Interfacing With EASYGroup™ Guide

Published November 18, 2020
The format of this document is 8.5 x 11"

© 2020 Optum.

All rights reserved.

This document is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this document, or any portions of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent under the law.

CPT® codes, descriptions, and other CPT® materials obtain a copyright of 2019 American Medical Association (AMA). All Rights Reserved. No fee schedules, basic units, relative values, or related listings are included in CPT®. The AMA does not directly or indirectly practice medicine or dispense medical services. The AMA assumes no liability for data contained or not contained herein. CPT® is a registered trademark of the American Medical Association.

Current Dental Terminology, © 2019 American Dental Association. All rights reserved.

Applicable FARS/DFARS Restrictions Apply to Government Use.

3M™ is a trademark of the 3M™ Company. The 3M™ Grouper Plus System (3M™ GPS) and the 3M™ Grouper Plus Content Services (GPCS) along with the 3M™ Enhanced Ambulatory Patient Grouping System (3M™ EAPGS) and its logic are proprietary to the 3M™ Company and are subject to the terms and conditions of the software licensing agreement between 3M™ and Optum.

© 2020, American Hospital Association (AHA), Chicago, Illinois. Reproduced with permission. No portion of this publication may be reproduced, sorted in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior express, written consent of AHA.

Optum and the Optum logo are registered trademarks of Optum. All other brand or product names or trademarks are registered marks of their respective owners. Because we are continuously improving our products and services, Optum reserves the right to change specifications without prior notice. Optum is an equal opportunity employer.

EDC Analyzer™ - U.S. Patent No. 10,417,382

Optum
11000 Optum Circle
Eden Prairie, MN 55344

client.services@optum.com
# Table of Contents

Chapter 1: Overview ................................................................. 2
  Introduction to This Guide ............................................. 3
  Organization of This Guide ........................................... 3
  Document Conventions .................................................. 4
  About Optum ................................................................. 4
  Contact Us ................................................................. 5

Chapter 2: Introduction ........................................................... 7
  EASYGroup™ Architecture ............................................. 8
  EASYGroup™ Interface Options ..................................... 9
  Hardware/Software Requirements .................................. 11

Chapter 3: Optimizer Overview .................................................. 16
  Overview of EASYGroup™ Optimizer ............................. 17
  Using Rate Files With the Optimizer ............................... 22
  Optimizer Process Summary ......................................... 26

Chapter 4: Optimizer Integration and Configuration .......................... 28
  Configuring the C Platform Optimizer ............................ 29
  Configuring the COBOL Platform Optimizer ................... 38

Chapter 5: Server Overview ..................................................... 42
  Server Overview .......................................................... 43

Chapter 6: Server Integration and Configuration ................................ 45
  Server Integrations ....................................................... 46
  Server Configuration ................................................... 57

Chapter 7: EASYGroup™ Web Service Overview .............................. 66
  Introduction to the EASYGroup™ Web Service ................ 67
  Data Formatting Conventions ....................................... 80
  Enabling Enhanced Web Security ................................ 84

Chapter 8: EASYGroup™ Web Service Tester Overview ....................... 91
  Tester Main Functions/ScreenGs ................................. 92
  Tester Hot Keys .......................................................... 99

Chapter 9: Optum Exchange PPS (OEPPS) Overview ........................ 100
  Overview of the OEPPS Application .............................. 101
  OEPPS Dependencies .................................................. 104

Chapter 10: OEPPS Configuration and Utilization .............................. 105
  Launching the OEPPS Application ................................. 106
  OEPPS User Interface (UI) ............................................. 107
  File Watch Mover (FWM) Utility ................................. 119

Chapter 11: OEPPS Mapping ...................................................... 120
  Mapping ................................................................. 121

Index ................................................................. 136
1 Overview

This chapter provides an overview of this manual and how to contact Optum. It contains the following sections:

- Introduction to This Guide
- Intended Audience
- Organization of This Guide
- Document Conventions
- About Optum
- Contact Us
  - Corporate Address
  - Need Assistance? Contact Optum Client Services
  - Optum Portals
  - Found an Error in This Guide?
1.1 Introduction to This Guide

The Interfacing With EasyGroup™ Guide contains all of the information needed to interface with the following products: the EasyGroup™ Optimizer, the EasyGroup™ Server, the EasyGroup™ Web Service, the ECM Pro™ Web Services, and the Optum Exchange PPS (OEPPS). Consider this guide your textbook, a ready reference source should you forget a procedure or encounter a problem.

1.1.1 Intended Audience

This guide is directed to:
- Information Technology (IT) Personnel
- Network Administrators
- Software Development Teams


1.2 Organization of This Guide

Table 1-1: Guide Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1, Overview</td>
<td>Overview of Optum and of this guide.</td>
</tr>
<tr>
<td>Chapter 2, Introduction</td>
<td>Introduction to the EASYGroup™ interface options and architecture.</td>
</tr>
<tr>
<td>Chapter 3, Optimizer Overview</td>
<td>Overview of the EASYGroup™ Optimizer features and functionality.</td>
</tr>
<tr>
<td>Chapter 4, Optimizer Integration and Configuration</td>
<td>Overview of how to integrate and configure your system for the EASYGroup™ Optimizer.</td>
</tr>
<tr>
<td>Chapter 5, Server Overview</td>
<td>Overview of the EASYGroup™ Server features and functionality.</td>
</tr>
<tr>
<td>Chapter 6, Server Integration and Configuration</td>
<td>Overview of how to integrate and configure your system for the EASYGroup™ Server.</td>
</tr>
<tr>
<td>Chapter 7, EASYGroup™ Web Service Overview</td>
<td>Overview of the EASYGroup™ Web Service methods and functionality.</td>
</tr>
<tr>
<td>Chapter 8, EASYGroup™ Web Service Tester Overview</td>
<td>Introduction to the EASYGroup™ Web Service Tester.</td>
</tr>
<tr>
<td>Chapter 9, Optum Exchange PPS (OEPPS) Overview</td>
<td>Overview of the Optum Exchange PPS (OEPPS) features and functionality.</td>
</tr>
</tbody>
</table>
1.3 Document Conventions

This guide uses the following conventions:

- Any screen fields, buttons, tabs, or other controls that you can manipulate are printed in bold type. Keys that you press on the keyboard are also printed in bold type. For example:
  - Press the Exit button.
  - Press the Enter key.
- Keyboard keys that you must press simultaneously are printed in bold type and separated by a plus (+) sign. For example:
  - Press Ctrl + C.
- Links embedded in the text that you can select to jump to another section are in orange text. For example:
  - For further information refer to Chapter 2.
- Field names and filenames are italicized. For example:
  - pricer_rtn_code
  - EASYGroup.exe
- Field descriptions are printed in bold type:
  - NICU Accreditation Indicator

1.4 About Optum

At Optum, we focus on improving connections across all parts of health care to deliver better outcomes. We help make healthier happen in three key ways: modernizing infrastructure, advancing care, and empowering consumers. Optum delivers secure data-driven solutions for those who provide care and those who pay for care.
1.5 Contact Us

1.5.1 Corporate Address
Optum
11000 Optum Circle
Eden Prairie, MN. 55344
T 1 + (888) 445-8745
www.optum.com
info@optum.com

1.5.2 Need Assistance? Contact Optum Client Services
We welcome you as a valued client. Please contact Optum Client Services using one of the methods detailed below.

When opening a ticket with Optum Client Services you will be issued a ticket number. These ticket numbers correlate to individual issues. If you are experiencing multiple issues, it is recommended that you obtain individual ticket numbers.

When calling Optum Client Services regarding a previously opened ticket, have your ticket number available. If you misplaced or did not receive a ticket number, please ask the technician to provide it to you.

Optum Client Services Phone: 800-999-DRGS (3747)
1. Calls are answered in the order that they are received. If there is a high call volume, calls are held in a queue until a technician becomes available.
2. Calls classified as an industry expert category (i.e., case and reimbursement, logic encoder, etc.) will be escalated to Optum experts.
3. Technicians are available 24/7.

After selecting **Option 6 for Technical Support** you will hear the following choices:

Table 1-2: Technical Support Options

<table>
<thead>
<tr>
<th>Option #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1</td>
<td>For password reset, login issues, or expiration error.</td>
</tr>
<tr>
<td>Option 2</td>
<td>For all other issues.</td>
</tr>
</tbody>
</table>

Email: **Optum Client.Services**

1. Include name and number and detailed description of product issue.
2. Response time to email is generally within a few business hours.
3. Service technician has ability to do prior research before calling back.

1.5.3 Optum Portals
For access to announcements, user documentation, notices, release schedules, and much more please visit the Optum Client Portal and/or the Regulatory Portal.

1.5.4 Found an Error in This Guide?
Please feel free to contact our EASYGroup™ Documentation team with any errors you may have found within this user’s guide:
EASYGroup_Documentation
We welcome feedback from our clients.
2 Introduction

This chapter provides an overview for all the different products that can be used to interface with EASYGroup™.

Note
For hardware/software requirements for the EASYGroup™ Payment System Interface (PSI) please refer to the EASYGroup™ PSI Installation Guide.

It contains the following sections:

- EASYGroup™ Architecture
- EASYGroup™ Interface Options
  - Optimizer
  - Server
  - EASYGroup™ Web Service
  - ECM Pro™ Web Services
  - Optum Exchange PPS (OEPPS)
  - EASYGroup™ Component Overview
  - Rate Manager Overview
- Hardware/Software Requirements
  - Optimizer
  - Server
  - EASYGroup™ Web Service
  - ECM Pro™ Web Services
  - Optum Exchange PPS (OEPPS)
2.1 EASYGroup™ Architecture

Below is a diagram of the EASYGroup™ architecture.

Figure 2-1. EASYGroup™ Architecture

As you can see above, the Optimizer is the brain of EASYGroup™. It enables communication among all of the EASYGroup™ components including Editors, Groupers, Mappers, and Pricers. It does this by breaking down each request and passing it to a set of middle-tier control programs dedicated to a specific function (map control, edit control, group control, price control, etc.). These middle-tier control programs are in turn responsible for invoking the individual EASYGroup™ components to perform the appropriate business function (mapping, editing, grouping, pricing).

The Optimizer provides a single point of communication between the EASYGroup™ components and an outside application. Clients can choose to either directly interface with the Optimizer or to interface with the EASYGroup™ Server, OEPPS, ECM Pro™ Web Service or EASYGroup™ Web Service, all of which directly communicate with the Optimizer. All of these various interface options are described in detail below.
2.2 EASYGroup™ Interface Options

2.2.1 Optimizer

The EASYGroup™ Optimizer uses the rate data files (from Rate Manager or the Optum-supplied rate files (i.e., NMPRFs or State Rate Files)) and information on the claim to determine which EASYGroup™ components are needed to process the claim. The user-defined rules for grouping, pricing, mapping, and editing are specific to a facility (i.e., hospital or provider), payer (i.e., line of business or provider contract/insurance company), and period of time. The Optimizer, along with various control programs, manages the editing, mapping, grouping, and pricing of patient data. This component determines which Editors, Groupers, and Pricers are appropriate for the patient records being processed, determines if mapping is required, processes records as indicated, and returns appropriate results. The EASYGroup™ Optimizer is a single point of communication for all EASYGroup™ components. Input and output requirements are listed in the associated EASYGroup™ Input & Output Parameter Blocks User’s Guides and Input & Output Parameter Blocks Workbook (which are posted on the Optum Client Portal and/or the Regulatory Portal).

2.2.2.1 COBOL Platform

The set of programs that make up the Optimizer are written in standard COBOL, using a compiler that conforms to INCITS/ISO/IEC 1989-2002 COBOL standard, “High” level specifications. The COBOL file types used are SEQUENTIAL and INDEXED.

2.2.3 Server

The Server runs on a node of the TCP/IP network. Clients can access the Server by connecting to a predefined port and issuing a request. For each request, the Server will perform an Optimizer function call and return the results to the client. The Server can run as an application or as a Windows® service. Input and output requirements are listed in the associated EASYGroup™ Input & Output Parameter Blocks User’s Guides and Input & Output Parameter Blocks Workbook (which are posted on the Optum Client Portal and/or the Regulatory Portal).

2.2.4 EASYGroup™ Web Service

The EASYGroup™ Web Service is a Representational State Transfer (REST)-based web service. The EASYGroup™ Web Service is comprised of five methods written in ASP.Net Model View Controller (MVC) and distributed to run on the Microsoft® .NET platform and on Internet Information Services (IIS). Each method will accept the input data and reformat for submission to the EASYGroup™ Optimizer. Return data from EASYGroup™ is parsed and relevant fields are returned to the requesting operation. Input and output requirements for each operation are listed in the associated EASYGroup™ Web Service Input & Output Spreadsheet and the Input & Output Parameter
Blocks Workbook (which are posted on the Optum Client Portal and/or the Regulatory Portal).

2.2.5 ECM Pro™ Web Services

The ECM Pro™ Web Services are SOAP (Simple Object Access Protocol)-based web services. The ECM Pro™ Web Service is written in Microsoft® C# and distributed to run on the Microsoft®.NET platform and on IIS.

There are five different ECM Pro™ Web Services based on patient setting, as well as one for labels:

- Inpatient
- Outpatient
- Physician
- Skilled Nursing Facility (SNF)
- Inpatient Rehabilitation Facility (IRF)
- Label

Each web service will accept the input data and reformat for submission to the EASYGroup™ Optimizer. Return data from EASYGroup™ is parsed and relevant fields are returned to the requesting operation. Input and output requirements for each web service are listed in the ECM Pro™ Web Service User's Guides (which are posted on the Optum Client Portal and/or the Regulatory Portal).

2.2.6 Optum Exchange PPS (OEPPS)

OEPPS is a Windows® service written in Microsoft® C# and distributed to run on the Microsoft® .NET 4.0 Platform. The available operations communicate with the EASYGroup™ Optimizer. Input and output requirements are listed in the associated EASYGroup™ Input & Output Parameter Blocks User's Guides and Input & Output Parameter Blocks Workbook (which are posted on the Optum Client Portal and/or the Regulatory Portal).

2.2.7 EASYGroup™ Component Overview

EASYGroup™ is a portable software library that supports the implementation of Prospective Payment Systems (PPS). The EASYGroup™ library contains Editors, Groupers, Mappers, Pricers, the Optimizer and a number of supporting data files. These EASYGroup™ modules are generally packaged into payment systems which support the complete editing, grouping, and pricing needs of a specific payer (e.g., Medicare, Medicaid, TRICARE, or commercial).

- Editors:

Editors ensure that valid ICD-9, ICD-10, and HCPCS/CPT® codes are billed for the claims’ service dates. Editors perform basic claims data validation edits, as well as, the Medicare Code Edits (MCEs), Outpatient Code Edits
(OCEs), Correct Coding Initiative (CCI) edits, Medically Unlikely Edits (MUEs), Local/National Coverage Determination (LCD/NCD) edits, and Optum proprietary clinical and reimbursement edits.

- **Groupers:**
  Groupers provide accurate classification and case-mix assignment. Groupers provide support for ICD-10 and ICD-9 DRGs, APCs, APGs, CMGs, HHRGs, and RUGs.

- **Mappers:**
  Mappers verify that the codes supplied are consistent with the Grouper coding requirements. If codes are not consistent with requirements, the Mappers translate them into codes that are recognized by the Grouper.

- **Pricers:**
  Pricers support reimbursement calculations, including calculations for inliers, outliers, transfers, and other payer-specific rules.

- **Supporting Data Files:**
  Fee schedules, case-mix weights, codes and titles/descriptions, and others.

- **National Medicare Provider Rate Files (NMPRF) and Medicaid State Rate Files (SRF):**
  Rate files are a comprehensive source of Medicare and Medicaid reimbursement information enabling the calculation of accurate reimbursement for all supported Medicare and Medicaid providers.

### 2.2.8 Rate Manager Overview

The Optimizer references external data files that contain processing rules for various providers, payers, and effective dates. Editing, grouping, and pricing rules are stored in these data files which are created and maintained using Rate Manager. Procedures for installing and using the Rate Manager are detailed in the Rate Manager Installation Guide and the Rate Manager User’s Guide.

### 2.3 Hardware/Software Requirements

To use the following EASYGroup™ products, your operating environment must meet or exceed the following hardware and software requirements. **Please refer to the EASYGroup™ Installation Guide for any additional hardware/software requirements specific to the components that you license.**
2.3.1 Optimizer

To successfully install and integrate the EASYGroup™ Optimizer, your operating system must meet the following minimum requirements:

**Note**
The Optimizer is only supported on 64-bit environments as of the V2002.01 release.

2.3.2.1 Hardware

- Processor: 1GHz Pentium or equivalent (Minimum); 2.4GHz or faster (Recommended)
- RAM: 1GB (Minimum); 2GB or higher (Recommended)
- Hard Disk: 5GB available space (Minimum); 10GB available space (Recommended)
- Network Adapter: 100BASE-T (Minimum); 1000BASE-T (Recommended)

2.3.3.2 Software

- Microsoft® Windows® Server 2019
- Microsoft® Windows® Server 2012 (including R2) (SP2)
- Microsoft® Windows® Server 2016
- Microsoft® Windows® 7 (SP1)
- Microsoft® Windows® 10

2.3.4 Server

To successfully install and integrate the EASYGroup™ Server, your operating system must meet the following minimum requirements:

**Note**
The Server is only supported on 64-bit environments as of the V2005.01 release.

2.3.5.1 Hardware

- Processor: 1GHz Pentium or equivalent (Minimum); 2.4GHz or faster (Recommended)
- RAM: 1GB (Minimum); 2GB or higher (Recommended)
- Hard Disk: 5GB available space (Minimum); 10GB available space (Recommended)
- Network Adapter: 100BASE-T (Minimum); 1000BASE-T (Recommended)

2.3.6.2 Software

- Microsoft® Windows® Server 2012 (including R2) (SP2)
- Microsoft® Windows® Server 2016
• Microsoft® Windows® 7 Service Pack 1 (SP1)
• Microsoft® Windows® 10
• Microsoft®.NET Framework 3.5 or higher

2.3.7 EASYGroup™ Web Service
To successfully install and integrate the EASYGroup™ Web Service, your operating system must meet the following minimum requirements:

Note
It is advised to follow normal industry-standard best practices to configure the software and Server to prevent unauthorized use of the software.

The EASYGroup™ Web Service is only supported in 64-bit environments.

2.3.8.1 Hardware
• Processor: 2.4GHz Dual Core (Minimum), 3.0GHz Quad Core (Recommended):
  - Intel® Core™ i5 or higher
  - AMD Athlon™ X64 or higher
• RAM: 4GB RAM (Minimum), 8GB RAM or higher (Recommended)
• Hard Disk: 10GB free disk space (Minimum), 20GB free disk space (Recommended)
• Network Adapter: 100 M-BITS/S (Minimum)

2.3.9.2 Software
• Microsoft® Windows® Server 2012 (including R2) Service Pack 2 (SP2)
• Microsoft® Windows® Server 2016
• Microsoft® Windows® 10
• Microsoft® Windows® 7 Service Pack 1 (SP1)
• Microsoft®.NET Framework V4.6.1 or higher
• Internet Information Services (IIS) V7.5 or higher
• Google Chrome™ browser
• Microsoft® Internet Explorer™
• Supported Browsers for EASYGroup™ Web Service Tester:
  - Mozilla® Firefox® browser V59 or higher
  - Google Chrome™ browser V60 or higher
  - Microsoft® Internet Explorer™ V11
2.3.10 ECM Pro™ Web Services

To successfully install and integrate the ECM Pro™ Web Services, your operating system must meet the following minimum requirements:

2.3.11.1 Hardware

- Processor: 1GHz Pentium or equivalent (Minimum); 2.4GHz Dual Pentium (Recommended)
- RAM: 1GB (Minimum); 2GB or higher (Recommended)
- Hard Disk: 5 GB available space (Minimum); 10 GB available space (Recommended)
- Network Adapter: 100BASE-T (Minimum); 1000BASE-T (Recommended)

2.3.12.2 Software

- Microsoft® Windows® Server 2012 (including R2) Service Pack 2 (SP2), 64-bit
- Microsoft® Windows® Server 2016
- Microsoft® Windows® 7 Service Pack 1 (SP1) (64-bit)
- Microsoft® Windows® 10
- Microsoft® .NET Framework 3.5 (Minimum); Microsoft® .NET Framework 4.5 Service Pack 1 (Recommended)
- Internet Information Services (IIS) 7.0, 7.5, 8.0, and 8.5.

2.3.13 Optum Exchange PPS (OEPPS)

To successfully install and integrate the OEPPS, your operating system must meet the following minimum requirements:

Note

OEPPS is only supported on 64-bit environments.

2.3.14.1 Hardware (Server)

- Processor: 2.0GHz Dual Core (Minimum), 3.0GHz Quad Core (Recommended)
- RAM: 4GB RAM or higher plus 500MB per process*

Note

*6GB of RAM dedicated and always available (the 4GB + 500MB per process) outside of other programs running.

The number of processes is set using the MaxNumberOfProcesses key in the configuration screen.

- Hard Disk: 10GB free disk space (Minimum), SCSI HDD w/10+ GB free disk space (Recommended)
Note
Each file processed requires enough space to store the entire input and output file.

- Network Adapter: 100 M-BITS/S (Minimum), DUAL 100 M-BITS/S Adapter (for load balancing) (Recommended)

2.3.15.2 Software
- Microsoft® Windows® Server 2012 (including R2) Service Pack 2 (SP2)
- Microsoft® Windows® Server 2016
- Microsoft® Windows® 7 Service Pack 1 (SP1)
- Microsoft® Windows® 10
- Microsoft®.NET Framework V4.6.1 or higher
- Microsoft® Visual C++ Redistributable V2015 or higher
3 Optimizer Overview

This chapter provides an introduction to the EASYGroup™ Optimizer. It contains the following sections:

- Overview of EASYGroup™ Optimizer
  - Optimizer Programs and Data Files
  - Optimizer Architecture
- Using Rate Files With the Optimizer
  - Data Handling
  - Providers
  - Payers
  - Effective Date
  - Price/Group Date
  - Grouper Type
  - Grouper Version
  - Analyzer Type
  - Analyzer Version
  - Pricer/Payer Type
  - Pricing Variables and Rate Data
  - Editor Request Flags
- Optimizer Process Summary
  - Read Patient Data
  - Select Appropriate Date
  - Retrieve Rate Calculator Data
  - Identify Requirements for Analyzer, Editor, Grouper, Mapper, and Pricer
  - Invoke Control Programs
  - Perform Appropriate Functions
3.1 Overview of EASYGroup™ Optimizer

The Optimizer provides a single interface between a claims processing application or host system and the various EASYGroup™ components that:

- Examine claims for coding errors or inconsistencies (Analyzers)
- Edit patient data (Editors)
- Map ICD-9-CM/ICD-10-CM/PCS diagnosis and procedure codes from different time periods (Mappers)
- Assign case-mix measures such as DRGs and APCs (Groupers)
- Calculate expected reimbursement (Pricers)

The Optimizer has the intelligence to triage among available EASYGroup™ components and provide customized analyzing, editing, mapping, grouping, and pricing solutions based on user-defined rules. The EASYGroup™ components and associated data files are licensed individually and are provided on separate distributions.

The user-defined rules for analyzing, grouping, pricing, mapping, and editing are specific to a facility (i.e., hospital or provider), payer (i.e., line of business or provider contract) and period of time. These rules are stored in data files which also contain data to drive the claims processing functions. The Optimizer distribution relies on Rate Manager to set-up and maintain these files.

When calling the Optimizer, the user can specify which functions should be performed for the patient record being processed. Choices include options to examine claims (Analyze), edit patient data (Edit); to map inpatient codes across time periods (Map); to assign various case-mix measures (Group); and to calculate expected reimbursement (Price).

3.1.1 Optimizer Programs and Data Files

The Optimizer works with an integrated collection of EASYGroup™ components, consisting of programs and data files, whose functions are summarized below:

3.1.1.1 Optimizer Program

The Optimizer program, along with various control programs, manage the analyzing, editing, mapping, grouping, and pricing of patient data. This program determines which Analyzers, Editors, Groupers, and Pricers are appropriate for the patient records being processed, determines if mapping is required, processes records as indicated, and returns appropriate results.

3.1.1.2 Optimizer Test Programs

Various test programs and drivers are included with the Optimizer distribution to verify the Optimizer and other products are compiled and installed correctly.
3.1.1.3 Sample Optimizer Interface Program
A sample interface program is included with the Optimizer distribution to assist with the development of an interface to the Optimizer. It can be used as a reference when writing a new interface or be modified and directly incorporated into a user application.

3.1.1.4 Utility Programs (COBOL only)
Utility programs provide functions for the Analyzer, Grouper, Pricer, and Editor table building and file maintenance.

3.1.1.5 Logging Program and Data Files
(logcntl, LOGCNTL) The Logging Control Program contains logging functionality for all EASYGroup™ components to assist in troubleshooting any undesired claims processing results. This program allows logging functionality to be modified without requiring an update to the Optimizer program and also allows for logging to be completed from any EASYGroup™ shared control program. Please refer to the EASYGroup™ Technical Reference Guide for further information on EASYGroup™ logging.

3.1.1.6 Modeling Program and Data Files
(mdlcntl, MDLCNTL) The Optimizer will call this program when a modeling function is requested by the user.

One function of the Model Control Program is to model how the choice of a principal diagnosis affects DRG assignment and anticipated reimbursement. This is accomplished by calculating the DRG assignments and reimbursements that result when each diagnosis on the patient record is coded as the principal diagnosis. This modeling function is available for inpatient processing only.

To model a record, each secondary diagnosis is exchanged with the principal diagnosis. After each exchange, the Grouper is invoked to assign the DRG that results when the “exchanged” secondary diagnosis is listed as principal. Following DRG assignment, the Pricer is invoked to calculate the applicable reimbursement for that DRG and other patient characteristics (transfer, outlier, etc.). In short, the Model Control Program can be thought of as a special purpose program which invokes the Grouper and Pricer multiple times for each patient record, according to a pre-specified set of “diagnosis exchanging” rules. Results of these multiple groupings and pricings are returned to the user for display and/or analysis.

The Optimizer does not return all possible Grouper and Pricer return fields for each “exchanged” secondary diagnosis. Rather it focuses on returning those fields which are key to DRG assignment and reimbursement calculation. However, it does preserve and pass back to the calling program the full set of Grouper and Pricer return fields, as derived using the record's original principal diagnosis.
3.1.1.7 Retrieve Payer Program and Data Files

(*rtvpyr, RTVPYR*) The Retrieve Payer control module is called regardless of the Operation Code (*opcode1, OPCODE1*) sent to the Optimizer. All calls to the Optimizer will have access to the retrieve payer functionality. If the Retrieve Payer field is left blank then the Retrieve Payer program will follow the current set of rules (i.e., executing or bypassing the payer lookup based on the Operation Code). If the user sets the Retrieve Payer field to a value of 1 then the Retrieve Payer program will perform the payer lookup regardless of the Operation Code.

3.1.1.8 File I/O

(*iocnrtl, IOCNRTL*) This module is a pointer-based API and allows greater flexibility in making enhancements while reducing the potential impact to EASYGroup™ clients.
3.1.2 Optimizer Architecture

The below diagrams illustrate the architecture and program flow of the Optimizer and various EASYGroup™ components:

Figure 3-1. Optimizer Architecture - C Platform
Figure 3-2. Optimizer Architecture - COBOL Platform
3.2 Using Rate Files With the Optimizer

For most Optimizer functions to work correctly, the user must identify the relevant providers (i.e., hospitals) and payers (i.e., provider contracts) that are associated with the patient data to be processed. For each provider and payer combination, the user must identify a set of rules which can be defined in the Rate Manager data files. These data files tell the Optimizer and the control programs:

- Which Grouper type and Grouper version should be used for case-mix assignment (e.g., DRG, APC, or APG).
- Whether ICD-9-CM or ICD-10-CM/PCS code mapping should be performed before invoking the selected Grouper (inpatient only).
- Whether editing should be performed and, if so which Editor should be used.
- Whether analyzing should be performed and, if so which Analyzer should be used.
- Whether pricing should be performed and, if so which Pricer should be used to calculate expected reimbursements.

Each set of Optimizer rules is linked to an effective date. Any time one or more of these rules change, a new set of rules must be created with the effective date of that change. Generally, multiple sets of rules will be created for any provider and payer combination. Rules become active on the specified effective date, and remain active until another set of rules with a more current effective date is created. If a user requests an AnalyzeOnly, Analyze/Group, MapOnly, or Analyze/Edit function, and the input parameters include the required input data in the ECB [ezg_cntl_block]/ECB-EZG-CNTL-BLOCK structure(s), it is not necessary to supply the data files described above.

3.2.1 Data Handling

For Optimizer functions that require data files with rules, the rules are stored in one or four data files; a Hospital Rate Calculator File, a Configuration File, sometimes a Payers File, and sometimes an Extended Hospital Rate Calculator File. These files are created and maintained using Rate Manager. Procedures for installing and using Rate Manager are detailed in the Rate Manager User’s and Installation Guides.

Rate Manager is also used to collect and maintain the provider and payer-specific variables required by the Pricers. This pricing data is also stored in the Hospital Rate Calculator File and sometimes the Extended Hospital Rate Calculator File.

In addition, the Pricers require:

- An additional file with case-mix-specific rate data such as weights, means, and trims (e.g., rate.dat/wghtrate.dat contains DRG-specific weight).
• Optional files such as fee schedule data (e.g., *fsyyyy.dat/fsr**.dat and *fst**.dat - where yyyy represents the calendar year or other fee schedule identifier) and a Zip Code File, which are used for APC/SNF pricing may also be included.

• Analyzers, Editors, Groupers, Mappers, and Pricers often require additional data files. These data files are defined in the EASYGroup™ Technical Reference Guide.

3.2.2 Providers
In the user’s patient data files, each patient episode of care is associated with a particular facility or provider. A Unique Provider Identification Number (UPIN) or National Provider Identifier (NPI) typically identifies these providers. These codes must be defined to the Optimizer. UPINs can be alphanumeric and up to 16 characters in length. NPIs are numeric only and are 10-digits in length. NPIs have replaced the UPIN as the required identifier for Medicare services, and are used by payers. The transition to the NPI was mandated as part of the Administrative Simplifications portion of HIPAA, and CMS began issuing NPIs in October 2006. The Optimizer can process patient data from a single provider only, or it can process data from many different providers.

3.2.3 Payers
For each provider, there will be one or more organizations that pay for the patient care provided by that provider. For each provider, the Optimizer user must identify all the payers for which separate grouping and pricing rules will be maintained. Individual payers can be identified by up to 13 alphabetic and/or numeric characters. Often, state or federal provider identifiers are used. Typical payers will include Medicare, Medicaid, Blue Cross plans, managed care groups, and various commercial organizations.

3.2.4 Effective Date
Payers periodically change grouping and pricing rules. In addition, provider-specific rates or conversion factors, as well as DRG, APC, APG or other case-mix specific weights, can vary over time. Each change in grouping or pricing rules, or in the rate variables, must be associated with an effective date. Generally, effective dates correspond to the start date of a provider’s fiscal year, the start date of the federal fiscal year, or the start or renewal date of various payer contracts.

3.2.5 Price/Group Date
Patient grouping and reimbursement will be based upon the rules that are in effect at the time that services were delivered to the patient. Two dates are passed to the Optimizer: patient admission/from date and patient discharge/thru date. The Optimizer user must specify which of these dates should be used in selecting the appropriate set of grouping and pricing rules.
3.2.6 Grouper Type
For each provider, payer, and effective date combination, the user must specify the type of Grouper that should be used for assigning DRGs or other case-mix measures. Examples of these Grouper types include: Medicare DRG, TRICARE/CHAMPUS DRG, AP-DRG, and APR-DRG.

3.2.7 Grouper Version
Groupers are updated periodically due to changes in medical practice, changes in statistical or grouping procedures, and changes in the underlying clinical data coding schemes. Each Grouper update is identified by a version number. A payer may utilize the same Grouper type over multiple years, but vary the version of the selected Grouper annually. For example, the Medicare DRG Grouper is updated every year, and each update is assigned a new version number. For FY 2017, which was effective on October 01, 2016, the appropriate Medicare DRG Grouper version number was 34. The user must specify the Grouper type and version number that is appropriate for a particular provider, payer, and effective date.

3.2.8 Analyzer Type
For each provider, payer, and effective date combination, the user must specify the type of Analyzer that should be used for examining claims, if requested. An example of an Analyzer type is: EDC Analyzer™.

3.2.9 Analyzer Version
Analyzers are updated periodically due to changes in medical practice or changes in analyzing procedures. Each Analyzer update is identified by a version number. The user must specify the Analyzer type and version number that is appropriate for a particular provider, payer, and effective date.

3.2.10 Pricer/Payer Type
Each payer reimburses for patient care based on a set of pricing rules with contractual or regulatory requirements. These payer reimbursement rules fall into basic categories known as payer or Pricer types. Examples of these Pricer types include: Medicare, Medicaid, commercial organizations, and various state-specific initiatives. Pricers incorporate formulas for many of these Pricer types. To apply the appropriate reimbursement rules, the Optimizer user must identify which Pricer type to use for each provider, payer, and effective date.

3.2.11 Pricing Variables and Rate Data
If pricing is required, then after the Pricer/Payer Type is identified, as described above, the user must supply associated reimbursement and rate data. These are the values that are input into the reimbursement formulas to drive expected payment calculations. Usually, pricing requires provider or payer specific variables, as well as case-mix-specific data, such as rates, weights, and high and low trim points.
3.2.12 Editor Request Flags

If editing is needed, an Editor Requests Flag (shown below in Table 3-1) needs to be set to identify which Editor needs to be invoked. These variables are specific to the Editor being called.

Table 3-1: Editor Request Flags

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
<th>Position</th>
<th>Applicable Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>edit_req; ECB-EDIT-MCE-SW</td>
<td>0 = No Edits Requested</td>
<td>76</td>
<td>Date-Sensitive Code Editor (DSC)</td>
</tr>
<tr>
<td>edit_req; ECB-EDIT-EZ-SW</td>
<td>0 = No Edits Requested</td>
<td></td>
<td>EASYEdit™ Editor</td>
</tr>
<tr>
<td>edit_req; ECB-EDIT-CCI-SW</td>
<td>0 = No Edits Requested</td>
<td>78</td>
<td>Ambulatory Code Editor™ (ACE)</td>
</tr>
<tr>
<td>edit_req; ECB-EDIT-OCE-SW</td>
<td>0 = No Edits Requested</td>
<td>79</td>
<td>ACE</td>
</tr>
<tr>
<td>edit_req; ECB-EDIT-OCE-CCI-SW</td>
<td>0 = No Edits Requested</td>
<td>80</td>
<td>ACE</td>
</tr>
<tr>
<td>edit_req; ECB-EDIT-LCD-SW</td>
<td>0 = No Edits Requested</td>
<td>81</td>
<td>Local Coverage Determination (LCD) Editor</td>
</tr>
<tr>
<td>edit_req; ECB-EDIT-NOCE-SW</td>
<td>0 = No Edits Requested</td>
<td>82</td>
<td>ACE</td>
</tr>
<tr>
<td>edit_req; ECB-EDIT-POA-SW</td>
<td>0 = No Edits Requested</td>
<td>83</td>
<td>DSC</td>
</tr>
<tr>
<td>edit_req; ECB-EDIT-HAC-SW</td>
<td>0 = No Edits Requested</td>
<td>84</td>
<td>DSC</td>
</tr>
<tr>
<td>edit_req; ECB-EDIT-TRICARE-SW</td>
<td>0 = No Edits Requested</td>
<td>85</td>
<td>ACE</td>
</tr>
<tr>
<td>edit_req2; ECB-EDIT-PHYS-SW</td>
<td>0 = No Edits Requested</td>
<td>86</td>
<td>Physician Editor</td>
</tr>
<tr>
<td>edit_req2; ECB-EDIT-MDCD-SW</td>
<td>0 = No Edits Requested</td>
<td>87</td>
<td>DSC</td>
</tr>
<tr>
<td>edit_req2; ECB-EDIT-MAXMUE</td>
<td>00 = No Edits Requested</td>
<td>88</td>
<td>Physician Editor</td>
</tr>
</tbody>
</table>
Table 3-1: Editor Request Flags

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
<th>Position</th>
<th>Applicable Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>edit_req2</td>
<td>0 = No Edits Requested</td>
<td>89</td>
<td>MOE</td>
</tr>
<tr>
<td></td>
<td>1 = Request Medicaid Outpatient Edits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please refer to the Input & Output Parameter Blocks User’s Guide, as well as the Input & Output Parameter Blocks Workbook for further information.

### 3.3 Optimizer Process Summary

Once the Optimizer data files are set-up via Rate Manager, the Optimizer can perform the following functions:

#### 3.3.1 Read Patient Data

Data describing a patient encounter is passed to the Optimizer, formatted as required by the standard input and output parameter structures (refer to the Input & Output Parameter Block’s User’s Guide).

#### 3.3.2 Select Appropriate Date

The Optimizer reads the provider and payer code from the patient data and locates the corresponding record in the Configuration File. This Configuration File includes the indicator specifying the patient date (admission/from or discharge/thru) that is to be used as the basis for selecting grouping and pricing rules.

#### 3.3.3 Retrieve Rate Calculator Data

Using the date from the previous step, plus the provider identifier and payer code from the patient data, the control program locates and reads the appropriate record from the Configuration File, the Hospital Rate Calculator File, and sometimes the Extended Hospital Rate Calculator File.

#### 3.3.4 Identify Requirements for Analyzer, Editor, Grouper, Mapper, and Pricer

Within the Configuration File records are the variables that indicate the appropriate Analyzer type, Analyzer version, Grouper type, Grouper version, and Pricer type for use with this provider/payer/effective date. The mapping flags for use with inpatient data are also stored on these records, as well as the Editor Request flags.

#### 3.3.5 Invoke Control Programs

The Optimizer calls appropriate control programs to retrieve data when needed.
3.3.6 Perform Appropriate Functions

When the user calls the Optimizer, one of the calling parameters passed is an Operation Code 1 which identifies what the Optimizer should do: analyze, map, edit, group, and price. Based on this field, the Optimizer will then perform the requested functions. If the requested tasks include inpatient grouping, the Optimizer may map codes, as needed. If pricing is required, the Optimizer may retrieve additional information from the case-mix specific rate file and pass it along to the Pricer. For outpatient pricing, the Optimizer may also retrieve fee schedule rate table data from the appropriate fee schedule data files. All results are stored in the appropriate output sections of the input and output parameter structures.
4 Optimizer Integration and Configuration

In this chapter clients will learn the steps required to interface with the EASYGroup™ Optimizer. This chapter includes the following sections:

• Configuring the C Platform Optimizer
  - Reading Patient Records and Specifying Which Functions to Perform
  - Calling the Optimizer
  - Storing Results
  - Error Handling
  - Compiling Your Interface
  - Running the Sample Interface With the Optimizer
  - Optimizer Data Files
  - Optimizer Minimum Required Input Fields
  - Allocating Memory

• Configuring the COBOL Platform Optimizer
  - Driver Program Overview
  - Writing Your Own Driver
  - Calling the Optimizer
  - Using the EASYGroup™ Drivers
  - Determining Applicable Grouper Types and Version
4.1 Configuring the C Platform Optimizer

To utilize the Optimizer with your own patient data, users may create a new interface or call the Optimizer as a subroutine from an existing interface. A sample interface program (sample.c) is included with the Optimizer distribution to assist with the development of an interface to the Optimizer. It can be used as a reference when writing a new interface or be modified and directly incorporated into a user application. Generally, this sample program performs the following functions:

1. Defines and initializes structures/variables to be used in the program.
2. Reads the patient record from the user application and formats the data into the standard input and output parameter structure format (refer to the Input & Output Parameter Blocks User’s Guide).
3. Specifies which functions to perform to each record (i.e., Analyze, Map, Edit, Group, and/or Price). These functions are passed as Optimizer Operation Codes.
4. Calls the Optimizer.
5. Translates the output results from the standard Optimizer input and output parameter structure format back to the user application.
6. Returns error codes for troubleshooting issues when claim processing fails.

In the following section we will describe the steps outlined above that must be performed by all client interface programs. The sample interface program (sample.c) performs a basic inpatient and outpatient call utilizing the ICD-10 Medicare DRG V33 Grouper, the APC V15 Grouper, the Medicare DRG Pricer, and the APC-HOPD Pricer, along with sample rate files.

4.1.1 Reading Patient Records and Specifying Which Functions to Perform

4.1.1.1 Defining Variables
Patient and claims data is passed to the Optimizer using the EASYGroup™-specific input and output parameter structure format. These parameter structures must be initialized and defined by the interface or the user application prior to invoking the Optimizer. Data passed to the Optimizer must be coded exactly as defined in the file: optcntl.h. A summary of structure characteristics are as follows:

- Can consist of input data fields only, output only, or a combination of both input and output.
- Some structures are required, others are optional, and some may not be in use.
- Some structures are fixed length while others are variable in length (e.g., repeating rows for each diagnosis or procedure, etc.).
• Can consist of claim-level data, as well as line-level data for each diagnosis or procedure.
• The ECB [ezg_cntl_block] structure is generally used to control Optimizer processing.

Refer to the Input & Output Parameter Blocks User’s Guide for a list of structures and specifications. The Summary of Data Structures chapter of the Input & Output Parameter Blocks User’s Guide includes a list of data structures that will be used based on the Patient Type and the function requested.

4.1.1.2 Specifying the Patient Type
To make an Edit, Group, and or Price Optimizer request, clients must specify the type of data they wish to pass. This is done via the Patient Type (pattype) field located in the ECB [ezg_cntl_block] structure. Possible values for this field are:

• 01 = Inpatient
• 02 = Outpatient
• 03 = Inpatient Rehabilitation Facility (IRF)
• 04 = Physician
• 06 = Skilled Nursing Facility (SNF)

4.1.1.3 Specifying Which Function to Perform
Along with the Patient Type, clients must specify which function to perform. This is done via the Operation Code 1 (opcode1) field located in the ECB [ezg_cntl_block] structure.

Possible values for this field include:

01 = Analyze/Edit
02 - 10 = Reserved
11 = Analyze/Group
12 = Analyze/Price (assumes case-mix measure already assigned)
13 = Reserved
14 = Model
15 = Analyze/Group/Price
16 = Analyze/Edit/Group/Price
17 = Map only
18 = Analyze only
R1 = Retrieve payer files
O1 = Open files
C1 = Close files
4.1.1.4 Specifying the Type of Editing
For Analyze/Edit and Analyze/Edit/Group/Price functions, clients must also specify which type of editing is to be performed (e.g., CCI only, CCI and OCE, etc.). This is done via the Editor Requests (edit_req and edit_req2) field(s) located in the ECB [ezg_cntl_block] structure. More than one type of editing can be requested. Refer to the Input & Output Parameter Blocks User’s Guide for a complete list of possible values for this field.

Clients may also specify which level of error will be acceptable to continue grouping and pricing via the Acceptable Level of Error (accept_if) field located in the PCB1 [patient_claim_data] structure (for APC and Physician Pricing only).

4.1.1.5 Viewing the Above Specifications in the Sample Interface Program
Collectively, the sample interface program defines/initializes certain variables and sets the ECB [ezg_cntl_block] structure for Patient Type and Operation Code 1, as well as populates the patient and code structures as shown below.
Figure 4-1. Sample.c - Defining Data and Functions

```c
int ProcessInpatient(void)
{
    int nResult = 0;
    int nIndex;
    int numdx = 0;
    int numop = 0;
    struct ezg_blocks blocks;
    SET_P(&blocks, '\0');

    /* Set diagnosis and procedure counts */
    numdx = 1;
    numop = 1;

    /* Allocate and initialize blocks */
    CLEAR_P(blocks.pob1, '0', 1);
    CLEAR_P(blocks.pob2, '0', numdx);
    CLEAR_P(blocks.pcb1, '1', 1);
    CLEAR_P(blocks.pcb2, '1', 1);
    CLEAR_P(blocks.pdx, numdx);
    CLEAR_P(blocks.pop, numop);
    CLEAR_P(blocks.pgob1, numdx);
    CLEAR_P(blocks.pgob2, numop);
    CLEAR_P(blocks.pob1, '0', 1);
    CLEAR_P(blocks.pwax, '\0', 1);
    CLEAR_P(blocks.ppaths, '\0', 1);

    /* Set control block */
    OPT_SET(blocks.pecb->opcode, ECB_GROUP_PRICE, 0);
    OPT_SET(blocks.pecb->pattype, ECB_INPATIENT, 1);
    OPT_SET(blocks.pecb->code_class, "01", 1);
    OPT_SET(blocks.pecb->ext_blk_sw, "1", 1);

    /* Set patient block */
    OPT_SET(blocks.pcb1->facility, "020005", 1);
    OPT_SET(blocks.pcb1->paysrc, "09", 1);
    OPT_SET(blocks.pcb1->from_date, "20151001", 0);
    OPT_SET(blocks.pcb1->thru_date, "20151001", 0);
    OPT_SET(blocks.pcb1->age, "49", 0);
    OPT_SET(blocks.pcb1->sex, "1", 1);
    ITOLN(blocks.pcb1->numdx, numdx);
    ITOLN(blocks.pcb1->numop, numop);
    OPT_SET(blocks.pcb1->dstat, "1", 0);
    OPT_SET(blocks.pcb1->los, "2", 0);

    /* Set inpatient block */
```

```c
```
4.1.2 Calling the Optimizer

Once the data parameters and Optimizer function calls have been set, the Optimizer can be called. The syntax for calling the Optimizer is as follows:

```c
int rc = optcntl (pBlocks);
```

The ECB [ezg_cntl_block] structure contains pointers to other structures. The structures within the ECB [ezg_cntl_block] structure are defined in the `optcntl.h` file and are described in the Input & Output Parameter Blocks User’s Guide. The sample interface program calls the Optimizer, as shown in Figure 4-2.

Figure 4-2. Sample.c - Calling and Returning Results From the Optimizer

```c
/* now call the optimizer */
result = optcntl (&blocks);

if (0 == result)
    {
        char szMDC[32];
        char szDRG[32];
        char szBaseRate[32];
        char szAddOn[32];
        char szTotal[32];
        char szMessage[256];
        /* handle optimizer output */
        OPT_GET(szMDC, blocks.psoBI->drg.mdc, 0);
        OPT_GET(szDRG, blocks.psoBI->drg.drg, 0);
        OPT_GET(szBaseRate, blocks.psoBI->drg.base, 2);
        OPT_GET(szAddOn, blocks.psoBI->drg.addon, 2);
        OPT_GET(szTotal, blocks.psoBI->drg.total, 2);
        printf("szMDC=%s, szDRG=%s, szBaseRate=%s, szAddOn=%s, szTotal=%s\n", szMDC, szDRG, szBaseRate, szAddOn, szTotal);
        printf("%s\n", szMessage);
    }
#else WIN32
    MessageBox(NULL, szMessage, "optimizer sample - Inpatient", MB_ICONINFOR
#endif /* WIN32 */

result = optimizererror(result, &blocks);

safefree(blocks.pCB);
safefree(blocks.pCB1);
safefree(blocks.pCB2);
safefree(blocks.pCBx);
safefree(blocks.pCBp);
safefree(blocks.pCBb);
safefree(blocks.pCBg);
safefree(blocks.pCBh);
safefree(blocks.pCBw);
safefree(blocks.pCBx);
return nResult;
```

4.1.3 Storing Results

Once the Optimizer has been called and results are returned, the interface can store the results. Once the Optimizer completes its processing, results and other output data is passed back from the Optimizer using the EASYGroup™ specific input and output parameter structure format (refer to the Input & Output Parameter Blocks User’s Guide). These parameter
structures must be defined by the interface or the user application prior to invoking the Optimizer. Refer back to the Defining Variables section for more information on required and optional output parameter structures.

4.1.3.1 Storing Results and Clearing Memory With the Sample Interface Program
The sample interface program stores results and clears memory for the next record processing as shown in Figure 4-2 above.

4.1.4 Error Handling
An important feature of the Optimizer is the reporting of error codes in situations where the Optimizer cannot process a patient record. The Optimizer has a two-tier architecture for returning error codes: First-Tier Optimizer Return Codes typically indicate high level errors in the initialization or processing such as missing files (e.g., Optimizer Return Code 05 (Error Calling Editor Control Program (edtcntl))) or memory issues.

Optimizer Return Codes 11 - 19, 21, and 22 (e.g., Optimizer Return Code 12 (Non-Zero Return Code from DSC Editor)) represent Second-Tier Optimizer Return Codes that typically indicate errors with the patient data or record that cause an Edit, Group, and/or Price request to fail. These errors include invalid principal diagnosis (e.g., DRG Grouper Return Code 07) or invalid dates of service (e.g., APC Pricer Return Code 23). In these cases, the Optimizer will first return a Return Code of 11 - 19, 21, or 22, then return the Editor, Grouper, Mapper, or Pricer Return Codes, as appropriate.

Note
All Optimizer Return Codes are returned in the Optimizer Return Code (opt_rtn_code) field located in the OOB1 [opt_output_block1] structure.

Please refer to the EASYGroup™ Technical Reference Guide for a list of all available Return Codes.

4.1.5 Compiling Your Interface
The following steps describe how to compile an interface or calling program to the Optimizer.

1. Copy your interface or calling program (or modified sample.c) to your working environment.

2. For non-Windows® platform users, also copy the following Optimizer Source files into your working environment: cntdll.c, optcntl.h, and static.h.

3. Compile using commands appropriate for your working environment.
4.1.6 Running the Sample Interface With the Optimizer

The sample interface program was modeled to call the Optimizer for a simple Group and Price function with the Medicare Inpatient and Outpatient Prospective Payment Systems (PPS). To run the sample interface program, you must have the following EASYGroup™ products installed:

- ICD-10 Medicare DRG V33 or higher Grouper
- APC V15 or higher Grouper
- Medicare DRG Pricer
- APC-HOPD Pricer
- Inpatient National Medicare Provider Rate Files (NMPRF)
- Outpatient National Medicare Provider Rate Files (ONMPRF)

These products must be individually installed and tested before they can be called with the sample interface program and the Optimizer. Please refer to the EASYGroup™ Installation Guide for further information on installation procedures. If the Optimizer and all applicable EASYGroup™ products are installed and communicating successfully, a call from the sample interface program will produce results.

4.1.7 Optimizer Data Files

Depending on the operation being requested, the Optimizer may require specific data files. These data files are used to determine which additional EASYGroup™ components to call, and in some cases to retrieve additional data required by these products. All Optimizer data files are typically stored in the same directory. The path of this directory is passed to the Optimizer.

The Optimizer distribution contains sample data files which are used for testing only. For production processing, these data files should be assembled from the following sources:

- From Rate Manager: `payors.dat`, `payout.dat`, `payirf.dat`, `paysnf.dat`, `payphys.dat`, `medcalc.dat`, `medout.dat`, `medext.dat`, `medirf.dat`, `medsnf.dat`, `medphys.dat`, `rate.dat`, `rateout.dat`, `ratephys.dat`, `rateirf.dat`, `ratesnf.dat`, `config.dat`, `cfgphys.dat`
- From the Fee Schedule Data Files: `fs***yy.dat`, `ex***yy.dat`, `zipcode.dat`
- From the Data Files: `code***.dat`
### Note

To view a list of all available EASYGroup™ Fee Schedule Data Files and Data Files please refer to the EASYGroup™ User’s Guide, as well as the Rate Manager User’s Guide and Rate Manager Installation Guide for detailed instructions on the preparation and installation of these data files.

### 4.1.8 Optimizer Minimum Required Input Fields

This section lists the minimum input fields that are required for basic inpatient and outpatient claims processing for the C Platform. These tables are intended to be used in conjunction with the input and output parameters structures defined in the Input & Output Parameter Blocks User’s Guide.

#### 4.1.8.1 Inpatient

Table 4-1: Optimizer Inpatient Minimum Required Input

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Field Description</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB</td>
<td>Operation Code</td>
<td>opcode1</td>
</tr>
<tr>
<td>ECB</td>
<td>Patient Type</td>
<td>pattype</td>
</tr>
<tr>
<td>PCB1</td>
<td>Provider Identifier</td>
<td>facility</td>
</tr>
<tr>
<td>PCB1</td>
<td>National Provider Identifier</td>
<td>npi</td>
</tr>
<tr>
<td>PCB1</td>
<td>Taxonomy Code</td>
<td>taxonomy</td>
</tr>
<tr>
<td>PCB1</td>
<td>Payer ID or Contract Code</td>
<td>paysrc</td>
</tr>
<tr>
<td>PCB1</td>
<td>UB-04 Bill Type</td>
<td>billtype</td>
</tr>
<tr>
<td>PCB1</td>
<td>From or Admission Date</td>
<td>from_date</td>
</tr>
<tr>
<td>PCB1</td>
<td>Thru or Discharge Date</td>
<td>thru_date</td>
</tr>
<tr>
<td>PCB1</td>
<td>Date of Birth or Age</td>
<td>birth_date</td>
</tr>
<tr>
<td>PCB1</td>
<td>UB-04 Admit Source</td>
<td>admsource</td>
</tr>
<tr>
<td>PCB1</td>
<td>Sex</td>
<td>sex</td>
</tr>
<tr>
<td>PCB1</td>
<td>Discharge Disposition</td>
<td>dstat</td>
</tr>
<tr>
<td>PCB1</td>
<td>Total Covered Charges</td>
<td>tot_chg</td>
</tr>
<tr>
<td>PCB1</td>
<td>UB-04 Condition Codes</td>
<td>condcd</td>
</tr>
<tr>
<td>PCB1</td>
<td>Length of Stay</td>
<td>los</td>
</tr>
<tr>
<td>PCB1</td>
<td>Admission Diagnosis</td>
<td>admit_dx</td>
</tr>
<tr>
<td>DX</td>
<td>All Diagnosis Codes</td>
<td>dx</td>
</tr>
<tr>
<td>OP</td>
<td>All Procedure Codes</td>
<td>op</td>
</tr>
<tr>
<td>LINE</td>
<td>Claim Line Data:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Revenue Code</td>
<td>rev</td>
</tr>
<tr>
<td></td>
<td>HCPCS Code (if present)</td>
<td>hcpcs</td>
</tr>
<tr>
<td></td>
<td>Modifiers (if present)</td>
<td>mod</td>
</tr>
<tr>
<td></td>
<td>Units</td>
<td>units</td>
</tr>
<tr>
<td></td>
<td>Charges</td>
<td>charges</td>
</tr>
<tr>
<td></td>
<td>Service Date</td>
<td>date</td>
</tr>
<tr>
<td>PCB1</td>
<td>All UB-04 Value Code/Value Amount Pairs</td>
<td>valcode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>valamt</td>
</tr>
</tbody>
</table>
### 4.1.8.2 Outpatient

**Table 4-2: Optimizer Outpatient Minimum Required Input**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Field Description</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCB1</td>
<td>All UB-04 Occurrence Code/Occurrence Date Pairs</td>
<td>occur_code, occur_date</td>
</tr>
<tr>
<td>PCB1</td>
<td>All UB-04 Occurrence Span/Occurrence Span Dates</td>
<td>span_code, span_date1, span_date2</td>
</tr>
</tbody>
</table>

**Table 4-1: Optimizer Inpatient Minimum Required Input**

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Field Description</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB</td>
<td>Operation Code</td>
<td>opcode1</td>
</tr>
<tr>
<td>ECB</td>
<td>Patient Type</td>
<td>pattype</td>
</tr>
<tr>
<td>PCB1</td>
<td>National Provider Identifier</td>
<td>facility</td>
</tr>
<tr>
<td>PCB1</td>
<td>National Provider Identifier</td>
<td>npi</td>
</tr>
<tr>
<td>PCB1</td>
<td>Taxonomy Code</td>
<td>taxonomy</td>
</tr>
<tr>
<td>PCB1</td>
<td>Payer ID or Contract Code</td>
<td>paysrc</td>
</tr>
<tr>
<td>PCB1</td>
<td>UB-04 Bill Type (APC only)</td>
<td>billtype</td>
</tr>
<tr>
<td>PCB1</td>
<td>From or Admission Date</td>
<td>from_date</td>
</tr>
<tr>
<td>PCB1</td>
<td>Thru or Discharge Date</td>
<td>thru_date</td>
</tr>
<tr>
<td>PCB1</td>
<td>Date of Birth or Age</td>
<td>birth_date, age</td>
</tr>
<tr>
<td>PCB1</td>
<td>Sex</td>
<td>sex</td>
</tr>
<tr>
<td>PCB1</td>
<td>Discharge Disposition (APC only)</td>
<td>dstat</td>
</tr>
<tr>
<td>PCB1</td>
<td>Total Covered Charges</td>
<td>tot_chg</td>
</tr>
<tr>
<td>PCB1</td>
<td>UB-04 Condition Codes (APC only)</td>
<td>condcd</td>
</tr>
<tr>
<td>PCB1</td>
<td>UB-04 Treatment Authorization Code (HHA only, bill types 032x and 033x (where x equals an alphanumeric character))</td>
<td>tx_authcode</td>
</tr>
<tr>
<td>PCB2</td>
<td>UB-04 Reason for Visit Diagnosis (APC only)</td>
<td>rfvdx</td>
</tr>
<tr>
<td>DX</td>
<td>All Diagnosis Codes</td>
<td>dx</td>
</tr>
<tr>
<td>LINE</td>
<td>Claim Line Data: Revenue Code (APC only)</td>
<td>rev</td>
</tr>
<tr>
<td></td>
<td>HCPCS Code (if present)</td>
<td>hcpcs</td>
</tr>
<tr>
<td></td>
<td>Modifiers (if present)</td>
<td>mod</td>
</tr>
<tr>
<td></td>
<td>Units</td>
<td>units</td>
</tr>
<tr>
<td></td>
<td>Charges</td>
<td>charges</td>
</tr>
<tr>
<td></td>
<td>Service Date</td>
<td>date</td>
</tr>
<tr>
<td>PCB1</td>
<td>All UB-04 Value Code/Value Amount Pairs (APC only)</td>
<td>valcode, valamt</td>
</tr>
<tr>
<td>PCB1</td>
<td>All UB-04 Occurrence Code/Occurrence Date Pairs (APC only)</td>
<td>occur_code, occur_date</td>
</tr>
</tbody>
</table>
4.1.9 Allocating Memory

The Optimizer previously (prior to V1510.02) allocated memory for most EASYGroup™ output structures if memory was not allocated for those structures by the calling program. With the release of the V1510.02 and V1712.00 Optimizer, the Optimizer no longer allocated memory for the calling program. Clients who have not allocated memory appropriately will receive Optimizer Return Code 95 (Parameter Error). Clients should be allocating memory for all EASYGroup™ input and output structures.

4.2 Configuring the COBOL Platform Optimizer

4.2.1 Driver Program Overview

After you have successfully assembled the Optimizer, you must create a Driver program to invoke the Optimizer for your own database or databases. You can also call the Optimizer as a subroutine from one of your existing programs.

4.2.1.1 Function of the Driver

The purpose of the Driver program is to read records from a database, organize data from that record into a form that can be used by the Optimizer, request the optimizing process and associated information, and store (or make some other use of) the resulting Optimizer information. The Driver program also has two housekeeping functions. It must initially tell the Optimizer Control program to open files and must also direct the Optimizer Control program to close all files before the Driver program terminates.

4.2.2 Writing Your Own Driver

4.2.2.1 Writing Your Own Driver - Batch

To add the Optimizer subroutine call to your own batch programs, you must proceed as follows:

1. Use the copy members supplied with the distribution of the Optimizer and any EASYGroup™ Analyzer, Grouper, Pricer, Editor, or Mapper in the Working Storage section of your program. Refer to the applicable Microsoft® Excel spreadsheet(s) that are available on the Optum Client Portal and/or the Regulatory Portal for a list of available copybooks.

2. To call the Optimizer (after a successful open), you must first move all required input variables into the appropriate data structures, which will

---

Table 4-2: Optimizer Outpatient Minimum Required Input

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Field Description</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCB1</td>
<td>All UB-04 Occurrence Span/Occurrence Span Dates (APC only)</td>
<td>span_code, span_date1, span_date2</td>
</tr>
</tbody>
</table>
be used to send and receive information from the Optimizer control program.

3. Using the header file HEZGPTR, you must set your pointers. This structure contains a block of pointers corresponding to all existing EASYGroup™ structures. Prior to calling the Optimizer, add the following code:

```
MOVE LOW-VALUES TO ESP-EZG-STRUCT-PTRS.
```

```
SET ESP-EZB TO ADDRESS OF ECB-EZG-CNTL-BLOCK.
SET ESP-FCB1 TO ADDRESS OF ECB-PATIENT-CHAIN-BLOCK1.
SET ESP-FCB2 TO ADDRESS OF ECB-PATIENT-CHAIN-BLOCK2.
SET ESP-CCB TO ADDRESS OF ECB-DL-CODES-BLOCK.
SET ESP-IIB TO ADDRESS OF RCT-ICPCS-CODE-TBL.
SET ESP-MBD1 TO ADDRESS OF NCB1-MAP-OUTPUT-BLOCK1.
SET ESP-MBD2 TO ADDRESS OF NCB2-MAP-OUTPUT-BLOCK2.
SET ESP-MBD3 TO ADDRESS OF NCB3-MAP-OUTPUT-BLOCK3.
SET ESP-GOB1 TO ADDRESS OF GCB1-GRFR-OUTPUT-BLOCK1.
SET ESP-GOB2 TO ADDRESS OF GCB2-GRFR-OUTPUT-BLOCK2.
SET ESP-GOB3 TO ADDRESS OF GCB3-GRFR-OUTPUT-BLOCK3.
SET ESP-GOB4 TO ADDRESS OF GCB4-GRFR-OUTPUT-BLOCK4.
SET ESP-FOB1 TO ADDRESS OF POB1-PCHR-OUTPUT-BLOCK1.
SET ESP-FOB2 TO ADDRESS OF POB2-PCHR-OUTPUT-BLOCK2.
SET ESP-FOB3 TO ADDRESS OF POB3-PCHR-OUTPUT-BLOCK3.
SET ESP-COB1 TO ADDRESS OF OCB1-OUT-OUTPUT-BLOCK1.
SET ESP-COB2 TO ADDRESS OF OCB2-OUT-OUTPUT-BLOCK2.
SET ESP-MED1 TO ADDRESS OF NCB1-MCE-EDITOR-BLOCK1.
SET ESP-MED2 TO ADDRESS OF NCB2-MCE-EDITOR-BLOCK2.
SET ESP-MED3 TO ADDRESS OF NCB3-MCE-EDITOR-BLOCK3.
SET ESP-MED4 TO ADDRESS OF NCB4-MCE-EDITOR-BLOCK4.
SET ESP-ED1 TO ADDRESS OF ESB1-ZEDIT-EDITOR-BLOCK1.
SET ESP-ED2 TO ADDRESS OF ESB2-ZEDIT-EDITOR-BLOCK2.
SET ESP-ED3 TO ADDRESS OF ESB3-ZEDIT-EDITOR-BLOCK3.
SET ESP-ED4 TO ADDRESS OF ESB4-ZEDIT-EDITOR-BLOCK4.
SET ESP-AED1 TO ADDRESS OF ABEL-ACE-EDIT-BLOCK1.
SET ESP-AED2 TO ADDRESS OF ABEL-ACE-EDIT-BLOCK2.
SET ESP-AED3 TO ADDRESS OF ABEL-ACE-EDIT-BLOCK3.
SET ESP-AED4 TO ADDRESS OF ABEL-ACE-EDIT-BLOCK4.
SET ESP-AED5 TO ADDRESS OF ABEL-ACE-EDIT-BLOCK5.
SET ESP-LEB1 TO ADDRESS OF LEB1-LCD-EDIT-BLOCK1.
SET ESP-LEB2 TO ADDRESS OF LEB2-LCD-EDIT-BLOCK2.
SET ESP-LEB3 TO ADDRESS OF LEB3-LCD-EDIT-BLOCK3.
SET ESP-FRB TO ADDRESS OF FEB-FUNC-RTN-BLOCK.
SET ESP-FVB1 TO ADDRESS OF FWB1-PRC-RXSRT-1.
SET ESP-FEB1 TO ADDRESS OF FEB1-PSY-EDIT-BLOCK1.
SET ESP-FEB2 TO ADDRESS OF FEB2-PSY-EDIT-BLOCK2.
SET ESP-FEB3 TO ADDRESS OF FEB3-PSY-EDIT-BLOCK3.
```

4. The specific functions that the Optimizer will perform when processing individual patient records depends on the patient type and operation requested, as specified by the values of the ECB-OPCODE1 and ECB-PATTYPE fields which are passed as input to the Optimizer in the ECB-EZG-CNTL-BLOCK.

5. The Optimizer can return Analyzer output data, Grouper output data, Pricer output data, Editor output data, Mapper output data, and Optimizer output data. The specific data returned depends upon what function the Optimizer is performing, and whether inpatient or outpatient data is being processed. Remember that no matter what task is requested of the Optimizer, the returned Grouper and Pricer data reflects case-mix assignment variables and pricing information based on the original, incoming diagnosis and procedures. You may
want to move some or all of this data into your record before you write the record back to your database.

6. The Optimizer Control Program has been designed to automatically open files. To initiate a manual open, your Driver can call OPTCNTL with an O1 (letter O, number 1) in ECB-OPCODE1 in the Optimizer control structure. If problems occur during the open process, the Optimizer will return a 16 in the Optimizer Return Code field OOB-OPT-RTN-CODE.

To close all opened files, your Optimizer driver must call OPTCNTL with a C1 in the ECB-OPCODE1 field. Your Driver should perform the close command after all patient data has been processed.

After you have compiled your Driver program, you must link it to the other EASYGroup™ components.

7. Multiple sets of edits can be requested with a single Optimizer request. To do this users must set the Edit OP codes. A value of 01 or 10 will represent an edit request. The user then has the option of invoking any combination of Editors with one call by setting a bank of edit switches.

To set switches:

Call the Optimizer: Set the OPCODE field and edit switches accordingly.

The Optimizer has embedded logic to set default edit values for an EditGroupPrice request (MCE edits for inpatient, OCE edits for APC, SNF Part B, ESRD, and HHA, CCI edits for APG and ASC, Physician edits for Physician). Users do not need to specify the individual edit settings when using these default editing settings.

4.2.2.2 Writing Your Own Driver - CICS
Adding the Optimizer subroutine call to your own CICS programs follows the same basic procedures as those outlined in the batch Driver section. The primary difference is that in CICS you do not have to code your Driver to initiate any explicit file open and close requests. Those functions are not applicable to a CICS environment and as such are not supported in the online Optimizer source code.

4.2.3 Calling the Optimizer
The basic COBOL calling syntax for a batch implementation of the Optimizer is as follows:

```
CALL ‘OPTCNTL’ USING ESP-EZG-STRUCT-PTRS.
```

The basic COBOL calling syntax for a CICS implementation is as follows:

```
CALL ‘COPTCNTL’ USING DFHEIBLK DFHCOMMAREA
                  ESP-EZG-STRUCT-PTRS.
```

The appropriate input fields of each parameter structure must be filled in based on the type of operation you are requesting of the Optimizer. Please
refer to the Input & Output Parameter Blocks User’s Guide for further information. The Optimizer will populate the results fields before returning to your Driver program.

4.2.4 Using the EASYGroup™ Drivers

Individual product drivers have embedded calls for invoking and testing the Optimizer. Use the sample batch drivers included in the respective distributions (i.e., Groupers, Pricers, Editors, etc).

4.2.5 Determining Applicable Grouper Types and Version

The Grouper type and version to be used for a particular patient record is determined by using information stored in the Configuration File. A description of this file can be found in this guide for both the inpatient and outpatient EASYGroup™ Pricers. The Hospital Rate Calculator File also serves a role within the Optimizer, as described below.

First, the Hospital Rate Calculator File contains the hospital and paysource specific rate information needed by the EASYGroup™ Pricers to calculate a patient’s anticipated reimbursement. Each record in this file has an effective date which indicates the earliest date upon which the rate information in this record should be utilized. If the Hospital Rate Calculator File has more than one record for a particular hospital and paysource, the Pricers utilize the most current record that is before or equal to the patient's date of hospitalization.

Second, each record in the Configuration File contains the Grouper type and version that should be used for this facility and paysource on or after the specified effective date. These fields are used by the Optimizer to determine which Grouper type and version should be used for a particular record. Grouper type and version are not required for patient pricing.

When creating your Hospital Rate Calculator File and Configuration File remember that a new record must be created each time reimbursement rate variables change. Place into the Effective Date field the date that these new rate variables go into effect. If the Grouper type and version to be used for this hospital and paysource has not changed, then enter the same values contained on the previous record. Similarly, a new record must be created any time a new Grouper type and/or version is put into effect. If reimbursement rate variables have not changed, copy these fields from the previous record.
5 Server Overview

This chapter provides an introduction to the EASYGroup™ Server. It contains the following sections:

- Server Overview
  - Features Supported
5.1 Server Overview

The Server provides access to the Optimizer as a Windows® service. Through the Optimizer, the full set of EASYGroup™ components are accessible (including Groupers, Pricers, Editors, etc.). The Server runs on a node of a TCP/IP network. It contains a service component that listens to a user-defined port number on the TCP/IP network. This component uses an internal Daemon object and dynamically creates internal TCP objects. Clients can access the Server by connecting to a pre-defined port and issuing a request. For each request, the Server will perform an Optimizer function call (grouping, pricing, editing, etc.) and return the results to the client. The Server can run as either a server application or as a system service on a machine capable of supporting the system requirements listed in this guide and the EASYGroup™ Installation Guide.

5.1.1 Features Supported

The Server includes the following features:

- Accessibility Across Platforms:
  The Server can be called by applications running on any platform that has access to the Server via TCP/IP, including Windows®, UNIX®, AIX, Linux®, VAX, IBM® mainframe (requires ASCII encoding), or other platforms.

- Integration:
  The syntax for calling the Server is straightforward and calls are easily integrated into the client application.

- Installation:
  Installation of the Server and associated EASYGroup™ components is effortless. No compilation is necessary and testing is automated using the EASYGroup™ (TCP/IP) Client. For further information on EASYGroup™ installation procedures please refer to the EASYGroup™ Installation Guide.

- Product Updates:
  Updates to any of the EASYGroup™ editing, mapping, grouping, and pricing components managed by the Server are installed on the Server machine. Generally, no changes are necessary to the client programs.

- Processing Speed:
  With the Server, all EASYGroup™ components are run on a centralized machine. Processing speed is linked to this central server only so that a server upgrade will improve performance for all client applications, including those coming from lower-end machines. Wherever possible, data files are memory-mapped for greatest throughput.
Note

The Server is **not** compatible with high-availability cluster environments at this time.

- **Problem Solving:**
  
  With the Server, EASYGroup™ components are not embedded within the calling application and are external to the calling (client) applications. This makes it easier to identify the source of any problems that might arise (in the EASYGroup™ components or the client application).

- **Alternate Rate Paths:**
  
  The Server supports user-defined rate paths that allow the user to reference an alternate set of rate data, which can reside in a directory other than the default **Data** directory (\C:\EASYGroup\Server\Data or \C:\EASYGroup\Server64\Data). There are three paths in the **ezg_paths** structure: 1) System, 2) Rates, and 3) User. **System** is the default directory; all data files reside in this location. Alternatively, the Rate Path(s) can be used for Rate Manager data files and the User Path can be used for EASYGroup™ component data files (e.g., aceedit, cciedit, ccimi, ocecci2, ocemi, ocecci, and mue used by the Ambulatory Code Editor™ (ACE)).

- **National Provider Identifier (NPI):**
  
  The Server currently supports the use of hospital rate data identified by legacy provider number (i.e., OSCAR number) or the National Provider Identifier (NPI) and associated Taxonomy Code. If the incoming claim contains both legacy and NPI hospital identifiers, both can be submitted to the Server. The underlying rate retrieval programs will attempt to locate rates first using the NPI and any Taxonomy code (if applicable), along with the first nine bytes of the payer code. If this lookup fails, the second attempt will use the legacy hospital identifier along with the full payer code.
6 Server Integration and Configuration

This chapter provides the integration and configuration procedures for the Server. It contains the following sections:

- Server Integrations
  - Server Setup
  - Passing Data to Send and Receive From the Server
  - Running the Sample Program with the Server
  - Incomplete Message Testing
  - Additional Testing
- Server Configuration
  - Using the Instance Wizard
  - Command Syntax Overview
6.1 Server Integrations

To utilize the EASYGroup™ Server with your own patient data, users may create a new program or modify an existing program to manage requests or messages to and from the Server. A sample interface program (sample.exe plus sample.cs) is included with the Server distribution to assist with the development of an interface to the Server. It can be used as a reference when writing a new interface or be modified and directly incorporated into a user application. Generally, this sample program performs the following functions:

1. Defines the Server name and location (port number) and establishes the connection with the Server.
2. Converts data (such as patient records) into strings based on the standard Optimizer input and output parameter block format (Input and Output Parameter Blocks User’s Guide).
3. Combines strings into CTL messages and sends the request to the Server for Optimizer processing (i.e., map, edit, group, and/or price).
4. Receives CTL message response from the Server and converts message back to strings.
5. Displays results, including error codes for troubleshooting issues when claim processing fails.

This section will focus primarily on the sample interface program to describe a basic inpatient Group and Price request to the Server using hard-coded values. The sample program also includes functions for a basic outpatient group and price request.

6.1.1 Server Setup

6.1.1.1 Defining Server Name and Location

As part of the initial installation process, the Server is assigned a host name and port number (by default, LOCALHOST for host name and 4321 for port number). The host name identifies the machine on which the Server is running and the port number identifies the TCP/IP port that has been defined to send requests or messages to the Server for processing.

In the Sample Interface Program (sample.cs, lines 406-408) these are defined, as follows (Figure 6-1):

Figure 6-1. sample.cs (Lines 406-408) - Defining the Server

```c
string ServerName = "localhost";
int ServerPort = 4321;
string command = "";
```

6.1.1.2 Establishing Connection to Server

Once the ServerName and Port location have been defined, a connection to the Server can be attempted. To test that the connection is successful, you can use the VER message type, as described in the Server Overview chapter of this guide.
In the Sample Interface Program (sample.cs, lines 417-425) the Server connection is established and tested, as follows (Figure 6-2):

Figure 6-2. sample.cs (Lines 417-425) - Establishing Connection

```csharp
TcpClient Server = new TcpClient(ServerName, ServerPort);
Server.ReceiveTimeout = 30000;
NetworkStream Stream = Server.GetStream();

// Send VER message
string[] blocks = null;
Send(Stream, "VER", null);
Receive(Stream, ref command, ref blocks);
Print(command, blocks);
```

6.1.2 Passing Data to Send and Receive From the Server

6.1.2.1 Strings and Message Types

Patient and claim data is passed to the Server and Optimizer via CTL messages. All message types are passed in the following format: nnnnnnCTLdata, where nnnnnn (6 ASCII characters) represents the total length of the message not including the 6-digit byte count, CTL (3 ASCII characters) represents the message type, and data represents the input or output structures, each of which is followed by a vertical bar (|) (used as a separation delimiter). Refer to the Server Overview chapter of this guide for more information on the CTL message type and command syntax.

Returning to the sample CTL message that was provided in the Server Overview chapter of this guide, Table 6-1 features a basic inpatient CTL
request to the Server, and the resulting CTL output message response from the Server (via the Sample Interface Program).

Table 6-1: CTL Message Request/Response

<table>
<thead>
<tr>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>000175CTL</td>
<td>000987CTL00 DRG</td>
</tr>
<tr>
<td>09 2001100120011003</td>
<td>09 2001100120011003 0401</td>
</tr>
<tr>
<td>0401</td>
<td>0401</td>
</tr>
<tr>
<td>01001 01</td>
<td>01001 01</td>
</tr>
<tr>
<td>0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000987CTL00 DRG</td>
</tr>
</tbody>
</table>

To make the process of building CTL messages simpler, the Sample Interface Program includes functions to dynamically build Server messages. The Opt function (lines 67-106) performs two basic tasks: 1) calculates field positions, lengths, and formats (alphanumeric and decimal) based on the standard Optimizer input and output structures, and 2) builds the strings to be used in the CTL message (Figure 6-3).
Then the Sample function (lines 109-165) combines the strings, calculates the total message length, and builds the CTL message to send to and receive from the Server (Figure 6-4).
6.1.2.2 Inpatient Processing

The Sample Interface Program features a basic Group and Price request to the Server using the sample test data provided for an inpatient claim (i.e., Facility ID, dates of service, age, sex, diagnosis codes, etc.), as well as the required Optimizer fields (i.e., type of data (pattype) and type of function to perform (opcode1)) defined in the ECB [ezg_cntl_block] structure. Refer to the Input & Output Parameter Blocks User's Guide for a list of patient type, function, and the associated data structure requirements.

Function Process Inpatient (lines 229-298) of the Sample Interface Program is scripted, as follows.

Lines 229-270 define the data values for the inpatient claim and specifies which strings will be combined together for the overall Server message request (Figure 6-5).
Then the Server request is sent and a response (and errors, if applicable) is received (lines 272-281) (Figure 6-6).
6.1.3 Running the Sample Program with the Server

The Sample Interface Program was modeled to send a request for a simple Group and Price function with the EASYGroup™ Medicare Inpatient and Outpatient Payment Systems. To run the sample program, you must have the following EASYGroup™ components:

- Medicare DRG Grouper
- APC Grouper
- Medicare DRG Pricer
- APC-HOPD Pricer

These products must be individually installed in the Server directory before they can be used with the sample program. Refer to the EASYGroup™ Installation Guide for instructions.

If the Optimizer and EASYGroup™ components are installed and communicating successfully, the sample program will produce the following results:

```csharp
if (0 == Result)
{
    //Print(Command, Blocks);
    Send(stream, Command, Blocks);
    Receive(stream, ref Command, ref Blocks);
    //Print(Command, Blocks);
    Result = OptimizerError(Blocks);
    if (0 != Result || Blocks.Length < 10)
        Print(Command, Blocks);
}
if (0 == Result && Blocks.Length >= 10)
{
    Opt Gob1 = new Opt(206, Blocks[6]);
    Opt Pob1 = new Opt(304, Blocks[10]);
    StringBuilder Text = new StringBuilder();
    Text.AppendFormat("MDC={0}, DRG={1}, ",
    Gob1.Get(20, 2, 0), Gob1.Get(22, 5, 0));
    Text.AppendFormat("BaseRate={0}, AddOn={1}, Total={2},
    Pob1.Get(14, 10, 2), Pob1.Get(24, 10, 2), Pob1.Get(44, 10, 2));
    Console.WriteLine(Text);
    return Result;
}```
6.1.4 Incomplete Message Testing

The EASYGroup™ (TCP/IP) Client can also be used to simulate the conditions related to sending an incomplete message, either due to message truncation or incorrect formatting. The Server will detect any message that is shorter than the supplied record length. To do this, enter the word timeout in the Output File field of the EASYGroup™ Client screen and press the Save button.

After thirty seconds, you will receive a time-out error, which will also be recorded in the log.
6.1.5 Additional Testing

The Server distribution includes sample input data (*srv*.txt) files and expected results or output data (*srv*.sav). This data can be used for additional testing of the Server.

**Note**
The *srv* sample files require the same EASYGroup™ products as the Sample Interface Program described above. Be sure these products are installed to the appropriate directory prior to using these files.

**6.1.5.1 Changing the Data File Location**
To validate the Server, you will need to change the Server **Data** directory to the **Test** directory that contains the inpatient and outpatient test files.

The recommended default Server **Data** directory upon installation is:
- C:\EASYGroup\Server\Data
- C:\EASYGroup\Server64\Data

The recommended default Server **Test** directory upon installation is:
- C:\EASYGroup\Server\Test
- C:\EASYGroup\Server64\Test

**If Running the Server as an Application:**
1. Select from the Windows® **Start** Menu:
    ```
    Start>All Programs>EASYGroup™ Server
    ```
2. Select the **Stop** button.
3. Edit the **Data Files Folder** field to reference the **Test** directory containing the sample test files (by default: C:\EASYGroup\Server\Test or C:\EASYGroup\Server64\Test).
4. Select the **Apply** button.
5. Re-start the Server by selecting the **Start** button.
Figure 6-11. Stopping the Server

Server start at port 4321. Server Logging is disabled.
Server Closed.
If Running the Server as a Service:

1. Stop the service using the Windows® Services Manager.

**Note**
The default name for the service is HSS Optimizer Server. If multiple instances have been created using the Instance Wizard, the name of the service was defined in the Instance Wizard Setup. Please refer to the Server Overview chapter of this guide.

2. Enter the following command at the RUN prompt: `optsrv /c`

3. Edit the **Data Files Folder** field to resource the **Test** directory containing the sample test files (by default: `C:\EASYGroup\Server\Test` or `C:\EASYGroup\Server64\Test`).

4. Select the **Apply** button.

5. Re-start the Server using the Windows® Services Manager.

**Figure 6-12. Optimizer Server Configuration**

### 6.1.5.2 Using the EASYGroup™ (TCP/IP) Client

The EASYGroup™ (TCP/IP) Client is provided to validate the installation of the Server and related EASYGroup™ components. Results from the test data can be compared against previously supplied results for that data.

**Note**
The following example is for inpatient testing. For outpatient testing, replace `srvinp` with `srvout` and follow the same procedures.

1. Select from the Windows® **Start** Menu:
   
   Start> All Program> EASYGroup™ Server> EASYGroup™ Client.

2. Double-click the field name **Output File** to enable the compare feature of the EASYGroup™ (TCP/IP) Client. The field name will change to **Compare File** when it is enabled.

3. In the **Input File** field, enter the directory path for the `srvinp.txt` test file (by default: `C:\EASYGroup\Server\Test\srvinp.txt` or `C:\EASYGroup\Server64\Test\srvinp.txt`).
4. In the **Compare File** field, enter the directory path for the `srvinp.sav` test file (by default: `C:\EASYGroup\Server\Test\srvinp.sav` or `C:\EASYGroup\Server64\Test\srvinp.sav`).

5. Select the **Send** button to initiate the compare.

6. Double-click **Compare File** to switch back to **Output File** and disable the compare function.

7. Change the Server **Data files** folder location back to the original setting (by default: `C:\EASYGroup\Server\Data` or `C:\EASYGroup\Server64\Data`) once validation and testing have been completed.

---

**Figure 6-11. EASYGroup™ (TCP/IP) Client Screen**

A successful test will report the count of records that were run in the test file as well as an elapsed time for this query in the bottom-right corner of the **EASYGroup™ (TCP/IP) Client** screen. If the **EASYGroup™ (TCP/IP) Client** detects any mismatches between the actual results and the expected results, it will notify you of the mismatch and the record number.

### 6.2 Server Configuration

#### 6.2.1 Using the Instance Wizard

If you wish to set-up more than one instance of the Server on the same machine, you can use the Instance Wizard (`instance.exe`) to create and maintain additional **EASYGroup™** Servers. The most common reason to do this is to support both testing and production servers on the same machine.

The first installation should be the primary installation. Typically, this installation is used for testing. Updates to the Server will automatically be applied to this primary installation. Subsequent installations or "instances" can
be created for quality assurance, training, or production purposes via the Instance Wizard. All instances will be updated in relation to the primary installation.

The installation programs for the various EASYGroup™ components (i.e., Editors, Mappers, Groupers, and Pricers) will automatically install the appropriate programs and data files to the primary Server directories. Refer to the EASYGroup™ Installation Guide for information on these installation procedures. The Instance Wizard will then copy these products to the additional Server instances.

After completing the primary Server installation, the installation of additional instances involves two steps: (1) creating the additional instances and (2) synchronizing those instances with the primary installation.

To create an instance of the Server:

1. Invoke the Instance Wizard using the Windows® Start menu:
   Start>All Programs>EASYGroup Server>Instance Wizard

2. Choose C:\EASYGroup\Server64 from the From drop-down menu to update the instances using 64-bit EASYGroup™ component files. Choose C:\EASYGroup\Server from the From drop-down menu to update the instances using 32-bit EASYGroup™ component files. This feature can be used to update 32-bit instances in a 64-bit environment.

Figure 6-12. Instance Wizard Window
3. If additional instances have already been created, the instances will be listed here. If you don’t see anything listed and wish to create a new instance, select **New** and the following prompt will appear.

Figure 6-13. New Instance Property Window

- **Name:**
- **Folder:**
- **Data Folder:**
- **Server Port:**
- **Run Server as Service**

4. **Next, specify a Name** for the instance (e.g., Quality Assurance, Production, Training, etc.).

5. **Browse...** for the location of the Server program and **Data** directories.

6. **Enter the Server Port** number.

7. **Choose to run the Server as an application or as an unattended Windows® service by selecting the Run Server as Service check box.**
The following example features the instance as an application.

**Note**

The name that is specified in this *New Instance Properties* window will be the name that will appear in the Windows® Services Manager.

Figure 6-14. Additional Instance Example (as an Application)
8. Select **OK** and the following prompt will appear:

Figure 6-15. Instance Wizard Window - Update Required

9. If **Update** is shown under the **Status** column, that indicates that the additional instance is out-of-sync with the primary Server installation and requires an update. For example, the primary Server has an updated version of the Ambulatory Code Editor™ (ACE) that the additional instance does not.

10. To synchronize the primary and instance Server installations, highlight the instance **Name** and select the **Copy** button.
11. The user should then see **Done** under the **Status** column, indicating that the instance is synchronized with the primary installation.

Figure 6-16. Instance Wizard Window - Update Completed

The **Status** column has the following options:

- **Done** - Indicates that the Instance Wizard has:
  - Copied all programs (i.e., extensions *.dll or *.exe) from the primary Server folder to the instance Server folder.
  - Copied all appropriate EASYGroup™ data files from the primary Server Data folder to the instance Data folder.
- **Up to date** - Indicates that the instance is in-sync with the primary Server installation.
- **Unknown** - Indicates that the instance is not supported by the primary Server installation.

**Note**
Rate Manager data files are not automatically copied to each instance. These must be manually copied to the appropriate Data directory. Refer to the Using HssSetup.ini for Synchronization section below for options with this task.
12. To check the status of all instances, invoke the Instance Wizard at any time. It will display a list of all instances and their status.

Figure 6-17. Instance Wizard Window

6.2.1.1 Using HssSetup.ini for Synchronization
The Instance Wizard utilizes a configuration file in the Windows directory titled HssSetup.ini. This file maintains a history of EASYGroup™ component installations and Server instances. Each time an additional Server instance is created, several lines are added to the HssSetup.ini file.

The Instance Wizard uses additional entries in HssSetup.ini to control synchronization. The installation program for several EASYGroup™ components update these entries. For example, if ACE has been installed, the following entries will be added:

```ini
[EASYGroup Instance Files]
Aceedit.dat=ACE Editor Data
Cciedit.dat=ACE Editor National Medicare CCI Data
Cciedt2.dat=ACE Editor National Medicaid CCI Data
CcimI.dat=Michigan Medicaid CCI Data
Copyline.txt=CPT Copyright
Copyrite.txt=CPT Copyright
Mue.dat=MUE Values
OceCci.dat=ACE Editor National Medicare OCE/CCI Data
OceCci2.dat=ACE Editor National Medicaid OCE/CCI Data
```
If you would like to automatically copy the Rate Manager data files or any other data files from the primary installation Data folder to the instance Data folder(s), you can do so by manually adding the desired files to the list in the HssSetup.ini as follows:

1. Add a new line with the filename and an equal (=) sign, followed by a brief file description.
2. If this file exists in the primary installation Data folder, it will be copied to the instance Data folder(s).

6.2.2 Command Syntax Overview

Once the Server is running, the user must set-up a TCP/IP client that can connect to the Server. This client is defined by the host name or an IP address on a specified port, and will send requests or messages to the Server for processing.

Note

The Server uses the Comprehensive Message Type (CTL) with the EASYGroup™ input and output structures.

All message types are passed in the following format: nnnnnnXXXdata, where “nnnnnn” (6 ASCII characters) represents the total length of the message not including the 6-digit byte count, “XXX” (3 ASCII characters) represents the message type, and “data” represents the input data structures. Output messages are passed in the same format as the input message.

6.2.2.1 Message Types

The following table includes basic message types and their descriptions:

Table 6-2: Message Types

<table>
<thead>
<tr>
<th>IN</th>
<th>OUT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000003LOG</td>
<td>nnnnnnLOG0 or nnnnnnLOG1</td>
<td>Toggles detailed logging</td>
</tr>
<tr>
<td>000003VER</td>
<td>nnnnnnVERxxxx</td>
<td>Returns the current Server version, where xxxx = version number</td>
</tr>
<tr>
<td>000003BYE</td>
<td></td>
<td>Does not return any data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Breaks the connection</td>
</tr>
<tr>
<td>000003RLS</td>
<td></td>
<td>Does not return any data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Releases most Server resources</td>
</tr>
<tr>
<td>Various messages</td>
<td>nnnnnnERRnnn</td>
<td>Provides the Optimizer Return Code. May append a description of the error</td>
</tr>
<tr>
<td>nnnnnnCTLnnn</td>
<td>nnnnnnCTLnnn</td>
<td>CTL request to/response from Server</td>
</tr>
</tbody>
</table>
6.2.2.2 CTL Message Type With UB-04/ICD-10 Structures

The CTL message supports all facility and physician claims for supported payment systems, including all UB-04 and National Provider Identifier (NPI) functionality, using UB-04/ICD-10 message segments. For the CTL message, data structures are separated by a vertical bar (|) (used as a separation delimiter). If a particular data structure is not used by the requested application, or if the data structure is a return or output structure (output from the Server), supply the delimiter only. Trailing spaces in each structure can be removed, as well as trailing vertical bars. Please note that in the returned structures trailing spaces are removed by the Server.

The following is a sample CTL input message for a basic inpatient request to the Server, and the resulting CTL output message response from the Server:

<table>
<thead>
<tr>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>000175CTL</td>
<td>00987CTL00 DRG 15 01N 01 24 A</td>
</tr>
<tr>
<td>09  2001100120011003 0401</td>
<td>09  2001100120011003 0401</td>
</tr>
<tr>
<td>001001 01 0002</td>
<td></td>
</tr>
</tbody>
</table>

Table 6-3: CTL Message Request/Response
7 EASYGroup™ Web Service Overview

This chapter provides an introduction to the EASYGroup™ Web Service. It contains the following sections:

• Introduction to the EASYGroup™ Web Service
  - API Model Definitions
  - Sample Code for JAVA™ and C# .NET Interfaces

• Web Service Operations
  - PostToOptimizer
  - PostToOptimizerReturnIO
  - PostToMessages
  - PostToLcdDetails
  - PostToRateDetails
  - PostToPricingWorksheet
  - PostResultsToPricingWorksheet
  - PostToAnalyzerWorksheet
  - PostResultsToAnalyzerWorksheet
  - PostToCheckVersions

• Data Formatting Conventions
  - Input Fields
  - Output Fields

• Enabling Enhanced Web Security
7.1 Introduction to the EASYGroup™ Web Service

The EASYGroup™ Web Service is a Representational State Transfer (REST)-based web service. RESTful web services are lightweight, highly scalable, maintainable, and are commonly used to create Application Programming Interfaces (APIs) for web based applications. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

The EASYGroup™ Web Service invokes all pricing, editing, grouping, and mapping via the PostToOptimizer/PostToOptimizerReturnIO methods. Additional functionality can be returned by means of other methods, which are outlined below.

- API Model Definitions

The following development tools have been added to allow easier integration for the EASYGroup™ Web Service.

- Java™ Objects

The EASYGroup™ Web Service contains Java™ Archive Files (JAR). These files define the models utilized across all methods within the EASYGroup™ Web Service and are located within the file structure: ...EZGWebService\Utilities\JavaFiles.

- Swagger Definitions

The EASYGroup™ Web Service contains Swagger Definitions. These files define the models utilized across all methods within the EASYGroup™ Web Service and are located within the file structure: ...EZGWebService\Utilities\SwaggerFiles.

Below is a list of the ways to use the files provided in the EASYGroup™ Web Service to build an interface:

Table 7-1: File Types and Usage

<table>
<thead>
<tr>
<th>File Type</th>
<th>Language</th>
<th>Usage</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>nuGet Package</td>
<td>C#</td>
<td>Consume Optum</td>
<td>jFrog Artifactory</td>
</tr>
<tr>
<td>XSD Files</td>
<td>C</td>
<td>Build C Structures From Provided XSD Files</td>
<td>gSOAP XSD.exe</td>
</tr>
<tr>
<td></td>
<td>C++</td>
<td>Build C++ Objects From Provided XSD Files</td>
<td></td>
</tr>
<tr>
<td>JAR File</td>
<td>Java™</td>
<td>Prebuilt Model Packages in JAR File</td>
<td>gSON</td>
</tr>
</tbody>
</table>
• Sample Code for JAVA™ and C# .NET Interfaces

If you are building your interface for the first time, or want to compare your code against our recommendations, Optum has provided two sets of sample code, one for JAVA™ and another for C# .NET interfaces, for guidance. This sample code goes over how to hit our PostToOptimizer endpoint using an example Medicare Inpatient and Medicare Outpatient claim.

To utilize the C# .NET sample code, you will need a target runtime of .NET Framework 4.6.1 or above in your development environment.

To utilize the JAVA™ sample code, you will need to follow the import steps noted in ImportSteps.txt provided in the root of the zip file.

The Sample Code Files (EZGWebClaimBuilderC#Example.zip and EZGWebCalimBuilderJavaExample.zip) will be available in the following locations:

• Optum Client Portal
• Regulatory Portal

Note
You will need to have the EASYGroup™ Web Service installed alongside the Medicare Inpatient Payment System and Medicare Outpatient Payment System (including rate files) to validate the example claims on your system.

### 7.1.1 Web Service Operations

The operations (methods) that comprise the EASYGroup™ Web Service are:

- PostToOptimizer
- PostToOptimizerReturnIO
- PostToMessages
- PostToLcdDetails
- PostToRateDetails
- PostToPricingWorksheet
- PostResultsToPricingWorksheet
- PostToAnalyzerWorksheet
- PostResultsToAnalyzerWorksheet

<table>
<thead>
<tr>
<th>File Type</th>
<th>Language</th>
<th>Usage</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>YAML</td>
<td>C#, Java™, PHP, Python, Scala, Typescript-Angular</td>
<td>Generate Code From Provided YAML or JSON File</td>
<td>Swagger</td>
</tr>
<tr>
<td>JSON</td>
<td>C#, Java™, PHP, Python, Scala, Typescript-Angular</td>
<td>Generate Code From Provided YAML or JSON File</td>
<td>Swagger</td>
</tr>
</tbody>
</table>

Table 7-1: File Types and Usage
• PostToCheckVersions

Table 7-2: EASYGroup™ Web Service Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>URI</th>
<th>Request Object</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostResultsToAnalyzerWorksheet</td>
<td>.../EZGWebAPI/worksheet/PostResultsToAnalyzerWorksheet</td>
<td>ClaimInputOutput</td>
<td>HTML Page</td>
</tr>
<tr>
<td>PostResultsToPricingWorksheet</td>
<td>.../EZGWebAPI/worksheet/PostResultsToPricingWorksheet</td>
<td>ClaimInputOutput</td>
<td>HTML Page</td>
</tr>
<tr>
<td>PostToAnalyzerWorksheet</td>
<td>.../EZGWebAPI/worksheet/PostToAnalyzerWorksheet</td>
<td>ClaimInput</td>
<td>HTML Page</td>
</tr>
<tr>
<td>PostToCheckVersions</td>
<td>.../EZGWebAPI/api/Versions/PostToCheckVersions</td>
<td>VersionsDetailInput</td>
<td>VersionsDetailOutput</td>
</tr>
<tr>
<td>PostToLcdDetails</td>
<td>.../EZGWebAPI/api/lcd/PostToLcdDetails</td>
<td>LcdDetailsInput</td>
<td>LcdDetailsOutput</td>
</tr>
<tr>
<td>PostToMessages</td>
<td>.../EZGWebAPI/api/messages/PostToMessages</td>
<td>ClaimInputOutput</td>
<td>ClaimInputOutput</td>
</tr>
<tr>
<td>PostToOptimizer</td>
<td>.../EZGWebAPI/api/claim/PostToOptimizer</td>
<td>ClaimInput</td>
<td>ClaimOutput</td>
</tr>
<tr>
<td>PostToOptimizerReturnIO</td>
<td>.../EZGWebAPI/api/claim/PostToOptimizerReturnIO</td>
<td>ClaimInput</td>
<td>ClaimInputOutput</td>
</tr>
<tr>
<td>PostToPricingWorksheet</td>
<td>.../EZGWebAPI/worksheet/PostToPricingWorksheet</td>
<td>ClaimInput</td>
<td>HTML Page</td>
</tr>
<tr>
<td>PostToRateDetails</td>
<td>.../EZGWebAPI/api/rate/PostToRateDetails</td>
<td>RateDetailsInput</td>
<td>RateDetailsSummary</td>
</tr>
</tbody>
</table>
7.1.1.1 PostToOptimizer

Figure 7-1. PostToOptimizer Method

The PostToOptimizer method receives claim information from the claim processing system and translates that information to the EASYGroup™ Optimizer. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI/api/Claim/PostToOptimizer Uniform Resource Identifier (URI). Examples of this information would be: diagnosis codes, procedure codes, patient age, patient sex, etc. The EASYGroup™ Optimizer in turn calls the applicable EASYGroup™ component by utilizing the Operation Code (opcode1) and Patient Type (pattype) fields to continue to process the claim and return expected results.

Table 7-3: Operation Codes (opcode1)

<table>
<thead>
<tr>
<th>Operation Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Analyze/Edit</td>
<td>Calls applicable EASYGroup™ Analyzer/Editor</td>
</tr>
<tr>
<td>11</td>
<td>Analyze/Group</td>
<td>Calls applicable EASYGroup™ Grouper, Analyzer, and Mapper, if needed</td>
</tr>
</tbody>
</table>
For an overview of grouping, pricing, editing, and/or mapping please refer to the EASYGroup™ User’s Guide.

7.1.1.2 PostToOptimizerReturnIO
The PostToOptimizerReturnIO method receives claim information from the claim processing system, translates that information, and will return the input provided to the EASYGroup™ Optimizer. The returned ClaimInputOutput object will provide the messages, labels, and input calculations where applicable. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI/api/Claim/PostToOptimizerReturnIO URI. Examples of this information would be: diagnosis codes, procedure codes, patient age, patient sex, etc. The EASYGroup™ Optimizer in turn calls the applicable EASYGroup™ component by utilizing the Operation Code (opcode1) and Patient Type (pattype) fields to continue to process the claim and return expected results.

For an overview of grouping, pricing, editing, and/or mapping please refer to the EASYGroup™ User’s Guide.

<table>
<thead>
<tr>
<th>Operation Code</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 12             | Analyze/Price | Assumes case-mix measure already assigned
|                |             | Calls applicable EASYGroup™ Analyzer and Pricer |
| 14             | Model       | Calls the applicable EASYGroup™ Grouper, Pricer, and Mapper, if needed |
| 15             | Analyze/Group/Price | Calls applicable EASYGroup™ Analyzer, Grouper, Pricer, and Mapper, if needed, plus rate files from Rate Manager |
| 16             | Analyze/Edit/Group/Price | Calls applicable EASYGroup™ Analyzer, Editor, Grouper, Pricer, and Mapper, if needed, plus rate files from Rate Manager |
| 17             | Map Only    | Calls applicable EASYGroup™ Mapper |
| 18             | Analyze Only | Calls applicable EASYGroup™ Analyzer |

<table>
<thead>
<tr>
<th>Patient Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Inpatient</td>
</tr>
<tr>
<td>02</td>
<td>Outpatient</td>
</tr>
<tr>
<td>03</td>
<td>Inpatient Rehabilitation</td>
</tr>
<tr>
<td>04</td>
<td>Physician</td>
</tr>
<tr>
<td>06</td>
<td>Skilled Nursing</td>
</tr>
</tbody>
</table>
7.1.1.3 PostToLabels

Figure 7-2. PostToLabels Method

The PostToLabels method accesses the files listed below in Table 7-5 to provide current and accurate codes and descriptions based on submitted effective dates. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI/api/Label/PostToLabels URI. It retrieves ICD-10-CM/PCS and HCPCS labels, and identifies partial codes, rubrics, and codes that are not valid as of the supplied effective date. These data files are stored as standard ASCII text files. Finally, this method is designed to support web-based interactive applications for browsing codes and code descriptions.

Table 7-5: PostToLabels Reference Sources

<table>
<thead>
<tr>
<th>File Description(s)</th>
<th>File Name(s)</th>
<th>Description</th>
<th>Update Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>APC/APG Description File</td>
<td>apgdsc.dat</td>
<td>Contains abbreviated descriptions for each Ambulatory Payment Classification (APC), Ambulatory Payment Group (APG), or Home Health Resource Group (HHRG) for every supported Grouper/Reader type and version.</td>
<td>Every time a new Grouper or Grouper version becomes available</td>
</tr>
<tr>
<td>DRG/CMG Description File</td>
<td>drgdsc.dat</td>
<td>Contains abbreviated descriptions for each Diagnosis Related Group (DRG) or Case-Mix Group (CMG) for every supported Grouper type and version.</td>
<td>Every time a new Grouper or Grouper version becomes available</td>
</tr>
<tr>
<td>HAC Description File</td>
<td>hacdsc.dat</td>
<td>Contains 100-byte descriptions for each Hospital Acquired Condition (HAC). Descriptions are retrieved based on the HAC number.</td>
<td>Various</td>
</tr>
<tr>
<td>ICD-10-CM/PCS Codes and Titles File</td>
<td>i10title.dat</td>
<td>Contains all current and previous codes, with accompanying short (abbreviated), medium (abbreviated), and long (complete) titles or descriptions. Each code is tagged with a starting and ending date which shows exactly when that particular code or version of the code was valid.</td>
<td>Every October, with a potential update in April</td>
</tr>
<tr>
<td>HCPCS/HCPCS Plus Codes and Titles File</td>
<td>hctitle.dat</td>
<td></td>
<td>Every January, April, July, and October</td>
</tr>
</tbody>
</table>
Table 7-5: PostToLabels Reference Sources

<table>
<thead>
<tr>
<th>File Description(s)</th>
<th>File Name(s)</th>
<th>Description</th>
<th>Update Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDC/RIC Description File</td>
<td>mcdsc.dat</td>
<td>Contains abbreviated descriptions for each Major Diagnostic Category (MDC) or Rehabilitation Impairment Category (RIC) for every supported Grouper type and version.</td>
<td>Every time a new Grouper or Grouper version becomes available</td>
</tr>
<tr>
<td>RUG Description File</td>
<td>rugdsc.dat</td>
<td>Contains abbreviated descriptions for each Resource Utilization Group (RUG) for Medicare skilled nursing facilities.</td>
<td>Various</td>
</tr>
</tbody>
</table>

7.1.1.4 PostToMessages

The PostToMessages method allows the client to retrieve messages for claims that have been processed without having to reprocess those claims. This method accesses the files listed below in Table 7-6 to provide current and accurate codes and descriptions based on submitted effective dates. It is accessed via an HTTP POST command to the https://localhost:443/EZGWebAPI/api/messages/PostToMessages URI. It retrieves ICD-10-CM/PCS and HCPCS labels, and identifies partial codes, rubrics, and codes that are not valid as of the supplied effective date. The label files are stored as standard ASCII and the message file is stored as XML encoded in UTF-8.

Table 7-6: PostToMessages Reference Sources

<table>
<thead>
<tr>
<th>File Description(s)</th>
<th>File Name(s)</th>
<th>Description</th>
<th>Update Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>APC/APG Description File</td>
<td>apgdsc.dat</td>
<td>Contains abbreviated descriptions for each Ambulatory Payment Classification (APC), Ambulatory Payment Group (APG), Home Health Resource Group (HHRG), or Home Health Agency Patient-Driven Groupings Model (PDG) for every supported Grouper/Reader type and version.</td>
<td>Every time a new Grouper or Grouper version becomes available</td>
</tr>
<tr>
<td>DRG/CMG Description File</td>
<td>drgdsc.dat</td>
<td>Contains abbreviated descriptions for each Diagnosis Related Group (DRG) or Case-Mix Group (CMG) for every supported Grouper type and version.</td>
<td>Every time a new Grouper or Grouper version becomes available</td>
</tr>
<tr>
<td>HAC Description File</td>
<td>hacdsc.dat</td>
<td>Contains 100-byte descriptions for each Hospital Acquired Condition (HAC). Descriptions are retrieved based on the HAC number.</td>
<td>Various</td>
</tr>
</tbody>
</table>
### Table 7-6: PostToMessages Reference Sources

<table>
<thead>
<tr>
<th>File Description(s)</th>
<th>File Name(s)</th>
<th>Description</th>
<th>Update Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICD-10-CM/PCS Codes and Titles File</td>
<td>i10title.dat</td>
<td>Contains all current and previous codes, with accompanying short (abbreviated), medium (abbreviated), and long (complete) titles or descriptions. Each code is tagged with a starting and ending date which shows exactly when that particular code or version of the code was valid.</td>
<td>Every October, with a potential update in April</td>
</tr>
<tr>
<td>HCPCS/HCPCS Plus Codes and Titles File</td>
<td>hctitle.dat</td>
<td></td>
<td>Every January, April, July, and October</td>
</tr>
<tr>
<td>MDC/RIC Description File</td>
<td>mdcdsc.dat</td>
<td>Contains abbreviated descriptions for each Major Diagnostic Category (MDC) or Rehabilitation Impairment Category (RIC) for every supported Grouper type and version.</td>
<td>Every time a new Grouper or Grouper version becomes available</td>
</tr>
<tr>
<td>RUG Description File</td>
<td>rugdsc.dat</td>
<td>Contains abbreviated descriptions for each Resource Utilization Group (RUG) for Medicare skilled nursing facilities.</td>
<td>Various</td>
</tr>
<tr>
<td>EASYGroup™ Messages File</td>
<td>ezgmsg.xml</td>
<td>Contains error message descriptions for errors such as diagnosis/procedure code errors, Return Code descriptions, etc. (e.g., 18, Optimizer Return Code, Non-Zero Return Code From Retrieve Payer Program)</td>
<td>Various</td>
</tr>
</tbody>
</table>

### 7.1.1.5 PostToLcdDetails

Figure 7-3. PostToLcdDetails Method
The PostToLcdDetails method implements medical necessity (Local and National Coverage Determination) edits by accessing the files listed below in Table 7-7. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI/api/Lcd/PostToLcdDetails URI. This method returns medical necessity errors, supporting ICD-10-CM diagnoses that will satisfy existing policies that govern the service, policy descriptions, effective dates, sex and age requirements, flags indicating if associated procedures or secondary diagnoses are required, associated procedure codes required by the policy, and ICD-10-CM secondary diagnoses required to satisfy the policy.
<table>
<thead>
<tr>
<th>File Description(s)</th>
<th>File Name(s)</th>
<th>Description</th>
<th>Update Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associated Procedure File</td>
<td>ap.dat</td>
<td>Contains code identifying the originating Fiscal Intermediary, carrier, or MAC for the LCD rules to be applied for this provider, the policy ID plus optional suffix, the date on which the associated procedure code affiliated with this policy ID was effective/terminated, and the original procedure code associated with the policy ID number.</td>
<td>Every Time a new version of the LCD Editor becomes available</td>
</tr>
<tr>
<td>HCPCS/Diagnosis Pairs File</td>
<td>i10pairs.dat</td>
<td>Contains the code identifying the Fiscal Intermediary, carrier, or MAC for the LCD rules to be applied for this provider, the procedure code, the date the procedure/diagnosis code pair was effective/terminated, the ICD-10-CM diagnosis code which will support the procedure code under the LCD rules, the policy ID plus optional suffix, and the applicable Diagnosis Support Flag.</td>
<td>Every Time a new version of the LCD Editor becomes available</td>
</tr>
<tr>
<td>Index File</td>
<td>index.dat</td>
<td>Contains the code identifying the Fiscal Intermediary, carrier, or MAC for the LCD rules to be applied for this provider, the policy ID plus optional suffix, the date the policy was effective/terminated, the patient sex, applicable age requirements, associated procedure/diagnosis flags, the long description of the policy name of the LCD, and a fully qualified link to the full LCD on the web.</td>
<td>Every Time a new version of the LCD Editor becomes available</td>
</tr>
<tr>
<td>Secondary Diagnosis File</td>
<td>i10sd.dat</td>
<td>Contains the code identifying the Fiscal Intermediary, carrier, or MAC for the LCD rules to be applied for this provider, the policy ID plus optional suffix, the date on which the secondary diagnosis code associated with this policy ID was effective/terminated, and the ICD-10-CM diagnosis code that must also be on the claim with the original procedure code associated with the policy ID number for this LCD.</td>
<td>Every Time a new version of the LCD Editor becomes available</td>
</tr>
</tbody>
</table>
7.1.1.6 PostToRateDetails

The PostToRateDetails method provides access to the grouping and pricing rules stored in the Rate Manager data files during grouping assignment and the calculation of reimbursement. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI/api/Lcd/PostToRateDetails URI. This operation invokes the Optimizer to read the rate file data from Rate Manager. Examples of input data are: facility ID and/or National Provider Identifier (NPI) and taxonomy code, payer ID, and admission date/discharge date. Examples of output data are: effective date of the most recent set of processing rules, Grouper type, Grouper version, and Pricer type. This method also provides a list of Payers set up for a specified facility. Messages are also returned for select fields.
7.1.1.7 PostToPricingWorksheet

Figure 7-5. PostToPricingWorksheet Method

The PostToPricingWorksheet method provides step-by-step detailed reimbursement calculations for various pricing (e.g., Medicare Outpatient, Medicare ESRD, Medicare Inpatient, and New York Medicaid APR) in HTML format. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI WorkSheet/PostToPricingWorksheet URI. This operation utilizes the Rate Manager data files and pricing variables accessed by the Optimizer and applicable Pricers. Refer to the table below for a complete list of pricing worksheet types and corresponding Pricer types.

Table 7-8: Pricing Worksheet and Pricer Types

<table>
<thead>
<tr>
<th>Pricing Worksheet Type</th>
<th>Pricer Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medicare Ambulatory Surgical Center (ASC)</td>
<td>55 = Medicare ASC</td>
</tr>
<tr>
<td>Medicare End Stage Renal Disease (ESRD)</td>
<td>60 = Medicare ESRD</td>
</tr>
<tr>
<td>Medicare Home Health Agency (HHA)</td>
<td>62 = Medicare HHA</td>
</tr>
<tr>
<td>Medicare Inpatient</td>
<td>A = Medicare Inpatient</td>
</tr>
<tr>
<td>Medicare Inpatient Psychiatric Facility (IPF)</td>
<td>19 = Medicaid IPF</td>
</tr>
</tbody>
</table>
Table 7-8: Pricing Worksheet and Pricer Types

<table>
<thead>
<tr>
<th>Pricing Worksheet Type</th>
<th>Pricer Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medicare Outpatient</td>
<td>h = Medicare APC-HOPD</td>
</tr>
<tr>
<td>Medicaid APG Pro</td>
<td>44 = Medicaid APG Pro</td>
</tr>
<tr>
<td>Medicaid APR Pro</td>
<td>42 = Medicaid APR Pro</td>
</tr>
<tr>
<td>New York Medicaid APG</td>
<td>45 = Enhanced New York Medicaid APG</td>
</tr>
<tr>
<td>New York Medicaid APR</td>
<td>25 = New York Medicaid APR</td>
</tr>
</tbody>
</table>


7.1.1.8 PostResultsToPricingWorksheet
The PostResultsToPricingWorksheet method allows the client to retrieve worksheets for claims that have been processed without having to reprocess those claims. This method provides step-by-step detailed reimbursement calculations for various pricing (e.g., Medicare Outpatient, Medicare ESRD, Medicare Inpatient, and New York Medicaid APR) in HTML format. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI/worksheet/PostResultsToAnalyzerWorksheet URI. This operation utilizes the Rate Manager data files and pricing variables accessed by the Optimizer and applicable Pricers. Refer to the table above (Table 7-8) for a complete list of pricing worksheet types and corresponding Pricer types. Refer to the Input & Output Parameter Blocks User’s Guide for further information.

7.1.1.9 PostToAnalyzerWorksheet
The PostToAnalyzerWorksheet method provides step-by-step detail of the evaluation of outpatient claims (Medicare, Medicaid, and Commercial) for correct coding in HTML format. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI/worksheet/PostToAnalyzerWorksheet URI. Refer to the table below for a complete list of Analyzer worksheet types and corresponding Analyzer types.

Table 7-9: Analyze Worksheet and Analyzer Types

<table>
<thead>
<tr>
<th>Analyze Worksheet Type</th>
<th>Analyzer Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDC Analyzer™</td>
<td>01 = EDC Analyzer™</td>
</tr>
</tbody>
</table>


7.1.1.10 PostResultsToAnalyzerWorksheet
The PostResultsToAnalyzerWorksheet method allows the client to retrieve worksheets for claims that have been processed without having to reprocess those claims. This method provides step-by-step detail of the evaluation of outpatient claims (Medicare, Medicaid, and Commercial) for correct coding in HTML format. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI/worksheet/PostResultsToAnalyzerWorksheet URI.
Refer to the table above (Table 7-9) for a complete list of Analyzer worksheet types and corresponding Analyzer types.

7.1.1.11 PostToCheckVersions
The PostToCheckVersions method retrieves the file names, product versions, and modified timestamps of all content within the requested directories. It is accessed via an HTTP POST command to the https://localhost/EZGWebAPI/api/Versions/PostToCheckVersions URI. This method will return a full list of all the files and associated version numbers in a requested directory. The product version will not return for data files.

For security reasons Optum has restricted the path locations. If the path does not contain:

- [drive letter]:\Optum\*
- [drive letter]:\inetpub\wwwroot\HSS\*

Clients will receive a 400 error, similar to the following:

Path [C:\Windows] is invalid; input paths must be of the form [drive letter]:\Optum\* or [drive letter]:\inetpub\wwwroot\HSS\*

Any sub-folders created in the paths mentioned above can only contain the following characters

- A - Z (capital letters)
- a - z (lower case letters)
- 0 - 9 (numbers)
- _ (underscores)
- - (dashes)
- Spaces

7.2 Data Formatting Conventions

The following data conventions are used for input and output in the EASYGroup™ Web Service.

Note

Users only need to supply what values they want to change. Default values are supplied for your convenience. Users will not need to manually enter most parameters.

7.2.1 Input Fields
Note
For all input fields, all leading and trailing spaces are removed prior to any other action.

- **Text.** Alphanumeric text string. Any trailing spaces will be removed. Maximum length is provided in the EASYGroup™ Web Service Input & Output Spreadsheet and the Input & Output Parameter Blocks Workbook. Default is blank. Fixed length text should be removed.

- **Fixed length text.** Text string, leading spaces removed, required length is provided in the EASYGroup™ Web Service Input & Output Spreadsheet and the Input & Output Parameter Blocks Workbook. If required length is not provided, spaces will be assumed on the right. Default is blank.

- **Numeric text.** Numeric text string. Required length is provided in the EASYGroup™ Web Service Input & Output Spreadsheet and the Input & Output Parameter Blocks Workbook. All case-mix indicators (DRG, MDC, APC, APG, etc.) should be treated as numeric text. If required length is not provided, zeros will be added on the left. Default is blank.

- **Integers.** Numeric fields without decimals. Leading zeros not required. All integers should be positive numbers. Maximum size is provided in the EASYGroup™ Web Service Input & Output Spreadsheet and the Input & Output Parameter Blocks Workbook. Default is zero.

- **Doubles.** Leading zeros not required. Decimal not required. If no decimal is supplied, input is assumed to be in whole dollars. If decimal is supplied, zeros will be assumed on the right if necessary (e.g., 1.00, 1.50). Maximum size for all dollar fields is defined in the EASYGroup™ Web Service Input & Output Spreadsheet and the Input & Output Parameter Blocks Workbook. Default is zero.

- **ICD-10-CM codes.** Left-justified. Decimals not required but may be submitted. Maximum length is eight bytes with decimal. Default is blank.

- **ICD-10-PCS codes.** Left-justified. Maximum length is seven bytes. Default is blank.

- **HCPCS codes.** Five alphanumeric characters. Default is blank.

- **Dates.** Eight digits, mm-dd-yyyy format.

- **Boolean.** Switch variable. Default is false.

- **Enumeration.** Field name values. For example: Code Class 01 (ICD-10 Coded Claim), Patient Type 01 (Inpatient), Operation Code 16 (Edit, Group, and Price), etc.

### 7.2.2 Output Fields

- **Text.** Any leading or trailing spaces will be removed. Maximum length provided below. A blank result will be returned as an empty string.
• **Fixed length text.** Text string (set of alphanumeric values), leading spaces removed.

• **Numeric text.** Treat as fixed length text (string of alphanumeric values). Leading zeros included. All case-mix indicators (DRG, MDC, APC, APG, etc.) will be treated as numeric text.

• **Integers.** Numeric fields without decimals. Leading zeros not included. Maximum size provided below. All error counts will be treated as integers.

• **Doubles.** Pricing operations and factors will return without leading zeros, with explicit decimals, and zero dollars or factors will be returned as zero. Maximum size for all doubles is defined in the EASYGroup™ Web Service Input & Output Spreadsheet and the Input & Output Parameter Blocks Workbook.

• **ICD-10-CM codes.** Left-justified. Decimals will be returned if supplied. Maximum length is eight bytes plus decimal.

• **ICD-10-PCS codes.** Left-justified. Maximum length is seven bytes.

• **HCPCS codes.** Five alphanumeric characters.

• **Error codes.** These will be returned as fixed length numeric fields.

• **Error messages.** These will be returned as text fields. Format of error messages includes the error code, a space and a comma, plus a text description. Error codes will be numeric text fields, with a required length. For the descriptions, no maximum length is specified.

Error messages are required only if the appropriate flags have been set within the EASYGroup™ Web Service. Types of error codes include: diagnosis/procedure code errors, Return Code descriptions, etc. For example, 18, Optimizer Return Code, Non-Zero Return Code From Retrieve Payer Program. Labels are retrieved from the ezgmsg.xml file, which is included in the distribution in located in the EZGWebAPI/EZGFiles directory.

• **Negative numeric or decimal fields.** These will be returned with an explicit negative sign (e.g., -450.25).

• **Enumeration.** Field name values. For example: Label Code Type 01 (APC).

**Note**
For further information please refer to the EASYGroup™ Web Service Input & Output Spreadsheet and the Input & Output Parameter Blocks Workbook.

**7.2.3.1 Custom Formatting for JSON Objects**
The EASYGroup™ Web Service supports the EASYGroup™ formatting conventions when requests are sent using Content-Type: application/json. You can choose to use the EASYGroup™ conventions when sending input and output, rather than the standard JSON conventions.
**Note**

The following related to input is for the PostToOptimizer, PostToOptimizerReturnIO, PostToRateDetails, PostToLCDDetails, PostToMessages, PostToPricingWorksheet, PostResultsToPricingWorksheet, PostToAnalyzerWorksheet, and PostResultsToAnalyzerWorksheet methods.
To choose to use EASYGroup™ input conventions, you must turn On CustomFormatSupport in the web.config file.

**Note**

The following related to output is for the PostToOptimizer, PostToOptimizerReturnIO, PostToRateDetails, PostToLCDDetails, and PostToMessages methods.

To choose to use EASYGroup™ output conventions, you must set the EzgControl.output_format output field to a 1 (JsonEZGStrings).

If you decide to use the EASYGroup™ conventions, the following formats will be utilized:

- **Dates.** Without dashes (e.g., yyyyMMdd).
- **Integers.** With leading zeros and within quotes (e.g., “025”).
- **Doubles.** With leading zeros, trailing zeros, and within quotes (e.g., “000012599”).
- **Enumerations.** With quotes as their generic value (e.g., “16”).
- **Booleans.** “0” for false and “1” for true.

### 7.3 Enabling Enhanced Web Security

**Note**

Please note that the below instructions are only for Microsoft® Windows® authentication.

If you wish to enable Internet Information Services (IIS) Windows® Authentication for the EASYGroup™ Web Service and the Tester you may do so by following the below instructions. Once these features are enabled, you can test these features by using the Tester. For example, you will be prompted to log-in with a user name and password when launching the Tester. Once logged in, the same credentials will pass to the EASYGroup™ Web Service. The Windows® Authentication feature will allow you to be connected immediately to the Tester if you are already logged into Microsoft® Windows®. If set-up correctly, an interface to the EASYGroup™ Web Service should behave the same.

**Note**

These instructions are optional, but Optum recommends securing the EASYGroup™ Web Service if being hosted via Intranet or Internet.

1. Navigate to **Start>Control Panel> Program and Features**.
2. Select **Turn Windows features on or off** from the menu on the left-hand side of the screen.
3. You should now see the **Windows Features** screen. In this screen, select the check box next to **Internet Information Services** (IIS) and expand the options, as shown below.
4. Check the box next to **Windows Authentication**.

**Note**
The system may need to be re-started for the setting to take effect.

5. Select **OK** when complete.

6. Open the IIS Manager: **Start>Control Panel>Administrative Tools>Internet Information Services (IIS) Manager.**
7. Once you have the IIS Manager screen open, select **Authentication** from the main window.

8. You should now see the **Authentication** window.
9. In this window, disable **Anonymous Authentication** and enable **Windows Authentication** as shown above.

10. Once you have completed the above step, the service will require a Windows® user name and password.

Figure 7-10. Log-in Dialog Box

**Note**
Your credentials will automatically pass from the Tester to the EASYGroup™ Web Service. The same authentication measures should function with your interface, as well. A non-authenticated communication will result in a 401-Unauthorized response from the service or the Tester.

7.3.0.1 Enabling Windows® Authentication in Windows® 2012 and Windows® 2016

1. Select **Manage>Add** Roles and Features in Server Manager:
2. Select the appropriate server using the **Server Selection** option on the left-hand side of the screen.

3. Once the appropriate server has been chosen, select **Server Roles** on the left-hand side of the screen and scroll to the **Web Server** section.

4. Expand the **Web Server** section.

5. Expand the **Security** section.

6. Check the **Windows Authentication** check box under **Security**.

7. Select **Install**.

8. Skip to step 6 on page 86 above.
Figure 7-12. Select Server Roles Screen
8 EASYGroup™ Web Service Tester Overview

This chapter provides an introduction to the EASYGroup™ Web Service Tester (a.k.a., “the Tester”). It contains the following sections:

- Tester Main Functions/Screens
  - Optimizer Tab
  - Labels Tab
  - Rate Details Tab
  - LCD Details Tab
- Tester Hot Keys
8.1 Tester Main Functions/Screens

The Tester is an interactive tool that can be used to test the EASYGroup™ Web Service directly; meaning outside of any interface or application. The Tester can also be used as a guide when writing a new interface to the EASYGroup™ Web Service. By default, the Tester is available in the following location: https://localhost/EzgWebTester/. The Tester includes the below four tabs on the right-hand side of the screen:

- Optimizer (used to test the PostToOptimizer method)
- Labels (used to test the PostToLabels method)
- Rate Details (used to test the PostToRateDetails method)
- LCD Details (used to test the PostToLcdDetails method)

![Method Tabs](Image)

8.1.1 Optimizer Tab

The Optimizer tab can be used to test the PostToOptimizer method. Claims can be entered on this tab and submitted to the EASYGroup™ Web Service to obtain editing, grouping, mapping, analyzing, and pricing results.

**Note**

For further information on all of the specific fields and subsequent values available for the Optimizer tab please refer to the Input & Output Parameter Blocks User’s Guide.

1. If the user selects the Optimizer tab on the Tester main screen, they can set an Operation Code (`opcode1`) (i.e., EditGroupPrice, GroupPrice, Edit, Analyze, Group, etc.), choose a corresponding Patient Type (`pattype`) (i.e., Inpatient, Outpatient, Physician, etc.), and enter the claim data that they wish test.

2. Once all of the claim data has been entered, select Submit to show results on the right-hand side of the screen. Results can be viewed in tabular format on the Table tab, in JSON format on the JSON tab, and/or in XML format on the XML tab. Examples of output in all three formats are shown below.

**Note**

Output can be viewed under all tabs, in a simplified tabular format via the Table tab, in JSON via the JSON tab, and in XML via the XML tab.
Figure 8-2. The Tester Main Screen

Figure 8-3. Example of JSON Results
Figure 8-4. Example of XML Results

```xml
<?xml version="1.0" encoding="utf-8"?>
<Root>
  <Analyzer/>
  <Decoder/>
  <Doc/>
  <EasyEdit/>
  <Led/>
  <Physician/>
  <Edit/>
  <Group/>
  <Ape/>
  <AlertFlag/>
    <AlertFlag/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
    <AlertHelp/>
There are options to **Load** previously saved data, **Submit** the data for output, and **Clear** the form, all of which can be found at the top of the Input section (as shown below).

In addition, to allow clients to set logging from the Tester, a **log_req** check box is available in the Input section. When selected, the Log Request field will be set to 1 (Log File Requested). Since this log file should only be utilized on a single claim basis, the **log_req** check box will default to unchecked when the Submit or Clear buttons are selected.

### 8.1.1.1 Load/Save/Submit ECM Pro™ Input Data

The load functionality accepts ECM Pro™ Web Service formatted input data. This input data can be converted into the EASYGroup™ Web Service.
format(s) (using XML or JSON). To load/save/submit input data users must follow the below steps:

1. Go to the **Input** tab and paste the input data (XML, JSON, or ECM Pro™ format) into the empty box.
2. Select the **Load** button.
3. The data will be loaded and the **Table** tab will appear.
4. Select the **Submit** button to view the corresponding table output results on the right side of the screen.
5. To view/save the input data in XML or JSON formats, go to the **XML** tab or **JSON** tab.
6. Select the **Submit** button to view/save the corresponding XML or JSON output results on the right side of the screen.

### 8.1.2 Labels Tab

The **Labels** tab can be used to test the PostToLabels method. Claims can be entered on this tab and submitted to the EASYGroup™ Web Service to obtain the label types listed in Table 8-1 below to provide current and accurate codes and corresponding descriptions based on submitted effective dates. It will also identify partial codes, rubrics, and codes that are not valid as of the supplied effective date.

1. By selecting the **Labels** tab, users can view labels they wish to test by selecting the plus (+) and minus (−) signs next to the **labelcodetype** column.

#### Table 8-1: Label Types

<table>
<thead>
<tr>
<th>Label Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APC</td>
<td>Text descriptions</td>
</tr>
<tr>
<td>APG</td>
<td>Text descriptions</td>
</tr>
<tr>
<td>CMG</td>
<td>Text descriptions</td>
</tr>
<tr>
<td>CPT®-4 and HCPCS Level II Codes</td>
<td>Short or long labels</td>
</tr>
<tr>
<td>CPT®-4 and HCPCS Level II Modifiers</td>
<td>Short or long labels</td>
</tr>
<tr>
<td>DRG</td>
<td>Text descriptions</td>
</tr>
<tr>
<td>HHRG</td>
<td>Text descriptions</td>
</tr>
<tr>
<td>Hospital Acquired Conditions (HAC)</td>
<td>Text descriptions</td>
</tr>
<tr>
<td>ICD-10-CM/PCS Diagnosis and Procedure Codes</td>
<td>Short, medium, and long labels</td>
</tr>
<tr>
<td>MDC</td>
<td>Text descriptions</td>
</tr>
<tr>
<td>RIC</td>
<td>Text descriptions</td>
</tr>
<tr>
<td>RUG</td>
<td>Text descriptions</td>
</tr>
</tbody>
</table>
2. Once the label information has been entered, select **Submit** to show results.

Figure 8-6. Labels Tab

<table>
<thead>
<tr>
<th>APC</th>
<th>S16</th>
<th>02/20/16</th>
<th>02/20/19</th>
<th>02/20/19</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRG</td>
<td>R73</td>
<td>02/20/16</td>
<td>02/20/19</td>
<td>02/20/19</td>
</tr>
<tr>
<td>MAC</td>
<td>D84</td>
<td>02/20/16</td>
<td>02/20/19</td>
<td>02/20/19</td>
</tr>
<tr>
<td>SO50X</td>
<td>N20</td>
<td>02/20/16</td>
<td>Medium</td>
<td>02/20/16</td>
</tr>
<tr>
<td>HCPCS</td>
<td>31</td>
<td>02/20/16</td>
<td>02/20/16</td>
<td>02/20/16</td>
</tr>
<tr>
<td>HCPCS</td>
<td>01901</td>
<td>02/20/16</td>
<td>02/20/16</td>
<td>02/20/16</td>
</tr>
</tbody>
</table>

8.1.3 Rate Details Tab

The **Rate Details** tab can be used to test the PostToRateDetails method. Claims can be entered on this tab and submitted to the EASYGroup™ Web Service to obtain access to the grouping and pricing rules stored in the Rate Manager data files during grouping assignment and the calculation of reimbursement. This operation invokes the Optimizer to read the rate file data from Rate Manager. Examples of input data are: facility ID and/or National Provider Identifier (NPI) and taxonomy code, payer ID, and admission date/discharge date. Examples of output data are: effective date of the most recent set of processing rules, Grouper type, Grouper version, and Pricer type. This method also provides a list of payers set up for a specified facility.

1. Under the **Rate Details** tab, select the appropriate check box (**getraterules** or **getpayers**).
2. Next, select the appropriate Patient Type (**pattype**) and Effective Date (**effdate**).
3. Enter the Facility ID (**facility**), National Provider Identifier (**npi**), Taxonomy Code (**taxonomy**), Pay Source (**paysrc**), and Rate Path (**ratepath**) in the corresponding text boxes.
4. Select **Submit** to show results.
Important
If you enter a NPI and leave the taxonomy field empty, the Tester will return results for all taxonomies associated to that NPI. If you enter a NPI as well as a taxonomy, the Tester will return only the results associated with that taxonomy.

8.1.4 LCD Details Tab
The **LCD Details** tab can be used to test the PostToLcdDetails method. Claims can be entered on this tab and submitted to the EASYGroup™ Web Service to implement Medical Necessity (Local and National Coverage Determinations (NCDs)) edits. They are designed for outpatient cases only. In addition, this method will perform the following tasks:

- Returns the supporting ICD-10-CM diagnoses that will satisfy existing policies that govern the selected service.
- Returns policy description, effective dates, sex and age requirements, as well as flags indicating if associated procedures or secondary diagnoses are required.
- Returns associated procedures (i.e., HCPCS codes) required by the selected policy.
- Returns the ICD-10-CM secondary diagnoses required to satisfy the selected policy.

1. Under the **LCD Details** tab, users can enter all or some of the following criteria, depending on the desired outcome:
   - Medicare Administrative Contractor (MAC) Identifier (**macid**)
- Policy Identifier (policy_id)
- Procedure Code (hcpcs)
- Effective Date (date)
- Code Class (code_class)
- User Path (userinfo)

2. Select Submit to show results.

Figure 8-8. LCD Details Results

8.2 Tester Hot Keys

Table 8-2: Tester Hot Keys

<table>
<thead>
<tr>
<th>Function Key</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + Shift + Enter</td>
<td>Submit</td>
</tr>
<tr>
<td>Enter</td>
<td>Add Row</td>
</tr>
<tr>
<td>F2</td>
<td>Load Claim</td>
</tr>
<tr>
<td>F8</td>
<td>Save Claim</td>
</tr>
<tr>
<td>Shift + Enter</td>
<td>Expand/Collapse Widget</td>
</tr>
<tr>
<td>Shift + Tab</td>
<td>Move to Previous Field</td>
</tr>
<tr>
<td>Space Bar</td>
<td>Check/Uncheck Box</td>
</tr>
<tr>
<td>Tab</td>
<td>Move to Next Field</td>
</tr>
</tbody>
</table>
9 Optum Exchange PPS (OEPPS) Overview

This chapter provides an introduction to the OEPPS. It contains the following sections:

- Overview of the OEPPS Application
  - OEPPS Features & Functionality
  - File Watch Folder
  - OPTXML
  - OPTDLL
  - Overview of OEPPS Processing
- OEPPS Dependencies
9.1 Overview of the OEPPS Application

OEPPS is a Windows® Service that acts as a portable software library that supports the large scale implementation of prospective payment systems on Windows®-based platforms. The Windows® Service is written in Microsoft® C# and distributed to run on Microsoft's® .NET 4.6 Framework. The available operations access the underlying EASYGroup™ components, including Groupers for case-mix classification, Pricers for expected reimbursement calculations, Mappers for translation of ICD-9-CM and ICD-10-CM/PCS codes across fiscal periods, and Editors for validating inpatient and outpatient data, including ICD-10-CM/PCS and HCPCS/CPT® codes for specific time periods. These services are designed to be used by third-party applications such as record abstracting or billing systems, to implement coding, editing, and prospective reimbursement functionality.

OEPPS is comprised of three main components:

1. **OPTUMExchangePPSService**: A Windows® Service that runs the software. This Service module monitors and administers all incoming processing requests.

2. **OEPPS User Interface (UI)**: Used to configure and manage the behavior of OEPPS, and to establish the resources used in batch processing.

3. **OEPPS Processor**: Used to process files executed through the service (OPTUMExchangePPSService) or application (OEPPS UI).

OEPPS was designed to provide a flexible mechanism for integrating the prospective payment system methodologies by eliminating the requirement of building an unique, environment-specific driver that conforms to the Optum proprietary input/output structures found within EASYGroup™. OEPPS allows clients to leverage their own input/output layouts, and performs needed data manipulation to call EASYGroup™ and provide accurate results.

OEPPS has the capability to process multiple instances to either the limitation of the system or the limitation configured within the Exchange PPS Config screen. This enables parallel processing, so that multiple files can be processed or multiple pieces of the same file can be processed at the same time. Additionally, OEPPS has the capability to process multiple claims within each file simultaneously through multi-threading. With these features, OEPPS scales performance to the capabilities of the system.

OEPPS includes a function to map a client’s unique data format to existing EASYGroup™ input/output variables, thereby providing significant flexibility during integration.

**Note**

Please refer to Chapter 11 for further information on Mapping.
9.1.1 OEPPS Features & Functionality

The OEPPS application has the following features and functionality:

- Runs as an interactive application or in the background automatically as a Windows® Service.
- Supports multiple claim folders, each with it's own configuration.
- Can accept flexible and customizable data layouts.
- Able to return output data using a separate layout.
  - These layouts are comprised of only the fields the client is interested in using.
- Locate and move remote files for processing from a FTP server.
- Can process multiple files at a time

9.1.2 File Watch Folder

The File Watch folder is the folder that your downstream application places data files filled with claims and also a trigger file which tells OEPPS that a data file is ready for processing. After processing the file, OEPPS places output files and also a ready file which tells your downstream application that a file has completed processing in the Out folder.

OEPPS includes a remote file mover utility which can optionally be used to move input and output files across client servers using FTP or UNC.

**Note**

It is recommended that all processing are done within the sub directories.

9.1.3 OPTXML

This directory processes claim files which are mapped via the XML files created in the Mapping Utility. A trigger in pattern is constantly looked for by the Optum Exchange File Watch Process, which kicks off the processing engine once found.

9.1.4 OPTDLL

This directory processes claim files which are mapped via legacy DLL files custom built for clients. This feature is not supported for new clients. A trigger in pattern is constantly looked for by the Optum Exchange File Watch Process, which kicks off the processing engine once found.
9.1.5 Overview of OEPPS Processing

Figure 9-1. OEPPS Processing

The above diagram illustrates the OEPPS process flow and outlines the following steps:

1. OEPPS monitors a location for the presence of claims data.
2. Clients move the data to be processed to a monitored location (network or local drive).
3. OEPPS processing is initiated when the data is discovered.
4. The data is converted to the EASYGroup™ input structures.
5. Claims are processed by EASYGroup™.
6. Processed claims data is converted back to the client-specific format.
7. OEPPS moves processed data back to the original location (network or local drive).
8. Log files are generated with claim edit and error information.
9. Client’s downstream process is typically initiated at this point.
10. Data can be manually or automatically imported to a data warehouse or third party application.

9.2 OEPPS Dependencies

9.2.1 EASYGroup™ Components
OEPPS invokes the EASYGroup™ Optimizer, which in turn invokes the appropriate EASYGroup™ components (Groupers, Pricers, etc). EASYGroup™ components are separately licensed modules.

9.2.2 Rate Manager
The EASYGroup™ Optimizer references external data files that contain processing rules for various facilities, payers, and effective dates. These files are created and maintained using Rate Manager, which is installed separately. Procedures for installing and using Rate Manager are detailed in the Rate Manager Installation Guide. Rate Manager provides a user-friendly front-end to collect, edit, and maintain the facility and payer-specific variables.
10 OEPPS Configuration and Utilization

This chapter provides configuration information for the OEPPS application. It contains the following sections:

- Launching the OEPPS Application
- OEPPS User Interface (UI)
  - Home/ File Watcher Status Screen
  - Main Menus
- File Watch Mover (FWM) Utility
10.1 Launching the OEPPS Application

Launch the OEPPS application by selecting the `OEPPS.exe` icon located on your local machine or navigate to the installation path. The default installation path is:

```
C:\Optum\ExchangePPS\n```

**Note**

When OEPPS is in use, users may select the Optum icon located in the System Tray to re-open the UI. Users will not see OEPPS in the Task Bar while it is in use.

The user will see the following screen:

Figure 10-1. Initial Start-Up Screen
10.2 OEPPS User Interface (UI)

The OEPPS UI utilizes a user-friendly menu layout for ease of access to all common functions. OEPPS launches to the Home/File Watcher Status screen.

The Status, Settings, Admin, and Help menus each have drop-down selections. To access these selections hover the mouse over the applicable menu to show the associated drop-down items.

Figure 10-2. Main Menus

<table>
<thead>
<tr>
<th>Home</th>
<th>Status</th>
<th>Settings</th>
<th>Admin</th>
<th>Help</th>
</tr>
</thead>
</table>

10.2.1 Home/File Watcher Status Screen

The Home/File Watcher Status Screen acts as the main dashboard for OEPPS, where users can view the current state of the Service and the status of any ongoing file processing.

Users should take note of the File Watcher path. This shows where OEPPS is currently set to look for new files to process. This was set during the installation, but it can also be changed at any time under the Application option within the Settings Menu. The File Watcher path must be local to where OEPPS is installed. To map to other network locations, use the File Watch Mover (FWM) Utility.

At the top of the screen there are two buttons. Start Watching which will launch the watching process, and Start Processing which will launch the processing service. Beneath the Start Watching and Start Processing buttons there are three sections:

- Files in Processing List
- Files in Waiting List
- Files in Completed List

When matching data files and associated trigger files are placed in the File Watch folder, a running OEPPS will show the file name in the Files in Waiting List. When OEPPS begins processing the file, you will see the file move from the Files in Waiting List to the Files in Processing List. This indicates that the data file is currently being parsed and handled by the Service. When the output data is ready, you will see the file appear in the Files in Completed List.
10.2.2 Main Menus

10.2.2.1 Status Menu
Select **Status>File Watcher** to navigate to the **Home/File Watcher Status Screen** from any place in the application.

![Status Menu](image)

Please refer to the **Home/File Watcher Status Screen** section for further information on this screen.

10.2.2.2 Settings Menu
The **Settings** menu has two options:
- Application
- File Watcher

![Settings Menu](image)

Selecting **Settings>Application** will bring the user to all the configurable options for OEPPS. Each line consists of a **Key** and a **Value**.
The following options are available:

Table 10-1: Configuration Options

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataInFilePattern</td>
<td>Defines what data file extension/suffix to look for in the File Watch folder. The presence of a trigger file and a matching data file signals to OEPPS that the data file is ready for processing. The default data file extension is .dat. Suffixes should be prefixed with an asterisk (*).</td>
</tr>
<tr>
<td>DataOutFilePattern</td>
<td>Defines what data file extension/suffix is appended to files after processing. The default data file extension is .out. Suffixes should be prefixed with an asterisk (*).</td>
</tr>
<tr>
<td>DeleteFilesOlderThanDays</td>
<td>Sets the number of days to keep processed files. After the set number of days files will be automatically deleted from the out folder. Default value is 9999.</td>
</tr>
<tr>
<td>LogLevel</td>
<td>This setting changes the logging detail in the OPTXML File Watch directories which contains claim files to process. Acceptable values are 0 (no detail) - 4 (highest level). Default is 0.</td>
</tr>
<tr>
<td>MaxBatchThreadCnt</td>
<td>Sets the number threads OEPPS creates for each claim file. Acceptable values are 1 (no threading) - 10. Default value is 10.</td>
</tr>
</tbody>
</table>
Selecting Settings>File Watcher will bring the user to the File Watcher Configuration screen. In this screen, the user can adjust the configuration settings for the File Watch Utility. The path should point the Config folder within the File Watch OPTDLL or OPTXML directories (which was set during installation). If it does not, select the Browse button to re-set.
Selecting **Load File** with a valid path to the **Config** folder will show all the File Watcher configuration elements and their current settings. Expand each by selecting the respective arrow.

The File Watcher Configuration File screen is broken out into three separate sections (as shown above):

- **PPS Optimizer Directories**
- **Remote File Watch Directories**
- **PPS Default Values**

Default values are established for certain fields on the File Watcher Configuration File screen (as shown in Figure 10-7) to assist clients and create a more user-friendly experience. In order for clients to utilize these
default values they must add values to the Configuration File where the default values should be considered for processing.

The very first value in the **PPS Default Values** section: **Override Values in Claim File** defines whether the value in the claim file or the value in the Configuration File should be used during processing. By default (false), the claim file value will be used. Turning this on (true) will change the value within the claim file to that in the Configuration File for every claim processed through the directory. If a value is blank, the other will be utilized, regardless of the setting.

Other important fields to be aware of are:

- **PATHS_System** - The value of this element should be the path to your system, for example: `C:\inetpub\wwwroot\HSS\Data`.
- **PATHS_Rates** - The value of this element should be the path to your **Rate File** folder, for example: `C:\inetpub\wwwroot\HSS\Data`.
- **PATHS_User** - The value of this element should be the path to your **User File** folder, for example: `C:\inetpub\wwwroot\HSS\Data`.
- **Optimizer64Path** - The value of this element should be the path to your local **Optimizer** folder(s), for example: `C:\inetpub\wwwroot\HSS\Optimizer64`.
- **RemotInputDirectory**
- **RemoteOutputDirectory**

When users select a valid path in the **RemotInputDirectory** or the **RemoteOutputDirectory**, the FWM utility looks for files in that location. Each directory can be configured to have it’s own input/output directory. The FWM utility then sees a trigger input or output pattern in the directory, defined by the Configuration File, and pulls that file into the local File Watch directory to be processed. No file watch is enabled when either no paths are available or the paths are invalid. If a path is added or changed during the file moving process, the file will be moved to the new or adjusted path.

Users are also able to have multiple file watch locations point to the same remote directories. The Configuration File can have the same **RemotInputDirectory** or **RemoteOutputDirectory**. If the files share the same location, the input files will be copied to all applicable directories before the trigger pattern is broken and processed. The output files will be renamed (a single underscore from the root processing directory, and a single underscore between the folder and file name) according to the folder where it was processed.
Examples:

Table 10-2: Examples of Input/Output Renaming Process

<table>
<thead>
<tr>
<th>File Name</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>testdata.dat</td>
<td>OPTXML</td>
<td>OPTXML_testdata.out</td>
</tr>
<tr>
<td>testdata.dat</td>
<td>OPTXML\APC</td>
<td>OPTXML\APC_testdata.out</td>
</tr>
<tr>
<td>testdata.dat</td>
<td>OPTXML\APG</td>
<td>OPTXML\APG_testdata.out</td>
</tr>
</tbody>
</table>

Note
Clients should add these configurations if they do not appear on the screen. Users should contact Optum Client Services if they cannot locate the above-mentioned folders.

Important
Before modifying the additional fields on the File Watcher Configuration screen it is strongly recommended that clients contact Optum Client Services. Adjusting these fields can make the application unstable.

10.2.2.3 Admin Menu
The Admin menu has two options:

- Folder Structure
- Mapping

Note
To learn about the Mapping option please refer to Chapter 11.

Figure 10-8. Admin Menu

The Folder Structure screen has several important features and functions. The user will see two tabs on this screen:

- OPTDLL
- OPTXML

These two tabs allow the user to easily transition between the two primary methods of processing in OEPPS. The OPTDLL tab contains the Folder
Structure tools for setting up File Watch directories for the legacy Custom Driver Solution (CDS) mapping functionality. The OPTXML tab houses the File Watch folder functionality for the XML-based Mapping Utility functionality. The OPTXML tab is used in conjunction with the Mapping Utility to create XML layouts within the UI for the processing engine.

The paths listed on the top left-hand side of these tabs should show these two main File Watch directories. If these paths are incorrect refer to the Application option under the Settings Menu for the parent File Watch directory defined by the WatchFolder element.

Figure 10-9. Folder Structure Paths

OEPPS is designed to process claims with a variety of custom mappings, groupings, and rates. Users may require multiple mappings or rates for processing, which can be handled simply by using the UI. By creating separate File Watch folders, users can specify unique processing modes.

To Create a New File Watch Folder Within the Existing Directories:

1. Type in a new name in the file File Watch Folder Name text box.
2. If needed, select the **Use Folder Name For Grouper Settings** check box to have OEPPS to look for Grouper settings that match the given name. This will enable the **Grouper Type** and **Grouper Version** drop-down lists.

3. If using folder specific Grouper settings, use the drop-down lists to select the desired **Grouper Type** and **Grouper Version**. The appended name should not be altered, otherwise OEPPS will be unable to apply the correct Grouper type and Grouper version.

**Note**  
Grouper Type and Version selection is dynamic and reads from the EASYGroup™ Optimizer directory (this path is defined in the **Settings>File Watcher** screen) to only display the installed Grouper types and versions.
4. If there are rate files specific to the data to be processed in this new folder, check the **Use Local Rate Files** check box. This will copy all `.dat` files within the **PATHS_Rates** variable defined by the source path.

5. If custom field mapping is required for claims in these folders, select the appropriate files from the **Mapping DLL** (to apply a `.dll` file) (**OPTDLL** tab) or **Mapping Input XML/Mapping Output XML** (to apply `.xml` file(s)) (**OPTXML** tab) drop-downs. If users select the **Separate Input/Output Layouts** option on the **OPTXML** tab they can apply two separate XML files (one for input and one for output).
Note

OPTXML directories require an XML file to be set before the directory can be created.

Two rows for the same File Watch directory will display when separate input/output folders are chosen.

If one XML file handles both input and output, the Separate Input/Output Layouts check box should be unchecked.

6. When all the desired options have been chosen, select the Save button to create the new File Watch folder(s). Users should then see them listed in the File Watch Folder Path list(s).

Figure 10-15. Newly Created File Watch Folder - OPTDLL Tab

Figure 10-16. Newly Created File Watch Folder - OPTXML Tab

On the OPTXML tab, users will see an Update button when the version of the XML mapping file located in the OPTXML child directory (with the same name) has been updated. When Update is selected, the XML mapping file located in the path set in the MappingXmlDir element on the
Settings>Application screen will replace the file located in the specified OPTXML directory the user selects.

Figure 10-17. OPTXML Update Option

10.2.2.4 Help Menu
The Help menu has two options:
- References
- About

Figure 10-18. Help Menu

The References screen shows available documentation for OEPPS. To launch a document, hover and select. Online documents will launch in a browser window and require an Internet connection.

The About screen shows an informative pop-up displaying the name of the application and the current version installed on the local machine.
10.3 File Watch Mover (FWM) Utility

The FWM utility has been created and replaces the previous Remote File Watch utility. The FWM utility allows users to:

- Move claim files and output files back-and-forth between directories.
- Remotely monitor a directory that is not your File Watch directory, and move identified files to your File Watch directory for OEPPS processing:
  - Multiple remote directories may be monitored if needed.
  - Multiple File Watch directories can be used if needed.
11 OEPPS Mapping

This chapter provides details on the available Mapping functionality within the OEPPS application. It contains the following sections:

- EASYGroup™ Input/Output Panel
  - Searching
  - Adding Structures/Fields
  - Grouping
  - Adjusting Standard EASYGroup™ Date Formats
  - Specifying Occurrences
  - Hover Help
- XML File Layout Panel
  - XML File Preview
  - Right-Click Options
  - Additional Options
- Helpful Hints
11.1 Mapping

**Note**
When customizing the EASYGroup™ input/output variables users may want to have the Input & Output Parameter Blocks User’s Guide available for reference. This document is available on the [Optum Client Portal](#) and/or the [Regulatory Portal](#).

Please note OEPPS only functions on Windows®/C Platforms.

The **Mapping** screen allows users to create an XML file containing customized input/output parameter block structures (from EASYGroup™). In this mapping file, users are able to create custom fields, nicknames for existing EASYGroup™ fields, re-order fields, adjust field length, set occurrence rates, and arrange dates in different orders, etc.

**Note**
Fields must be less than or equal to the recommended EASYGroup™ length. XML mapping files will receive error code 1109 (Error Truncation Occurred) if the field length is greater than the length of the original EASYGroup™ field.

The Mapping Utility will highlight field lengths that are different than the recommended value. Please refer to the Input & Output Parameter Blocks User’s Guide for the recommended field length.

Navigate to Admin>Mapping to open the Mapping screen. Users will see two panels. The left-hand sidebar panel is a list of EASYGroup™ input/output parameter structures. The panel to the right is the layout of the XML mapping file that users can create with customized input/output fields.

Users can create one mapping file with fields from different EASYGroup™ structures. For example, a user can create a single layout with fields from the ECB [ezg_cntl_block] structure as well as the PCB1 [patient_claim_data] structure.
Figure 11-1. Mapping Screen
11.1.1 EASYGroup™ Input/Output Panel

The EASYGroup™ Input/Output Panel shows all of the input/output EASYGroup™ structures, as well as any custom fields the user creates. Users can toggle between Input, Output, and Input/Output by using the drop-down next to the Search text box.

Figure 11-2. EASYGroup™ Input/Output Panel
11.1.1.1 Searching
Users may search existing fields by typing search criteria into the Search text box. Users may use the entire name or a partial name to search. As you type you will see the applicable structures in the panel below.

11.1.1.2 Adding Structures/Fields
In the EASYGroup™ Input/Output Panel the user can add any fields from existing EASYGroup™ structures, as well as any Custom Fields (created by the user) to the mapping file by selecting the plus (+) sign next to the appropriate field.

To expand each structure simply select the caret on the right-hand side of the panel.

To minimize a structure simply select the caret again. User’s can also add an entire EASYGroup™ structure by selecting the plus (+) sign next to the structure name:
11.1.1.3 Grouping

- Grouping Multiple Structures

Users have the ability to group certain structures together that vary based on a reoccurring value or set of values by following the below steps:

1. Right-click the structure in the EASYGroup™ Input/Output Panel (far left panel) you wish to have grouped with another structure.

2. A **Group With** dialog box will appear. Choose two or more structures you would like to have grouped as shown in Figure 11-7.

**Note**

If users right-click the structure and the **Group With** dialog box does not appear, that structure cannot be grouped.

3. Once you have selected the groups, you can left-click to perform grouping.

4. To view the group, select the arrow of the first field in the grouped structure as shown in Figure 11-8.

5. The fields will now occur sequentially rather than subsequently across the multiple structures specified.

6. Right-click any structure that is part of the group and de-select the check box to remove the structure from the group.

- Grouping Individual Structures

Users have the ability to group certain fields together that vary based on a reoccurring value or set of values by following the below steps:

1. Add fields or occurrence values to the mapping layout of the recurring structure.

2. Right-click the field and select **G** (Group).

3. The field will now occur sequentially rather than subsequently.

4. To remove the grouping, simply right-click the field and select **E** (De-Group).
Figure 11-7. Group With Dialog Box
11.1.1.4 Adjusting Standard EASYGroup™ Date Formats

Users can re-arrange certain date fields (listed below):

1. By using the applicable date fields:
   - **Pcb1**: **UB-04 Occurrence Dates**
   - **Pcb1**: **UB-04 Occurrence Span Date #1**
   - **Pcb1**: **UB-04 Occurrence Span Date #2**
   - **Op**: **Procedure Date**
   - **Line**: **Date of Service**
   - **Leb_Op**: **Policy Original Effective Date**
   - **Leb_Op**: **Last Update Date**
Note
Make sure the selected fields are not in a group.

2. If the occurrences for the field or structure are greater than one.
3. The **Occur in Parts** radio button at the bottom of the **Date Format** section on the far right-hand side of the screen is available.

Figure 11-9. Example of Adjusting Standard EASYGroup™ Date Formats

4. Checking the data at the bottom of the screen will verify if the format matches a desired or existing claim file format.
5. Save the mapping layout by selecting the **Save** or **Save As** buttons.

11.1.1.5 Specifying Occurrences
Users can choose how to order fields which occur within a structure occurrence (i.e., the structure also repeats). Users can choose whether they would like each occurrence to occur within each field occurrence (i.e., if the structure occurs twice and the field occurs three times (numbered) the order would be: [1][1][2][2][3][3][3]) or if they would like each field occurrence to occur within each structure occurrence (i.e., if the structure occurs twice and the field occurs three times (numbered) the order would be: [1][2][3][1][2][3][3]).

1. The additional specificity only applies to the following fields:
   - Line: **Modifier**
   - Aeb_Dx_01: **Diagnosis Errors**
   - Aeb_Op_0: **Procedure Errors**
   - Mob2: **Code Type**
   - Mob2: **Code**
   - Mob3: **Code Type**
   - Mob3: **Code**
• Peb2: Diagnosis Edits
• Peb3: Procedure Edits

Figure 11-10. Example of Specifying Occurrences

2. Alternatively, the Occurrence Format can be set to Block within Field. Under this setting structure occurrences will repeat within each field occurrence.

   Example:

Table 11-1: Block Within Field Settings

<table>
<thead>
<tr>
<th>Field Occurs</th>
<th>Block Occurs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

3. By default, the Occurrence Format is set to Field within Block. Under this setting field occurrences repeat within each structure occurrence.

   Example:

Table 11-2: Field Within Block Settings

<table>
<thead>
<tr>
<th>Field Occurs</th>
<th>Block Occurs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
4. Save the mapping layout by selecting the Save or Save As buttons.

**11.1.1.6 Hover Help**

Another feature of the EASYGroup™ Input/Output Panel is hover help. By placing the mouse over the question mark next to each field users will see text displayed that gives some background information (variable name and field description) for the applicable field.

![Hover Help Additional Field Information](image)

**11.1.2 XML File Layout Panel**

The XML File Layout Panel on the right-hand side of the screen includes the field(s) that the user has added from the various EASYGroup™ input/output structures, as well as any user-defined custom fields, that they included in the mapping file. This panel shows the Start Index (for delimited layouts) or Position (for fixed layouts) for each individual field, the Field Name, the user-created Nickname, the field Length, and the Block or EASYGroup™ structure that the field was derived from.

![XML File Layout Panel](image)

**11.1.2.1 XML File Preview**

The XML File Layout Panel also shows the user a preview of the mapping file at the bottom of the screen:

<table>
<thead>
<tr>
<th>Start Index</th>
<th>Field Name</th>
<th>occur</th>
<th>Nick Name</th>
<th>Length</th>
<th>Block</th>
<th>occur</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Effective Date</td>
<td></td>
<td>Start</td>
<td>8</td>
<td>Pcb1</td>
<td></td>
</tr>
</tbody>
</table>
When the user selects a field on the screen, they will see that field’s value highlighted on the bottom of the screen, as shown in Figure 11-13. In this example, the user had the **From or Admission Date** field shown in Figure 11-15 highlighted on the screen. The user can adjust how this information appears to insure it matches the target claim file, by adjusting the options on the far right-hand side of the screen under the **Delimiter** heading.

**Note**

Claim data files using **Pipe (|)** or **Comma (,)** delimiters are required to have a final delimiter after the last value needed to process the claim.
11.1.2.2 Right-Click Options

If users right-click on any fields in the **XML File Layout Panel** they will see the following options:

- **X Remove** (removes the field from the custom XML layout)
- **R Remove Block** (removes the entire EASYGroup™ structure from the XML layout)
- **U Move Up** (moves the selected field up one position in the custom XML layout)
- **D Move Down** (moves the selected field down one position in the custom XML layout)
- **F Move to First** (moves the selected field to the first position in the custom XML layout)
- **L Move to Last** (moves the selected field to the last position in the custom XML layout)
- **T Move To** (moves the selected field to any position in the custom XML layout)
- **G Group** (only appears when the field selected can be grouped)
- **E De-Group** (only appears when the field selected has been grouped)
After selecting the desired option the field should move accordingly. These options are also available on the far right-hand side of the screen under **Reorder**:

- **X** Remove
- **R** Remove Block
- **U** Move Up
- **D** Move Down
- **F** Move to First
- **L** Move to Last
- **T** Move To
- **G** Group (only appears when the structure selected can be grouped)
- **E** De-Group (only appears when the structure selected has been grouped)
11.1.2.3 Additional Options

On the far right-hand side of the screen there are a few functions that users should be aware of:

- **New** - Selecting this button allows the user to reset the **Mapping** screen to its default state.
- **Open** - This button allows the user to open a previously saved XML file.
- **Save** - This option allows the user to save an updated XML file.
- **Save As** - This option allows the user to save an XML file. Users should select this option when first saving a new file or when re-naming an existing file.

**Note**

Selecting the **Save** or **Save As** options the mapping file will be saved to the path designated in the **MappingXmlDir** element located under **Settings>Application**. Please refer to Chapter 10 for further information.

11.1.3 Helpful Hints

To optimize OEPPS performance, the following is recommended:

- Limit the claim file(s) to only contain fields that are necessary.
- Since OEPPS processes left to right, position all regularly used fields at the beginning of the file and omit anything unused per claim at the end of the file. For example, if there are 100 line occurrences, group that block and position at the end of the XML file. This way, when only 5 lines are passed, the other 95 lines can be completely omitted (including the delimiters) from the end of the file.
- Use as little occurrences as possible for input and output. If possible, create a separate layout and FileWatch directory for outlier cases that exceed the normal distribution of occurrences.
• Separate input from output, or don’t supply the spaces or delimiters at the end of the input, so that OEPPS does not map the output during input mapping.

• Combine as many similar claims as possible in a single file before launching the processor. For example, a thousand claims in one file processes faster than ten claims in a hundred files.
Index

C
Contact Optum 4
Contact Us 5
Client Services 5
Corporate Address 5

E
EASYGroup Web Service 66

G
Guide Overview 2
Document Conventions 4
Introduction to This Guide 3
Intended Audience 3
Organization of This Guide 3

I
Integration 45
Server 46

O
Optimizer 16
Architecture 20
Overview 17
Process Summary 26
Program and Data Files 17
Using Rate Files
Data Handling 22
Effective Date 23
Grouper Type 24
Grouper Version 24
Payer/Pricer Type 24
Payers 23
Price/Group Date 23
Pricing Variables and Rate Data 24
Optimizer Configuration
C Platform
Calling the Optimizer 33
Compiling Your Interface 34
Data Files 35
Reading Patient Records 29
Storing Results 33
COBOL Platform
Calling the Optimizer 40
Determining Applicable Grouper Types and Versions 41
Driver Program Overview 38
Using EASYGroup Drivers 41
Writing Your Own Driver 38

S
Server 42

Configuration 57
Features Supported 43
Overview 43