized table allows simpler foreign-key specification and management throughout the database. The transactional views on top of the PARTIES table will shield the developers from the details of the denormalization.
2. **APPOINTMENTS:** Also a denormalized representation of the APPOINTMENT entity and its two subtypes. The APPOINTMENTS table is used to handle appointment requests and scheduled appointments. Having a denormalized structure reduced the overall number of joins that were needed in the database to handle appointment data of either kind. Again, views on top of the APPOINTMENTS table shield the developers from the details of this denormalization.
3. **RESOURCE\_SETS:** The logical model contains about one dozen entities in a complex normalized model that represents a RESOURCE SET and the items in it. Since these resource sets are a vital element in the RSA application, especially with respect to the search algorithm that looks for appointment opportunities, it was necessary to denormalize ten (10) resource set entities into a single table named RESOURCE\_SETS. This was a deliberate design decision made to ensure that the RSA search algorithm could meet performance goals.

The middle-tier component that manages RESOURCE\_SETS will verify that the data in the denormalized structure remains accurate.

Each diagram in the second set of transactional view diagrams is essentially a presentation of views that are needed by individual middle tier components. Each transactional view appears in one diagram in this set.

## 4.1 Physical Model Reports

The physical table/view models define a set of tables, views, columns, and foreign-key-relationships. They also include specifications of mandatory and optional columns as well as primary keys. Embedded in the Oracle Designer repository is additional data about the data types of each column and the specification of domains for columns. Hence, the table/view models are read in conjunction with standard reports that are generated from Oracle Designer. For the physical model there are four (4) such reports, each listed as a link to an Adobe PDF file below. After each link, there is a brief description of the content of the report. Note that the data in the reports overlaps some between the reports (i.e., the reports present different views of some of the same data). Which report one needs to examine will depend in part on what information is being sought.

### 4.1.1 [DatabaseDocs\cktc1.pdf](#)

This report contains comprehensive details of tables, views and materialized views in the RSA database model in the Designer repository. The information includes descriptions, volumes, column details and indexes. This report represents what is traditionally considered to be the data dictionary for the physical RSA database.

### 4.1.2 [DatabaseDocs\ckvwdef.pdf](#)

This report shows details of materialized views and views, together with their columns and their column derivations, within the RSA database model in the Designer repository. The data in this report includes:

1. **View Name:** The name of the materialized view or view.
2. **Alias:** The alias for the materialized view or view name.
3. **Type:** This will be either a 'View' or 'Materialized View' definition.
4. **Object Type:** The name of the Object Type the view is based on.
5. **Updatable ?:** For materialized views only, indicates whether INSERT, UPDATE and DELETE statements can be performed.
6. **Column Prefix:** For views only, a prefix for columns in the view. The column prefix is used by the Server Generator when generating the column names that are to be implemented on the database.
7. **Display Title:** The default title that Generator products can use to label the part of the application that is based on this materialized view or view. For example, generated forms

show block titles, generated reports show group titles and generated Visual Basic applications show prompts or captions.

8. **Comment:** A comment about the materialized view or view.
9. **Free Format Select Text?:** Indicates whether tables on which the materialized view or view is to be based are defined using a free format SELECT statement, enabling complex queries to be defined.
10. **Select Text:** The SELECT statement that specifies the base tables and columns from which the materialized view or view is derived.
11. **Optimizer Hint Clause Text:** A hint clause used by the optimizer where the materialized view or view is defined declaratively. The optimizer uses the hint to choose an execution statement plan for the SQL statement.
12. **Where/Validation Condition:** A restriction or join condition for the materialized view or view, specified in a WHERE clause. This clause defines a condition on the SELECT statement that defines the materialized view or view. This condition must be met in order for rows resulting from the query to be returned.
13. **Base Table Name:** The name of the table from which the materialized view or view is derived.
14. **Alias:** An alias for the materialized view or view. This alias can be used to qualify columns in the SELECT and WHERE clause.
15. **Used in Sub-Query?:** Indicates that the relation definition referred to by the materialized view or view is used within a sub-query of the WHERE clause.
16. **View Column:** The name of a column within the materialized view or view.
  - **Datatype:** The datatype defined for the column.
  - **Avg:** The estimated average length of data that can be stored in this column.
  - **Max:** The maximum length of data that can be stored in this column.
  - **Dp:** The number of decimal places allowed for the column.
  - **Base Column:** The name of the column in the base table on which the materialized view column or view column is based.
  - **Base Table:** The name of the table to which the base column belongs.
  - **Expression:** The derivation expression. This is the SELECT statement that specifies the derivation of the column's value, where it is a product of other columns in the same row.

#### 4.1.3 [DatabaseDocs\ckpkgdfrn.pdf](#)

This report lists details of the PL/SQL module definitions in the Oracle Designer repository for the RSA database. This report includes a large number of fields. The most significant fields for this system are:

1. **Short Name:** The short name of the specified PL/SQL module. This property is used as the basis for naming generated files and objects.
2. **Name:** The full name of the module.
3. **Type:** The type of PL/SQL definition, e.g., Cursor, Function, Package, Procedure or Trg-Logic.
4. **Purpose:** The purpose, or a short description, of the module.



5. **Status:** Indicates whether the design of this definition is complete. Possible values are Not Started, In Progress, or Completed.
6. **PL/SQL Block:** Defines in full the public declarations of subprograms and cursors listed in the package specification.
7. **Package Specification:** For modules of type Package, declares public functions, procedures, cursors, variables, constants and user-defined datatypes that can be accessed by subprograms and triggers outside the package.

#### 4.1.4 [DatabaseDocs\cddom1.pdf](#)

This report lists the tables and columns that exist in each domain for the RSA database model in the Designer repository. The data in this report includes:

1. **Domain:** The name of the domain.
2. **Table Name:** The name of the table within the listed domain.
3. **Column Name:** The name of the column within the listed domain.
4. **Data Type:** The datatype of the listed column.
5. **Max Length:** The maximum length of the datatype
6. **Dec Places:** The number of decimal places allowed for this column.

## 5.0 GLOBAL

There is one global report that is produced from the Oracle Designer repository. This is the Domain Definition report. This report is considered global since it is used in understanding reports in the logical model as well as in the physical model. The design modeling methodology being followed in the development of the RSA database requires that every attribute in every entity and every column in every table either be defined against an explicit domain or have specified a set of allowable values. Except for the very few columns in the RSA physical database that are generated by the Database Transformer when it processes an entity with subtypes, every column has an explicit domain.

#### [DatabaseDocs\ckdomdef.pdf](#)

The Domain Definition reports includes the following data about the domains defined in the RSA repository for the RSA database:

1. **Name:** The name of the specified domain.
2. **Supertype:** The name of the supertype of the domain (if a hierarchy exists).
3. **Description:** A brief description of the domain, listing the element which the domain is created from (e.g., column, program data, attribute).
4. **Comment:** Comments about the domain.
5. **Attribute Format:** The format of any attributes in this domain.
6. **Length:** The length of any attributes in this domain.
7. **U.O.M.:** The unit of measurement for the attributes in this domain.



8. **Column Datatype:** The datatype of any columns, program data, attributes in this domain.
9. **Max Length:** The maximum length of any columns, program data, attributes in this domain.
10. **Owned by Container:** The name of the container that owns this domain.
11. **Authority:** The level of authority needed to update this domain. The default is ANY.
12. **Default Value:** The default value for the attributes or columns in this domain.
13. **Null Value:** The value that indicates that the attribute or column in the domain is not present for an occurrence of the entity or table.
14. **Derivation:** The derivation of attributes within the domain.
15. **Domain Values**
  - **Value:** A valid value for this domain. This field defines the low value if a range of values is defined.
  - **High Value:** A high value for the domain, if a range of values is defined.
  - **Abbreviation:** An abbreviation or code, commonly used for a value or range of values.
  - **Meaning:** The full meaning for the abbreviation.
16. **Domain Validation Rules:** Validation rules for use when testing and implementing the system.