****

Connect5

Database Design

*Agencyport Software, Inc*

v1.2.0.00011

Table of Contents

[Overview 3](#_Toc367377074)

[General Terminology 3](#_Toc367377075)

[SDK Tables 4](#_Toc367377076)

[PF\_EIP\_TRANSPORTLOG 4](#_Toc367377077)

[PF\_EIP\_TRANSACTSEQUENCE 5](#_Toc367377078)

[PF\_EIP\_PARAMLOG 5](#_Toc367377079)

[PF\_EIP\_STAGELOG 5](#_Toc367377080)

[PF\_EIP\_TRANSACTIONLOG 6](#_Toc367377081)

[PF\_EIP\_USERS 6](#_Toc367377082)

[PF\_EIP\_CONFIG 6](#_Toc367377083)

[AP\_NONCE 7](#_Toc367377084)

[PF\_EIP\_ERRORLOG 7](#_Toc367377085)

[PF\_EIP\_LOCKS 7](#_Toc367377086)

# Overview

This document details the tables that are in place as part of the Connect5 implementation.

Not all tables are used by all implementations.

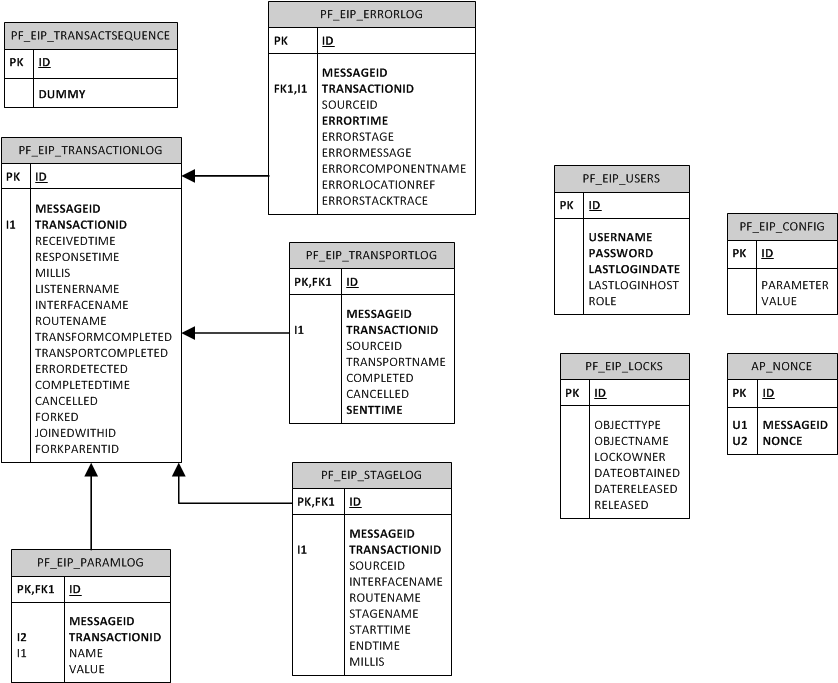
The diagrams have been generated from a Microsoft SQL Server instance so please note that some data types may not match exactly with your implementation.

Tables that are required by the framework start with “PF”; tables that were added for specific modules (e.g. WebService Security) start with “AP”; carrier-specific tables should follow table naming conventions as set forth by the carrier.

## General Terminology

* **Interface** – An Interface is a series of routes designed to represent a business process. Sometimes interfaces are “named”, whereas other times they are the “ROOT” interface (though this is discouraged).
  + **Message** – A Message is an invocation of an Interface.
* **Route** – A Route is a dedicated business unit of work in an Interface for the purpose of making a service call, or acting as a self-contained unit of work.
  + **Transaction** – A Transaction is an invocation of a Route.
* **Format** – Contains all Transformation logic & artifacts for inbound/outbound services.
* **Module** – A configurable component that is one of the following types:
  + **Listener** – Entry point of a Route (Source).
  + **Processor** – A Processor is an ad-hoc step within the Route.
  + **Transport** – Exit point of a Route (Target).
    - **Stage** – A Stage is an invocation of a Module that is part of a Route. For example when running an XSLT that’s a straight “Processor” Stage, but when making a call out to HTTP or another route that would be a “Transport” Stage.

# SDK Tables



## PF\_EIP\_TRANSPORTLOG

The PF\_EIP\_TRANSPORTLOG table is responsible for capturing pertinent information about the Transport invocations that happen as part of an interface execution.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| MESSAGEID | varchar(128) | ID of Message which executed Transaction |
| TRANSACTIONID | varchar(128) | ID of Transaction which executed Transport |
| SOURCEID | varchar(128) | Original Transaction ID in the event that a fork happened, which would end up causing the original Transaction ID to be lost |
| TRANSPORTNAME | varchar(128) | Name of Transport that was executed |
| COMPLETED | int | Indicator that Transport completed |
| CANCELLED | int | Indicator that Transport was cancelled |
| SENTTIME | datetime | Time Transport was invoked |

## PF\_EIP\_TRANSACTSEQUENCE

The PF\_EIP\_TRANSACTSEQUENCE table is responsible for being a shell table to manage Transaction IDs

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| DUMMY | varchar(128) | Dummy column to make table more than just an identity column |

## PF\_EIP\_PARAMLOG

The PF\_EIP\_PARAMLOG table is responsible for storing the parameters that the application determines are worth storing for the purpose of displaying those fields back as part of the Admin Console. The fields that make it to this table are set up in the report\_param.xml file.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| MESSAGEID | varchar(128) | ID of Message which executed Transaction |
| TRANSACTIONID | varchar(128) | ID of Transaction which was the last to update the parameter |
| NAME | varchar(128) | Parameter name |
| VALUE | varchar(128) | Parameter value |

## PF\_EIP\_STAGELOG

The PF\_EIP\_STAGELOG table is responsible for capturing the relevant information from the execution of a Processor.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| MESSAGEID | varchar(128) | ID of Message which executed Transaction |
| TRANSACTIONID | varchar(128) | ID of Transaction which executed Stage |
| SOURCEID | varchar(128) | Original Transaction ID in the event that a fork happened, which would end up causing the original Transaction ID to be lost |
| INTERFACENAME | varchar(128) | Name of Interface which owns Route |
| ROUTENAME | varchar(128) | Name of Rotue which owns Stage |
| STAGENAME | varchar(128) | Name of Stage |
| STARTTIME | datetime | Start Time |
| ENDTIME | datetime | End Time |
| MILLIS | int | Duration in milliseconds |

## PF\_EIP\_TRANSACTIONLOG

The PF\_EIP\_TRANSACTIONLOG table is responsible for auditing the execution of the Routes in the Interfaces.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| MESSAGEID | varchar(128) | ID of Message which executed Transaction |
| TRANSACTIONID | varchar(128) | ID of Transaction – logically this is the primary key |
| RECEIVEDTIME | datetime | Start Time |
| RESPONSETIME | datetime | End Time |
| MILLIS | int | Duration in milliseconds |
| LISTENERNAME | varchar(128) | Name of Listener on Route that was used to “enter” the Route |
| INTERFACENAME | varchar(128) | Name of Interface which owns Route |
| ROUTENAME | int | Name of Rotue which was executed |
| TRANSFORMCOMPLETED | int | Indicator that Transformation was completed |
| TRANSPORTCOMPLETED | int | Indicator that Transport was completed |
| ERRORDETECTED | int | Indicator that an error occurred |
| COMPLETEDTIME | datetime | Completion time |
| CANCELLED | int | Indicator whether Transaction was cancelled or not |
| FORKED | int | Indicator whether this Transaction was forked |
| JOINEDWITH | varchar(128) | If Transaction was joined, the name of the Transaction with which it was joined |
| FORKPARENTID | varchar(128) | If this Transaction was Forked, the ID of the parent Transaction |

## PF\_EIP\_USERS

The PF\_EIP\_USERS table is responsible for managing user information.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| USERNAME | varchar(128) | User’s login |
| PASSWORD | varchar(128) | User’s password |
| LASTLOGINDATE | datetime | Last login timestamp |
| LASTLOGINHOST | varchar(128) | Host from which last login occurred |
| ROLE | int | Role identifier |

## PF\_EIP\_CONFIG

The PF\_EIP\_CONFIG table is responsible for the product’s internal configuration. This table serves as an extension point to the application’s configuration.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| PARAMETER | varchar(128) | Application configuration property |
| VALUE | varchar(128) | Application configuration value |

## AP\_NONCE

The AP\_NONCE table is responsible for storing unique Nonces as they’re associated to a Interface invocation.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| MESSAGEID | varchar(128) | ID of the Message for which the NONCE is being stored |
| NONCE | varchar(128) | Unique NONCE associated with Message |

## PF\_EIP\_ERRORLOG

The PF\_EIP\_ERRORLOG table is responsible for capturing information around error generation.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| MESSAGEID | varchar(128) | ID of Message which executed Transaction |
| TRANSACTIONID | varchar(128) | ID of Transaction which executed Stage (if applicable) |
| SOURCEID | varchar(128) | Original Transaction ID in the event that a fork happened, which would end up causing the original Transaction ID to be lost |
| ERRORTIME | datetime | Timestamp of when error occurred |
| ERRORSTAGE | varchar(128) | Type of activity that was happening when error occurred: Transport or Processor, for example |
| ERRORMESSAGE | varchar(128) | Specific error message |
| ERRORCOMPONENTNAME | varchar(128) | ID of Stage during which error happened |
| ERRORLOCATIONREF | varchar(128) | A combination of ERRORSTAGE and ERRORCOMPONENTNAME. |
| ERRORSTACKTRACE | varchar(4000) | Entire stack of the error |

## PF\_EIP\_LOCKS

The PF\_EIP\_LOCKS table is responsible for managing the dedicated locks that are needed by the application.

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | int | Primary key identifier |
| OBJECTYPE | varchar(128) | Type of object that owns the lock |
| OBJECTNAME | varchar(128) | Name of the object that owns the lock |
| LOCKOWNER | varchar(128) | Owner of the lock |
| DATEOBTAINED | datetime | Timestamp of when the lock was acquired |
| DATERELEASED | datetime | Timestamp of when the lock was released |
| RELEASED | int | Indicator as to whether lock was released |