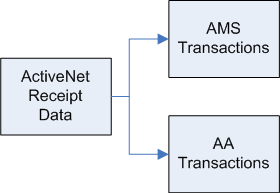
# Basics

## Reconciliation period

Reconciliation is performed for a calendar month. However, the reconciliation process can reconcile over arbitrary periods. At the very beginning (in 2009), there were four quarterly reconciliations. But now, it’s always monthly.

## Triad

The accounting people will talk about reconciling a “triad”. This consists of a source application (ActiveNet), the electronic payments processed (AMS), and the remittances to the orgs (AA = Active Accounting). So for ActiveNet, the triad consists of two dataflows which need to be reconciled:



Although the data flow isn't strictly unidirectional (e.g., AMS results flow back into AN), for simplicity each path is considered to have a source system (AN) and a destination system (AMS).

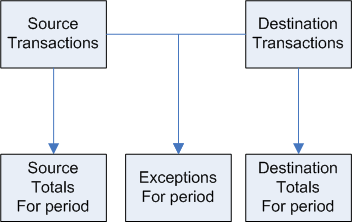
## Reconciled data

* For AMS, only cash data is reconciled.
* For AA, both cash and revenue data are reconciled.
* These data are subtotalled and compared by currency code.

## Exceptions

An exception is a specific mismatch between the two systems for a specific key during the reconciliation period.

For each reconciliation path, the basic process is the same:



* Transaction totals are generated for the source table for the reconciliation period
* Transaction totals are generated for the destination table for the reconciliation period
* The source and destination tables are compared for the reconciliation period using some common key, and a table of exceptions is generated.

This procedure is considered internally consistent only if the differences of the totals (by currency or currency plus increase/reduction) is exactly accounted for by the corresponding exceptions. This is not a question of source or destination data quality; it is only a question of the quality of the exception generating algorithm.

## Time skew exceptions

There is a difference in how the three different systems record times:

* AMS transactions are always recorded in Pacific time
* AN transactions are always recorded at the time they occurred in the org’s timezone.
* AA transactions keep the AN datestamp, so are also recorded in the org’s timezone.

This leads to the most common type of exception (by number) are “time skew” exceptions. For example, a transaction which took place in a org in New York, at 1:00AM on the 1st of the reconciliation month, say July. It would have been processed by AMS at 10:00PM PT on June 30th. This results in the July AMS totals (which don’t include this amount) mismatching the AN totals (which do).

Therefore the exception generating process generates one exception record for each such time skew, to account to that

## Deferred capture

A second factor which can cause a significant difference between AMS time and AN time is if the org is configured with deferred capture, which can be up to 12 hours. In this case, an AMS auth is issued at the time the receipt is saved, but the AMS capture will be some number of hours later. The capture transactions are the ones we get, so this can also lead to time skew exceptions.

## Defects

Exceptions are categorized as to whether they are “defects” or not. Defects are exceptions which represent actual financial differences between the systems, which must therefore be explained by adding addition information into the reconciliation spreadsheet. For example, the time skew exceptions just described are not defects. But if there is an AN receipt paid by CC, but no corresponding AMS transaction, that is a defect.

## Currency code

All financial transactions are given a currency code, and are subtotaled on the spreadsheet by currency code. Currently, there are only active transactions in USD (US dollars) and CAD (Canadian dollars), although we had a few test transactions go through from an org setup with AUD (Australian dollars).

## Inc/Red

The initial audit requirements were to also match on Increases (positive amounts) vs. Reductions (negative amounts). Once this requirement went away, I eliminated I/R processing. However, I found or AMS, this actually led to more false positives rather than less, so now, AA processing is done without I/R, and AMS processing is done with I/R.

# Basic data flow



## Data sources

There are two basic sources of data to reconcile:

**1. Reconciliation database:** On the reconciliation server, there is a database called “Reconciliation”. It contains two tables which we use:

* **aa\_transactions:** This table contains a copy of the Active Accounting transactions for all applications, exported from the actual AA system.
* **ams\_transactions:** Similarly, this table contains a copy of the AMS transactions for all applications, exported from the actual AMS system.

**2. ActiveNet rpt\_rev\_v2 tables:** The primary source of AN data to reconcile is the rpt\_rev\_v2 table in the ActiveNetSites DB in each data center. We access it via linked servers DBANET01VS and DBPRDCDC.

**3. ActiveNet org databases:** In addition, some information is extracted directly from tables in the org databases, in order to determine whether certain types of exceptions are actually defects or not. Since Las Vegas has a second SQL Server with production sites on it, there is a third linked server DBANET10VS.

## Import process

The first part of the process is importing data from these three source systems into corresponding tables in the ActiveNet database.

**4. rec\_set\_month MM,YY:** Executing this proc sets three values in the system\_info table, which control how all the other procs work:

|  |  |  |
| --- | --- | --- |
| Keyword | Example | Meaning |
| period\_beg\_date | 2016-6-1 | First day of the month being reconciled |
| period\_end\_date | 2016-7-1 | First day of the next month |
| import\_beg\_date | 2016-5-30 | First day imported. Currently, this is two days before the beginning of the period, to ensure we the data necessary to determine whether a mismatch is a timeskew or a defect. (Actually, one day would be adequate). |

**5. rec\_import\_aa:** This proc imports data from the Reconciliation database aa\_transactions table into the ActiveNet database imported\_aa table. When doing so, it

* It only imports records with data\_set\_name = ‘activenet’.
* It only imported records on or after the import\_beg\_date.
* It puts standard data into columns with some standard column names, so generic procs and do total and exception generation (detailed below).
* Because the AA key is the rev\_id (PK) from rpt\_rev\_v2, it should be unique. If it is not, the proc gives an error.

**6. rec\_import\_ams:** This proc imports data from the Reconciliation database ams\_transactions table into the ActiveNet database imported\_am table. When doing so, it

* It only imports records with data\_set\_name = ‘activenet’.
* It only imported records on or after the import\_beg\_date.
* It handles AMS voids, bringing them into imported\_ams.
* It puts standard data into columns with some standard column names, so generic procs and do total and exception generation (detailed below).
* It captures some other AMS columns which will be used later.

It fires a second proc rec\_import\_ams\_fix, which also can be re-run later, which does

* It parses the cct\_order\_id, to get the receipt number into a separate column, which will be used as part of the common key between AN and AMS.
* It takes the ma\_name (AMS account name) and converts it into the AN org name. If certain default patterns were followed (e.g., ma\_name = “Recnet – “ + orgname), this happens automatically. However, if the AMS account creation was done with a non-standard name (maybe ¼ of cases), the proc fixes it.

**7. rec\_import\_rpt:** This proc imports data from the rpt\_rev\_v2 table in each data center:

* It only imported records on or after the import\_beg\_date.
* It puts standard data into columns with some standard column names, so generic procs and do total and exception generation (detailed below).
* It generates the Inc\_red column

**8. rec\_import\_org\_data:** This proc uses the ActivenetSites orgs and orgsites table to go through all production orgs in each data center, and then import certain data from their tables:

* **imported\_ecp:** contains one record for each ECP payment for each org. It is used to determine whether certain apparent defects are actually expected, based on the different between how AMS and AN handle ECP cancellations.
* **imported\_org\_info:** contains one record per org with time zone information, currency code, and a few other columns.
* **imported\_icverifylog:** contains key columns from the ICVerifyLog table in all org databases for the selected period. Not used in any procs, but sometimes useful for trouble-shooting.

## Total and exception generation

**9. rec\_totals:** This proc generates four tables of totals by currency\_code and sometimes I/R for the reconciliation period, then displays four result sets in a form which can be directly pasted into the Totals tab. It uses the proc sub\_gen\_totals to generate each case.

**10. rec\_exceptions\_aa:** This proc generates the table exceptions\_aa, which contains all the exceptions between AN and AA for the reconciliation period. It then displays it in a resultset which can be directly pasted into the AA Exceptions tab. After the generic exception proc runs, it demotes a few defects as follows:

|  |  |  |
| --- | --- | --- |
| **Conditions** | **Changes to exception** | **Defect?** |
| org is AMStest or AMStestCA | Descript Explanation='Test site' | 0 |
| org is ‘USASynchro’ | Explanation='Org manually invoiced quarterly; not fed to AA' | 0 |

**11. rec\_exceptions\_ams:** This proc generates the table exceptions\_aa, which contains all the exceptions between AN and AA for the reconciliation period. It then displays it in a resultset which can be directly pasted into the AA Exceptions tab. After the generic exception proc runs, it demotes or recategorizes a few exceptions as follows:

|  |  |  |
| --- | --- | --- |
| **Conditions** | **Changes to exception** | **Defect?** |
| Defect exists, but matching record exists in imported\_ecp table | Description = ECP failure | 0 |
| AN missing in AMS defect, with $2 amount | Explanation = ANE-30410 Extra explanation = 'Zero dollar receipt with $2 convenience fee and no AMS' | 1 |
| cct\_order\_descriptor = 'Terminal Transaction' | Description='Terminal Transaction' | 0 |
| Org like ‘AMSTest%’ or ‘AMSLiveTest’ or ‘ANETLiveUSVerify’ | Explanatiom = ‘Test site’ | 0 |
| Two defects exist for same org/receipt, one an I and one an R, with same dollar amount but inverted sign | Description = 'Receipt split between I and R' | 0 |

## Explanation (automatic and proc-assisted)

**12. rec\_explain\_defects:** This proc fills in the “explanation” and “extra explanation” column for some common current cases, so manual research (using rec\_investigate\_receipt) is only necessary on a much smaller number of cases:

<TBD: Table of cases here>

# Details of exception processing

## Common key

The purpose of the exception system is to explain the aggregate difference between two systems (e.g., AN and AMS) as the sum of a number of specific differences involving one or a small number of records in each system. For this purpose, it is necessary to have a common key between the systems, to be able to match records. The AN-AA reconciliation and the AN-AMS reconciliation use different common keys as follows:

|  |  |  |
| --- | --- | --- |
| **Reconciliation** | **Common key** | **Notes** |
| AN – AA | Rev\_id | The rpt\_rev\_v2 primary key, rev\_id, is brought into AA as its tran\_source\_id, so we can use it to reliably match the records between the systems 1-1. |
| AN - AMS | Org : receiptnumber | The AMS interface is not a simple 1-1 data feed, and it’s quite possible to have more than one AMS record when only one is expected. So the comparison is based on receipt number for that org. |

## Types of generic exceptions

The exception proc used by exceptions\_ams and exceptions\_aa (sub\_gen\_exceptions), generates these generic exceptions (where XX = AMS or AA):

|  |  |  |  |
| --- | --- | --- | --- |
| **Exception type** | **Defect?** | **Description field** |  |
| Amounts mismatch | 1 | Amounts mismatch | For a given effective key, AN and XX record(s) exist during the reconciliation period; however, the totals of the cash\_amt and rev\_amt fields for these records don't match. |
| "Not In" types | 1 | AN not in XX | A given value of the effective key exists in the AN table, but doesn't exist at all in the destination table. |
|  | 1 | XX not in AN | Similarly, a given value of the effective key exists in the destination table, but not in AN. |
| "Time skew" types | 0 | Time skew: AN before period | For a given effective key, XX record(s) exist during the reconciliation period; AN record(s) exist, but are before the period. |
|  | 0 | Time skew: AN after period | For a given effective key, XX record(s) exist during the reconciliation period; AN record(s) exist, but are after the period. |
|  | 0 | Time skew: XX before period | For a given effective key, AN record(s) exist during the reconciliation period; XX record(s) exist, but are before the period. |
|  | 0 | Time skew: XX after period | For a given effective key, AN record(s) exist during the  reconciliation period; XX record(s) exist, but are after the  period. |

Also, for "Not In" and "Time skew" exceptions, no exception is generated if the cash and revenue values are zero.

## ECP defects

# Old documentation not updated

The following section is documentation which is several years old, and has not been updated/rewritten. It’s generally, but not always, accurate.

## Explanation of defects - general

* For each defect on the AMS and AA Exceptions tabs, the developer must provide an explanation.
  + The basic explanation code (often a Jira ticket number) should be in the column "Developer's explanation summary (defects only)"
  + Any additional detail should be in the column "Additional information from developer".
* The spreadsheet computes an effective explanation (column B) as follows:
  + If the exception is not a defect, the description is used as the explanation
  + If the exception is a defect, and a summary explanation is provided, the explanation is the concatenation of the generic description and the developer's explanation.
  + If the exception is a defect, but no summary explanation is provided, the explanation is "Unexplained".
* The summary explanation is used in the Explanations tab, which totals the exceptions by cause (explanation).
* As defects are explained, the summary tab will adjust to show the $ totals of explained and remaining (unexplained) exceptions:
  + When the process is done, the "AA-AN Remaining Exceptions" block should be all 0
  + When the process is done, the "Remaining exceptions" column in the "AMS-AN Deltas" block should be all 0.

## Explanation of AA defects

* **Site-specific remittance problem:**Some of the income from the month couldn't be fed into AA, because the org was configured for site-specific remittance, but the site receiving the income wasn't configured with EBS accounts. This is generally a problem on the ActiveNet side, of fees being configured to use the Internet site. It could also be a failure to configure the site EBS configuration.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Overview Generating these exception lists is by far the most algorithm in the reconciliation process (the proc is ~500 lines). When I first worked on ActiveNet reconciliation to AMS, I developed a set of very Activenet-specific reconciliation procedures which would combine to generate the exception list. Later, when I worked on RegOnline reconciliation, I developed a much more general approach, but still with a separate proc for each of the two paths initially using it. I then folded this approach back into ActiveNet, rewriting it again as a general purpose parameterized procedure which could be used in all cases. Currently, both ActiveNet reconciliation paths and all five ROL paths generate their exception list using this common proc.  In order to generate spurious "Not In " defect exceptions which require investigation, rather than benign "Time Skew" non-defect exceptions, it is recommended that all data sets have at least the data from the day following the end of the period. (More about those exceptions below). Spreadsheet generation The procedure below is a set of manual steps of running procs and cutting results into a spreadsheet. However, the process of generating a spreadsheet for a system for a month typically takes under 15 minutes, once the exported data is in place, so I haven't judged it cost-effective to automate the full spreadsheet generation, especially since the format is still a moving target.   * Within the application database, a series of stored procedures are executed. * Each proc generates an output table; the proc drops the table if it already exists, and then creates it, so the schema is self-contained and documented in the proc. * For the totals and exception lists, these procs generate result sets which are designed to exactly match the data required by the spreadsheet, so they can be pasted directly into the spreadsheet. * In a diagnostic tab, the exception totals are compared against the aggregate deltas, to allow development to ensure that the generated data is internally consistent. * Formulas in the spreadsheet compute aggregate deltas and roll up the exception details for accounting.  Explanation of exceptions  * The exceptions are categorized as to whether they are "defects" or not. A defect represents a true financial mismatch between the system; non-defects are generally exceptions such as timing differences which are necessary to explain the aggregate delta but don't represent a system problem. * In the exception lists in the spreadsheet, development annotates all the defects with an additional "explanation"; non-defects are automatically considered to be explained. * The spreadsheet summarizes for accounting the total differences between the source and destination systems, and the amount of the difference which is explained and unexplained.  Spreadsheet operation A copy of sample ROL reconciliation spreadsheet is attached to illustrate the spreadsheet design. Template format For each application, there is a template spreadsheet which contains no data, but does contain all the necessary formulas.  The template is organized into two tabs for accounting, and a series of tabs for development. This allows us to hide the development tabs prior to submitting it to accounting. All the proc data is pasted into the development tabs; from there it flows into the accounting tabs. These tabs are:   |  |  |  | | --- | --- | --- | | **Consumer** | **Tab** | **Contents** | | Accounting | Summary | Summary of totals from all systems, and a comparison of reconciliation deltas against the "explained" exceptions. | | Exception Summary | Pivot tables breaking down the exceptions data from each reconciliation path by exception type. | | Development | Totals | Development pastes all system totals for periods into blocks on this tab. | | Exceptions (1 or more tabs) | For each reconciliation path, there is an exception tab. Development pastes the exception lists into this tab. | | Diagnostics | Contains   * An internal audit block for each reconciliation path, with the aggregate deltas, the total of the corresponding exceptions, and their difference (which should always be 0). * A metrics block, which gives the defect rates ($ of defects / total $ transacted). |  Exception tabs For each reconciliation path, a proc is used to generate a set of exceptions for the financial system, which are mismatches between the source and destination tables, based on some common key. The result set of this proc is pasted into the corresponding exception tab in the template; here is an example of some exception rows from ActiveNet:   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Defect Amt** | **Explanation** | **Defect** | **Group** | **Discrepancy** | **Receipt** | **Description** | **Developer's   explanation summary (defects only)** | **Additional   information from developer** | | $        390.00 | AMS   missing in AN: JIRA AN-21717 | 1 | R-USD | $           (390.00) | 1001401.002 | AMS missing in AN | JIRA AN-21717 | This   looks like a bug in either AN or AMS.    We are currently researching. | | $        135.00 | Unexplained | 1 | R-USD | $           (135.00) | 1000677.011 | AMS missing in AN |  |  | | $        300.00 | AN   missing in AMS: JIRA AN-21728 Permit Refund Void | 1 | I-USD | $           (300.00) | 2002046.002 | AN missing in AMS | JIRA AN-21728 Permit   Refund Void | Voided   permit refund.  When a permit refund is   voided, the system warns the user that the charge WILL NOT be added back to   AMS.  We can look at adding this to AN   in the future. | | $               - | Time   skew: AMS before period | 0 | I-USD | $             (35.00) | 1080102.002 | Time skew: AMS before   period |  |  | | $               - | Time   skew: AMS before period | 0 | I-CAD | $           (224.46) | 1000609.008 | Time skew: AMS before   period |  |  | | $               - | Time   skew: AN after period | 0 | I-USD | $              54.50 | 1022702.002 | Time skew: AN after   period |  |  | | $               - | Time   skew: AN after period | 0 | I-USD | $              54.50 | 1022703.002 | Time skew: AN after   period |  |  | | $               - | Time   skew: AN after period | 0 | I-USD | $              54.50 | 1022704.002 | Time skew: AN after   period |  |  | | $               - | Time   skew: AN after period | 0 | I-USD | $              54.50 | 1022705.002 | Time skew: AN after   period |  |  | | $               - | Time   skew: AN after period | 0 | I-USD | $              30.00 | 1022706.002 | Time skew: AN after   period |  |  |  * Columns from the proc pasted in:   **Defect:**Whether or not this exception is a defect, representing an financial mismatch requiring further explanation by development.  **Group:**Subtotalling key, in this case a combination of I/R (increase/reduction) and currency code.  **Discrepancy:**Actual difference between the two systems  **Description:**Generic type of exception as determined by the proc.  **Application-specific columns:**Depending on the application, additional columns are output to provide further explanatory information for development. In this case, to fit on the page, it is represented by just the Receipt column, although there are several more columns in the AN output.   * Columns added by developer:   **Explanation summary:**A short summary of the cause. Should include a Jira ticket number if a known code defect or under investigation by development. Should be short and standardized, because it is fed into the Exception Summary tab as a pivot table key.  **Additional information:**Further more detailed information about a specific exception or group of exceptions.   * Columns computed by spreadsheet:   **Defect amount:** This figure is totalled into the defect metrics on the Diagnostics tab. For a defect, is it the absolute value of the discrepancy; for non-defects, it is zero.  **Explanation:** The effective explanation fed into the Exception Summary tab.  For non-defects, it is just the description.  For defects with an explanation, it is a concatenation of the description and the explanation  For defects without explanations, it is "Unexplained". Diagnostics tab  * **Reconciliation metrics block:** This block contains a summary of the total currency value of defects compared to the total cash transacted, as a high-level metric of system quality. These are computed from the exception tab data, and currently, are arbitrarily added up regardless of currency code. Here is an ROL example:  |  |  | | --- | --- | | **Reconciliation Metrics** | | | **Total cash** | $          14,192,637.71 | | Total payment   defects | $                 81,441.05 | | **Payment defect   rate** | 0.5738% | | Total INV-TR   defects | $                 18,951.53 | | Total EBS-INV   defects | $                   1,312.24 | | Total   accounting defects | $                 20,263.77 | | **Accounting   defect rate** | 0.1428% |  * Exception auditing blocks: For each reconciliation path, there is an exception auditing block, which takes input from the relevant transaction total blocks and exception tab. This is used by development to confirm that the exceptions generated account for the entire aggregate delta; the "Remaining" column should be zero in all cells:  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **PT -   ROL Exceptions** |  |  |  |  |  | | **Currency** | **PT** | **RO** | **Delta** | **Exceptions** | **Remaining** | | AUD | $             627,018.63 | $         629,602.00 | $             (2,583.37) | $       (2,583.37) | $         0.00 | | CAD | $             469,967.06 | $         473,540.38 | $           (3,573.32) | $       (3,573.32) | $        (0.00) | | CHF | $              59,264.10 | $           58,784.10 | $               480.00 | $           480.00 | $              - | | EUR | $          1,674,577.34 | $       1,673,213.71 | $            1,363.63 | $        1,363.63 | $         0.00 | | GBP | $          1,024,559.29 | $       1,023,194.16 | $              1,365.13 | $        1,365.13 | $         0.00 | | HKD | $             111,107.70 | $         110,024.80 | $            1,082.90 | $        1,082.90 | $        (0.00) | | NZD | $             229,322.65 | $         232,518.61 | $           (3,195.96) | $       (3,195.96) | $         0.00 | | SGD | $             157,841.80 | $         157,784.74 | $                 57.06 | $            57.06 | $        (0.00) | | USD | $          9,349,029.17 | $       9,651,334.34 | $          (302,305.17) | $ (302,305.17) | $         0.00 | | ZAR | $             183,035.87 | $         182,640.87 | $               395.00 | $           395.00 | $              - | | **All units** | **$      13,885,723.61** | **$   14,192,637.71** | **$        (306,914.10)** | **$ (306,914.10)** | **$         0.00** |  Summary tab The summary tab contains:   * For each transaction source, there is a totals block with data from the from the totals tab:  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **EBS   Data** |  |  |  |  | |  | **Currency** | **Revenue** | **Liability** | **Cash** | | Invoice Totals | AUD | 46,032.23 | 583,569.76 | 629,601.99 | | CAD | 35,657.50 | 437,882.88 | 473,540.38 | | CHF | 2,822.10 | 56,442.00 | 59,264.10 | | EUR | 89,780.92 | 1,584,453.78 | 1,674,234.70 | | GBP | 67,439.48 | 957,560.53 | 1,025,000.01 | | HKD | 6,586.60 | 103,438.20 | 110,024.80 | | NZD | 12,973.72 | 219,544.89 | 232,518.61 | | SGD | 5,855.87 | 151,971.89 | 157,827.76 | | USD | 742,473.14 | 8,922,815.04 | 9,665,288.18 | | ZAR | 8,740.14 | 173,900.73 | 182,640.87 | |  | 1,018,361.70 | 13,191,579.70 | 14,209,941.40 |  * For each reconciliation path, a deltas blocks with comparison to the explained exceptions (those which are not "Unexplained").  |  |  |  |  |  | | --- | --- | --- | --- | --- | | Source-PT   Deltas |  |  |  |  | | Type | Currency | Cash | Explained exceptions | Unexplained | | Invoice Totals | AUD | 33,855.07 | 33,855.07 | - | | CAD | (4,047.73) | (4,047.73) | 0.00 | | CHF | (2.00) | (2.00) | - | | EUR | (2,038.87) | (2,038.87) | 0.00 | | GBP | 2,839.39 | 2,839.39 | 0.00 | | HKD | 1,018.72 | 1,018.72 | 0.00 | | NZD | (152.78) | (152.78) | 0.00 | | SGD | 1,376.00 | 1,376.00 | - | | USD | 160,029.08 | 160,029.08 | - | | ZAR | (2.00) | (2.00) | - | | ALL UNITS | 192,874.88 | 192,874.88 | - |  Exception summary tab The exception summary tab contains pivot tables of exceptions (regardless of defect status), by explanation; here is a typical example:   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | I-CAD |  | I-USD |  | Total Count | Total Amount | | Row Labels | Count | Amount | Count | Amount |  |  | | Time skew: AMS before period | 49 | ($6,060.89) | 2245 | ($169,869.79) | 2294 | ($175,930.68) | | Time skew: AN after period | 82 | $31,563.12 | 1913 | $126,155.96 | 1995 | $157,719.08 | | Time skew: AN before period |  |  | 1 | $72.92 | 1 | $72.92 | | Rounding error: Jira AN-21700 |  |  | 19 | $0.20 | 19 | $0.20 | | Receipt split between I and R |  |  | 7 | $300.50 | 7 | $300.50 | | AMS missing in AN: JIRA   AN-21754 |  |  | 4 | $24.76 | 4 | $24.76 | | AN   missing in AMS: JIRA AN-21728 Permit Refund Void | |  | 4 | ($611.00) | 4 | ($611.00) | | AN   missing in AMS: AMS Data Missing from Amazon DB | |  | 5 | ($105.00) | 5 | ($105.00) | | Grand Total | 131 | $25,502.23 | 4198 | ($44,031.45) | 4329 | ($18,529.22) |  Exception generating procPurpose This general purpose exception generation algorithm is ~500 line long, and has quite a bit of subtlety, especially dealing with cases in which the common key between the transaction tables in not unique. Therefore is desirable to be able to develop this proc once and use it in multiple settings, and not keep copying and modifying it. As shown in the next section, this makes it possible to have a very simple proc to generate the exceptions for a given reconciliation path. Currently this proc is used in 6 of the 7 reconciliation paths in AN and ROL.  The proc is made made reusable in two ways:   * Making some assumptions about the names and data types of transaction table columns * Having a bunch of parameters on the proc.  Basic Usage model Here is a very simple example of the logic in an exception generating proc, taken from ROL transaction to Paymentech exception proc:  -- Generic exception generator  exec Activenet.dbo.sub\_gen\_exceptions      @period\_beg\_date, @period\_end\_date,     'rol.dbo.imported\_rol\_transactions', 'order\_id', 'ROL',      'rol.dbo.imported\_pt\_merchant', 'order\_id', 'PT',      'rol.dbo.exceptions\_pt\_merchant', 0  -- Display result  select defect, currency\_code, adj\_cash\_amt, trans\_id, description, dst\_date, src\_date from exceptions\_pt\_merchant order by defect desc, description, inc\_red, currency\_code, trans\_id   * The first code block calls the proc sub\_gen\_exceptions. This proc is actually in the ActiveNet database, and is being shared by ROL, so the proc name is qualified. Moreover, since it is executing in ActiveNet database, the input and output table names are qualified back to the ROL database. * The second code block generates a result set which can be directly pasted into the ROL template.   Some more complex examples of usage are included below. Proc parameters Here are the parameters for sub\_gen\_exceptions:   |  |  |  |  | | --- | --- | --- | --- | | **Parameter** | **Data type** | **Purpose** | **Example above** | | @period\_beg\_date | varchar(20) | First date of reconciliation period; e.g., '1/1/10' | @period\_beg\_date | | @period\_end\_date | varchar(20) | First date of month after reconciliation period; e.g., '2/1/10' | @period\_end\_date | | @src\_table | varchar(50) | Name of the source transaction table | 'rol.dbo.imported\_rol\_transactions' | | @src\_key | varchar(50) | Name of the common key in the source table | 'order\_id' | | @src\_abbr | varchar(20) | An abbreviation of the name of the source system to include in exception descriptions. In the example above, this is used to generate descriptions such as "ROL not in PT", when there is a RegOnline transaction missing its corresponding Paymentech transactions | 'ROL' | | @dst\_table | varchar(50) | Name of the destination transaction table | 'rol.dbo.imported\_pt\_merchant' | | @dst\_key | varchar(50) | Name of the common key in the destination table | 'order\_id' | | @dst\_abbr | varchar(20) | |  | | --- | |  | | An abbreviation of the name of the destination system to include in  exception descriptions. | | 'PT' | | @exc\_table | varchar(50) | Name of exception table to create | 'rol.dbo.exceptions\_pt\_merchant' | | @include\_revenue | bit | 1 = process and compare revenue and cash, 0 = just compare cash | 0 |  Transactional table schema assumptions In order to simplify the exception generating proc, the imported transaction tables are assumed to have columns with certain names, so the import procs must honor these conventions:   |  |  |  | | --- | --- | --- | | **Column** | **Type** | **Purpose** | | id | int identity(1,1) | ids  from the source and destination transaction tables are included in the  exception table, so it is possible to join back to the source tables to  get application-specific data to display in the result set. Assumed to  be an int; identity(1,1) is a convenience. | | import\_id | varchar(100) null | An optional key back to the data which was imported, for traceability. | | inc\_red | varchar(1) null | Increase/reduction  flag. If the source application does not use I/R logic (AN does, ROL  does not), this should be blank. Otherwise, it should be "I" if  cash\_amt>=0 and "R" if cash\_amt<0 | | currency\_code | varchar(3) null | Currency code | | trans\_date | datetime null | Transaction date | | cash\_amt | money not null default 0 | Cash amount | | rev\_amt | money not null default 0 | Revenue amount | | <common key> | (not assumed) | For  exception generation, there must be a common key between the two  tables. Because the same table may participate in more than one  reconciliation path, which may have different common keys, it was not  possible to enforce a column name on the common key; | | additional application-specific columns | any | Any  number of application-specific columns may also be in the schema, and included in the result set copied to the exception tab, by joining to  the id from the exception table. |  Exception table schema The output of the exception generating proc is table with a requested name, with the following schema:   |  |  |  | | --- | --- | --- | | **Column** | **Data type** | **Description** | | period\_beg\_date | datetime | Beginning date of period being reconciled | | period\_end\_date | datetime | Ending date of period being reconciled | | defect | int | 1 if exception is a defect; 0 otherwise | | inc\_red | varchar(1) | inc\_red from source and destination records (which might be blank); together with currency\_code, this is the comparison grouping | | currency\_code | varchar(3) | currency\_code from the source and destination records | | trans\_id | varchar(255) | common key value | | src\_id | int null | id  of source record, or null if no matching source record for destination  key; if there are more than one matching source record, this is the min | | src\_cash\_amt | money not null default 0 | total of cash\_amt for source records for common key | | src\_rev\_amt | money not null default 0 | total of rev\_amt for source records for common key; if revenue is not being compared on this reconciliation path, 0 is used | | src\_date | datetime null | trans\_date  of source record, or null if no matching source record for destination  key; if there are more than one matching source record, this is the min | | dst\_id | int null | id of destination record, or null if no matching destination record for key; if there are more than one matching destination record, this is the min | | dst\_cash\_amt | money not null default 0 | total of cash\_amt for destination records for common key | | dst\_rev\_amt | money not null default 0 | total of rev\_amt for source destination for common key; if revenue is not being compared on this reconciliation path, 0 is used | | dst\_date | datetime null | trans\_date  of destination record, or null if no matching destination record for  source key; if there are more than one matching destination record, this  is the min | | adj\_cash\_amt | money not null default 0 | dst\_cash\_amt - src\_cash\_amt | | adj\_rev\_amt | money not null default 0 | dst\_rev\_amt - src\_rev\_amt | | description | varchar(255) | The automatically generated description of the type of exception (described below). |  General logic At at a high level, the exception generation works by using some of the standardized transaction table field described above:   |  |  | | --- | --- | | **Column** | **Purpose** | | trans\_date | This is used to determine select the relevant records from the source and destination tables for the reconciliation period, based on the @period\_beg\_date and @period\_end\_date parameters. | | effective key =  inc\_red, currency\_code, <common key> | The @src\_key column in the source table is compared to the @dst\_key column in the destination table to determine which records in the two data sets should correspond. If a given value of the common key has records for more than one value  of inc\_red / currency code, these are compared separately. For example,  an Activenet receipt number may have both a charge (I) and a void (R) in  the ActiveNet data and AMS, so these are compared separately. (Multiple currencies on the same key value aren't expected, but the logic would also compare them separately).  So the actual key compared is inc\_red + currency\_code + common key. | | cash\_amt | When corresponding records do exist in the period, the cash\_amt totals for the common key are compared. | | rev\_amt | Similarly for revenue amount, but only if @include\_revenue<>0. If revenue is not being compared (e.g., for reconciliation to credit card gateway), the revenue is internally zeroed out so it always matches. |  Common key considerations The simplest case is one in which a given value of the common key is unique in both the source and destination tables. In this case, there is no additional complexity, and the above logic is straight-forward.  However, in many cases, the available key isn't unique. In this case, the logic has to ensure that each financial number is only counted once in the exception process to generate a an exception list which exactly accounts for the aggregate delta. Very occasionally, if the data is particularly bad in terms of being scattered across periods for a common key, this creates exceptions which are confusing. So the following points summarize the key points of this logic:   * For "Not In" exceptions, the cash and revenue totals for the existing record are totalled by the common key, and only one exception generated. * For Time Skew cases:   + If a given key has matching records in the period but also out of the period, no exception is generated.   + If a given key has no matching records in the period, but matching records both before and after the period, only a before exception is generated. * For Amounts Mismatch exceptions, what is compared are the cash and revenue totals of the matching records in the period for a given key.  Variations on usage Here are few variations on usages taken from Activenet:  **Including application-specific columns in result set:**The exception table schema is entirely generic, so if it is necessary to include application-specific columns in the result set for the template, this can be done by joining back to the source or destination table using the ids. In this case, exceptions\_aa is the exception table for the AN -> AA reconciliation path, and imported\_rpt is the AN source data:  -- Detail to cut into exceptions tab select e.defect, e.adj\_rev\_amt, e.adj\_cash\_amt, a.org, a.receipt\_number,      e.description, e.src\_date, e.src\_rev\_amt, e.src\_cash\_amt, e.dst\_date, e.dst\_rev\_amt, e.dst\_cash\_amt from exceptions\_aa e left outer join imported\_rpt a     on e.src\_id=a.id order by defect desc, description, trans\_id  **Post-processing of exceptions:**The exception generating proc takes a very generic view of exceptions. However, for the Activenet to AMS reconciliation, it was desirable to add some additional columns to that result set to help development research exceptions, and to change the characterization of certain exceptions so development doesn't have to research known cases. This can be done by adding application-specific code between the generic exception generation and the result set display:  -- generate table using generic exception generator  exec sub\_gen\_exceptions      @period\_beg\_date, @period\_end\_date,      'imported\_rpt', 'org\_receipt', 'AN',      'imported\_ams', 'org\_receipt', 'AMS',      'exceptions\_ams', 0  -- Add columns to split out org and receipt number from trans\_id  alter table exceptions\_ams add org varchar(30)  update exceptions\_ams set org = LEFT(trans\_id, PATINDEX('%:%', trans\_id)-1)  alter table exceptions\_ams add receipt\_number varchar(20) update exceptions\_ams set receipt\_number = SUBSTRING(trans\_id, PATINDEX('%:%', trans\_id)+2, 100)  -- Add explanation column so we can generate some of the developer explanations  alter table exceptions\_ams add explanation varchar(100) not null default ''  -- Find any records which match ECP mismatches and turn them into non-defects  update exceptions\_ams  set description = 'ECP Failure', defect=0 where exists (select 1 from imported\_ecp ecp     where exceptions\_ams.trans\_id = ecp.org + ': ' + ecp.receipt\_number       and exceptions\_ams.src\_cash\_amt = ecp.cash\_amt)  -- Convert "AMS missing in AN" messages to "Terminal transaction" when true  update exceptions\_ams  set      description='Terminal Transaction',     defect = 0 where description='AMS missing in AN' and     (select is\_terminal\_trans from imported\_ams a where a.id = exceptions\_ams.dst\_id) = 1  -- Identify rounding errors  update exceptions\_ams set description = 'Rounding error',   explanation =  'Jira AN-21700' where description='Amounts mismatch' and abs(adj\_cash\_amt)<.10  -- Display results  -- Detail to cut into exceptions tab select defect, adj\_cash\_amt, org, receipt\_number,      description, src\_date, src\_cash\_amt, dst\_date, dst\_cash\_amt,      (select min(import\_id) from imported\_ams a where a.org\_receipt=e.trans\_id) as transaction\_id, explanation from exceptions\_ams e order by defect desc, description, trans\_id Totals generating proc The totals generating process is much simpler than the exception generating process, so the number of lines of code which are saved is a lot less. However, there is now also a generate purpose procedure for generating the totals data. It makes the same assumptions about the transaction table format, and also generates an output table which then can be used to display a result set to be pasted into the spreadsheet. Usage model Here is a simple example of totals generation from ROL. It does have a call to an ROL-specific proc to generate entries for any missing currencies of the 10 which ROL supports, to ensure it can be pasted directly into the spreadsheet:  -- Generate generic totals  exec ActiveNet.dbo.sub\_gen\_totals 'rol.dbo.imported\_pt\_merchant', @period\_beg\_date, @period\_end\_date,      'rol.dbo.totals\_pt\_merchant', ''  -- Add any records for missing currencies, so we'll be able to paste this into a spreadsheet  exec sub\_add\_missing\_currencies 'totals\_pt\_merchant', @period\_beg\_date, @period\_end\_date, ''  -- Display results  select 'PT', period\_beg\_date, period\_end\_date, inc\_red, currency\_code, cash\_total as cash from totals\_pt\_merchant order by inc\_red, currency\_code Procedure parameters  |  |  |  |  | | --- | --- | --- | --- | | **Parameter** | **Data type** | **Purpose** | **Example above** | | @transaction\_table | varchar(50) | Name of the source transaction table | 'rol.dbo.imported\_rol\_pt\_merchant' | | @period\_beg\_date | varchar(20) | First date of reconciliation period; e.g., '1/1/10' | @period\_beg\_date | | @period\_end\_date | varchar(20) | First date of month after reconciliation period; e.g., '2/1/10' | @period\_end\_date | | @totals\_table | varchar(50) | Name of totals table to create | 'rol.dbo.totals\_pt\_merchant' | | @where\_clause | varchar(255) | Additional where clause to restrict the set of records to include from period. Blank if no where clause.  (I actually don't use this anymore, and use a view on the source table instead which works better in the one case in which I did use the where clause) | ' ' |  Totals table schema The output of the total generating proc is table with a requested name, with the following schema:   |  |  |  | | --- | --- | --- | | **Column** | **Data type** | **Description** | | period\_beg\_date | datetime | Beginning date of period being reconciled | | period\_end\_date | datetime | Ending date of period being reconciled | | inc\_red | varchar(1) | inc\_red from source and destination records (which might be blank); together with currency\_code, this is the comparison grouping | | currency\_code | varchar(3) | currency\_code from the source and destination records | | cash\_total | money | Total of cash\_amt column for inc\_red / currency\_code, for selected period | | liability\_total | money | cash\_total - rev\_total, computed for convenience to be able to drop into spreadsheet | | revenue\_total | money | Total of rev\_amt column for inc\_red / currency\_code, for selected period | |