AWK - Loops

This chapter explains AWK's loops with suitable example. Loops are used to execute a set of actions in a repeated manner. The loop execution continues as long as the loop condition is true.

**For Loop**

The syntax of **for** loop is −

**Syntax**

for (initialisation; condition; increment/decrement)

action

Initially, the **for** statement performs initialization action, then it checks the condition. If the condition is true, it executes actions, thereafter it performs increment or decrement operation. The loop execution continues as long as the condition is true. For instance, the following example prints 1 to 5 using **for** loop −

**Example**

[jerry]$ awk 'BEGIN { for (i = 1; i <= 5; ++i) print i }'

On executing this code, you get the following result −

**Output**

1

2

3

4

5

**While Loop**

The **while** loop keeps executing the action until a particular logical condition evaluates to true. Here is the syntax of **while** loop −

**Syntax**

while (condition)

action

AWK first checks the condition; if the condition is true, it executes the action. This process repeats as long as the loop condition evaluates to true. For instance, the following example prints 1 to 5 using **while** loop −

**Example**

[jerry]$ awk 'BEGIN {i = 1; while (i < 6) { print i; ++i } }'

On executing this code, you get the following result −

**Output**

1

2

3

4

5

**Do-While Loop**

The **do-while** loop is similar to the while loop, except that the test condition is evaluated at the end of the loop. Here is the syntax of **do-while**loop −

**Syntax**

do

action

while (condition)

In a **do-while** loop, the action statement gets executed at least once even when the condition statement evaluates to false. For instance, the following example prints 1 to 5 numbers using **do-while** loop −

**Example**

[jerry]$ awk 'BEGIN {i = 1; do { print i; ++i } while (i < 6) }'

On executing this code, you get the following result −

**Output**

1

2

3

4

5

**Break Statement**

As its name suggests, it is used to end the loop execution. Here is an example which ends the loop when the sum becomes greater than 50.

**Example**

[jerry]$ awk 'BEGIN {

sum = 0; for (i = 0; i < 20; ++i) {

sum += i; if (sum > 50) break; else print "Sum =", sum

}

}'

On executing this code, you get the following result −

**Output**

Sum = 0

Sum = 1

Sum = 3

Sum = 6

Sum = 10

Sum = 15

Sum = 21

Sum = 28

Sum = 36

Sum = 45

**Continue Statement**

The **continue** statement is used inside a loop to skip to the next iteration of the loop. It is useful when you wish to skip the processing of some data inside the loop. For instance, the following example uses **continue** statement to print the even numbers between 1 to 20.

**Example**

[jerry]$ awk 'BEGIN {

for (i = 1; i <= 20; ++i) {

if (i % 2 == 0) print i ; else continue

}

}'

On executing this code, you get the following result −

**Output**

2

4

6

8

10

12

14

16

18

20

**Exit Statement**

It is used to stop the execution of the script. It accepts an integer as an argument which is the exit status code for AWK process. If no argument is supplied, **exit** returns status zero. Here is an example that stops the execution when the sum becomes greater than 50.

**Example**

[jerry]$ awk 'BEGIN {

sum = 0; for (i = 0; i < 20; ++i) {

sum += i; if (sum > 50) exit(10); else print "Sum =", sum

}

}'

**Output**

On executing this code, you get the following result −

Sum = 0

Sum = 1

Sum = 3

Sum = 6

Sum = 10

Sum = 15

Sum = 21

Sum = 28

Sum = 36

Sum = 45

Let us check the return status of the script.

**Example**

[jerry]$ echo $?

On executing this code, you get the following result −

**Output**

10

**AWK - Built-in Functions**

AWK has a number of functions built into it that are always available to the programmer. This chapter describes Arithmetic, String, Time, Bit manipulation, and other miscellaneous functions with suitable examples.

|  |  |
| --- | --- |
| **S.No.** | **Built in functions & Description** |
| 1 | [**Arithmetic Functions**](https://www.tutorialspoint.com/awk/awk_arithmetic_functions.htm)  AWK has the following built-in arithmetic functions. |
| 2 | [**String Functions**](https://www.tutorialspoint.com/awk/awk_string_functions.htm)  AWK has the following built-in String functions. |
| 3 | [**Time Functions**](https://www.tutorialspoint.com/awk/awk_time_functions.htm)  AWK has the following built-in time functions. |
| 4 | [**Bit Manipulation Functions**](https://www.tutorialspoint.com/awk/awk_bit_manipulation_functions.htm)  AWK has the following built-in bit manipulation functions. |
| 5 | [**Miscellaneous Functions**](https://www.tutorialspoint.com/awk/awk_miscellaneous_functions.htm)  AWK has the following miscellaneous functions. |

**AWK - Arithmetic Functions**

AWK has the following built-in arithmetic functions −

**atan2(y, x)**

It returns the arctangent of (y/x) in radians. The following example demonstrates this −

**Example**

[jerry]$ awk 'BEGIN {

PI = 3.14159265

x = -10

y = 10

result = atan2 (y,x) \* 180 / PI;

printf "The arc tangent for (x=%f, y=%f) is %f degrees\n", x, y, result

}'

On executing this code, you get the following result −

**Output**

The arc tangent for (x=-10.000000, y=10.000000) is 135.000000 degrees

**cos(expr)**

This function returns the cosine of **expr**, which is expressed in radians. The following example demonstrates this −

**Example**

[jerry]$ awk 'BEGIN {

PI = 3.14159265

param = 60

result = cos(param \* PI / 180.0);

printf "The cosine of %f degrees is %f.\n", param, result

}'

On executing this code, you get the following result −

**Output**

The cosine of 60.000000 degrees is 0.500000.

**exp(expr)**

This function is used to find the exponential value of a variable.

**Example**

[jerry]$ awk 'BEGIN {

param = 5

result = exp(param);

printf "The exponential value of %f is %f.\n", param, result

}'

On executing this code, you get the following result −

**Output**

The exponential value of 5.000000 is 148.413159.

**int(expr)**

This function truncates the **expr** to an integer value. The following example demonstrates this −

[jerry]$ awk 'BEGIN {

param = 5.12345

result = int(param)

print "Truncated value =", result

}'

On executing this code, you get the following result −

Truncated value = 5

**log(expr)**

This function calculates the natural logarithm of a variable.

**Example**

[jerry]$ awk 'BEGIN {

param = 5.5

result = log (param)

printf "log(%f) = %f\n", param, result

}'

On executing this code, you get the following result −

**Output**

log(5.500000) = 1.704748

**rand**

This function returns a random number N, between 0 and 1, such that 0 <= N < 1. For instance, the following example generates three random numbers

**Example**

[jerry]$ awk 'BEGIN {

print "Random num1 =" , rand()

print "Random num2 =" , rand()

print "Random num3 =" , rand()

}'

On executing this code, you get the following result −

**Output**

Random num1 = 0.237788

Random num2 = 0.291066

Random num3 = 0.845814

**sin(expr)**

This function returns the sine of **expr**, which is expressed in radians. The following example demonstrates this −

**Example**

[jerry]$ awk 'BEGIN {

PI = 3.14159265

param = 30.0

result = sin(param \* PI /180)

printf "The sine of %f degrees is %f.\n", param, result

}'

On executing this code, you get the following result −

**Output**

The sine of 30.000000 degrees is 0.500000.

**sqrt(expr)**

This function returns the square root of **expr**.

**Example**

[jerry]$ awk 'BEGIN {

param = 1024.0

result = sqrt(param)

printf "sqrt(%f) = %f\n", param, result

}'

On executing this code, you get the following result −

**Output**

sqrt(1024.000000) = 32.000000

**srand([expr])**

This function generates a random number using seed value. It uses **expr** as the new seed for the random number generator. In the absence of expr, it uses the time of day as the seed value.

**Example**

[jerry]$ awk 'BEGIN {

param = 10

printf "srand() = %d\n", srand()

printf "srand(%d) = %d\n", param, srand(param)

}'

On executing this code, you get the following result−

**Output**

srand() = 1

srand(10) = 1417959587

**AWK - String Functions**

AWK has the following built-in String functions −

**asort(arr [, d [, how] ])**

This function sorts the contents of **arr** using GAWK's normal rules for comparing values, and replaces the indexes of the sorted values **arr** with sequential integers starting with 1.

**Example**

[jerry]$ awk 'BEGIN {

arr[0] = "Three"

arr[1] = "One"

arr[2] = "Two"

print "Array elements before sorting:"

for (i in arr) {

print arr[i]

}

asort(arr)

print "Array elements after sorting:"

for (i in arr) {

print arr[i]

}

}'

On executing this code, you get the following result −

**Output**

Array elements before sorting:

Three

One

Two

Array elements after sorting:

One

Three

Two

**asorti(arr [, d [, how] ])**

The behavior of this function is the same as that of **asort()**, except that the array indexes are used for sorting.

**Example**

[jerry]$ awk 'BEGIN {

arr["Two"] = 1

arr["One"] = 2

arr["Three"] = 3

asorti(arr)

print "Array indices after sorting:"

for (i in arr) {

print arr[i]

}

}'

On executing this code, you get the following result −

**Output**

Array indices after sorting:

One

Three

Two

**gsub(regex, sub, string)**

**gsub** stands for global substitution. It replaces every occurrence of **sub** with **regex**. The third parameter is optional. If it is omitted, then $0 is used.

**Example**

[jerry]$ awk 'BEGIN {

str = "Hello, World"

print "String before replacement = " str

gsub("World", "Jerry", str)

print "String after replacement = " str

}'

On executing this code, you get the following result −

**Output**

String before replacement = Hello, World

String after replacement = Hello, Jerry

**index(str, sub)**

It checks whether **sub** is a substring of **str** or not. On success, it returns the position where sub starts; otherwise it returns 0. The first character of **str** is at position 1.

**Example**

[jerry]$ awk 'BEGIN {

str = "One Two Three"

subs = "Two"

ret = index(str, subs)

printf "Substring \"%s\" found at %d location.\n", subs, ret

}'

On executing this code, you get the following result −

**Output**

Substring "Two" found at 5 location.

**length(str)**

It returns the length of a string.

**Example**

[jerry]$ awk 'BEGIN {

str = "Hello, World !!!"

print "Length = ", length(str)

}'

On executing this code, you get the following result −

Length = 16

**match(str, regex)**

It returns the index of the first longest match of **regex** in string **str**. It returns 0 if no match found.

**Example**

[jerry]$ awk 'BEGIN {

str = "One Two Three"

subs = "Two"

ret = match(str, subs)

printf "Substring \"%s\" found at %d location.\n", subs, ret

}'

On executing this code, you get the following result −

**Output**

Substring "Two" found at 5 location

**split(str, arr, regex)**

This function splits the string **str** into fields by regular expression **regex** and the fields are loaded into the array **arr**. If **regex** is omitted, then FS is used.

**Example**

[jerry]$ awk 'BEGIN {

str = "One,Two,Three,Four"

split(str, arr, ",")

print "Array contains following values"

for (i in arr) {

print arr[i]

}

}'

On executing this code, you get the following result −

**Output**

Array contains following values

One

Two

Three

Four

**sprintf(format, expr-list)**

This function returns a string constructed from **expr-list** according to format.

**Example**

[jerry]$ awk 'BEGIN {

param = 1024.0

result = sqrt(param)

printf "sqrt(%f) = %f\n", param, result

}'

On executing this code, you get the following result −

**Output**

Hello, World !!!

**strtonum(str)**

This function examines **str** and return its numeric value. If str begins with a leading 0, it is treated as an octal number. If str begins with a leading 0x or 0X, it is taken as a hexadecimal number. Otherwise, assume it is a decimal number.

**Example**

[jerry]$ awk 'BEGIN {

print "Decimal num = " strtonum("123")

print "Octal num = " strtonum("0123")

print "Hexadecimal num = " strtonum("0x123")

}'

On executing this code, you get the following result −

**Output**

Decimal num = 123

Octal num = 83

Hexadecimal num = 291

**sub(regex, sub, string)**

This function performs single substitution. It replaces the first occurrence of **sub** with **regex**. The third parameter is optional. If it is omitted, $0 is used.

**Example**

[jerry]$ awk 'BEGIN {

str = "Hello, World"

print "String before replacement = " str

sub("World", "Jerry", str)

print "String after replacement = " str

}'

On executing this code, you get the following result −

**Output**

String before replacement = Hello, World

String after replacement = Hello, Jerry

**substr(str, start, l)**

This function returns the substring of string **str**, starting at index **start** of length **l**. If length is omitted, the suffix of **str** starting at index **start** is returned.

**Example**

[jerry]$ awk 'BEGIN {

str = "Hello, World !!!"

subs = substr(str, 1, 5)

print "Substring = " subs

}'

On executing this code, you get the following result −

**Output**

Substring = Hello

**tolower(str)**

This function returns a copy of string **str** with all upper-case characters converted to lower-case.

**Example**

[jerry]$ awk 'BEGIN {

str = "HELLO, WORLD !!!"

print "Lowercase string = " tolower(str)

}'

On executing this code, you get the following result −

**Output**

Lowercase string = hello, world !!!

**toupper(str)**

This function returns a copy of string **str** with all lower-case characters converted to upper case.

**Example**

[jerry]$ awk 'BEGIN {

str = "hello, world !!!"

print "Uppercase string = " toupper(str)

}'

On executing this code, you get the following result −

**Output**

Uppercase string = HELLO, WORLD !!!

**Medical Billing Terminology**

**A**

**Allowed Amount:** The sum an insurance company will reimburse to cover a healthcare service or procedure. The patient typically pays the remaining balance if there is any amount left over after the allowed amount has been paid. This amount should not to be confused with co-pay or deductibles owed by a patient.

**American Medical Association (AMA):** The AMA is the largest organization of physicians in the U.S. dedicated to improving the quality of healthcare administered by providers across the country. The current procedural technology (CPT) code set is maintained and revised by the AMA in accordance with federal guidelines.

**Aging:**A formal medical billing term that refers to insurance claims that haven’t been paid or balances owed by patients overdue by more than 30 days. Aging claims may become denied if they aren’t filed in time with a health insurance company.

**Ancillary Services:** Any service administered in a hospital or other healthcare facility other than room and board, including biometrics tests, physical therapy, and physician consultations among other services.

**Appeal:**Appeal occurs when a patient or a provider tries to convince an insurance company to pay for healthcare after it has decided not to cover costs for someone on a claim. Medical billing specialists deal with appeals after a claim has been denied or rejected by an insurance company.

**Applied to Deductible (ATD):**This term refers to the amount of money a patient owes a provider that goes to paying their yearly deductible. A patient’s deductible is determined by their insurance plan and can range in price.

**Assignment of Benefits (AOB):**This term refers to insurance payments made directly to a healthcare provider for medical services received by the patient. Assignment of benefits occurs after a claim has been successfully processed with an insurance company.

**Application Service Provider (ASP):** ASP is a digital network that allows healthcare providers to access quality medical billing software and technologies without needing to purchase and maintain it themselves. Providers who use ASP typically pay a monthly fee to the company that maintains the billing software.

**Authorization:**This term refers to when a patient’s health insurance plan requires them to get permission from their insurance providers before receiving certain healthcare services. A patient may be denied coverage if they see a provider for a service that needed authorization without first consulting the insurance company.

**B**

**Beneficiary:**The beneficiary is the person who receives benefits and/or coverage under a healthcare plan. The beneficiary of an insurance plan may not be the person paying for the plan, as is the case for young children covered under their parents’ plans.

**Blue Cross Blue Shield:**Blue Cross Blue Shield is a federation of 38 health insurance companies in the U.S. (some of which are non-profit companies) that offer health insurance options to eligible persons in their area. Blue Cross Blue Shield offers healthcare plans to over 100 million people in the U.S.

**C**

**Capitation:**A fixed payment that a patient makes to a health insurance company or provider to recoup costs incurred from various healthcare services. A capitation is different from a deductible or co-pay.

**Civilian Health and Medical Program of Uniform Services (CHAMPUS):**CHAMPUS (now known as TRICARE) is the federal health insurance program for active and retired service members, their families, and the survivors of service members.

**Charity Care:**This type of care is administered at reduced or zero cost to patients who cannot afford healthcare. Providers may offer charity care at their discretion.

**Clean Claim:**This refers to a medical claim filed with a health insurance company that is free of errors and processed in a timely manner. Some providers may send claims to organizations that specialize in producing clean claims, like clearinghouses.

**Clearinghouse:**Clearinghouses are facilities that review and correct medical claims as necessary before sending them to insurance companies for final processing. This meticulous editing process for claims is known in the medical billing industry as “scrubbing.”

**Centers for Medicare and Medicaid Services (CMS):**The CMS is the federal entity that manages and administers healthcare coverage through Medicare and Medicaid. CMS coordinates with providers and enrollees to provide healthcare to over 100 million Americans.

**CMS 1500:**The CMS 1500 is a paper medical claim form used for transmitting claims based on coverage by Medicare and Medicaid plans. Commercial insurance providers often require that providers use CMS 1500 forms to process their own paper claims.

**Coding:**Coding is the process of translating a physician’s documentation about a patient’s medical condition and health services rendered into medical codes that are then plugged into a claim for processing with an insurance company. Medical billing specialists must be familiar with many code sets in order to perform their job duties.

**COBRA Insurance:**A federal program that allows a person terminated from their employer to retain health insurance they had with that employer for up to 18 months, or 36 months if the former employee is disabled.

**Co-Insurance:**The percentage of coverage that a patient is responsible for paying after an insurance company pays the portion agreed upon in a health plan. Co-insurance percentages vary depending on the health plan.

**Collection Ratio:**This refers to the ratio of payments received relative to the total amount owed to providers.

**Contractual Adjustment:**This refers to a binding agree between a provider, patient, and insurance company wherein the provider agrees to charges that it will write off on behalf of the patient. Contractual adjustments may occur when there is a discrepancy between what a provider charges for healthcare services and what an insurance company has decided to pay for that service.

**Coordination of Benefits (COB):**COB occurs when a patient is covered by more than one insurance plan. In this situation one insurance company will become the primary carrier and all other companies will be considered secondary and tertiary carriers that may cover costs left after the primary carrier has paid.

**Co-Pay:**A patient’s co-pay is the amount that must be paid to a provider before they receive any treatment or services. Co-pays are separate from a deductible, and will vary depending on a person’s insurance plan.

**Current Procedural Technology (CPT) Code:**CPT codes represent treatments and procedures performed by a physician in a 5-digit format. CPT codes are entered together with ICD-9 codes that explain a patient’s diagnosis. Medical billing specialists will enter CPT codes into claims so insurance companies understand the nature of healthcare a patient received with a provider.

**Credentialing:**The application process for a provider to coordinate with an insurance company. Once providers have become credentialed with an insurance company, they have the opportunity to work with that company in providing affordable healthcare to patients.

**Credit Balance:**Refers to the sum shown in the “balance” column of a billing statement that reflects the amount due for services rendered.

**Crossover Claim:**When claim information is sent from a primary insurance carrier to a secondary insurance carrier, or vice versa.

**D**

**Date of Service (DOS):**The date when a provider performed healthcare services and procedures.

**Day Sheet:** A document that summarizes the services, treatments, payments, and charges that a patient received on a given day.

**Deductible:**The amount a patient must pay before an insurance carrier starts their healthcare coverage. Deductibles range in price according to terms set in a person’s health plan.

**Demographics:**The patient’s information required for filing a claim, such as age, sex, address, and family information. An insurance company may deny a claim if it contains inaccurate demographics.

**Durable Medical Equipment (DME):**This refers to medical implements that can be reused such as stretchers, wheelchairs, canes, crutches, and bedpans.

**Date of Birth (DOB):** The exact date a patient was born.

**Downcoding:**Downcoding occurs when an insurance company finds there is insufficient evidence on a claim to prove that a provider performed coded medical services and so they reduce or remove those codes. Downcoding usually reduces the cost of a claim.

**Duplicate Coverage Inquiry (DCI):**A formal request typically submitted by an insurance carrier to determine if other health coverage exists for a patient.

**Dx:**The abbreviation for diagnosis codes, also known as ICD-9 codes.

**E**

**Electronic Claim:**A claim sent electronically to an insurance carrier from a provider’s billing software. The format of electronic claims must adhere to medical billing regulations set forth by the federal government.

**Electronic Funds Transfer:**A method of transferring money electronically from a patient’s bank account to a provider or an insurance carrier.

**Evaluation and Management (E/M):**E/M refers to the section of CPT codes most used by healthcare personnel to describe a patient’s medical needs.

**Electronic Medical Records (EMR):** EMR is a digitized medical record for a patient managed by a provider onsite. EMRs may also be referred to as electronic health records (EHRs).

**Enrollee:**A person covered by a health insurance plan.

Explanation of Benefits (EOB):A document attached to a processed medical claim wherein the insurance company explains the services they will cover for a patient’s healthcare treatments. EOBs may also explain what is wrong with a claim if it’s denied.

**Electronic Remittance Advice (ERA):**The digital version of EOB, which specifies the details of payments made on a claim either by an insurance company or required by the patient.

**ERISA:**Stands for the Employee Retirement Income Security Act of 1974. This act established guidelines and requirements for health and life insurance policies including appeals and disclosure of grievances.

**F**

**Fee for Service:**This refers to a type of health insurance wherein the provider is paid for every service they perform. People with fee-for-service plans typically can choose whatever hospitals and physicians they want to receive care in exchange for higher deductibles and co-pays.

**Fee Schedule:**A document that outlines the costs associated for each medical service designated by a CPT code.

**Financial Responsibility:**Whoever owes the healthcare provider money has financial responsibility for the services rendered. Insurance companies or patients themselves may be financially responsible for the costs associated with care, and these responsibilities are typically outlined in a healthcare plan contract.

**Fiscal Intermediary (FI):**The name for Medicare representatives who process Medicare claims.

**Formulary:**A table or list provided by an insurance carrier that explains what prescription drugs are covered under their health plans.

**Fraud:**Providers, patients, or insurance companies may be found fraudulent if they are deliberately achieving their ends through misrepresentation, dishonesty, and general illegal activity. Medical billing specialists who deliberately enter incorrect or misleading information on claims may be charged with fraud.

**G**

**Group Health Plan (GPH):**A plan provided by an employer to provide healthcare options to a large group of employees.

**Group Name:**The name of the group, insurance carrier, or insurance plan that covers a patient.

**Group Number:**A number given to a patient by their insurance carrier that identifies the group or plan under which they are covered.

**Guarantor:**The party paying for an insurance plan who is not the patient. Parents, for example, would be the guarantors for their children’s health insurance.

**H**

**Healthcare Financing Administration:**The former name for what is now the CMS.

**Healthcare Financing Administration Common Procedure Coding System (HCPCS):** HCPCS is a three-tier coding system used to explain services, devices, and diagnoses administered in the healthcare system. Medical billing specialists utilize codes in the HCPCS on a daily basis to file claims.

**Healthcare Insurance:**This is insurance offered to a group or an individual to cover costs associated with medical care and treatment. Those covered by healthcare insurance typically must pay a premium for their plans in addition to various co-pays and/or deductibles.

**Healthcare Provider:**These are the entities that offer healthcare services to patients, including hospitals, physicians, and private clinics, hospices, nursing homes, and other healthcare facilities.

**Healthcare Reform Act:**The major healthcare legislation passed in 2010 designed to make healthcare accessible and less expensive for more Americans.

**Health Insurance Claim:**The unique number ascribed to an individual to identify them as a beneficiary of Medicare.

**Health Insurance Portability and Accountability Act (HIPAA):** HIPAA was a law passed in 1996 with an aim to improve the scope of healthcare services and establish regulations for securing healthcare records from unwanted parties.

**Health Maintenance Organization (HMO):**HMOs are networks of healthcare providers that offer healthcare plans to people for medical services exclusively in their network.

**Hospice:**This refers to medical care and treatment for persons who are terminally ill.

**I**

**ICD-9 Codes:**ICD-9 codes are an international set of codes that represent diagnoses of patients’ medical conditions as determined by physicians. Medical billing specialists may translate a physician’s diagnoses into ICD-9 codes and then input those codes into a claim for processing.

**ICD-10 Codes:** ICD-10 codes are the updated international set of codes based on the preceding ICD-9 codes. ICD-10 codes are estimated to be mandatory in the American healthcare system by October 2014.

Incremental Nursing Charge:A fee for nursing services a patient is charged during the course of receiving healthcare.

**Indemnity:**A type of health insurance plan whereby a patient can receive care with any provider in exchange for higher deductibles and co-pays. Indemnity is also known as fee-for-service insurance.

**In-Network:**This term refers to a provider’s relationship with a health insurance company. A group of providers may contract with an insurance company to form a network of healthcare professionals that a person can choose from when enrolled in that insurance company’s health plan.

**Inpatient:**Inpatient care occurs when a person has a stay at a healthcare facility for more than 24 hours.

Independent Practice Association (IPA):The IPA is a professional organization of physicians who have a contract with an HMO.

**Intensive Care:**Intensive care is the unit of a hospital reserved for patients that need immediate treatment and close monitoring by healthcare professionals for serious illnesses, conditions, and injuries.

**M**

**Medicare Administrative Contractor (MAC):** MACs are contract with the federal government to process Medicare claims.

**Managed Care Plan:**A health insurance plan whereby patients can only receive coverage if they see providers who operate in the insurance company’s network.

**Maximum Out of Pocket:**The amount a patient is required to pay. After a patient reaches their maximum out of pocket, their healthcare costs should be covered by their plan.

**Medical Assistant:**An employee in the healthcare system such as a physian’s assistant or a nurse practitioner who perform duties in administration, nursing, and other ancillary care.

**Medical Coder:**A medical coder is responsible for assigning various medical codes to services and healthcare plans described by a physician on a patient’s superbill.

**Medical Billing Specialist:**A medical billing specialist is responsible for using information regarding services and treatments performed by a healthcare provider to complete a claim for filing with an insurance company so the provider can be paid.

**Medical Necessity:**This term refers to healthcare services or treatments that a patient requires to treat a serious medical condition or illness. This does not include cosmetic or investigative services.

**Medical Record Number:**A unique number ascribed to a person’s medical record so it can be differentiated from other medical records.

**Medicare Secondary Payer:**The insurance company that covers any remaining expenses after Medicare has paid for a patient’s coverage.

**Medical Savings Account (MSA):**AnMSA is an optional health insurance payments plan whereby a person apportions part of their untaxed earnings to an account reserved for healthcare expenses. A person with an MSA can only contribute a certain amount of their earnings per year. Any unused funds in an MSA at the end of the year will roll over to the next.

**Medical Transcription:**The process of converting dictated or handwritten instructions, observations, and documentation into digital text formats.

**Medicare:**Medicare is a government insurance program started in 1965 to provide healthcare coverage for persons over 65 and eligible people with disabilities.

**Medicare Coinsurance Days:**Referring to 61st through 90th days of inpatient treatment, the law requires that patients pay for a portion of their healthcare during Medicare coinsurance days.

**Medicare Donut Hole:**This term refers to the discrepancy between the limits of healthcare insurance coverage and the Medicare Part D coverage limits for prescription drugs.

**Medicaid:**Medicaid is a joint federal and state assistance program started in 1965 to provide health insurance to lower-income persons. Both state and federal governments fund Medicaid programs, but each state is responsible for running its own version of Medicaid within the minimum requirements established by federal law.

**Medigap:**Medigap is supplemental health insurance under Medicaid for eligible persons who need help covering co-pays, deductibles, and other large fees.

**Modifier:**Modifiers are additions to CPT codes that explain alterations and modifications to an otherwise routine treatment, exam, or service.

**N**

**Non-Covered Charge (N/C):**N/Cs are procedures and services not covered by a person’s health insurance plan.

**Not Elsewhere Classifiable (NEC):**A term used to describe a procedure or service that can’t be described within the available code set.

**Network Provider:**A provider within a health insurance company’s network that has contracted with the company to provide discounted services to a patient covered under the company’s plan.

**Non-participation:**This is when a provider refuses to accept Medicare payments as a sufficient amount for the services rendered to a patient.

**Not Otherwise Specified (NOS):**This term is used in ICD-9 codes to describe conditions with unspecified diagnoses.

**National Provider Identifier (NPI) Number:**A unique 10-digit number ascribed to every healthcare provider in the U.S. as mandated by HIPAA.

**O**

**Office of Inspector General (OIG):**The organization responsible for establishing guidelines and investigating fraud and misinformation within the healthcare industry. The OIG is part of the Department of Health and Human Services.

**Out-of-Network:**Out-of-network refers to providers outside of an established network of providers who contract with an insurance company to offer patients healthcare at a discounted rate. People who go to out-of-network providers typically have to pay more money to receive care.

**Outpatient:**This term refers to healthcare treatment that doesn’t require an overnight hospital stay, including a routine visit to a primary care doctor or a non-invasive surgery.

**P**

**Palmetto GBA:**A MAC based in Columbia, South Carolina that is also a subsidiary of Blue Cross Blue Shield.

Patient Responsibility:This refers to the amount a patient owes a provider after an insurance company pays for their portion of the medical expenses.

**Primary Care Physician (PCP):**The physician who provides the basic healthcare services for a patient and recommends additional care for more serious treatments as necessary.

**Point of Service Plans:**A plan whereby patients with HMO membership may receive care at non-HMO providers in exchange for a referral and paying a higher deductible.

**Place of Service Code:**A two-digit code used on claims to explain what type of provider performed healthcare services on a patient.

**Preferred Provider Organization (PPO):**A plan similar to an HMO whereby a patient can receive healthcare from providers within an established network set up by an insurance company.

**Practice Management Software:**Software used for scheduling, billing, and recordkeeping at a provider’s office.

**Preauthorization:**Some insurance plans require that a patient receive preauthorization from the insurance company prior to receiving certain medical services to make sure the company will cover expenses associated with those services.

**Pre-Certification:**A process similar to preauthorization whereby patients must check with insurance companies to see if a desired healthcare treatment or service is deemed medically necessary (and thus covered) by the company.

**Pre-determination:**A maximum sum as explained in a healthcare plan an insurance company will pay for certain services or treatments.

**Pre-existing Condition (PEC):**PEC is a medical condition a patient had before receiving coverage from an insurance company. A person might become ineligible for certain healthcare plans depending on the severity and length of their PEC.

**Pre-exisiting Condition Exclusion:**The existence of a PEC denies a person certain coverage in some health insurance plans.

**Premium:**The sum a person pays to an insurance company on a regular (usually monthly or yearly) basis to receive health insurance.

**Privacy Rule:**Standards for privacy regarding a patient’s medical history and all related events, treatments, and data as outlined by HIPAA.

**Provider:**A provider is the healthcare facility that administered healthcare to an individual. Physicians, clinics, and hospitals are all considered providers.

**Provider Transaction Access Number (PTAN):**This refers to a provider’s current legacy provider number with Medicare.

**R**

**Referral:**This is when a provider recommends another provider to a patient to receive specialized treatment.

**Remittance Advice (R/A):**The R/A is also known as the EOB, which is the document attached to a processed claim that explains the information regarding coverage and payments on a claim.

**Responsible Party:**The person who pays for a patient’s medical expenses, also known as the guarantor.

**Revenue Code:**A three-digit code used on medical bills that explains the kind of facility in which a patient received treatment.

**Relative Value Amount (RVA):**The median amount Medicare will repay a provider for certain services and treatments.

**S**

**Scrubbing:**A process by which insurance claims are checked for errors before being sent to an insurance company for final processing. Providers scrub claims in an attempt to reduce the number of denied or rejected claims.

**Self-Referral:**When a patient does their own research to find a provider and acts outside of their primary care physician’s referral.

**Self-Pay:**Payment made by the patient for healthcare at the time they receive it at a provider’s facilities.

**Secondary Insurance Claim:**The claim filed with the secondary insurance company after the primary insurance company pays for their portion of healthcare costs.

**Secondary Procedure:**This is when provider performs another procedure on a patient covered by a CPT code after first performing a different CPT procedure on them.

**Security Standard:**The security standard serves as the guidelines for policies and practices necessary to reduce security risks within the healthcare system. The security standard policies work in concert with the security guidelines set in place with the passage of HIPAA.

**Skilled Nursing Facility:**These are facilities for the severely ill or elderly that provide specialized long-term care for recovering patients. Skilled nursing facilities are alternative healthcare establishments to extended hospital stays and may be covered by eligible patients’ insurance policies.

**Signature on File (SOF):**A patient’s official signature on file for the purpose of billing and claims processing.

**Software as a Service (SAAS):**Medical billing software hosted off site by another company and only accessible with Internet access. SAAS is useful for providers who don’t want to maintain and update in-house medical billing software.

**Specialist:**A physician or medical assistant with expertise in a specific area of medicine. Oncologists, pediatricians, and neurologists are among the many specialists in the medical field.

**Subscriber:**The subscriber is the individual covered under a group policy. For instance, an employee of a company with a group health policy would be one of many subscribers on that policy.

**Superbill:**A document used by healthcare staff and physicians to write down information about a patient receiving care. The superbill can contain demographic information, insurance information, and especially any diagnoses or healthcare plans written by the physician. A medical billing specialist inputs the information on a patient’s superbill into a claim.

**Supplemental Insurance:**Supplemental insurance can be a secondary policy or another insurance company that covers a patient’s healthcare costs after receiving coverage from their primary insurance. Supplemental insurance policies typically help patients cover expensive deductibles and copays.

**T**

**Treatment Authorization Request (TAR):**A unique number the insurance company gives the provider for billing purposes. A provider must receive the insurance company’s TAR number before administering healthcare to a patient covered by the company.

**Taxonomy Code:**Medical billing specialists utilize this unique codeset for identifying a healthcare provider’s specialty field.

**Term Date:**The end date for an insurance policy contract, or the date after which a person no longer receives or is no longer eligible for health insurance with company. Term dates are typically determined on a case-by-case basis.

**Tertiary Insurance Claim:**A claim filed by a provider after they have filed claims for primary and secondary health insurance coverage on behalf of a patient. Tertiary insurance claims often cover the remaining healthcare costs such as deductibles and co-pays left over after the primary and secondary claims have been processed.

**Third Party Administrator (TPA):**The name for the organization or individual that manages healthcare group benefits, claims, and administrative duties on behalf of a group plan or a company with a group plan.

**Tax Identification Number (TIN):**A unique number a patient or a company may have to produce for billing purposes in order to receive healthcare from a provider. The TIN is also known as the employment identification number (EIN).

**Triple Option Plan (TOP):**Also referred to as the cafeteria plan, this plan gives an enrolled individual the options to choose between an HMO, a PPO, or a traditional point of service plan for their health insurance. Some companies offer triple option plans to their employees to accommodate the needs of a diverse staff.

**Type of Service (TOS):**A field on a claim for describing what kind of healthcare services or procedures a provider administered.

**TRICARE:**TRICARE is the federal health insurance plan for active service members, retired service members, and their families, in addition to survivors of service members. TRICARE was previously known as CHAMPUS.

**U**

**UB04:**A form used by providers for filing claims with insurance companies. The UB04 form has a format similar to that of the CMS 1500 form.

**Unbundling:**This term refers to the fraudulent practice of ascribing more than one code to a service or procedure on a superbill or claim form when only one is necessary.

**Untimely Submission:**Claims have a specific timeframe in which they can be sent off to an insurance company for processing. If a provider fails to file a claim with an insurance company in that timeframe, it is marked for untimely submission and will be denied by the company.

**Upcoding:**Upcoding is the fraudulent practice of ascribing a higher ICD-9 code to a healthcare procedure in an attempt to get more money than necessary from the insurance company or patient.

**Unique Physician Identification Number (UPIN):**A unique six-digit identification number given to physicians and other healthcare personnel, which has subsequently been replaced by a national provider identifier (NPI) number.

**Usual Customary and Reasonable (UCR):**The UCR is the amount of money stipulated in a contract that an insurance company agrees to pay for healthcare costs. After passing the UCR a patient is typically responsible for covering their healthcare costs.

**Utilization Limit:**The limit per year for coverage under certain available healthcare services for Medicare enrollees. Once a patient passes the utilization limit for a service, Medicare may no longer cover them.

**Utilization Review (UR):**An investigation or audit performed to optimize the number of inpatient and outpatient services a provider performs.

**V**

**V-Codes:**A codeset under ICD-9-CM used to organize healthcare services rendered for reasons other than illness or injury.

**W**

**Worker’s Compensation:**Worker’s compensation is paid by an employer when an employee becomes ill or injured while performing routine job duties. Most states have laws requiring that companies provide worker’s compensation.

**Write-Off:**This term refers to the discrepancy between a provider’s fee for healthcare services and the amount that an insurance company is willing to pay for those services that a patient is not responsible for. The write-off amount may be categorized as “not covered” amounts for billing purposes.

|  |  |
| --- | --- |
| **Output is Deterministic** | Specify only when the source output does not change between session runs. |
| **Output is Repeatable** | Specify only when the order of the source output is same between the session runs. |

To be simple on these two terms, deterministic (means, the output will remains the same with the same input with every session run) and Repeatable (means, the order of the output remains the same for the same input with every session run). So if it’s true, then this session, can be recovered, otherwise not. These constitute, that that there will no change, across every run.

**Output is Deterministic:**

Relational source or transformation output that does not change between session runs when the input data is consistent between runs. When you configure this property, the Integration Service does not stage source data for recovery if transformations in the pipeline always produce repeatable data.

**Output is Repeatable:**

Relational source or transformation output that is in the same order between session runs when the order of the input data is consistent. When output is deterministic and output is repeatable, the Integration Service does not stage source data for recovery.

As per my understanding, if you configure a session for recovery integration service will store the source data in shared storage directory whenever you run the session. This is regular process. Whenever the session gets failed and if you rerun the session in recovery mode it will process the data from shared storage directory but not from your actual source table.

If you select these options ('Output is Deterministic' and 'Output is  Repeatable') in Source Qualifier transformation, the integration service does not store the data in shared storage directory every time. It will store the data in shared storage directory for the first time and it will use that data in subsequent session runs.

By specifying these options you are telling to integration service that the data is same between session runs.

**Change Data Capture (CDC) Made Easy (Using Mapping Variables)**

At times we may need to implement [Change Data Capture](http://www.disoln.org/2013/03/Change-Data-Capture-Implementation-for-Multi-Sourced-ETL-Process.html) for small data integration projects which includes just couple of workflows.  Introducing a [Change Data Capture framework](http://www.disoln.org/2012/10/An-ETL-Framework-for-Change-Data-Capture-CDC.html) for such project is not a recommended way to handle this, just because of the efforts required to build the framework may not be justified. Here in this article lets discuss about a simple, easy approach handle Change Data Capture.

We will be  using Informatica Mapping Variables to building our Change Data Capture logic. Before even we talk about the implementation, lets understand the Mapping Variable

## Informatica Mapping Variable

### What is Mapping Variable

These are variables created in PowerCenter Designer, which you can use in any expression in a mapping, and you can also use the mapping variables in a source qualifier filter, user-defined join, or extract override, and in the Expression Editor of reusable transformations.

### Mapping Variable Starting Value

Mapping variable can take the starting value from

* 1. Parameter file
  2. Pre-session variable assignment
  3. Value saved in the repository
  4. Initial value
  5. Default Value

The Integration Service looks for the start value in the order mentioned above. Value of the mapping variable can be changed with in the session using an expression and the final value of the variable will be saved into the repository. The saved value from the repository is retrieved in the next session run and used as the session start value.

### Setting Mapping Variable Value

You can change the mapping variable value with in the mapping or session using the Set Function. We need to use the set function based on the Aggregation Type of the variable. Aggregation Type of the variable can be set when the variable is declared in the mapping.

* **SetMaxVariable.** Sets the variable to the maximum value of a group of values. To use the SetMaxVariable with a mapping variable, the aggregation type of the mapping variable must be set to Max.
* **SetMinVariable.**Sets the variable to the minimum value of a group of values. use the SetMinVariable with a mapping variable, the aggregation type of the mapping variable must be set to Min.
* **SetCountVariable.**Increments the variable value by one. In other words, it adds one to the variable value when a row is marked for insertion, and subtracts one when the row is marked for deletion. To use the SetCountVariable with a mapping variable, the aggregation type of the mapping variable must be set to Count.
* **SetVariable.**Sets the variable to the configured value. At the end of a session, it compares the final current value of the variable to the start value of the variable. Based on the aggregate type of the variable, it saves a final value to the repository.

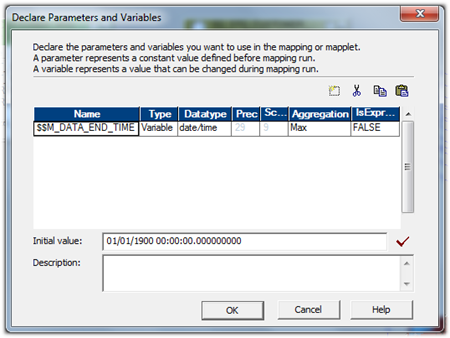
## Change Data Capture Implementation

Now we understand the mapping variables, lets go ahead and start building our mapping with [Change Data Capture](http://www.disoln.org/2013/03/Change-Data-Capture-Implementation-for-Multi-Sourced-ETL-Process.html). 

Here we are going to implement [Change Data Capture](http://www.disoln.org/2012/12/Change-Data-Capture-CDC-Implementation-Using-CHECKSUM-Number.html) for CUSTOMER data load. We need to load any new customer or changed customers data to a flat file. Since the column UPDATE\_TS value changes for any new or updated customer record, we will be able to find the new or changed customer records using UPDATE\_TS column. 

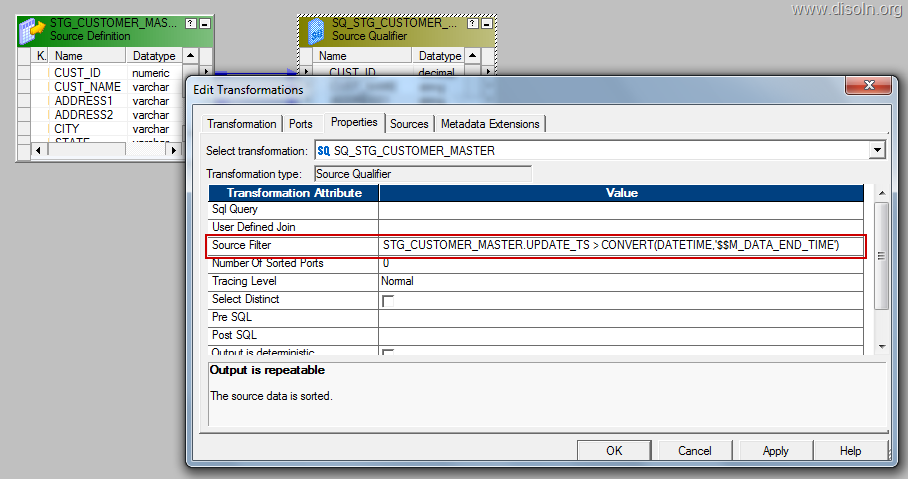
As the first step lets start the mapping and create a mapping variable as shown in below image.

* + $$M\_DATA\_END\_TIME as Date/Time

[](http://lh6.ggpht.com/-_sHvDtOQOIM/UHes3ESSEjI/AAAAAAAAFp8/cA8W_xx8ELE/s1600-h/image%255B11%255D.png)

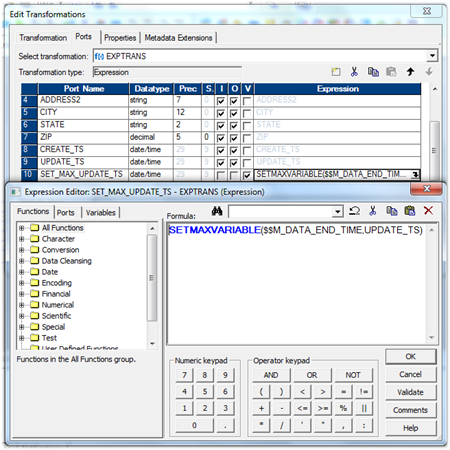
Now bring in the source and source qualified to the mapping designer workspace. Open the source qualifier and give the filter condition to get the latest data from the source as shown below.

* + STG\_CUSTOMER\_MASTER.UPDATE\_TS > CONVERT(DATETIME,'$$M\_DATA\_END\_TIME')



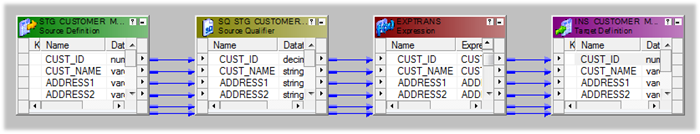
Note : This filter condition will make sure that, latest data is pulled from the source table each and every time. Latest value for the variable $M\_DATA\_END\_TIME is retrieved from the repository every time the session is run.   
Now map the column UPDATE\_TS to an expression transformation and create a variable expression as below.

* SETMAXVARIABLE($M\_DATA\_END\_TIME,UPDATE\_TS)

[](http://lh5.ggpht.com/-wkXWeSOXjU0/UHes30RqY7I/AAAAAAAAFqM/tK8ZTh6CFnU/s1600-h/image%255B20%255D.png)

Note : This expression will make sure that, latest value from the the column UPDATE\_TS is stored into the repository after the successful completion of the session run.

Now you can map all the remaining columns to the down stream transformation and complete all other transformation required in the mapping.

[](http://lh3.ggpht.com/-bqC0GTJxHxU/UHes4_NIJfI/AAAAAAAAFqc/IHzZOZq_gcY/s1600-h/image%255B25%255D.png)

That’s all you need to configure Change Data Capture, Now create your workflow and run the workflow.

Once you look into the session log file you can see the mapping variable value is retrieved from the repository and used in the source SQL, just like shown in the image below.

[http://lh6.ggpht.com/-boEI6Hod5wo/UHes6ZXxL9I/AAAAAAAAFq0/vFnBgzKrMmc/image_thumb%25255B15%25255D.png?imgmax=800](http://lh4.ggpht.com/-f79dyLcritU/UHes5y55zII/AAAAAAAAFqs/lKfOaVYlUfs/s1600-h/image%255B29%255D.png)

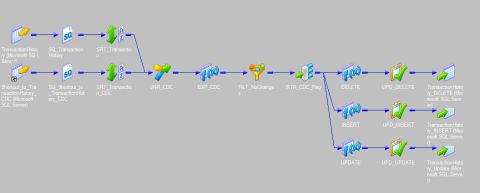
You can look at the mapping variable value stored in the repository, from workflow manager. Choose the session from the workspace, right click and select 'View Persistent Value'. You get the mapping variable in a pop up window, like shown below.

## **Change Data Capture – Informatica Mapping Logic for CDC Implementation (Using JOIN on Source and Target)**

Requirement – To track on the data change in the source system with the earlier data pulled into the warehouse and to track on Updates, new Inserts and Delete on the data that is no more in the source. I would continue on this CDC implementation with an another article on loading of data forward into a table to save the change history for the data user to keep a track on the change history.

**Mapping Design:**

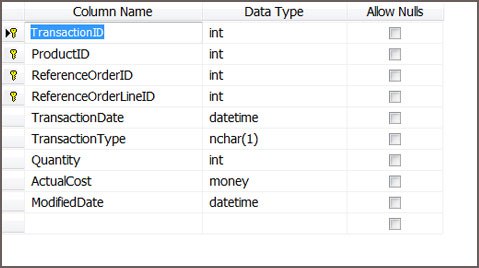
The mapping I have come up with for the CDC logic would be as below,

[](https://mahaveersingh.files.wordpress.com/2014/10/cdc_data_load_img21.jpg)

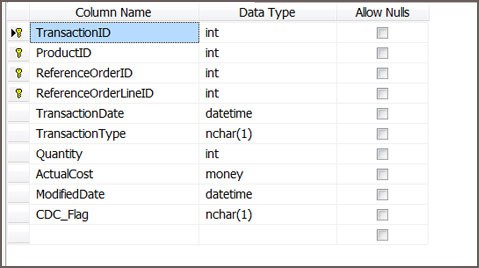
(Click on the image for a better view)

The mapping would involve the Staging/source table which would be a daily truncate load from the source system as is with the current status on the transactions from a source. And the data are then to be sorted with respect to the primary key columns, sample table used would be as below. The sample implies on the target CDC table to have the same composite primary key columns.

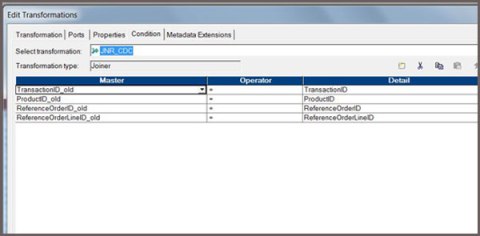
**Source Table:**

[](https://mahaveersingh.files.wordpress.com/2014/10/cdc_data_load_img3.jpg)

**Target Table:**

[](https://mahaveersingh.files.wordpress.com/2014/10/cdc_data_load_img4.jpg)

The mapping would involve a simple logic to have a joiner (Full Outer Join) with the source table and the Target CDC table and to check for the matching and non matching records on the Primary key columns as is illustrated below,

[](https://mahaveersingh.files.wordpress.com/2014/10/cdc_data_load_img5.jpg)

This being a Full Outer Join, enables the loading of all matching and non matching transactions to flow through and then the following three expressions should be defined to check on the Change type,

Flag\_DELETE:

**ISNULL**(TransactionID) AND **ISNULL**(ProductID) AND **ISNULL**(ReferenceOrderID) and **ISNULL**(ReferenceOrderLineID)

Flag\_INSERT:

ISNULL(TransactionID\_old) AND ISNULL(ProductID\_old) AND ISNULL(ReferenceOrderID\_old) AND ISNULL(ReferenceOrderLineID\_old)

Flag\_UPDATE:

TransactionID=TransactionID\_old AND

ProductID=ProductID\_old AND

ReferenceOrderID=ReferenceOrderID\_old AND

ReferenceOrderLineID=ReferenceOrderLineID\_old AND

(TransactionDate != TransactionDate\_old OR

TransactionType != TransactionType\_old OR

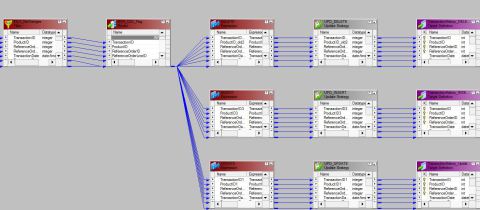
Quantity != Quantity\_old OR

**ABS**(ActualCost – ActualCost\_old)>1)

Flag => IIF(v\_Flag\_Update=1,-1,IIF(v\_Flag\_Insert=1,-2,IIF(v\_Flag\_Delete=1,-3,0)))

where ‘0’ is for no change records, and then to use a filter transformation to filter out all the no change transactions with Flag=0.

Then a Router to define the grouping as per the result of the above expression on checking if the Transaction is an INSERT, UPDATE or DELETE.

[](https://mahaveersingh.files.wordpress.com/2014/10/cdc_data_load_img11.jpg)

(Click on the image for a better view)

And the Update strategy transformation for each to perform and Update or Insert for each, and to note even for DELETE flags I will be going for a DD\_UPDATE, which would update the CDC flag for the transactions not available in the source system to ‘D’ in the target table.

Also to note, that the Delete flow, would involve pulling the ports of CDC source table from the Router to the target and Update on the key column matching records with CDC\_Flag=’D’.

And finally to make the session level changes for each target to define ‘Update as Update’ or ‘Update as Insert’ for the targets respectively. Upon running the mapping for the first time all transactions would be tagged as INSERT and then the proceeding runs will then depend on the source system changes to be captured through the CDC logic of our mapping.

Note – The DELETE transactions are the one which are no more in the source table, and hence these once after being tagged as ‘D’ for the CDC\_Flag can be deleted from the target CDC table as well after making use of the change on to the History tracking table which will be covered more in detail with the next article shortly.

**What is Transaction Control Transformation?**

Transaction Control is an active and connected transformation which allows us to commit or rollback transactions during the execution of the mapping. Commit and rollback operations are of significant importance as it guarantees the availability of data.

When processing a high volume of data, there can be a situation when to commit the data to the target. If a commit is performed too frequently, then it will be an overhead to the system. If a commit is performed too late then in the case of failure there are chances of data loss.

So to provide flexibility Transaction control transformation is provided. There are five in-built variables available in this transformation to handle the operation.

1. TC\_CONTINUE\_TRANSACTION

In tc\_continue\_transaction there are no operations performed, the process of data load continues as it is

1. TC\_COMMIT\_BEFORE

In tc\_commit\_before when this flag is found set, a commit is performed before the processing of current row.

1. TC\_COMMIT\_AFTER

In tc\_commit\_after the current row is processed then a commit is performed.

1. TC\_ROLLBACK\_BEFORE

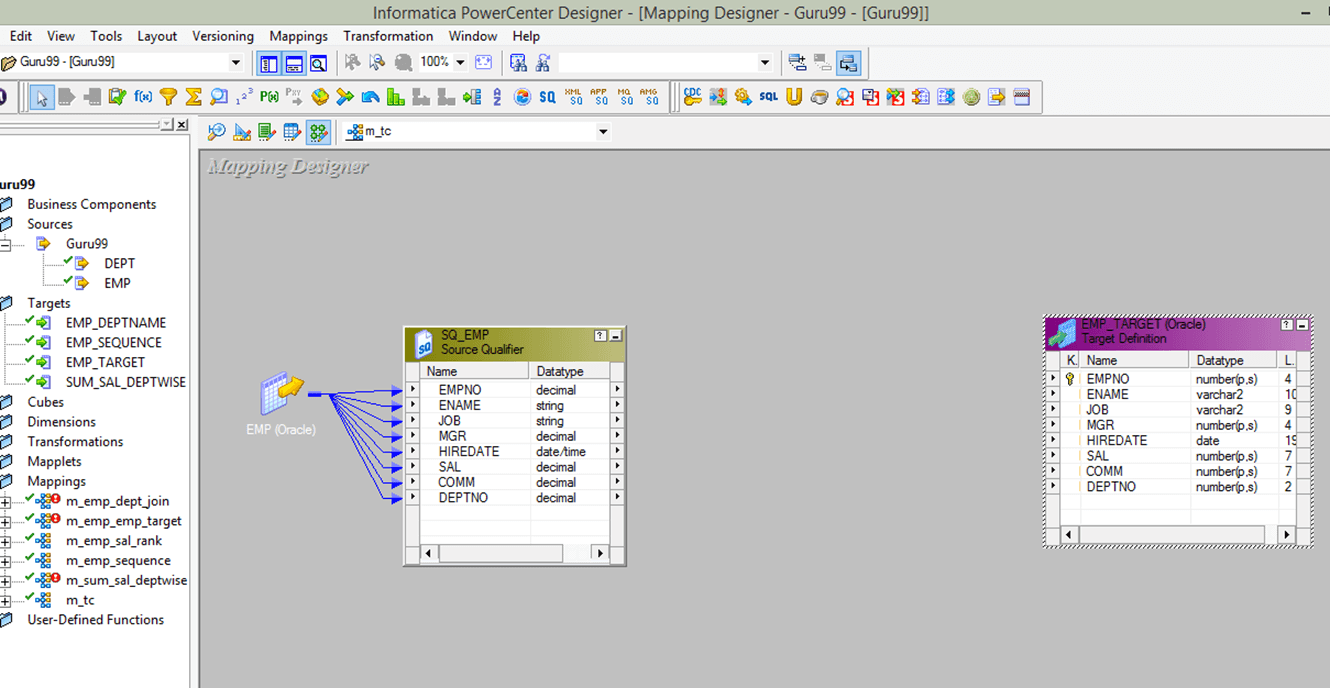
In tc\_rollback\_before, rollback is performed first then data is processed to write

1. TC\_ROLLBACK\_AFTER

In tc\_rollback\_after data is processed then the rollback is performed.

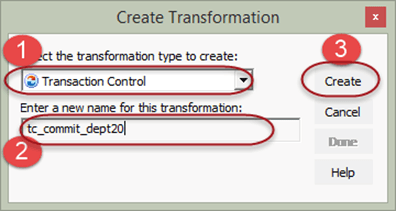
In this example, we will commit data to the target when dept no =20 condition is found true

**Step 1** – Create a mapping with EMP as source and EMP\_TARGET as target

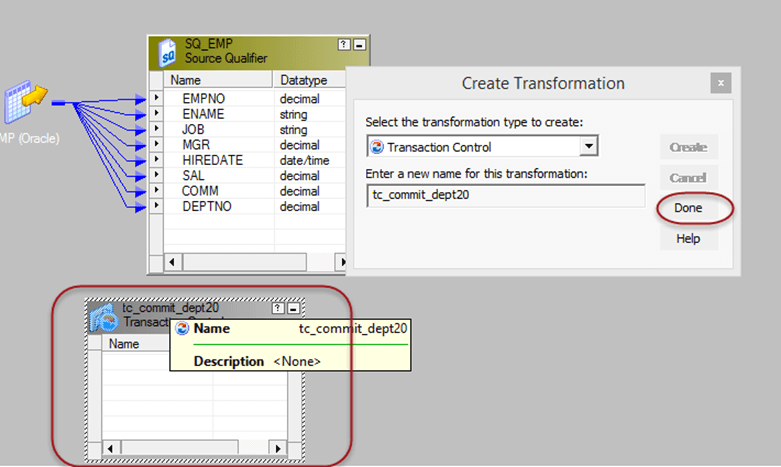
[](http://cdn.guru99.com/images/informatica/083115_1218_Transaction1.png)

**Step 2** – Create a new transformation using transformation menu, then

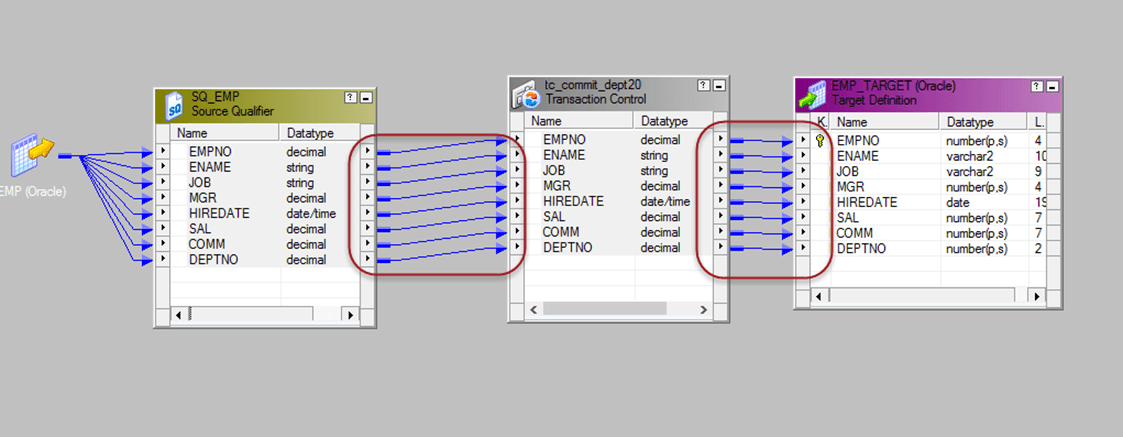
1. Select a transaction control as the new transformation
2. Enter transformation name "tc\_commit\_dept20"
3. Select create option

[](http://cdn.guru99.com/images/informatica/083115_1218_Transaction2.png)

**Step 3** – The transaction control transformation will be created, select done button

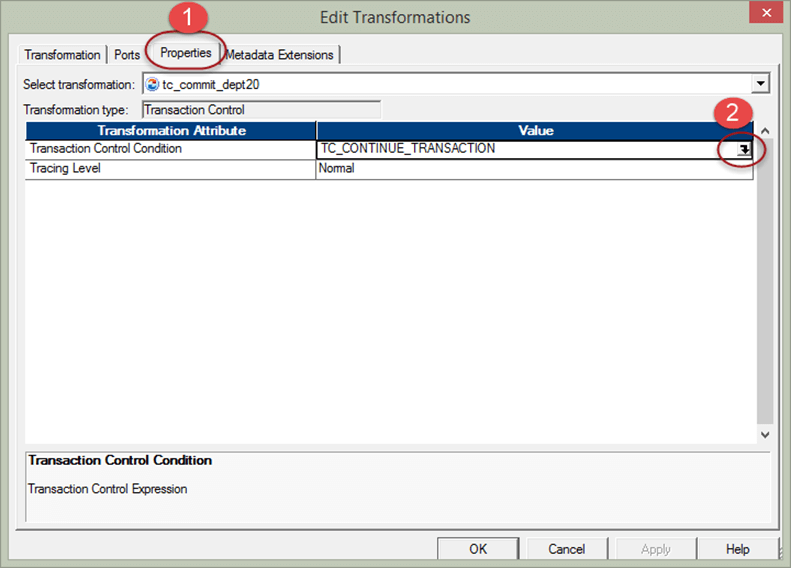
[](http://cdn.guru99.com/images/informatica/083115_1218_Transaction3.png)

**Step 4** - Drag and drop all the columns from source qualifier to the transaction control transformation then link all the columns from transaction control transformation to the target table

[](http://cdn.guru99.com/images/informatica/083115_1218_Transaction4.png)

**Step 5** – Double click on the transaction control transformation and then in the edit property window

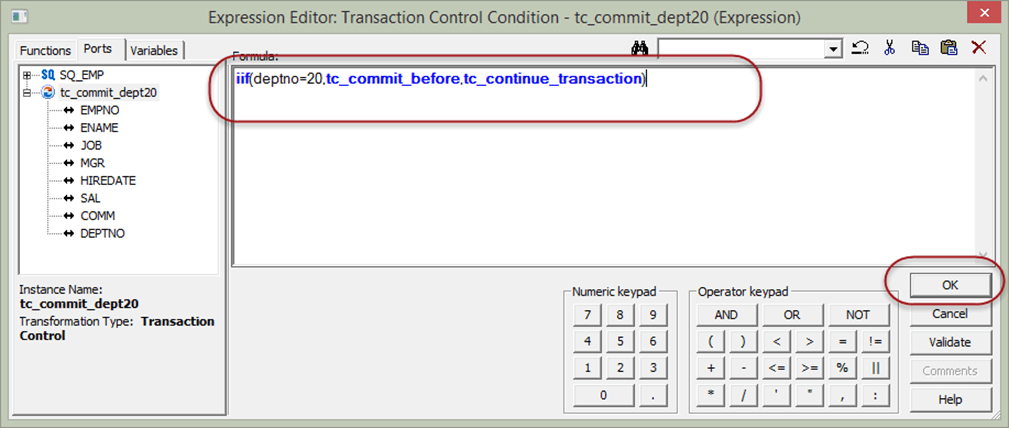
1. Select property tab
2. Click on the transaction control editor icon

[](http://cdn.guru99.com/images/informatica/083115_1218_Transaction5.png)

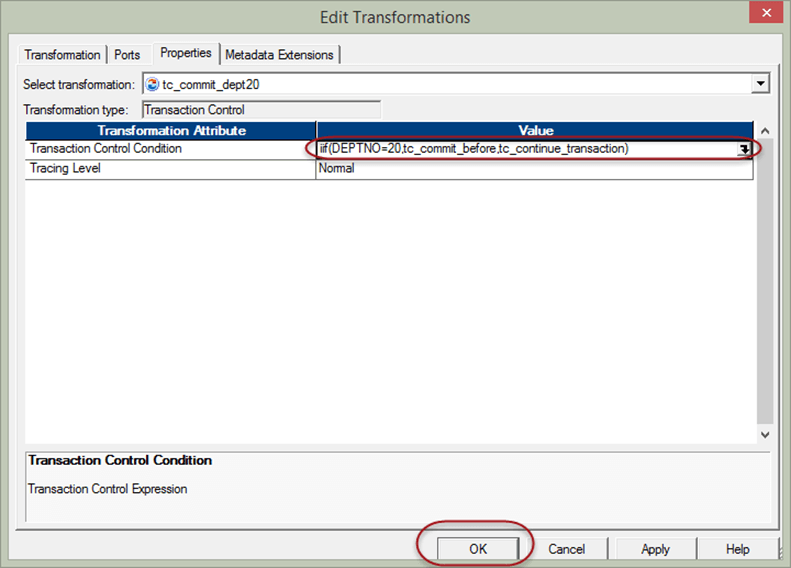
**Step 6** –in the expression editor enter the expression –

**"**iif(deptno=20,tc\_commit\_before,tc\_continue\_transaction)" and select OK

It means if deptno 20 is found then commit transaction in target, else continue the current processing.

[](http://cdn.guru99.com/images/informatica/083115_1218_Transaction6.png)

**Step 7** – Select OK in the previous window

[](http://cdn.guru99.com/images/informatica/083115_1218_Transaction7.png)

Now save the mapping and execute it after creating session and workflows. This mapping will commit the data to the target whenever department number 20 is found in the data.

**Diff 9.5 and 9.6**

1. 9.6.1 is compatible with latest version of the database products like Oracle 12c, SQLServer 2014 etc and as you may know already that 961 is cumulative of 951.

2. Informatica 9.6.1 HF2 which came out in Jan 2015

3. 9.6.x doesn't have any new on the PowerCenter end but now has the new feature of the kerberos authentication(both the user(client) and the server verify each other's identity), can import Business Glossary into the Analyst interface, Metadata Manager has many new enhancements.

**Diff 8x vs 9x**

1. Informatica 9.1 you can make lookup as active transformation... i.e., for one input row you can get all matching rows from lookup table..by makin the multiple match policy to return all rows.

2. Connection management

Database connections are centralized in the domain. We can create and view database connections in Informatica Administrator, Informatica Developer, or Informatica Analyst. Create, view, edit, and grant permissions on database connections in Informatica Administrator.

3. Passive transformation

We can configure the SQL transformation to run in passive mode instead of active mode. When the SQL transformation runs in passive mode, the SQL transformation returns one output row for each input row.

4. Limit the Session Log

You can limit the size of session logs for real-time sessions. You can limit the size by time or by file size. You can also limit the number of log files for a session.

5. Database deadlock resilience

In previous releases, when the Integration Service encountered a database deadlock during a lookup, the session failed. Effective in 9.0, the session will not fail. When a deadlock occurs, the Integration Service attempts to run the last statement in a lookup. You can configure the number of retry attempts and time period between attempts.

6. Enforcement of licensing restrictions

Powercenter will enforce the licensing restrictions based on the number of CPUs and repositories.

Also Informatica 9 supports data integration for the cloud as well as on premise. You can integrate the data in cloud applications, as well as run Informatica 9 on cloud infrastructure.

**VIVIERAE Project Work:**

1. Viverae is a data warehousing and reporting project. It consists of two main types of reports (Standard and Custom feeds).

The feeds are extracted as and when required by the clients of Viverae Inc.

The feeds relate to point system implemented by Viverae for the client’s employees and their spouses for workplace wellness.

The data is from files on ad-hoc basis depending upon requirement.

There are more than one hundred feeds.

Mappings were made in Informatica, database used was MYSQL, Jobs were scheduled using JobScheduler (SOS), GIT was used for version control.

The information included Employer, Employee, spouse, Health Plans taken, various tests the employee or his/her spouse had to take. Timeline by which the tests were to be conducted, points based on the number of tests conducted by employee or spouse as mentioned in the Health Plan.

The medical tests included:

|  |  |
| --- | --- |
| Biometric Screening | Blood Glucose |
| Cholesterol | Systolic Blood Pressure |
| HDL Cholesterol | Diastolic Blood Pressure |
| LDL Cholesterol | Height |
| Triglyceride | Weight |
|  | TOBACCO |

Employer details:

|  |
| --- |
| Employee Class |
| Location Name |
| Location Code |
| Department Name |
| Department Code |
| Division Name |
| Division Code |

Member Details:

|  |  |  |
| --- | --- | --- |
| lastname | address1 | hiredate |
| firstname | address2 | termdate |
| preferredName | contactby | membersince |
| gender | employeeclass | hearing |
| phone | education | vision |
| city | maritalstatus | ssn |
| state | email | relationship |
| zip | birthdate | cellphone |
|  |  | workphone |

FEED\_CONFIGURATION:

Used for generating parameter file manually.

|  |  |  |
| --- | --- | --- |
| Sesslog | tgt\_file\_to\_email | feed\_start\_date |
| Wflwlog | success\_email\_to | feed\_name |
| tgt\_filename\_ts | sftp\_out\_loc\_dir | feed\_end\_date |
| tgt\_filename\_pattern | sftp\_in\_loc\_dir | failure\_email\_to |
| tgt\_file\_to\_sftp | planyearId | email\_signature |
|  |  | email\_privacy |

# Basic Git commands

Here is a list of some basic Git commands to get you going with Git.

For more detail, check out the **[Atlassian Git Tutorials](http://atlassian.com/git?utm_source=basic-git-commands&utm_medium=link&utm_campaign=git-microsite)** for a visual introduction to Git commands and workflows, including examples.

|  |  |  |
| --- | --- | --- |
| **Git task** | **Notes** | **Git commands** |
| [**Tell Git who you are**](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config) | Configure the author name and email address to be used with your commits.  Note that Git [strips some characters](http://stackoverflow.com/questions/26159274/is-it-possible-to-have-a-trailing-period-in-user-name-in-git/26219423#26219423) (for example trailing periods) from user.name. | git config --global user.name "Sam Smith"  git config --global user.email sam@example.com |
| [**Create a new local repository**](http://atlassian.com/git/tutorial/git-basics#!init) |  | git init |
| [**Check out a repository**](http://atlassian.com/git/tutorial/git-basics#!clone) | Create a working copy of a local repository: | git clone /path/to/repository |
| For a remote server, use: | git clone username@host:/path/to/repository |
| [**Add files**](http://atlassian.com/git/tutorial/git-basics#!add) | Add one or more files to staging (index): | git add <filename>  git add \* |
| [**Commit**](http://atlassian.com/git/tutorial/git-basics#!commit) | Commit changes to head (but not yet to the remote repository): | git commit -m "Commit message" |
| Commit any files you've added with git add, and also commit any files you've changed since then: | git commit -a |
| [**Push**](http://atlassian.com/git/tutorial/remote-repositories#!push) | Send changes to the master branch of your remote repository: | git push origin master |
| [**Status**](http://atlassian.com/git/tutorial/git-basics#!status) | List the files you've changed and those you still need to add or commit: | git status |
| [**Connect to a remote repository**](http://atlassian.com/git/tutorial/remote-repositories#!remote) | If you haven't connected your local repository to a remote server, add the server to be able to push to it: | git remote add origin <server> |
| List all currently configured remote repositories: | git remote -v |
| [**Branches**](http://atlassian.com/git/tutorial/git-branches) | Create a new branch and switch to it: | git checkout -b <branchname> |
| Switch from one branch to another: | git checkout <branchname> |
| List all the branches in your repo, and also tell you what branch you're currently in: | git branch |
| Delete the feature branch: | git branch -d <branchname> |
| Push the branch to your remote repository, so others can use it: | git push origin <branchname> |
| Push all branches to your remote repository: | git push --all origin |
| Delete a branch on your remote repository: | git push origin :<branchname> |
| [**Update from the remote repository**](http://atlassian.com/git/tutorial/remote-repositories) | Fetch and merge changes on the remote server to your working directory: | git pull |
| To merge a different branch into your active branch: | git merge <branchname> |
| View all the merge conflicts:  View the conflicts against the base file:  Preview changes, before merging: | git diff  git diff --base <filename>  git diff <sourcebranch> <targetbranch> |
| After you have manually resolved any conflicts, you mark the changed file: | git add <filename> |
| **Tags** | You can use tagging to mark a significant changeset, such as a release: | git tag 1.0.0 <commitID> |
| CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using: | git log |
| Push all tags to remote repository: | git push --tags origin |
| [**Undo local changes**](http://atlassian.com/git/tutorial/undoing-changes) | If you mess up, you can replace the changes in your working tree with the last content in head:  Changes already added to the index, as well as new files, will be kept. | git checkout -- <filename> |
| Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this: | git fetch origin  git reset --hard origin/master |
| **Search** | Search the working directory for foo(): | git grep "foo()" |

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL

$ git init

Initialized empty Git repository in C:/VERSION\_CONTROL/.git/

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git init

Reinitialized existing Git repository in C:/VERSION\_CONTROL/.git/

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ clear

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git remote add origin "https://github.com/naveenbahuguna/GIT\_REP\_TST.git"

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git pull origin master

remote: Counting objects: 6, done.

remote: Compressing objects: 100% (4/4), done.

remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (6/6), done.

From https://github.com/naveenbahuguna/GIT\_REP\_TST

\* branch master -> FETCH\_HEAD

\* [new branch] master -> origin/master

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git commit

On branch master

nothing to commit, working tree clean

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git status

On branch master

nothing to commit, working tree clean

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

abc.txt

nothing added to commit but untracked files present (use "git add" to track)

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git add abc.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git status

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

new file: abc.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git commit -m "new file added. First commit in local"

[master 7ea1656] new file added. First commit in local

Committer: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 1 insertion(+)

create mode 100644 abc.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: abc.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

abc2.txt

abc3.txt

no changes added to commit (use "git add" and/or "git commit -a")

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git add -A

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git status

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: abc.txt

new file: abc2.txt

new file: abc3.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git commit -a -m "multiple files have been committed"

[master c330af4] multiple files have been committed

Committer: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

3 files changed, 3 insertions(+), 1 deletion(-)

create mode 100644 abc2.txt

create mode 100644 abc3.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git log

commit c330af4939954b4af9bae832e62256c22ccfb8f7

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:18:53 2017 +0530

multiple files have been committed

commit 7ea165693fd467e0470e01989a377992d61dec65

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:12:53 2017 +0530

new file added. First commit in local

commit cd81747cd4eee72d32868a4fe26e16b79946957a

Author: naveenbahuguna <na.bahuguna@gmail.com>

Date: Thu Feb 23 16:43:05 2017 +0530

Update README.md

commit 8dbd60b9aefad8197872be19a9c90b6532769ae5

Author: naveenbahuguna <na.bahuguna@gmail.com>

Date: Thu Feb 23 16:39:30 2017 +0530

Initial commit

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git branch Branch\_1

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git checkout Branch\_1

Switched to branch 'Branch\_1'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git status

On branch Branch\_1

Untracked files:

(use "git add <file>..." to include in what will be committed)

abc4.txt

nothing added to commit but untracked files present (use "git add" to track)

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git add abc4

fatal: pathspec 'abc4' did not match any files

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git add abc4.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git commit

Vim: Error reading input, exiting...

Vim: Finished.

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git commit -m "Branch\_1: abc4 added. committed"

[Branch\_1 09495cc] Branch\_1: abc4 added. committed

Committer: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 1 insertion(+)

create mode 100644 abc4.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ ls

abc.txt abc2.txt abc3.txt abc4.txt README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git checkout master

Switched to branch 'master'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ls

abc.txt abc2.txt abc3.txt README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git merge Branch\_1

Updating c330af4..09495cc

Fast-forward

abc4.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 abc4.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ls

abc.txt abc2.txt abc3.txt abc4.txt README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git checkout Branch\_1

Switched to branch 'Branch\_1'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git commit -a -m "modified abc4.txt"

[Branch\_1 3e9f42e] modified abc4.txt

Committer: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 1 insertion(+), 1 deletion(-)

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ cat abc4.txt

Branch\_1: "How" -Modified: "are you?"

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git checkout master

Switched to branch 'master'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat abc4.txt

Branch\_1: "How"

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git merger Batch\_1

git: 'merger' is not a git command. See 'git --help'.

Did you mean this?

merge

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git merge Batch\_1

merge: Batch\_1 - not something we can merge

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ls

abc.txt abc2.txt abc3.txt abc4.txt README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git log

commit 09495ccf38bfbaf65015c6662328f7cc713a91b0

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:28:08 2017 +0530

Branch\_1: abc4 added. committed

commit c330af4939954b4af9bae832e62256c22ccfb8f7

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:18:53 2017 +0530

multiple files have been committed

commit 7ea165693fd467e0470e01989a377992d61dec65

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:12:53 2017 +0530

new file added. First commit in local

commit cd81747cd4eee72d32868a4fe26e16b79946957a

Author: naveenbahuguna <na.bahuguna@gmail.com>

Date: Thu Feb 23 16:43:05 2017 +0530

Update README.md

commit 8dbd60b9aefad8197872be19a9c90b6532769ae5

Author: naveenbahuguna <na.bahuguna@gmail.com>

Date: Thu Feb 23 16:39:30 2017 +0530

Initial commit

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat abc4.txt

Branch\_1: "How"

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git merge Brahch\_1

merge: Brahch\_1 - not something we can merge

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git merge Branch\_1

Updating 09495cc..3e9f42e

Fast-forward

abc4.txt | 2 +-

1 file changed, 1 insertion(+), 1 deletion(-)

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ls

abc.txt abc2.txt abc3.txt abc4.txt README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat abc4.txt

Branch\_1: "How" -Modified: "are you?"

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git checkout Branch\_1

Switched to branch 'Branch\_1'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git add -m "adding abc5.txt and abc6.txt"

error: unknown switch `m'

usage: git add [<options>] [--] <pathspec>...

-n, --dry-run dry run

-v, --verbose be verbose

-i, --interactive interactive picking

-p, --patch select hunks interactively

-e, --edit edit current diff and apply

-f, --force allow adding otherwise ignored files

-u, --update update tracked files

-N, --intent-to-add record only the fact that the path will be added later

-A, --all add changes from all tracked and untracked files

--ignore-removal ignore paths removed in the working tree (same as --no-all)

--refresh don't add, only refresh the index

--ignore-errors just skip files which cannot be added because of errors

--ignore-missing check if - even missing - files are ignored in dry run

--chmod <(+/-)x> override the executable bit of the listed files

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git add -A

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git commit -a -m "adding abc5.txt and abc6.txt"

[Branch\_1 7dac5a2] adding abc5.txt and abc6.txt

Committer: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

2 files changed, 2 insertions(+)

create mode 100644 abc5.txt

create mode 100644 abc6.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ ls

abc.txt abc2.txt abc3.txt abc4.txt abc5.txt abc6.txt README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git checkout master

Switched to branch 'master'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ls

abc.txt abc2.txt abc3.txt abc4.txt README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git checkout Branch\_1

Switched to branch 'Branch\_1'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git rebase master

Current branch Branch\_1 is up to date.

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git checkout master

Switched to branch 'master'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git rebase Branch\_1

First, rewinding head to replay your work on top of it...

Fast-forwarded master to Branch\_1.

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ls

abc.txt abc2.txt abc3.txt abc4.txt abc5.txt abc6.txt README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/c/Users/naveen.bahuguna/.ssh/id\_rsa): abc7.txt

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in abc7.txt.

Your public key has been saved in abc7.txt.pub.

The key fingerprint is:

SHA256:FrIpUcHKsbB1VnxAnQ6831bIuGcPho0dyqifFXtGTto naveen.bahuguna@C40-GF98-

The key's randomart image is:

+---[RSA 2048]----+

| .o\*+o . |

| . o.+ + + |

| =.\*. .= o . |

| . +. +..o + . |

| . o S+.Ooo |

| . .. BOX |

| . +=Eo |

| . o o . |

| .o |

+----[SHA256]-----+

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat /c/Users/naveen.bahuguna/.ssh/id\_rsa/abc7.txt

cat: /c/Users/naveen.bahuguna/.ssh/id\_rsa/abc7.txt: Not a directory

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat /c/Users/naveen.bahuguna/.ssh/id\_rsa/abc7.txt

cat: /c/Users/naveen.bahuguna/.ssh/id\_rsa/abc7.txt: Not a directory

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat C:\VERSION\_CONTROL\abc7.txt

cat: 'C:VERSION\_CONTROLabc7.txt': No such file or directory

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/c/Users/naveen.bahuguna/.ssh/id\_rsa):

/c/Users/naveen.bahuguna/.ssh/id\_rsa already exists.

Overwrite (y/n)? y

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /c/Users/naveen.bahuguna/.ssh/id\_rsa.

Your public key has been saved in /c/Users/naveen.bahuguna/.ssh/id\_rsa.pub.

The key fingerprint is:

SHA256:HS72SwqRlkRCayic6+AbsPHDv1IRZS9TD1f/G9et2vQ naveen.bahuguna@C40-GF98-

The key's randomart image is:

+---[RSA 2048]----+

| .o.+ o ... |

|. ...= o + . |

|.o. o.+ . o . |

| ..... = o . .o|

|+. .= S o .=|

|== .. o o .+|

|.++. . o o. |

| +o . o . + . |

| . .o. . . . . E|

+----[SHA256]-----+

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat ^C

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat /c/Users/naveen.bahuguna/.ssh/id\_rsa.pub

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDGd8r6DniLHGUNaQGJot+mfmoSrooSH4w0NnJT/i3Pg5jSCNCGQoWKNAsf6UI3TXXD74QmRGruZgxDjy5Ds11GTHZBskb9zKZyxlVh6SIz60U85eNwKPLKh0T5raPpDMuohA41GpS5GgU2ggGa219G7Azix1KBGYtpLD2IbYLbUHRe3ewpuYpyB/4bSeggnkIhfL9aodwo28MiJ/bpx/2FFnSjmfj70lZ4Y3H0Jc0meaEwCACsU3uSa7d0T9GnvVfQwI/CqnqnQ+XBYYn80oqdA+NLt4TXaspYpsCOFvQzu6jGg+e7pQWzvHj3uDmorE0x953KbOHKrKC9Q3BEeH/5 naveen.bahuguna@C40-GF98-

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ssh -T git@github.com

The authenticity of host 'github.com (192.30.253.112)' can't be established.

RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.

Are you sure you want to continue connecting (yes/no)? Yes

Warning: Permanently added 'github.com,192.30.253.112' (RSA) to the list of known hosts.

Hi naveenbahuguna! You've successfully authenticated, but GitHub does not provide shell access.

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git checkout Branch\_1

Switched to branch 'Branch\_1'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git push origin Branch\_1

Counting objects: 18, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (10/10), done.

Writing objects: 100% (18/18), 1.42 KiB | 0 bytes/s, done.

Total 18 (delta 3), reused 0 (delta 0)

remote: Resolving deltas: 100% (3/3), done.

To https://github.com/naveenbahuguna/GIT\_REP\_TST.git

\* [new branch] Branch\_1 -> Branch\_1

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (Branch\_1)

$ git checkout master

Switched to branch 'master'

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git push origin master

Total 0 (delta 0), reused 0 (delta 0)

To https://github.com/naveenbahuguna/GIT\_REP\_TST.git

cd81747..7dac5a2 master -> master

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ls

abc.txt abc2.txt abc3.txt abc4.txt abc5.txt abc6.txt abc7.txt abc7.txt.pub README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git add abc7.txt -m "reverse"

error: unknown switch `m'

usage: git add [<options>] [--] <pathspec>...

-n, --dry-run dry run

-v, --verbose be verbose

-i, --interactive interactive picking

-p, --patch select hunks interactively

-e, --edit edit current diff and apply

-f, --force allow adding otherwise ignored files

-u, --update update tracked files

-N, --intent-to-add record only the fact that the path will be added later

-A, --all add changes from all tracked and untracked files

--ignore-removal ignore paths removed in the working tree (same as --no-all)

--refresh don't add, only refresh the index

--ignore-errors just skip files which cannot be added because of errors

--ignore-missing check if - even missing - files are ignored in dry run

--chmod <(+/-)x> override the executable bit of the listed files

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git add abc7.txt

warning: LF will be replaced by CRLF in abc7.txt.

The file will have its original line endings in your working directory.

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git commit -m "revert"

[master 2e79701] revert

Committer: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 27 insertions(+)

create mode 100644 abc7.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ ls

abc.txt abc2.txt abc3.txt abc4.txt abc5.txt abc6.txt abc7.txt README.md

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git commit -a -m "revert2"

[master efc8815] revert2

Committer: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

1 file changed, 1 insertion(+), 27 deletions(-)

rewrite abc7.txt (100%)

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat abc7.txt

changed

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git log

commit efc881527dab1cd1c3a02edefb4895e859b45d31

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 18:16:14 2017 +0530

revert2

commit 2e79701e8d6f0d35af5f19a07ce382bc1cae95d9

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 18:13:38 2017 +0530

revert

commit 7dac5a240845d030a1433cdad1ea5c92f6d2bfe0

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:49:54 2017 +0530

adding abc5.txt and abc6.txt

commit 3e9f42e0c7486588dd189ae0d8ab028c46dc6eaf

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:37:47 2017 +0530

modified abc4.txt

commit 09495ccf38bfbaf65015c6662328f7cc713a91b0

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:28:08 2017 +0530

Branch\_1: abc4 added. committed

commit c330af4939954b4af9bae832e62256c22ccfb8f7

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:18:53 2017 +0530

multiple files have been committed

commit 7ea165693fd467e0470e01989a377992d61dec65

Author: Naveen Bahuguna <Naveen.Bahuguna@rsystems.com>

Date: Thu Feb 23 17:12:53 2017 +0530

new file added. First commit in local

commit cd81747cd4eee72d32868a4fe26e16b79946957a

Author: naveenbahuguna <na.bahuguna@gmail.com>

Date: Thu Feb 23 16:43:05 2017 +0530

Update README.md

commit 8dbd60b9aefad8197872be19a9c90b6532769ae5

Author: naveenbahuguna <na.bahuguna@gmail.com>

Date: Thu Feb 23 16:39:30 2017 +0530

Initial commit

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ git checkout 2e79701e abc7.txt

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$ cat abc7.txt

-----BEGIN RSA PRIVATE KEY-----

MIIEowIBAAKCAQEA0pJSxzFMutJe0UlhiVJ3Zr2LInmScjs5P/PheQCMNJaMCWm5

ij1cGxw0rGWB5/szJTaWFbln5Gwv++Etv09Z6VWzhtcAzmhMVuuqkMn8QNuy2ziY

8RhprytcdbCBKIp1H8Sj+NHE/VJ3rXWqsanBtj4CSNo3b0YZqJY8tqhn2xucptM8

kT8dsZ4Jvo4HEE/vXxcZwU0y8Y2K3TwntBClpKYw5qYGIvz+FVjXkgDobIbSUMDY

hY8SN8reRwZdv3BiSzK1ra1UZ9t+O0aDfkCeJiQ1LQixKwUxRCbQRsXfq4pYfKOB

hOAoIxRHF3affk95jig6UTDI9U3kQDgNgrVJtQIDAQABAoIBAESos1cByywLViFV

lPkKq6YTXZ+TFl4q02L36Wfj9tNa1zRmhbn+gd7RFpezzWTPsoCKyOsydh6z6/1s

EdeUTxPW/t+TkU68EVAyLX5ibd7UBACY9r1+IU+dittQ/vwyETiXjIOgGmQjFKS5

fjvJx0Dd/S62AGdG6yzeN9OzG5IEWe/OAeWJHdQkf2mspl/8ktvdGrfnQlGa6f+H

CQYMGsq35YMa4xzKREUweMg0JJzCzYcf95emF5HTk3l2VQ8tcwBHO2KOwB9XYDOF

xZg5HZtagHp7+JMc0nZQ2bD9KAKUP5TbqzPJ5JoRynrrfdarnHUGzD930oY6Tejv

s7XVu6ECgYEA+1kh573CBWN72tBtPxOJHgLe4X6lWKuy6dKgJM+/Gg98Ld/FxeUN

fiQ/b2v0by3obxfEzk3vz5J97el2MuyaGgQiQ3dptRzRlOmj0SXpzPkk+1QUEeSB

2WWYcsUz/t47ZxMcSlcFxzTdsh6/Rj8dcLeamjTf4Q1UZoTOctaitL0CgYEA1nf+

oTQz6rr0pxx3QIWyQxSzGnRAxjhTanemlUAMMfNdyVfhN/3ZorbU7f4k823yU+m+

OgWfGhtA616PT3nnU+OzFxtX8bC07lPmawQx/iGH5T6+dcOLE05r3OQecGvJzsPO

ZlnLYAIt2ppENkZ/cpPhspSeOYZseSchC7VvhFkCgYBV47h1U1/IrZ02QcEDK+7Q

JZT6I2fnLA0gX8w/qAoaoq4BkxUnMcVxlHtk3NQTZfGmfM3Q3csT+HA163BrqHRk

bYMz4Mtri2SG6WwEXVtOotijVbDiCF7i5he5KHBDKLfN2jbxpFkiIIudgWpFW+34

hZU8GtNiOUbdHYbasUO6VQKBgQCCZuOD0OHgT1yCX2SC7se80dE0vIrF+2l8ytOW

3CKOHJNh4PUnvkmeDdzZ+DUvbiJxsyXe0Zwzm26VriElCdIFx+UUn6ORaQgaARmQ

6DQdePXQBZ54rDTbwEsFPA7l/CPwfE37HY3zCmAnErV26taDLDeHBaZPA9TgtVH6

dI8hwQKBgG3/1/DBujbYa92cLA2yZ3hQ39zEQQyIs7nexLDO9sryofJCq0LLl0F/

Uzh9JamHx/9Jay+pUOeyNf4hwR+zP0yrMS7JtDr4CUdY9j+HhNecappjX6Rp6yMz

ZUNFsaR28G7cq5XYtkdwlOxLPtxX2Wlc1pqMFavfSNwlI7bWPFos

-----END RSA PRIVATE KEY-----

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$

naveen.bahuguna@C40-GF98- MINGW64 /c/VERSION\_CONTROL (master)

$

* [The 101 Guide to Dimensional Data Modeling](https://dwbi.org/data-modelling/dimensional-model/1-dimensional-modeling-guide)
* [How to find out Expected Time of Completion for an Oracle Query](https://dwbi.org/database/oracle/35-how-to-find-out-expected-time-of-completion-for-an-oracle-query)
* [Understanding Oracle QUERY PLAN - A 10 minutes guide](https://dwbi.org/database/oracle/38-oracle-query-plan-a-10-minutes-guide)
* [Exception Handling While Reading Multiple XML Files in Data Services](https://dwbi.org/etl/sap-data-services/89-exception-handling-while-reading-multiple-xml-files-in-data-services)

## **Informatica $$Parameters and their scopes**

**Important Points:**

The parameters are detected by the Informatica IS with a $$ and takes the parameter name from the $$ till the occurrence of the ‘=’ symbol, anything between $$ and = shall be considered as the parameter name.

Similarly any value from the ‘=’ till the end of the line shall be considered as the value to the respective parameter.

The parameter files are saved with an extension .prm

There are different levels of precedence of the parameters,

**The mapping level precedence**

This is the least precedence level, defined as initial value in the mapping designer itself.

**The Global precedence:**

These are the parameters defined in the whole workflow’s parameter files, directly as parameters.

Example:

[gloabal]

$$Param\_test=2011

**The Session level precedence:**

The session level parameters are defined in the workflow’s parameter file with the session name, this has the precedence over the global parameters.

Example:

$$Param\_test=2011

[s\_sample\_session\_name]

$$Param\_test=2012

So here in the session ‘s\_sample\_session\_name’, the ‘’$$Param\_test” holds the value 2012 and not the 2011 for the higher precedence by the session level parameter.

**The Session Param file precedence:**

There can also be a separate parameter file that can be defined for a session, where the parameter value defined in this param file takes the higher precedence than the one defined in the workflow’s param file.

**The Path defined parameter precedence:**

There can be situations where a reusable session used in two different worklets can need two different param values at two instances. This can be very well achieved by indicating the specific session name through its worklet as in the below example.

Example:

If s\_session\_test1 is used in WKLT\_workletA and WKLT\_workletB with two different values, then we can define the parameter value as,

[FLD\_folder\_name.WF:wf\_workflow\_name.WT:WKLT\_workletA.S:s\_session\_test1]

$$Param\_name=2011

[FLD\_folder\_name.WF:wf\_workflow\_name.WT:WKLT\_workletB.S:s\_session\_test1]

$$Param\_name=2012

So the same session uses the same parameter, but with different values at different worklets.

**How to Use Error Handling Options and Techniques in Informatica PowerCenter**

[Data quality](http://www.disoln.org/2012/10/User-Defined-Error-Handling-in-Informatica-PowerCenter.html) is very critical to the success of every data warehouse projects. So ETL Architects and Data Architects spent a lot of time defining the [error handling](http://www.disoln.org/2012/07/error-handling-made-easy-using.html)approach. Informatica PowerCenter is given with a set of options to take care of the error handling in your ETL Jobs. In this article, lets see how do we leverage the PowerCenter options to handle your exceptions.

**Error Classification**

You have to deal with different type of errors in the ETL Job. When you run a session, the PowerCenter Integration Service can encounter fatal or non-fatal errors. Typical error handling includes:

* + **User Defined Exceptions**: Data issues critical to the data quality, which might get loaded to the database unless explicitly checked for quality. For example, a credit card transaction with a future transaction data can get loaded into the database unless the transaction date of every record is checked.
  + **Non-Fatal Exceptions**: Error which would get ignored by Informatica PowerCenter and cause the records dropout from target table otherwise handled in the ETL logic. For example,  a data conversion transformation error out and fail the record from loading to the target table.
  + **Fatal Exceptions**: Errors such as database connection errors, which forces Informatica PowerCenter to stop running the workflow.

I. User Defined Exceptions

Business users define the user defined user defined exception, which is critical to the data quality. We can setup the user defined error handling using;

* + - 1. [Error Handling Functions](http://www.disoln.org/2012/10/User-Defined-Error-Handling-in-Informatica-PowerCenter.html).
      2. User Defined Error Tables.

### 1. Error Handling Functions

We can use two functions provided by Informatica PowerCenter to define our user defined error capture logic.

**ERROR()** : This function Causes the PowerCenter Integration Service to skip a row and issue an error message, which you define. The error message displays in the session log or written to the error log tables based on the error logging type configuration in the session.

You can use ERROR in Expression transformations to validate data. Generally, you use ERROR within an IIF or DECODE function to set rules for skipping rows.

Eg : IIF(TRANS\_DATA > SYSDATE,ERROR('Invalid Transaction Date'))

Above expression raises an error and drops any record whose transaction data is greater than the current date from the ETL process and the target table.

**ABORT()**: Stops the session, and issues a specified error message to the session log file or written to the error log tables based on the error logging type configuration in the session. When the PowerCenter Integration Service encounters an ABORT function, it stops transforming data at that row. It processes any rows read before the session aborts.

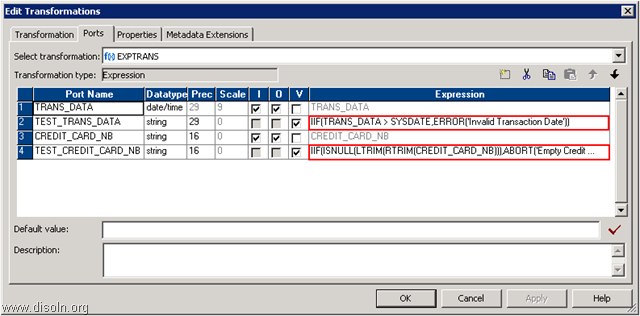
You can use ABORT in Expression transformations to validate data.

Eg : IIF(ISNULL(LTRIM(RTRIM(CREDIT\_CARD\_NB))),ABORT('Empty Credit Card Number'))

Above expression aborts the session if any one of the transaction records are coming with out a credit card number.

### Error Handling Function Use Case

Below shown is the configuration required in the expression transformation using ABORT() and ERROR() Function. This transformation is using the expressions as shown in above examples.

[](http://lh4.ggpht.com/-Ps9JKWcCkZ8/U0JGFXzYFyI/AAAAAAAAJS4/y5P3ShnPgoU/s1600-h/image%25255B36%25255D.png)

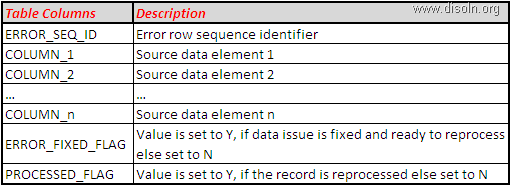
**Note**:- You need to use these two functions in a mapping along with a session configuration for [row error logging](http://www.disoln.org/2012/07/error-handling-made-easy-using.html) to capture the error data from the source system. Depending on the session configuration, source data will be collected into Informatica predefined PMERR error tables or files.

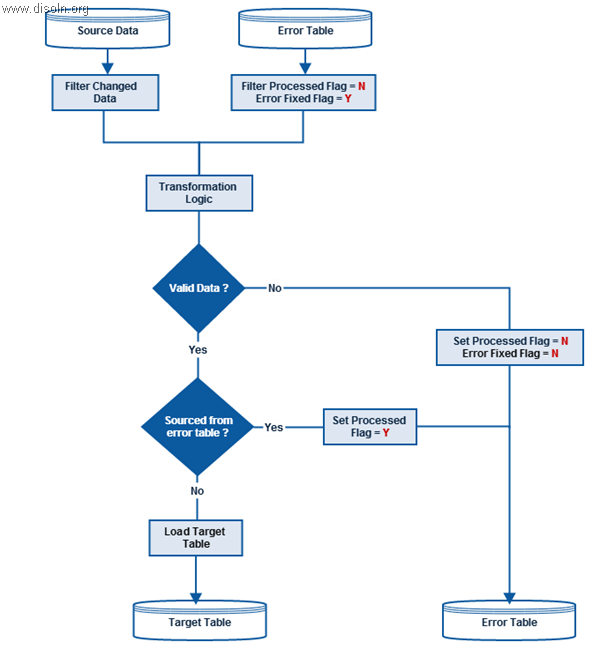
Please refer the article "[User Defined Error Handling in Informatica PowerCenter](http://www.disoln.org/2012/10/User-Defined-Error-Handling-in-Informatica-PowerCenter.html)**"**for more detailed level implementation information on user defined error handling.

### 2. User Defined Error Tables

Error Handling Functions are easy to implement with very less coding efforts, but at the same time there are some disadvantages such as readability of the error records from the PMERR tables and [performance](http://www.disoln.org/search/label/Performance%20Tips?max-results=8) impact. To avoid the disadvantages of error handling functions, you can create your own error log tables and capture the error records into it.

Typical approach is to create an error table which is similar in structure to the source table. Error tables will include additional columns to tag the records as "error fixed", "processed". Below is a sample error table. This error table includes all the columns from the source table and additional columns to identify the status of the error record.

[](http://lh3.ggpht.com/-SXg7jFaTW9I/U0Ixcb-DGxI/AAAAAAAAJR8/LOamayJqQ0E/s1600-h/image%25255B12%25255D.png)

Below is the high level design.[](http://lh5.ggpht.com/-eVoD2A6KJec/U0JCLWmtkAI/AAAAAAAAJSU/XqiOP7reDZM/s1600-h/Error%252520Processing%25255B12%25255D.png)

Typical [ETL Design](http://www.disoln.org/search/label/ETL%20Design?&max-results=15) will read error data from the error table along with the source data. During the data transformation, data quality will be checked and any record violating the quality check will be moved to error tables. Record flags will be used to identify the reprocessed and records which are fixed for reprocessing.

## **II. Non-Fatal Exceptions**

Non-fatal exception causes the records to be dropped out in the ETL process, which is critical to quality. You can handle non-fatal exceptions using;

* + - 1. Default Port Value Setting.
      2. Row Error Logging.
      3. Error Handling Settings.

### **1. Default Port Value Setting**

Using default value property is a good way to handle exceptions due to NULL values and unexpected transformation errors. The Designer assigns default values to handle null values and output transformation errors. [PowerCenter Designer](http://www.disoln.org/2012/08/Understand-Informatica-PowerCenter-Mapping-Designer.html) let you override the default value in input, output and input/output ports.

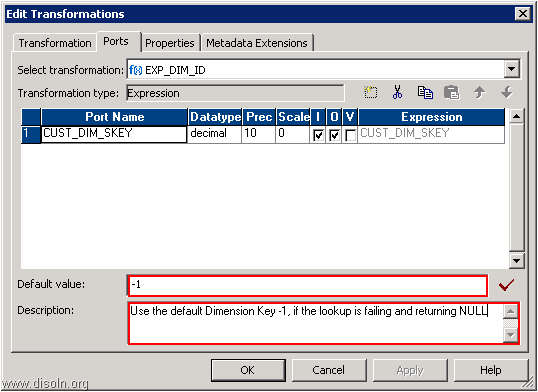
Default value property behaves differently for different port types;

* **Input ports** : Use default values if you do not want the Integration Service to treat null values as NULL.
* **Output ports** : Use default values if you do not want to skip the row due to transformation error or if you want to write a specific message with the skipped row to the session log.
* **Input/output ports** : Use default values if you do not want the Integration Service to treat null values as NULL. But no user-defined default values for output transformation errors in an input/output port.

### Default Value Use Case

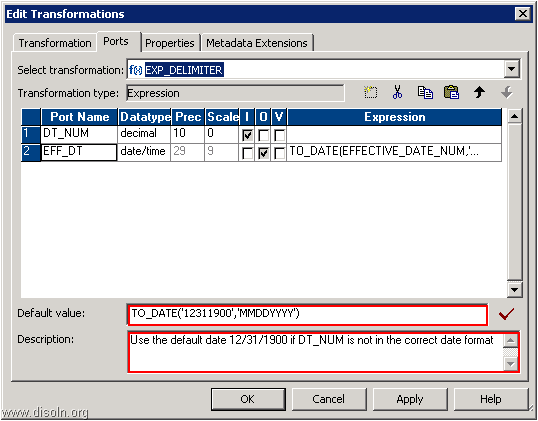
**Use Case 1**

Below shown is the setting required to handle NULL values. This setting converts any NULL value returned by the [dimension](http://www.disoln.org/2012/08/slowly-changing-dimension-type-2-implementation-using-informatica.html) lookup to the default value -1. This technique can be used to handle [late arriving dimensions](http://www.disoln.org/2013/12/Design-Approach-to-Handle-Late-Arriving-Dimensions-and-Late-Arriving-Facts.html)

[](http://lh3.ggpht.com/-iAbkIOrccS8/U0JCMQsTnNI/AAAAAAAAJSg/t8MWICQXrKs/s1600-h/image%25255B25%25255D.png)

**Use Case 2**

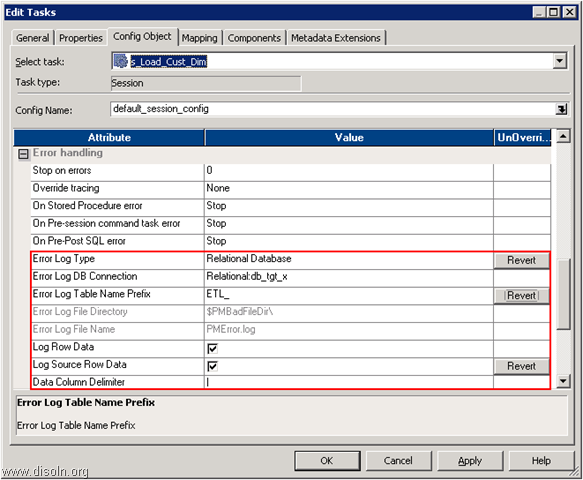
Below setting uses the default expression to convert the date if the incoming value is not in a valid date format.

[](http://lh4.ggpht.com/-Py56aSkSqLg/U0JSu_emPiI/AAAAAAAAJTQ/L-y8fPez8Vo/s1600-h/image%25255B47%25255D.png)

### **2. Row Error Logging**

Row error logging helps in capturing any exception, which is not consider during the design and coded in the mapping. It is the perfect way of capturing any unexpected errors.

Below shown session error handling setting will capture any un handled error into PMERR tables.

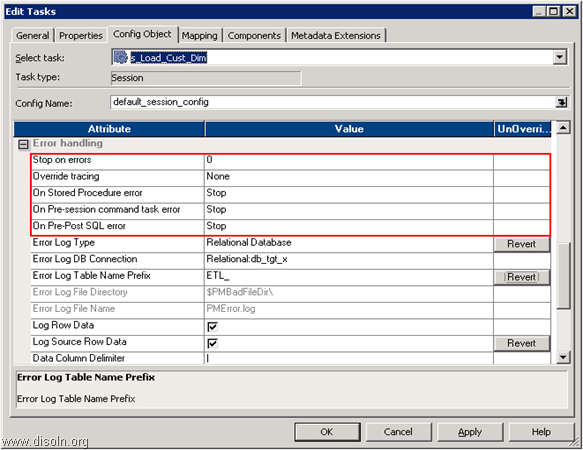
[](http://lh5.ggpht.com/-faZ4fR4Q-X8/U0JVzfnxdtI/AAAAAAAAJTk/txvtOHH0LQY/s1600-h/image%25255B62%25255D.png)

Please refer the article [Error Handling Made Easy Using Informatica Row Error Logging](http://www.disoln.org/2012/07/error-handling-made-easy-using.html) for more details.

### **3. Error Handling Settings**

Error handling properties at the session level is given with options such as Stop On Errors, Stored Procedure Error,  Pre-Session Command Task Error and Pre-Post SQL Error. You can use these properties to ignore or set the session to fail if any such error occurs.

* **Stop On Errors** : Indicates how many non-fatal errors the Integration Service can encounter before it stops the session.
* **On Stored Procedure Error** : If you select Stop Session, the Integration Service stops the session on errors executing a pre-session or post-session stored procedure.
* **On Pre-Session Command Task Error** : If you select Stop Session, the Integration Service stops the session on errors executing pre-session shell commands.
* **Pre-Post SQL Error** : If you select Stop Session, the Integration Service stops the session errors executing pre-session or post-session SQL.

[](http://lh3.ggpht.com/-qwTYuhVWnmA/U0JV0fNHNaI/AAAAAAAAJT0/YENBCiMO7dc/s1600-h/image%25255B73%25255D.png)

## **III. Fatal Exceptions**

A fatal error occurs when the Integration Service cannot access the source, target, or repository. When the session encounters fatal error, the PowerCenter Integration Service terminates the session. To handle fatal errors, you can either use a [restartable ETL design](http://www.disoln.org/2013/02/Restartability-Design-for-Different-Type-ETL-Loads.html) for your workflow or use the [workflow recovery features](http://www.disoln.org/2013/07/Workflow-Recovery-Configuration-for-Informatica-PowerCenter-Workflows.html) of Informatica PowerCenter

* + - 1. [Restartable ETL Design](http://www.disoln.org/2013/02/Restartability-Design-for-Different-Type-ETL-Loads.html)
      2. [Workflow Recovery](http://www.disoln.org/2013/07/Workflow-Recovery-Configuration-for-Informatica-PowerCenter-Workflows.html)

### 1. Restartable ETL Design

Restartability is the ability to restart an ETL job if a processing step fails to execute properly. This will avoid the need of any manual cleaning up before a failed job can restart. You want the ability to restart processing at the step where it failed as well as the ability to restart the entire ETL session.

Please refer the article "[Restartability Design Pattern for Different Type ETL Loads](http://www.disoln.org/2013/02/Restartability-Design-for-Different-Type-ETL-Loads.html)**"**for more details on restartable ETL design.

### 2. Workflow Recovery

Workflow recovery allows you to continue processing the workflow and workflow tasks from the point of interruption. During the workflow recovery process Integration Service access the workflow state, which is stored in memory or on disk based on the recovery configuration. The workflow state of operation includes the status of tasks in the workflow and workflow variable values.

MEDICINES –

1. Newpogen – cancer related drug

2. EPOGEN® is a prescription medicine used to treat a lower than normal number of red blood cells (anemia) caused by chronic kidney disease in patients

3. Prolia(Denosumab injection) – for issues with bones in men and women

4. [Repatha](https://www.drugs.com/repatha.html)

5. [Nplate](https://www.drugs.com/nplate.html) - platelet-stimulating agents

ROWID:

SELECT ROWID, ename

FROM emp

WHERE deptno = 20;

ROWID ENAME

------------------ ----------

AAAAqYAABAAAEPvAAA SMITH

AAAAqYAABAAAEPvAAD JONES

AAAAqYAABAAAEPvAAH SCOTT

AAAAqYAABAAAEPvAAK ADAMS

AAAAqYAABAAAEPvAAM FORD

ROWNUM

For each row returned by a query, the ROWNUM pseudo-column returns a number indicating the order in which Oracle selects the row from a table or set of joined rows. The first row selected has a ROWNUM of 1, the second has 2, and so on.

You can use ROWNUM to limit the number of rows returned by a query, as in this example:

SELECT \* FROM emp WHERE ROWNUM < 10;

Normalization Rule

Normalization rule are divided into following normal form.

1. First Normal Form – No repeating groups in the columns.
2. Second Normal Form – 1st + no partial dependency of any non-key field on the key field/s.
3. Third Normal Form – 2nd + no transitive dependency of non-key field on another non-key field.
4. BCNF – 3rd + there should not be any overlapping candidate keys.

**Patient Protection and Affordable Care Act**, often shortened to the **Affordable Care Act** (**ACA**) and nicknamed **Obamacare**

Affordable Care Act made Medicare more fiscally sound.

The life of the Medicare Trust fund will be extended to at least 2029 -- a 12-year extension due to reductions in waste, fraud and abuse, and Medicare costs, which will provide you with future savings on your premiums and coinsurance.

**Health Insurance Portability and Accountability Act of 1996 (“HIPAA”).**

**HIPAA Title II - Administrative Simplification**

* Four Rules of HIPAA:

1. HIPAA Privacy Rule – Saving, Accessing and Sharing of PHIs(Protected Health Information)
2. HIPAA Security Rule – Security standards for to protect PHIs

Administrative safeguards

Technical safeguards

Physical safeguards

1. HIPAA Enforcement Rule
2. HIPAA Breach Notification Rule

Covered Entity – Anyone who provides treatment, payments, operation in health care.

Business associate – anyone who has access to patient information.

Department of Health and Human Services (“HHS”)

**Protected health information (PHI) is “identifiable” health information acquired in the course of serving patients.**

**Any of the following data make health information “identifiable”:**

–Name

–Social security number

–Street and email addresses

–Employer

–Telephone and fax numbers

–Member or account numbers

– (e.g. medical record number, health plan identification number)

–Relatives’ names

–Date of service, birth or death

–Fingerprints, photographs, voice recordings

–Certificate or license numbers

–Any other linked number, code, characteristic (e.g. device identifiers, serial numbers)

**HMS Holdings Corp.** (**H**ealthcare **M**anagement **S**ystems) ([NASDAQ](https://en.wikipedia.org/wiki/NASDAQ):[HMSY](http://www.nasdaq.com/symbol/hmsy/real-time)) was set up in 1974 and now is based in [Irving, Texas](https://en.wikipedia.org/wiki/Irving,_Texas).

The company markets on cost containment services to healthcare payers and sponsors, including co-ordination of benefits services and program integrity services.

Both services aim at making the healthcare claims paid correctly.

Customers of the company involve both government and individuals, like

State Medicaid agencies (served for 47states in 2012),

[Centres for Medicare & Medicaid Services](https://en.wikipedia.org/wiki/Centers_for_Medicare_%26_Medicaid_Services) (CMS),

[Pharmacy Benefit Managers](https://en.wikipedia.org/wiki/Pharmacy_Benefit_Managers) (PBMs),

[Veterans Health Administration](https://en.wikipedia.org/wiki/Veterans_Health_Administration) ((VHA)).

In December 2012, the Company acquired MRM.[[1]](https://en.wikipedia.org/wiki/HMS_Holdings#cite_note-1)

Rep. Tom Price

Price's "[Empowering Patients First Act](https://tomprice.house.gov/HR2300)" would repeal Obamacare and replace it with a 401(k)-like plan where you'd be exposed to the ravages of the private insurance market and be given tax credits based on your age.

**Electronic Remittance Advice (ERA):**The digital version of EOB, which specifies the details of payments made on a claim either by an insurance company or required by the patient.

**Blue Cross Blue Shield:**Blue Cross Blue Shield is a federation of 38 health insurance companies in the U.S. (some of which are non-profit companies) that offer health insurance options to eligible persons in their area. Blue Cross Blue Shield offers healthcare plans to over 100 million people in the U.S.

**Civilian Health and Medical Program of Uniform Services (CHAMPUS):**CHAMPUS (now known as TRICARE) is the federal health insurance program for active and retired service members, their families, and the survivors of service members.

**Clean Claim:**This refers to a medical claim filed with a health insurance company that is free of errors and processed in a timely manner. Some providers may send claims to organizations that specialize in producing clean claims, like clearinghouses.

**Clearinghouse:**Clearinghouses are facilities that review and correct medical claims as necessary before sending them to insurance companies for final processing. This meticulous editing process for claims is known in the medical billing industry as “scrubbing.”

**Coding:**Coding is the process of translating a physician’s documentation about a patient’s medical condition and health services rendered into medical codes that are then plugged into a claim for processing with an insurance company. Medical billing specialists must be familiar with many code sets in order to perform their job duties.

**COBRA Insurance:**A federal program that allows a person terminated from their employer to retain health insurance they had with that employer for up to 18 months, or 36 months if the former employee is disabled.

**Date of Service (DOS):**The date when a provider performed healthcare services and procedures.

**Downcoding:**Downcoding occurs when an insurance company finds there is insufficient evidence on a claim to prove that a provider performed coded medical services and so they reduce or remove those codes. Downcoding usually reduces the cost of a claim.

**Duplicate Coverage Inquiry (DCI):**A formal request typically submitted by an insurance carrier to determine if other health coverage exists for a patient.

**Group Health Plan (GPH):**A plan provided by an employer to provide healthcare options to a large group of employees.

**Medical Coder:**A medical coder is responsible for assigning various medical codes to services and healthcare plans described by a physician on a patient’s superbill.

**Medicare:**Medicare is a government insurance program started in 1965 to provide healthcare coverage for persons over 65 and eligible people with disabilities.

**Medicaid:**Medicaid is a joint federal and state assistance program started in 1965 to provide health insurance to lower-income persons. Both state and federal governments fund Medicaid programs, but each state is responsible for running its own version of Medicaid within the minimum requirements established by federal law.

**Medigap:**Medigap is supplemental health insurance under Medicaid for eligible persons who need help covering co-pays, deductibles, and other large fees.

# How to run concurrent/multiple instances of an Informatica workflow?

By default Informatica workflows cannot be triggered again if they are already running. But there are scenarios where we need to trigger multiple instances of the same workflow. One such example could be when you need to load multiple source files through a single workflow, which gets triggered as soon as the file arrives.

So how do you enable a workflow to run multiple instances concurrently?

Well, there is a flag in the workflow properties which we need to turn on. Below are the steps to achieve this:-

**Step1**:- Open the workflow in workflow monitor and select the Edit Workflow/Worklet  
in the right click command menu for the workflow.

**Step2**:- In the general tab enable the property Configure Concurrent Execution.

**Step3**:- Open the Configure Concurrent Execution dialogue box. There are two options-  
a. Allow concurrent run with same instance name  
The workflow name remains same for every concurrent execution.  
b. Allow concurrent run only with unique instance names  
Here you will need to create at least one instance name in the lower half  
of this dialogue box. You can also specify the parameter file for each  
instance

Now you can run different instances by selecting the Start Workflow Advanced option in the workflow right click command menu. Here you can select which instance(s) you need to run.

However in production environment we execute the workflows from bat or shell scripts. You can also trigger multiple instances of the workflow from pmcmd command if the workflow is enabled for concurrent execution. In this case you can leave the property (a) in step 3 selected.

Below is the command using which you can start and name a particular instance of the same workflow.

pmcmd %workflow\_name% %informatica\_folder\_name% -paramfile %paramfilepathandname% -rin %instance\_name%

Using this command you can run as many instances of a single workflow with unique instance names appended to the original workflow name.

### [**SQL Transformation - Script Mode Static**](http://gowtham-informatica-reference.blogspot.in/2014/12/sql-transformation-script-mode-static.html)

The SQL transformation processes SQL queries midstream in a pipeline. You can insert, delete, update, and retrieve rows from a database. You can pass the database connection information to the SQL transformation as input data at run time. The transformation processes external SQL scripts or SQL queries that you create in an SQL editor. The SQL transformation processes the query and returns rows and database errors.

For example, you might need to create database tables before adding new transactions. You can create an SQL transformation to create the tables in a workflow. The SQL transformation returns database errors in an output port. You can configure another workflow to run if the SQL transformation returns no errors.

When you create an SQL transformation, you configure the following options:

**Script Mode:**

An SQL transformation running in script mode runs SQL scripts from text files. We pass each script file name from the source to the SQL transformation ScriptName port. The script file name contains the complete path to the script file.

When we configure the transformation to run in script mode, we create a passive transformation. The transformation returns one row for each input row. The output row contains results of the query and any database error.

When the SQL transformation runs in script mode, the query statement and query data do not change. When you need to run different queries in script mode, you pass the scripts in the source data. Use script mode to run data definition queries such as creating or dropping tables.

When we configure an SQL transformation to run in script mode, the Designer adds the ScriptName input port to the transformation. When you create a mapping, you connect the ScriptName port to a port that contains the name of a script to execute for each row. You can execute a different SQL script for each input row. The Designer creates default ports that return information about query results.

An SQL transformation configured for script mode has the following default ports:

| Port | Type | Description |
| --- | --- | --- |
| ScriptName | Input | Receives the name of the script to execute for the current row. |
| ScriptResult | Output | Returns PASSED if the script execution succeeds for the row. Otherwise contains FAILED. |
| ScriptError | Output | Returns errors that occur when a script fails for a row. |

**Note:**The input file contains the information of the sql file name and path which contain insert, update or delete statements.

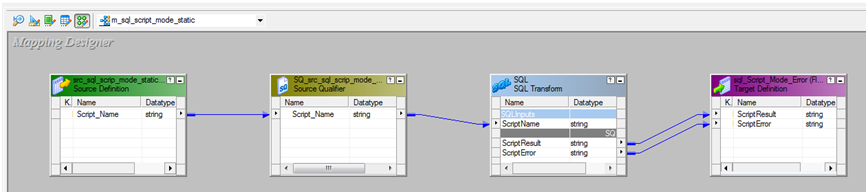
**Static Database Connection**

We can configure the SQL transformation to connect to a database with a static connection. A static database connection is a database connection defined in the Workflow Manager.

To use a static connection, choose a relational connection object when you configure the session. To avoid datatype conversion errors, use a relational connection for the same database type that is configured in the transformation.

Below is the sample mapping:

1. Create or import an source & target instance as below snapshot.

[](http://2.bp.blogspot.com/-w9ZZdGZchlk/VKJj7N_9ZtI/AAAAAAAAByg/7oY9WZvrjUQ/s1600/1.png)

Source: Script\_Name port of 255 scale

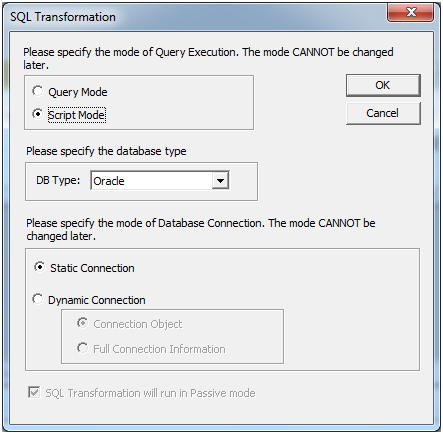
Target: ScriptResult port of 64 scale, ScriptError port of 4096 scale

2. Add SQL Transformation to the mapping.

Mode: Script Mode

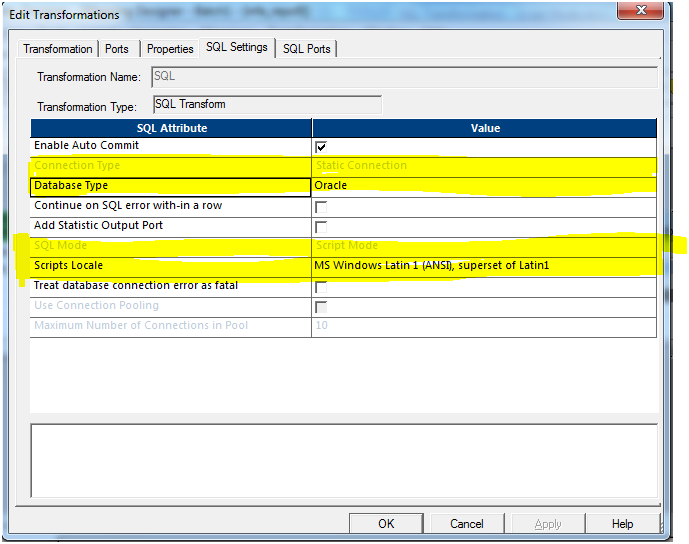
DB Type: Oracle (target database)

Database Connection: Static Connection

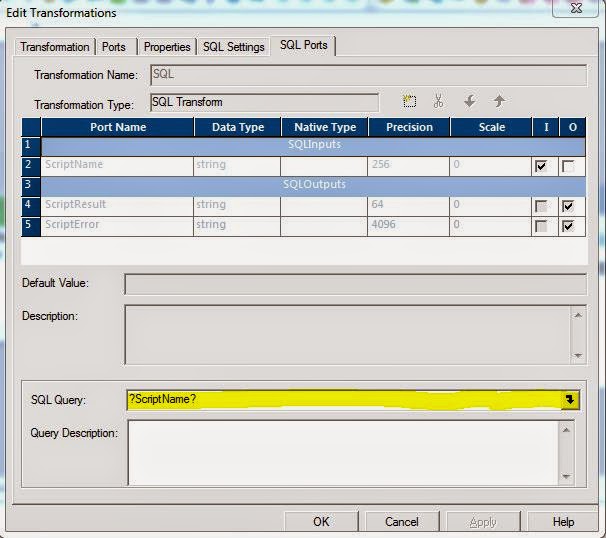
[](http://3.bp.blogspot.com/-1EIZ5-wGH74/VKJkApAqXpI/AAAAAAAABzg/NYlPBe_CQ7I/s1600/2.png)

3. Select 'SQL Transformation' edit it and go to tab: SQL Settings

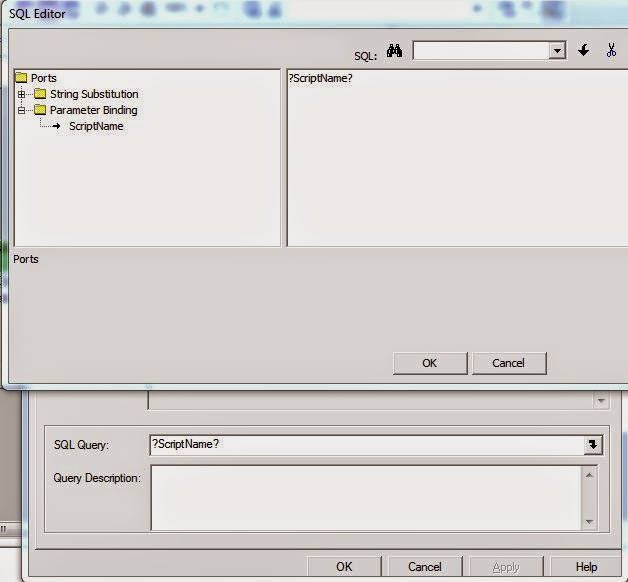
check the below highlighted information as your selection or not?

[](http://4.bp.blogspot.com/-gK9BIOaAQJE/VKJkH5-H4_I/AAAAAAAAB1A/X4hvwIeK_GE/s1600/3.png)

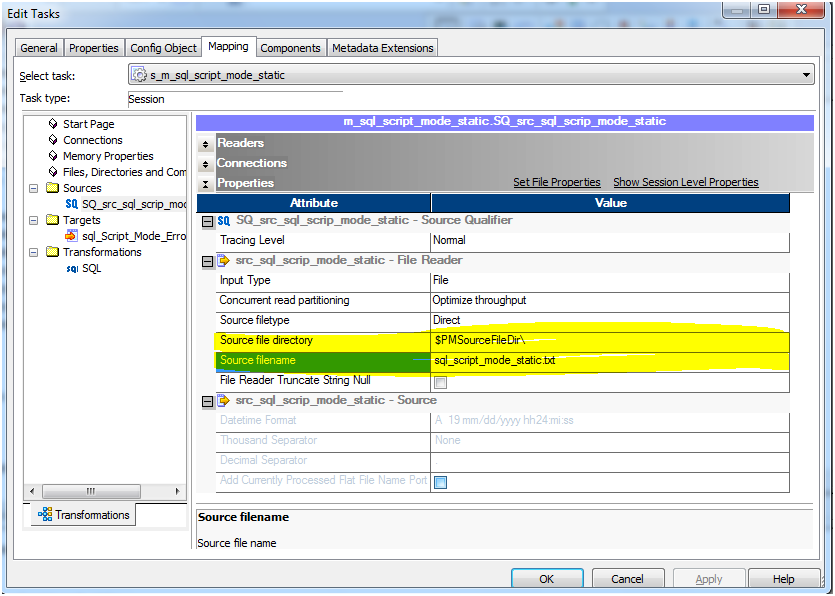
 Next to to SQL Ports tab:

[](http://1.bp.blogspot.com/-8hOtpqEWtTA/VKJobj0UgDI/AAAAAAAAB3g/qG-GlRlbNFQ/s1600/a.JPG)

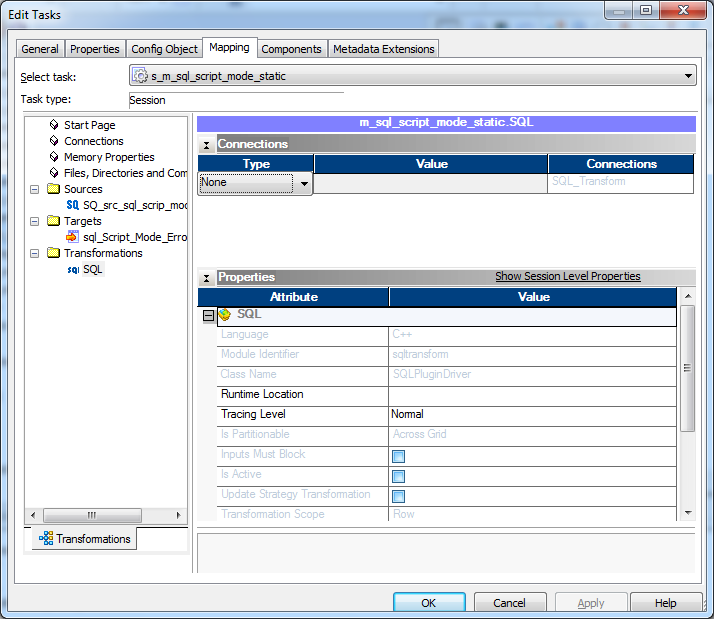
In SQL Query as the input port name as shown below

[](http://2.bp.blogspot.com/-U5-b4d93on0/VKJobR4egwI/AAAAAAAAB3c/sXOG_Gs__ms/s1600/b.JPG)

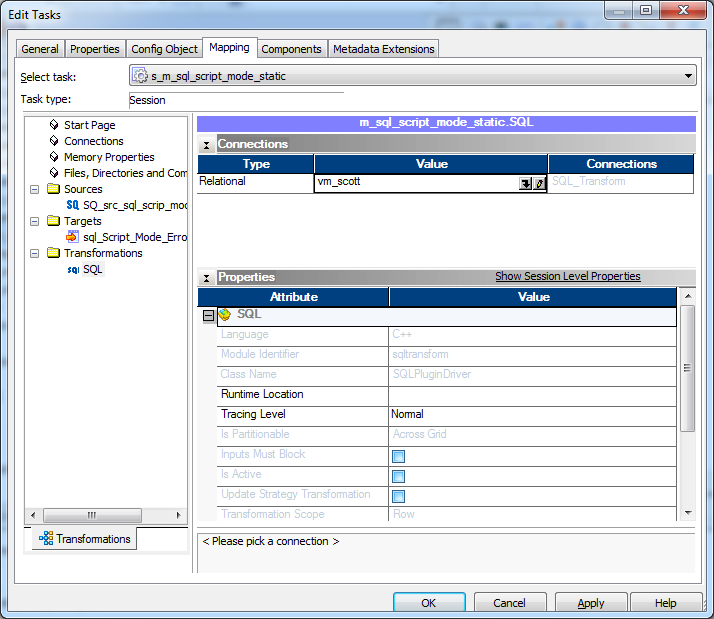
4. Then link the ports below SQ to SQL Transformation and SQL Transformation to Target instance.  
Validate the mapping and save it.  
  
5. Go the Workflow Manager and Create a workflow for the same.  
6. Create a session for the above mapping.  
7. Edit the session and go to Mapping tab, to pass source file details.

[](http://3.bp.blogspot.com/-fgnZaOLbID0/VKJkPLl5zMI/AAAAAAAAB3A/rXusd7uMfAA/s1600/5.png)

**Note: For Sql Transformation connection details passing.**  
1. Choose connection Type as None (In case you want to defined schema/dataname in the ddl's)

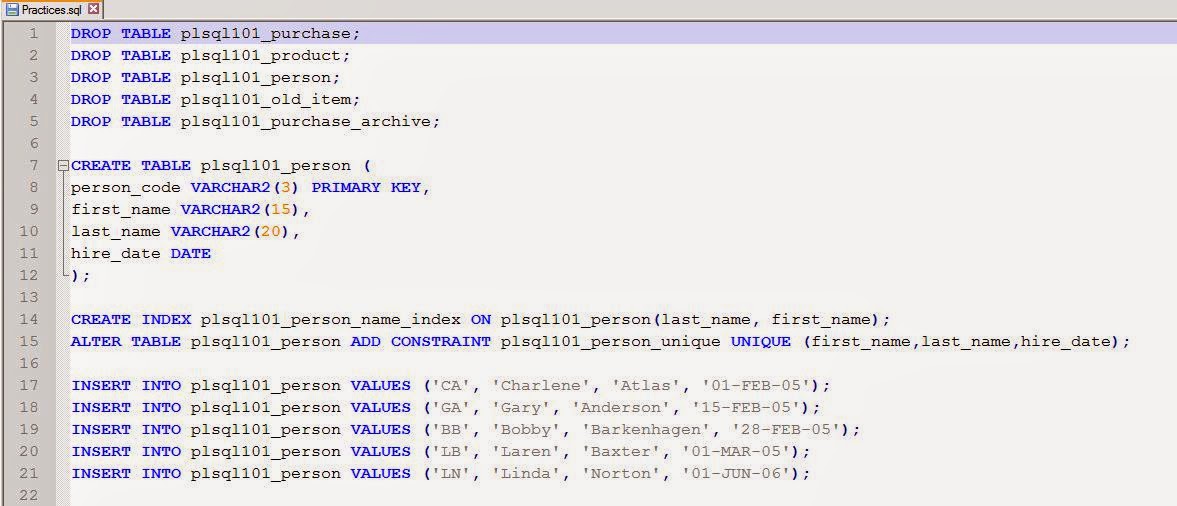
[](http://1.bp.blogspot.com/-dMPoU3yp1eE/VN3DJeVXAyI/AAAAAAAAB9o/UzeuxUoM10E/s1600/sql_script_s.png)

2 Choose connection Type as Relation (In case you not defined any schema/database name in ddl's)

[](http://4.bp.blogspot.com/-htmiYmLODxY/VN3D4CYEHpI/AAAAAAAAB9w/o1xFvnawOhY/s1600/sql_script_s1.png)

Source File: The Source file contain the information of the sql script file name and the path.

[http://1.bp.blogspot.com/-5n2Ln05Hbb0/VKJkPcQyOsI/AAAAAAAAB2c/U6SCQoiAqqI/s1600/6.png](http://1.bp.blogspot.com/-5n2Ln05Hbb0/VKJkPcQyOsI/AAAAAAAAB2c/U6SCQoiAqqI/s1600/6.png)  
And the Practices.sql may contain any sql statement like create, alter, drop, insert, update, delete or truncate as below:

[](http://4.bp.blogspot.com/-ZwuwSkiIOnU/VKJqaNdTOxI/AAAAAAAAB3w/CsVF7UXZUWQ/s1600/c.JPG)