**Customer creation using Application:**

1. **Define Customer**

**Defining a Customer Profile Classes:**

Use Customer Profiles to group customer accounts with similar creditworthiness, business

volume, payment cycles, and late charge policies. For each profile class you can define

information such as credit limits, payment terms, statement cycles, invoicing, and discount

information. You can also define amount limits for your late charges and statements for each

currency in which you do business.

Define your standard customer profiles in the Customer Profile Classes window. These profiles contain generic options that you can use to group your customers into broad categories.

For example, you might define three categories: one for prompt paying customers; one for

late paying customers with high late charge rates; and a third for customers who mostly pay

on time, with discount incentives for early payment. You can also use the profile class

'DEFAULT,' which Oracle Receivables provides.

You can assign a profile class to customer accounts and sites on the Create Customer page or

the Account Profile sub tab of the Account Overview page. The customer profile class you

assign provides the default values, which you can optionally customize to meet specific

requirements for each account or site.

Profile class options set at the account or site level take precedence over those defined at

the customer profile class level.

For statement site, dunning site, and late charges site profile amounts, Receivables uses the

profile amounts defined at the site level only if the site is assigned a dunning, statement, or

late charge business purpose and you set the AR: Use Statement, Dunning, and Late Charges

Site Profiles profile option to Yes. Otherwise, Receivables uses the profile amounts defined on

the billing site.

**Steps to define profile classes:**

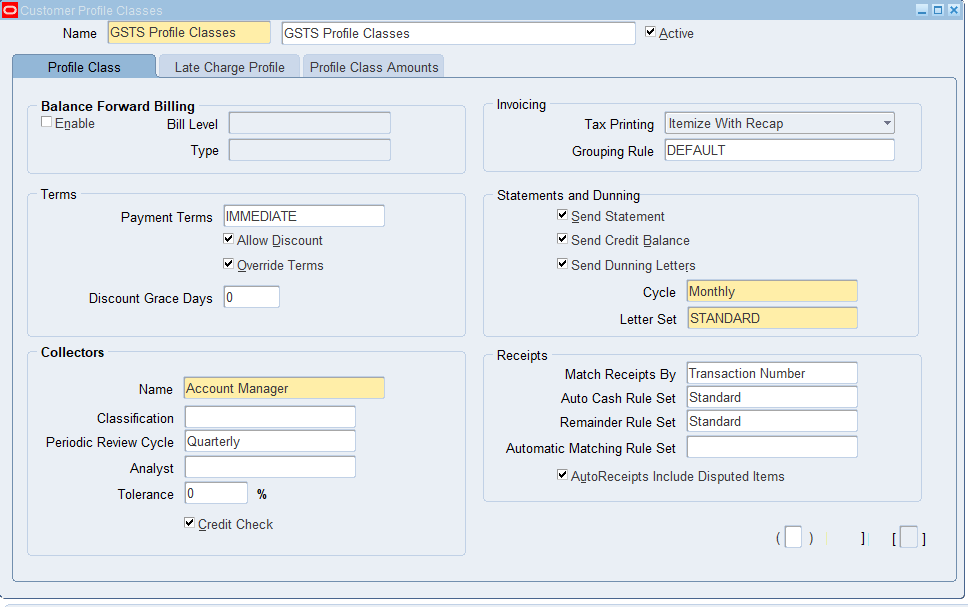
To define a new customer profile class:

* Navigate to the Customer Profile Classes window.
* Enter a Name and a brief description of this profile class.
* Check the Active check box to make this profile class active.

**Note:** Active profile classes appear in the Profile Class list on the Create Customer page or the Account Profile sub tab of the Account Overview page.

* Enter other profile class information.

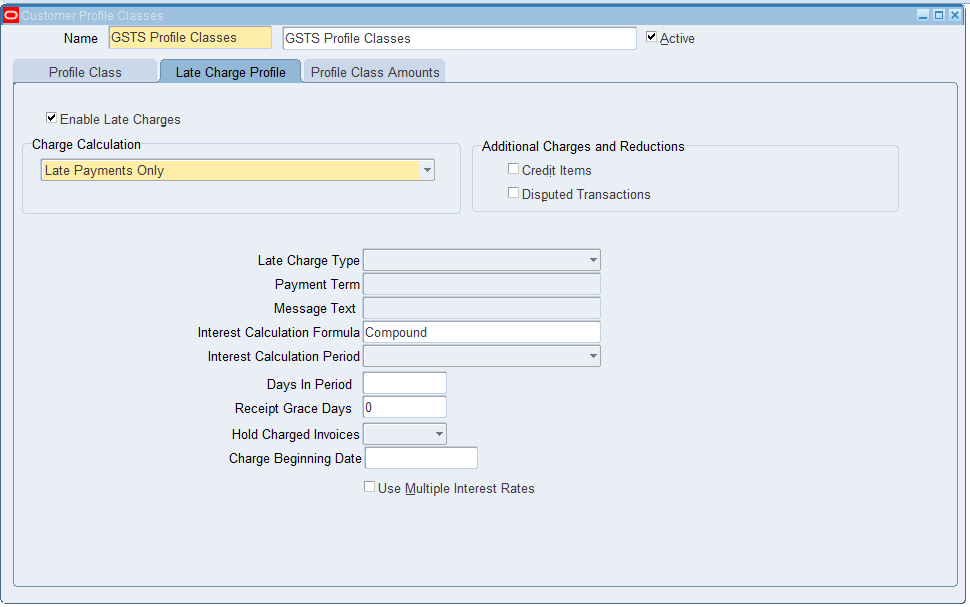
**Navigation: -** Receivables 🡪 Customers 🡪 Profile Classes



**Late Charge Profile Class**

Use the Late Charge Profile tab to enter default information for late charge

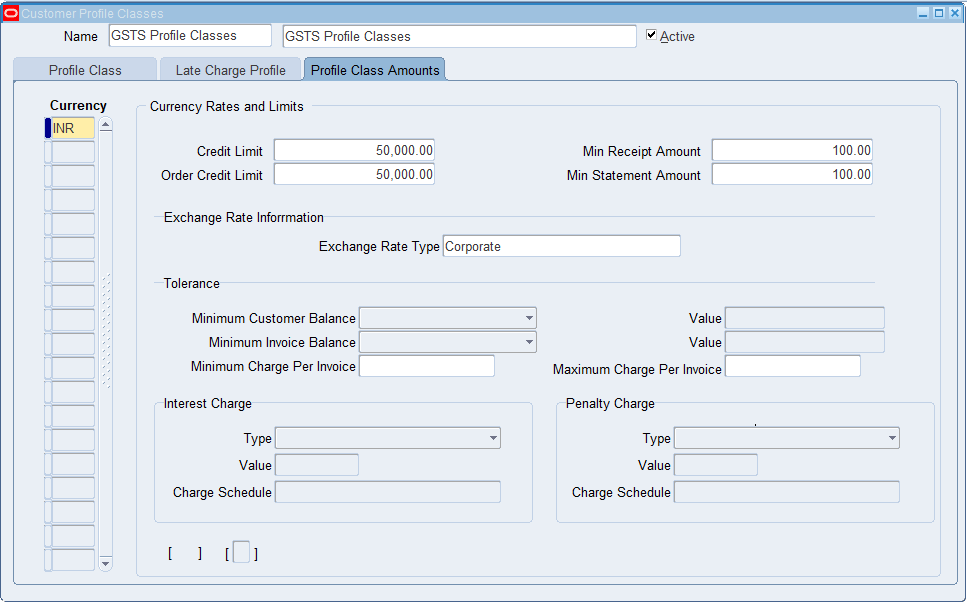
calculations.



**Profile Class Amounts**

Open the Profile Class Amounts tabbed region, then enter the Currency in which customers assigned to this profile class will do business. For each currency that you enter, define the currency

rates and limits for customers using this profile class, including Credit Limit and exchange rate information.



**Creating the Customer:**

This procedure lets you create:

* A new customer
* An account for the customer
* An account site for the account
* A business purpose for the account site

**To create a customer:**

**1. Navigate to the Create Customer page.**

The Create Customer page displays differently, depending upon your customer type

selection, Organization or Person, on the Customer Search page.

**2. Enter customer, account, and account site details.**

**Attention:** To create a customer, you must specify at least a customer name, account

number, and account site address. All other details are optional depending on the

business requirements at your enterprise.

**Note:** You can either enter a customer account number or enable Automatic Customer

Numbering.

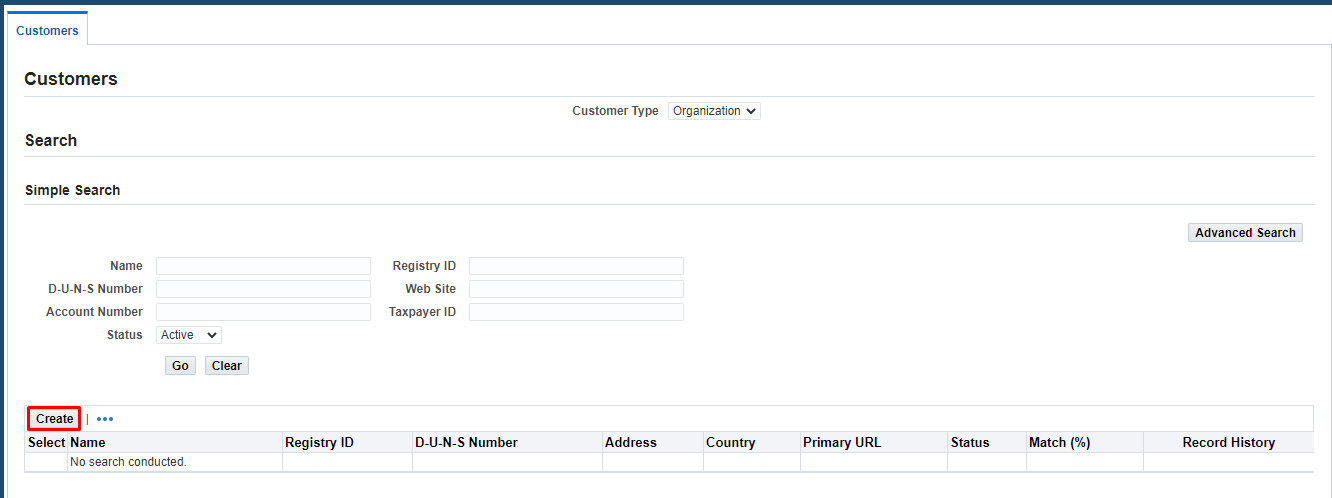
Even if you do not enable Automatic Customer Numbering, customer account numbers

are automatically generated if the MO: Default Operating Unit profile option does not

have a value.

**3. Click Save and Add Details or Apply.**

**Navigation: -** Receivables 🡪 Customers 🡪 Customers.

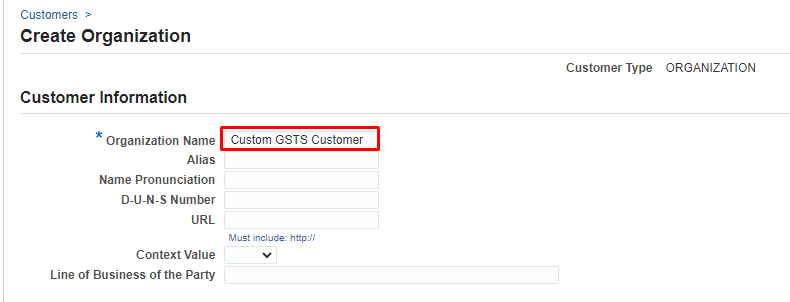


In oracle we have two of customers.

* Organization
* Person

Select which type of customer you want to create (if it is a company select Organization, if it an induvial select Person).

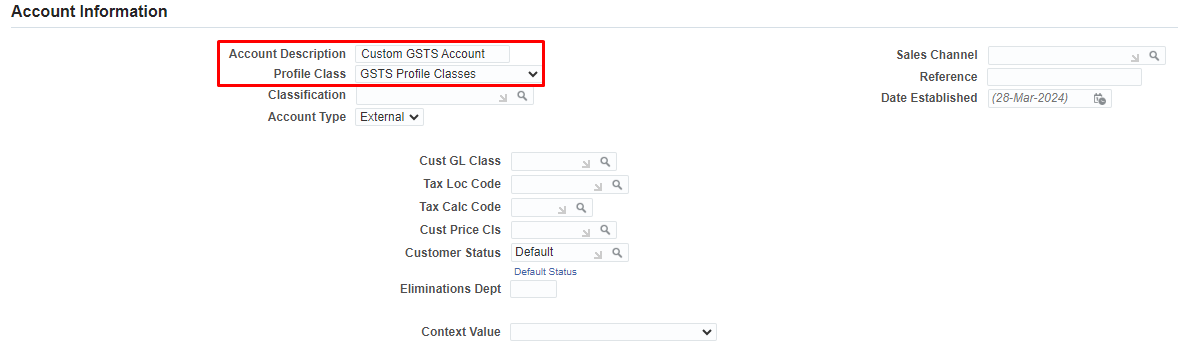
Click on Create Button to create a new customer.



In the Customer Information give mandatory fields. i.e. **Organization Name**

Enter Organization name as **Custom GSTS Customer**.

After entering the Customer information, we need t give the Account Information for the customer.



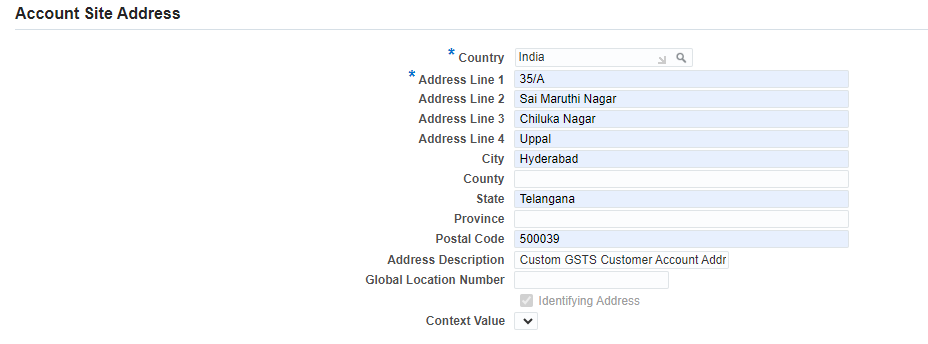
In the above form,

**Sales Channel:** A sales channel is a category of sales transaction that you can differentiate by customer type and distribution channel, such as web-shop or retail outlet.

Enter the address details.

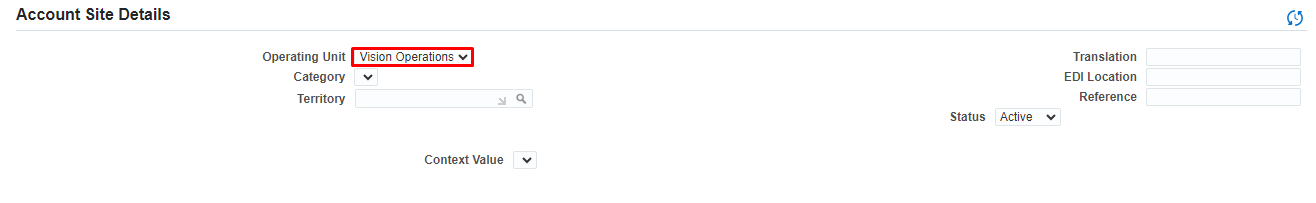
**Customer classification:** Customer classification in Oracle Apps R12 allows a utility to categorize customers into groups or "classifiers" for tracking or targeting purposes.

After entering the Account Information, we need to mention Account site Address.



After entering the Account site Address, we need to give the Account Site Details for the above address.

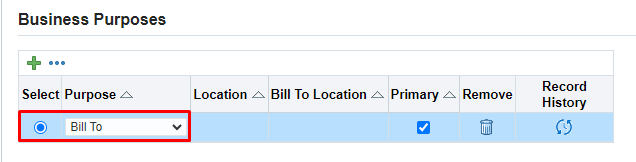
For general purpose, I’m selecting only operating unit. Based on the requirement we need to mention the appropriate details.

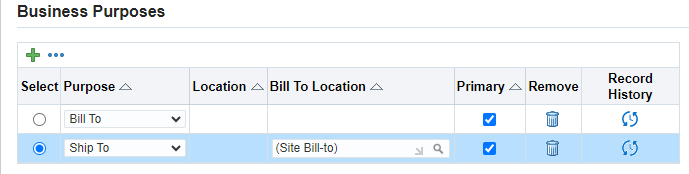


After entering the Account site Details, we need to select the business purpose of the address.

Ex: For which purpose we need to use this address (Bill to, Ship to, etc.)

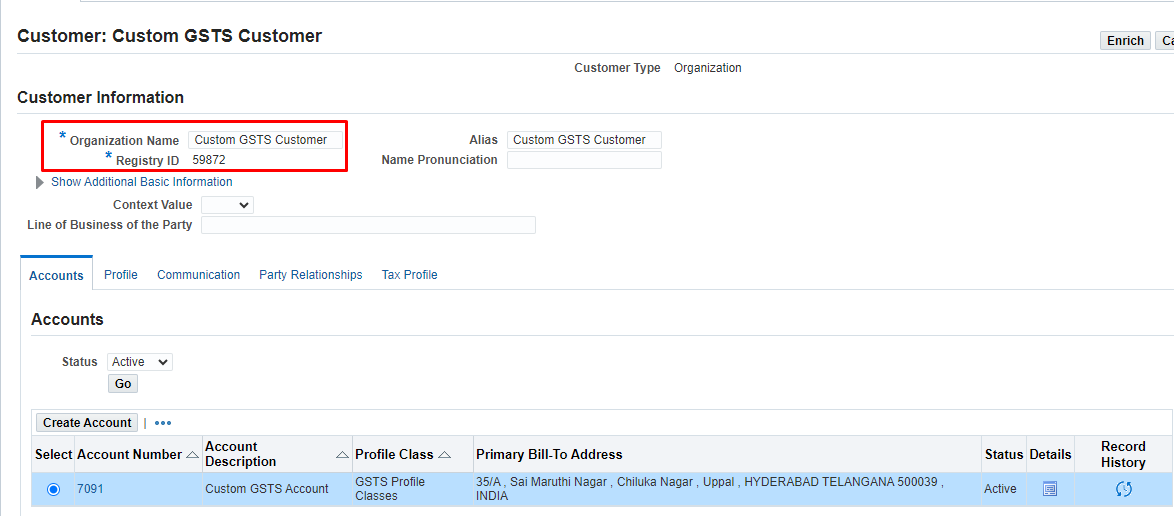
By default, it is bill to. You can select multiple business purposes for one Address. Or if you want to change the address, you change the address for different purposes. Here I’m creating Bill to and Ship to for same address.

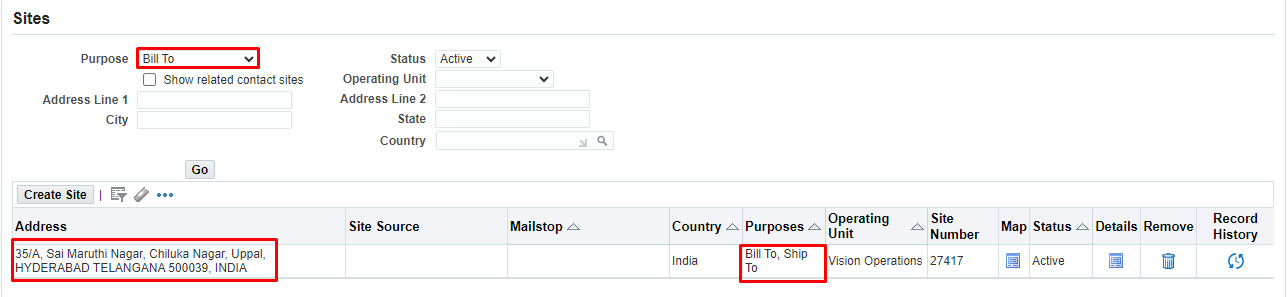




If you want to change the Account site address and Account site details, select the purpose and edit the details at above.

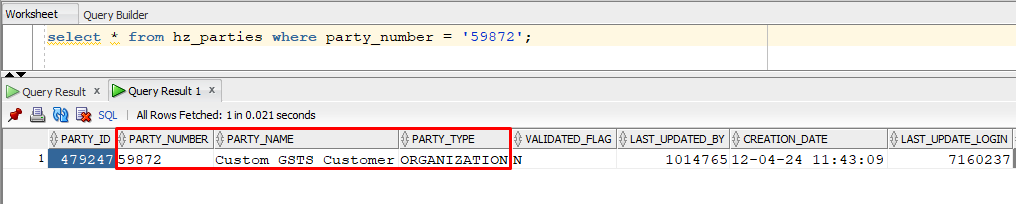
When you click on Apply, the customer details are showed as below.



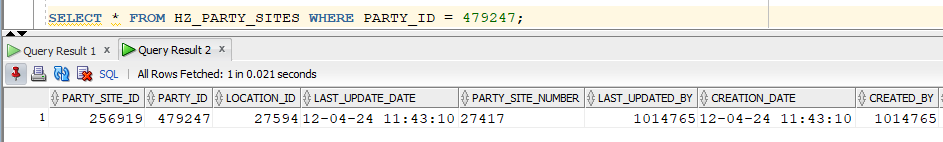


Here, some tables while creating the Customer.

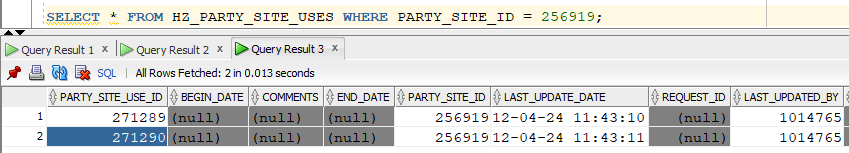
**SELECT \* FROM HZ\_PARTIES WHERE PARTY\_NUMBER = '59872';**



**SELECT \* FROM HZ\_PARTY\_SITES WHERE PARTY\_ID = 479247;**



**SELECT \* FROM HZ\_PARTY\_SITE\_USES WHERE PARTY\_SITE\_ID = 256919;**



**Customer Creation Using API Approach:**

List of API’s Used in the Creation of Customer and effected tables are listed below.

|  |  |  |
| --- | --- | --- |
| **Entity** | **API** | **Tables** |
| Party – Organization | HZ\_PARTY\_V2PUB.create\_organization | HZ\_PARTIES HZ\_ORGANIZATION\_PROFILES |
| Party - Person | HZ\_PARTY\_V2PUB.create\_person | HZ\_PARTIES HZ\_PERSON\_PROFILES |
| Location | HZ\_LOCATION\_V2PUB.create\_location | HZ\_LOCATIONS HZ\_LOCATIONS\_PROFILES |
| Party Site | HZ\_PARTY\_SITE\_V2PUB.create\_party\_site | HZ\_PARTY\_SITES |
| Party Site Use | HZ\_PARTY\_SITE\_V2PUB.create\_party\_site\_use | HZ\_PARTY\_SITE\_USES |
| Customer Account | HZ\_CUST\_ACCOUNT\_V2PUB.create\_cust\_account | HZ\_PARTIES HZ\_CUST\_ACCOUNTS HZ\_CUSTOMER\_PROFILES HZ\_CUST\_PROFILE\_AMTS |
| Account Site | HZ\_CUST\_ACCOUNT\_SITE\_V2PUB.create\_cust\_acct\_site | HZ\_CUST\_ACCT\_SITES\_ALL |
| Account Site Use | HZ\_CUST\_ACCOUNT\_SITE\_V2PUB.create\_cust\_site\_use | HZ\_CUST\_SITE\_USES\_ALL HZ\_CUSTOMER\_PROFILES HZ\_CUST\_PROFILE\_AMTS |

List of all Customer API’s



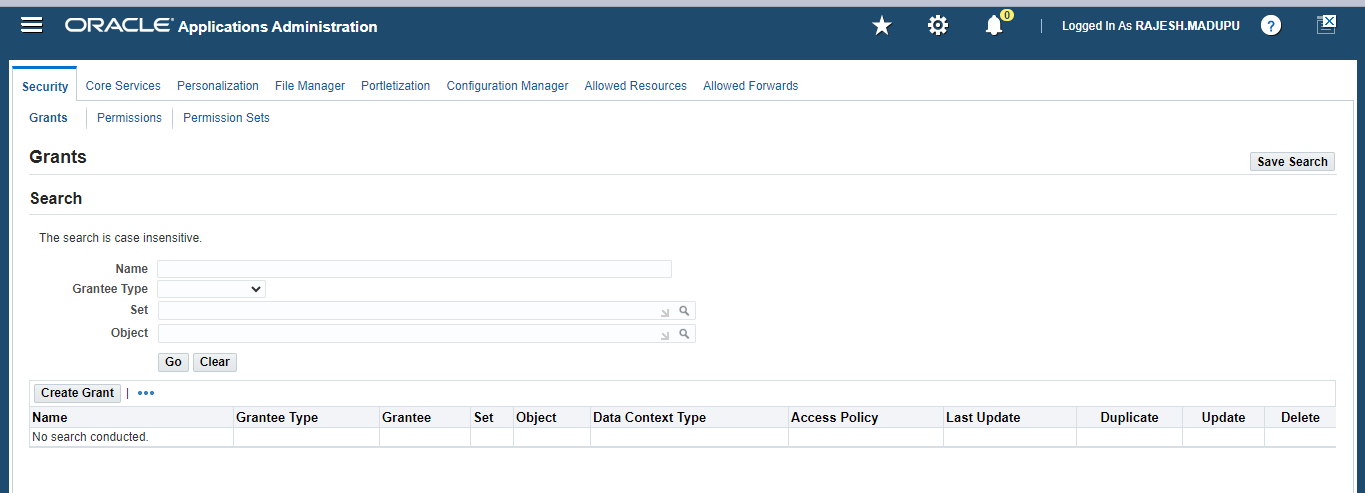
Customer is created using the API, the customer will create, but it is not visible from application,

To overcome this problem;

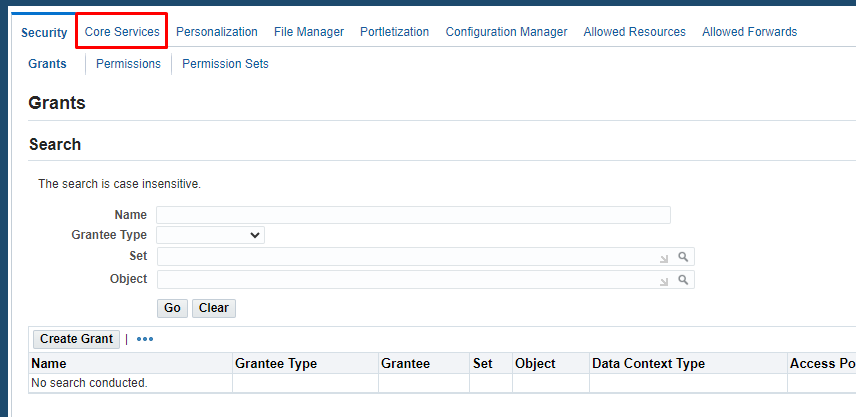
1. **Cleare Global Cache**

**Navigation🡪** Functional Administrator🡪 Core Services 🡪 Caching Framework 🡪 Global Configuration 🡪 Clear All Cache 🡪 Ignore warning message.

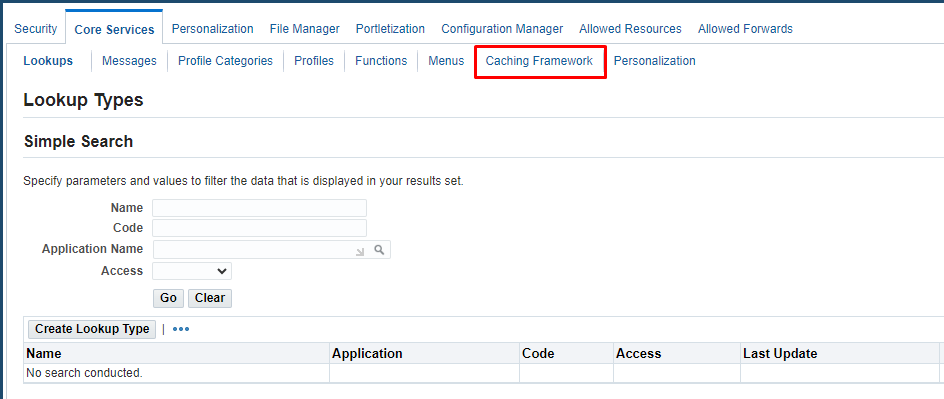
Functional Administrator responsibility.



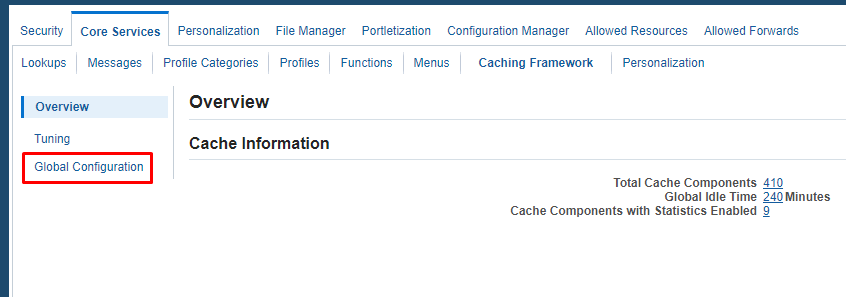
Click on Core Services



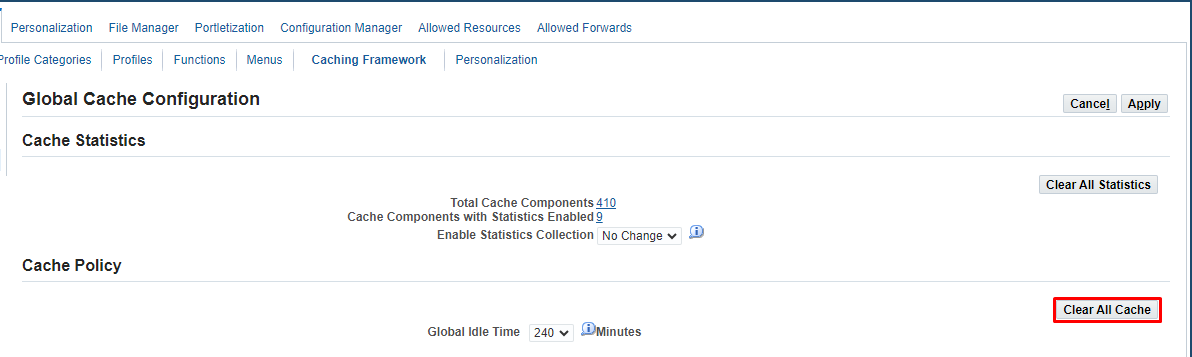
Click on Caching Framework



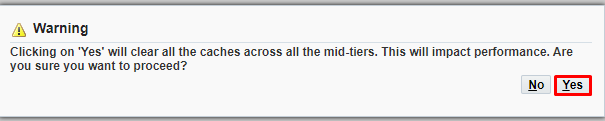
Click on Global Configuration



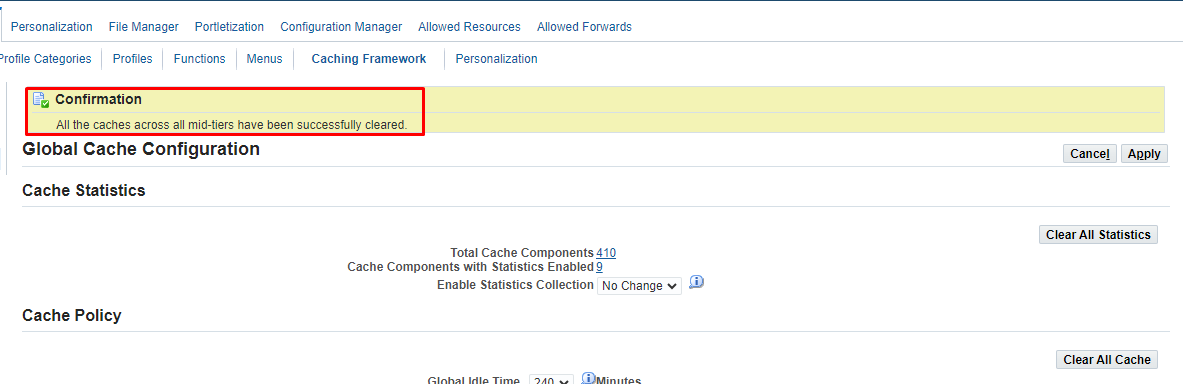
Click on clear cache



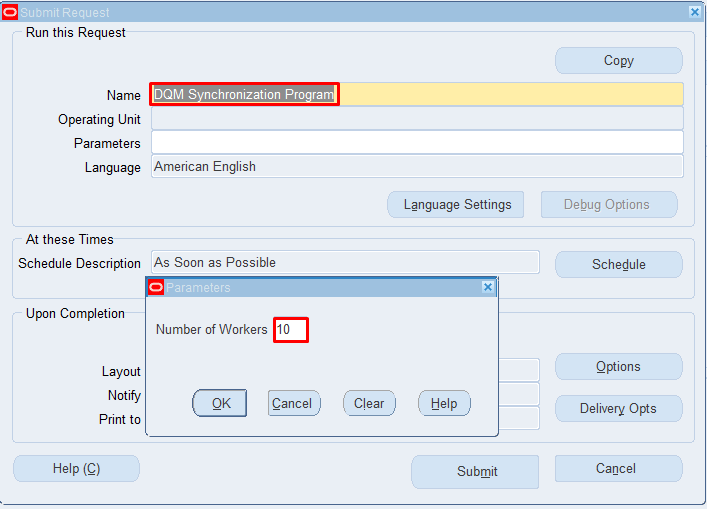
Click on Yes

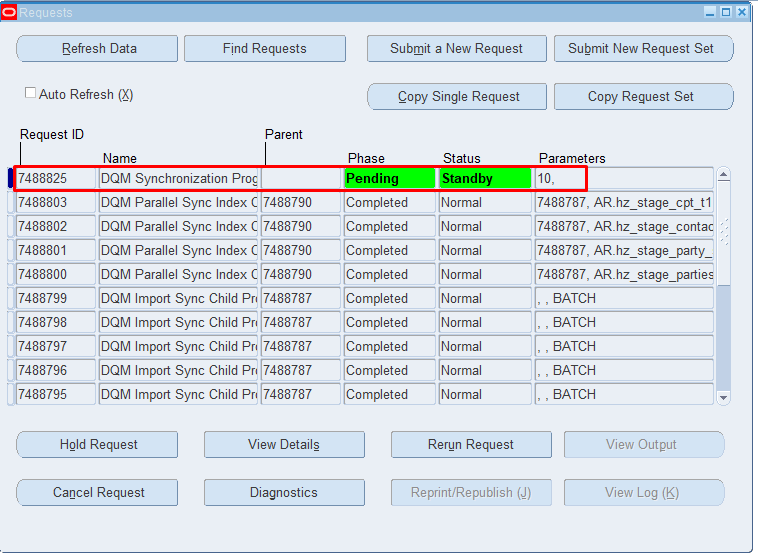


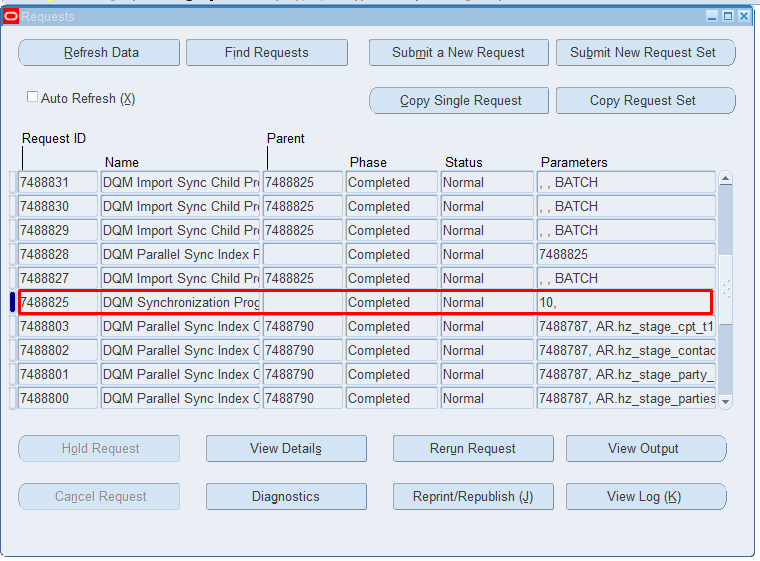
Confirmation message.



1. **Ran the "DQM Synchronization Program" manually from "Trading Community Manager" responsibility**







**CREATE PARTY AND CUSTOMER ACCOUNT:**

SET SERVEROUTPUT ON;

DECLARE

p\_cust\_account\_rec HZ\_CUST\_ACCOUNT\_V2PUB.CUST\_ACCOUNT\_REC\_TYPE;

p\_organization\_rec HZ\_PARTY\_V2PUB.ORGANIZATION\_REC\_TYPE;

p\_customer\_profile\_rec HZ\_CUSTOMER\_PROFILE\_V2PUB.CUSTOMER\_PROFILE\_REC\_TYPE;

x\_cust\_account\_id NUMBER;

x\_account\_number VARCHAR2(2000);

x\_party\_id NUMBER;

x\_party\_number VARCHAR2(2000);

x\_profile\_id NUMBER;

x\_return\_status VARCHAR2(2000);

x\_msg\_count NUMBER;

x\_msg\_data VARCHAR2(2000);

begin

p\_cust\_account\_rec.account\_name := ‘Custom GSTS Account 2’;

p\_cust\_account\_rec.created\_by\_module := 'BO\_API';

p\_organization\_rec.organization\_name := ’Custom GSTS Customer 2’;

p\_organization\_rec.created\_by\_module := 'BO\_API';

***DBMS\_OUTPUT.PUT\_LINE***('Calling the API hz\_cust\_account\_v2pub.create\_cust\_account');

HZ\_CUST\_ACCOUNT\_V2PUB.CREATE\_CUST\_ACCOUNT (

p\_init\_msg\_list => FND\_API.G\_TRUE,

p\_cust\_account\_rec =>p\_cust\_account\_rec,

p\_organization\_rec =>p\_organization\_rec,

p\_customer\_profile\_rec=>p\_customer\_profile\_rec,

p\_create\_profile\_amt =>FND\_API.G\_FALSE,

x\_cust\_account\_id =>x\_cust\_account\_id,

x\_account\_number =>x\_account\_number,

x\_party\_id =>x\_party\_id,

x\_party\_number =>x\_party\_number,

x\_profile\_id =>x\_profile\_id,

x\_return\_status =>x\_return\_status,

x\_msg\_count =>x\_msg\_count,

x\_msg\_data =>x\_msg\_data);

IF x\_return\_status = fnd\_api.g\_ret\_sts\_success THEN

COMMIT;

***DBMS\_OUTPUT.PUT\_LINE***('Creation of Party and customer account is Successful ');

***DBMS\_OUTPUT.PUT\_LINE***('Output information ....');

***DBMS\_OUTPUT.PUT\_LINE***('x\_cust\_account\_id : '||x\_cust\_account\_id);

***DBMS\_OUTPUT.PUT\_LINE***('x\_account\_number : '||x\_account\_number);

***DBMS\_OUTPUT.PUT\_LINE***('x\_party\_id : '||x\_party\_id);

***DBMS\_OUTPUT.PUT\_LINE***('x\_party\_number : '||x\_party\_number);

***DBMS\_OUTPUT.PUT\_LINE***('x\_profile\_id : '||x\_profile\_id);

ELSE

***DBMS\_OUTPUT.put\_line*** ('Creation of Party and customer account failed:'||x\_msg\_data);

ROLLBACK;

FOR i IN 1 .. x\_msg\_count LOOP

x\_msg\_data := oe\_msg\_pub.get( p\_msg\_index => i, p\_encoded => 'F');

***dbms\_output.put\_line***( i|| ') '|| x\_msg\_data);

END LOOP;

END IF;

***DBMS\_OUTPUT.PUT\_LINE***('Completion of API');

END;

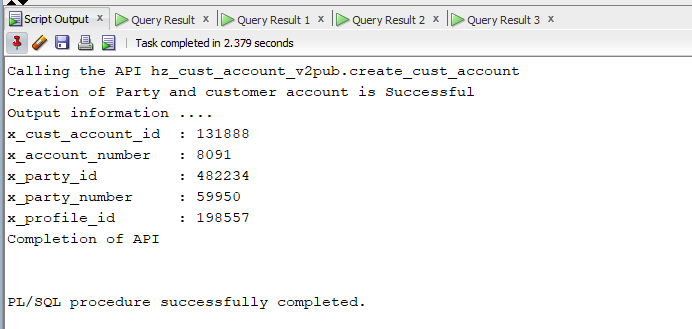
In this program we are using HZ\_CUST\_ACCOUNT\_V2PUB.CREATE\_CUST\_ACCOUNT (Using this API we can create Customer as Organization only; we cannot create Customer as a Person using this API) Api to create Customer Account and Party.

In this program we need to pass Account Name and Organization Name and Created by module is BO\_API.

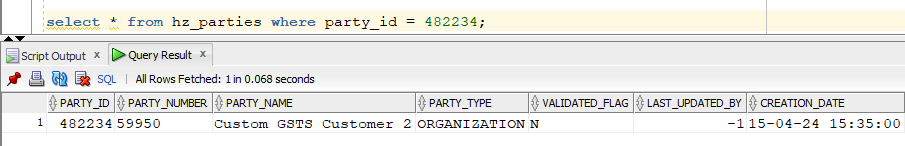
**BO\_API:** "BO\_API" typically refers to the Business Object Application Programming Interface. This API allows developers to interact programmatically with various business objects within the Oracle E-Business Suite environment. These business objects can include entities such as customers, suppliers, orders, invoices, and more.

The BO\_API provides a set of functions and procedures that enable developers to create, update, delete, and retrieve data from these business objects. It serves as a means to integrate custom applications or extensions with the Oracle E-Business Suite, allowing for automation of business **processes,** data synchronization, and other integration scenarios.

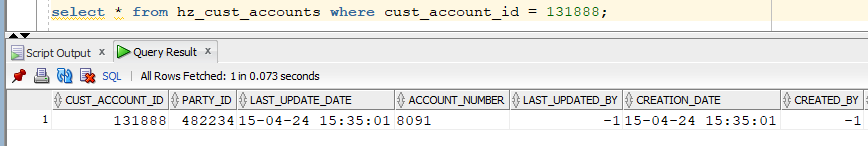
**Output:**



* Check whether the party id is created or not by using **HZ\_PARTIES** Table.



* Check whether the Cust Account id is created or not by using **HZ\_CUST\_ACCOUNTS** Table.



**CREATE LOCATIONS:**

set serveroutput on

DECLARE

l\_location\_id NUMBER;

l\_return\_status VARCHAR2 (200);

l\_msg\_count NUMBER;

l\_msg\_data VARCHAR2 (200);

l\_location\_rec hz\_location\_v2pub.location\_rec\_type;

BEGIN

l\_location\_rec.country := 'IN';

l\_location\_rec.address1 := ' CHC6+3V5';

l\_location\_rec.address2 := 'Adarsh Nagar';

l\_location\_rec.address3 := 'Uppal';

l\_location\_rec.address4 := 'Secunderabad';

l\_location\_rec.city := 'HYDERABAD';

l\_location\_rec.postal\_code := '500076';

l\_location\_rec.state := 'Telangana';

*--l\_location\_rec.province := NULL;*

*--l\_location\_rec.county := NULL;*

l\_location\_rec.created\_by\_module := 'BO\_API';

HZ\_LOCATION\_V2PUB.CREATE\_LOCATION (p\_init\_msg\_list => FND\_API.G\_FALSE,

p\_location\_rec => l\_location\_rec,

x\_location\_id => l\_location\_id,

x\_return\_status => l\_return\_status,

x\_msg\_count => l\_msg\_count,

x\_msg\_data => l\_msg\_data);

***DBMS\_OUTPUT.put\_line*** ('Status is ' || l\_return\_status||' - '||l\_location\_id);

FOR i IN 1 .. l\_msg\_count

LOOP

fnd\_msg\_pub.get (p\_msg\_index => i,

p\_encoded => 'F',

p\_data => l\_msg\_data,

p\_msg\_index\_out => l\_msg\_count);

***DBMS\_OUTPUT.put\_line*** (l\_msg\_data);

end loop;

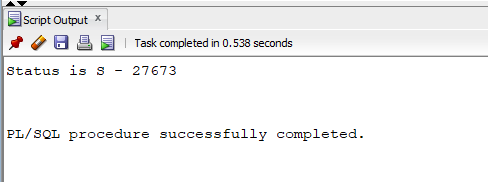
END;

In the above program we use HZ\_LOCATION\_V2PUB.CREATE\_LOCATION to create a location.

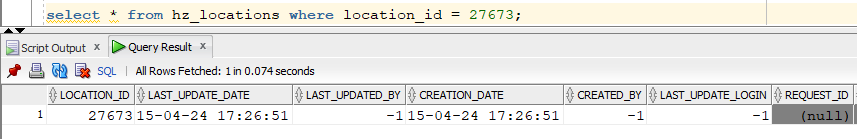
It is an Independent. No need to pass any Customer Information.

Here, we need to pass appropriate Address. If we give any random address, it won’t accept it.

Output:



* Check whether the location is created or not by using **HZ\_LOCATIONS** Table.



**CREATE PARTY SITE:**

set serveroutput on;

DECLARE

l\_party\_site\_id NUMBER;

l\_party\_site\_number VARCHAR2 (200);

l\_return\_status VARCHAR2 (200);

l\_msg\_count NUMBER;

l\_msg\_data VARCHAR2 (200);

l\_party\_site\_rec\_type hz\_party\_site\_v2pub.party\_site\_rec\_type;

BEGIN

l\_party\_site\_rec\_type.party\_id := 482234;

l\_party\_site\_rec\_type.location\_id := 27673;

l\_party\_site\_rec\_type.orig\_system\_reference := NULL;

l\_party\_site\_rec\_type.orig\_system := NULL;

l\_party\_site\_rec\_type.mailstop := NULL;

l\_party\_site\_rec\_type.identifying\_address\_flag := NULL;

l\_party\_site\_rec\_type.status := 'A';

l\_party\_site\_rec\_type.attribute\_category := NULL;

l\_party\_site\_rec\_type.attribute1 := NULL;

l\_party\_site\_rec\_type.attribute2 := NULL;

l\_party\_site\_rec\_type.attribute3 := NULL;

l\_party\_site\_rec\_type.attribute4 := NULL;

l\_party\_site\_rec\_type.attribute5 := NULL;

l\_party\_site\_rec\_type.attribute6 := NULL;

l\_party\_site\_rec\_type.attribute7 := NULL;

l\_party\_site\_rec\_type.attribute8 := NULL;

l\_party\_site\_rec\_type.attribute9 := NULL;

l\_party\_site\_rec\_type.attribute10 := NULL;

l\_party\_site\_rec\_type.attribute11 := NULL;

l\_party\_site\_rec\_type.attribute12 := NULL;

l\_party\_site\_rec\_type.attribute13 := NULL;

l\_party\_site\_rec\_type.attribute14 := NULL;

l\_party\_site\_rec\_type.attribute15 := NULL;

l\_party\_site\_rec\_type.attribute16 := NULL;

l\_party\_site\_rec\_type.attribute17 := NULL;

l\_party\_site\_rec\_type.attribute18 := NULL;

l\_party\_site\_rec\_type.attribute19 := NULL;

l\_party\_site\_rec\_type.attribute20 := NULL;

l\_party\_site\_rec\_type.language := NULL;

l\_party\_site\_rec\_type.addressee := NULL;

l\_party\_site\_rec\_type.created\_by\_module := 'BO\_API';

l\_party\_site\_rec\_type.application\_id := NULL;

l\_party\_site\_rec\_type.global\_location\_number := NULL;

l\_party\_site\_rec\_type.duns\_number\_c := NULL;

hz\_party\_site\_v2pub.create\_party\_site (

p\_init\_msg\_list => FND\_API.G\_FALSE,

p\_party\_site\_rec => l\_party\_site\_rec\_type,

x\_party\_site\_id => l\_party\_site\_id,

x\_party\_site\_number => l\_party\_site\_number,

x\_return\_status => l\_return\_status,

x\_msg\_count => l\_msg\_count,

x\_msg\_data => l\_msg\_data);

***DBMS\_OUTPUT.put\_line*** ('status is ' || l\_return\_status);

***DBMS\_OUTPUT.put\_line*** ('Party Site ID is ' || l\_party\_site\_id);

***dbms\_output.put\_line*** ('Party Site Number is ' || l\_party\_site\_number);

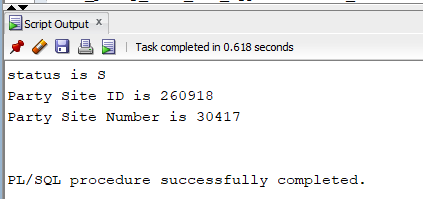
END;

In the above program we use HZ\_PARTY\_SITE\_V2PUB.CREATE\_PARTY\_SITE to create a party site,

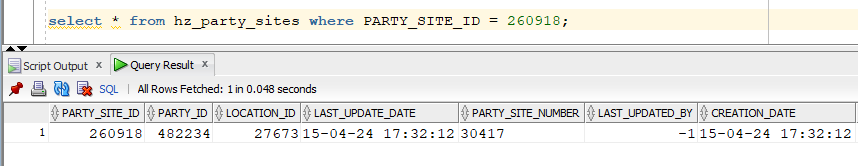
We can create Party site using Party and Location, without Party and Location we cannot create Party site.

So, in this program we need to pass Party\_id and Location\_id as a mandatory parameter’s.

Output:



* Check whether the party site is created or not by using **HZ\_PARTY\_SITES** Table.



**CREATE PARTY\_SITE\_USES:**

SET SERVEROUTPUT ON;

DECLARE

p\_party\_site\_use\_rec HZ\_PARTY\_SITE\_V2PUB.PARTY\_SITE\_USE\_REC\_TYPE;

x\_party\_site\_use\_id NUMBER;

x\_return\_status VARCHAR2(2000);

x\_msg\_count NUMBER;

x\_msg\_data VARCHAR2(2000);

BEGIN

*-- Initializing the Mandatory API parameters*

p\_party\_site\_use\_rec.site\_use\_type := upper('BILL\_TO');

p\_party\_site\_use\_rec.party\_site\_id := 260918;

p\_party\_site\_use\_rec.created\_by\_module := 'BO\_API';

***DBMS\_OUTPUT.PUT\_LINE***('Calling the API hz\_party\_site\_v2pub.create\_party\_site\_use');

HZ\_PARTY\_SITE\_V2PUB.CREATE\_PARTY\_SITE\_USE

(

p\_init\_msg\_list => FND\_API.G\_TRUE,

p\_party\_site\_use\_rec => p\_party\_site\_use\_rec,

x\_party\_site\_use\_id => x\_party\_site\_use\_id,

x\_return\_status => x\_return\_status,

x\_msg\_count => x\_msg\_count,

x\_msg\_data => x\_msg\_data

);

IF x\_return\_status = fnd\_api.g\_ret\_sts\_success THEN

COMMIT;

***DBMS\_OUTPUT.PUT\_LINE***('Creation of Party Site use is Successful ');

***DBMS\_OUTPUT.PUT\_LINE***('Output information ....');

***DBMS\_OUTPUT.PUT\_LINE***('Party Site Id = '||p\_party\_site\_use\_rec.party\_site\_id);

***DBMS\_OUTPUT.PUT\_LINE***('Site Use Id = '||x\_party\_site\_use\_id);

ELSE

***DBMS\_OUTPUT.put\_line*** ('Creation of Party Site use got failed:'||x\_msg\_data);

ROLLBACK;

FOR i IN 1 .. x\_msg\_count

LOOP

x\_msg\_data := fnd\_msg\_pub.get( p\_msg\_index => i, p\_encoded => 'F');

***dbms\_output.put\_line***( i|| ') '|| x\_msg\_data);

END LOOP;

END IF;

***DBMS\_OUTPUT.PUT\_LINE***('Completion of API');

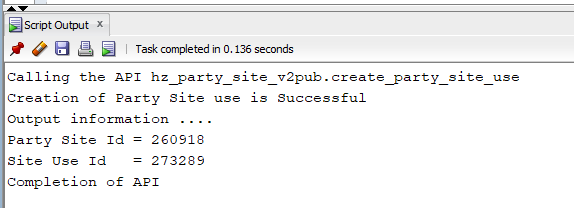
end;

In the above program we use HZ\_PARTY\_SITE\_V2PUB.CREATE\_PARTY\_SITE\_USE to create a use of the created party site.

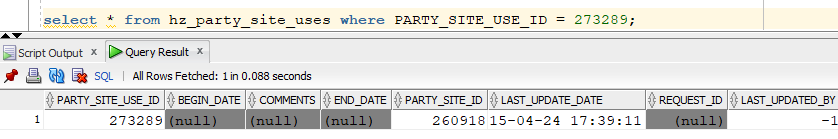
We cannot create the party site use without the party site.

So, in this program we give Site use type, Party site id and created by module.

Output:



* Check whether the party site use is created or not by using **HZ\_PARTY\_SITE\_USES** Table.



**CREATE ACCOUNT SITE:**

SET SERVEROUTPUT ON;

DECLARE

p\_cust\_acct\_site\_rec hz\_cust\_account\_site\_v2pub.cust\_acct\_site\_rec\_type;

x\_return\_status VARCHAR2(2000);

x\_msg\_count NUMBER;

x\_msg\_data VARCHAR2(2000);

x\_cust\_acct\_site\_id NUMBER;

BEGIN

*-- Setting the Context --*

mo\_global.init('AR');

fnd\_global.apps\_initialize ( user\_id => 1318

,resp\_id => 21623

,resp\_appl\_id => 660);

mo\_global.set\_policy\_context('S',204);

fnd\_global.set\_nls\_context('AMERICAN');

*-- Initializing the Mandatory API parameters*

p\_cust\_acct\_site\_rec.cust\_account\_id := 124734;

p\_cust\_acct\_site\_rec.party\_site\_id := 236632;

p\_cust\_acct\_site\_rec.created\_by\_module := 'BO\_API';

***DBMS\_OUTPUT.PUT\_LINE***('Calling the API hz\_cust\_account\_site\_v2pub.create\_cust\_acct\_site');

HZ\_CUST\_ACCOUNT\_SITE\_V2PUB.CREATE\_CUST\_ACCT\_SITE

(

p\_init\_msg\_list => FND\_API.G\_TRUE,

p\_cust\_acct\_site\_rec => p\_cust\_acct\_site\_rec,

x\_cust\_acct\_site\_id => x\_cust\_acct\_site\_id,

x\_return\_status => x\_return\_status,

x\_msg\_count => x\_msg\_count,

x\_msg\_data => x\_msg\_data

);

IF x\_return\_status = fnd\_api.g\_ret\_sts\_success THEN

COMMIT;

***DBMS\_OUTPUT.PUT\_LINE***('Creation of Customer Account Site is Successful ');

***DBMS\_OUTPUT.PUT\_LINE***('Output information ....');

***DBMS\_OUTPUT.PUT\_LINE***('Customer Account Site Id is = '||x\_cust\_acct\_site\_id);

ELSE

***DBMS\_OUTPUT.put\_line*** ('Creation of Customer Account Site got failed:'||x\_msg\_data);

ROLLBACK;

FOR i IN 1 .. x\_msg\_count

LOOP

x\_msg\_data := fnd\_msg\_pub.get( p\_msg\_index => i, p\_encoded => 'F');

***dbms\_output.put\_line***( i|| ') '|| x\_msg\_data);

END LOOP;

END IF;

***DBMS\_OUTPUT.PUT\_LINE***('Completion of API');

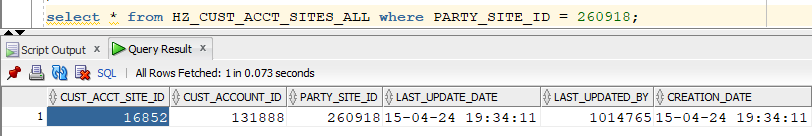
END;

/

In this program we need to use HZ\_CUST\_ACCOUNT\_SITE\_V2PUB.CREATE\_CUST\_ACCT\_SITE to create an Account site for created account. In this program first we need to initialize the user using

FND\_GLOBAL.APPS\_INITIALIZE. And we can pass cust\_account\_id and party\_site\_id to create an Account Site.

* Check whether the customer account site is created or not by using **HZ\_CUST\_ACCT\_SITES\_ALL** Table.



**CREATE CUSTOMER ACCOUNT SITE USES:**

SET SERVEROUTPUT ON;

DECLARE

p\_cust\_site\_use\_rec HZ\_CUST\_ACCOUNT\_SITE\_V2PUB.CUST\_SITE\_USE\_REC\_TYPE;

p\_customer\_profile\_rec HZ\_CUSTOMER\_PROFILE\_V2PUB.CUSTOMER\_PROFILE\_REC\_TYPE;

x\_site\_use\_id NUMBER;

x\_return\_status VARCHAR2(2000);

x\_msg\_count NUMBER;

x\_msg\_data VARCHAR2(2000);

BEGIN

*-- Setting the Context --*

mo\_global.init('AR');

fnd\_global.apps\_initialize ( user\_id => 1318

,resp\_id => 21623

,resp\_appl\_id => 660);

mo\_global.set\_policy\_context('S',204);

fnd\_global.set\_nls\_context('AMERICAN');

*-- Initializing the Mandatory API parameters*

p\_cust\_site\_use\_rec.cust\_acct\_site\_id := 11415;

p\_cust\_site\_use\_rec.site\_use\_code := 'SHIP\_TO';

*--p\_cust\_site\_use\_rec.location := 'NEWYORK';*

p\_cust\_site\_use\_rec.created\_by\_module := 'BO\_API'; *--TCA\_V1\_API*

***DBMS\_OUTPUT.PUT\_LINE***('Calling the API hz\_cust\_account\_site\_v2pub.create\_cust\_site\_use');

HZ\_CUST\_ACCOUNT\_SITE\_V2PUB.CREATE\_CUST\_SITE\_USE(

p\_init\_msg\_list => FND\_API.G\_TRUE,

p\_cust\_site\_use\_rec => p\_cust\_site\_use\_rec,

p\_customer\_profile\_rec => p\_customer\_profile\_rec,

p\_create\_profile => FND\_API.G\_TRUE,

p\_create\_profile\_amt => FND\_API.G\_TRUE,

x\_site\_use\_id => x\_site\_use\_id,

x\_return\_status => x\_return\_status,

x\_msg\_count => x\_msg\_count,

x\_msg\_data => x\_msg\_data);

IF x\_return\_status = fnd\_api.g\_ret\_sts\_success THEN

COMMIT;

***DBMS\_OUTPUT.PUT\_LINE***('Creation of Customer Accnt Site use is Successful');

***DBMS\_OUTPUT.PUT\_LINE***('Output information ....');

***DBMS\_OUTPUT.PUT\_LINE***('Site Use Id = '||x\_site\_use\_id);

***DBMS\_OUTPUT.PUT\_LINE***('Site Use = '||p\_cust\_site\_use\_rec.site\_use\_code);

ELSE

***DBMS\_OUTPUT.put\_line*** ('Creation of Customer Accnt Site use got failed:'||x\_msg\_data);

ROLLBACK;

FOR i IN 1 .. x\_msg\_count

LOOP

x\_msg\_data := fnd\_msg\_pub.get( p\_msg\_index => i, p\_encoded => 'F');

***dbms\_output.put\_line***( i|| ') '|| x\_msg\_data);

END LOOP;

END IF;

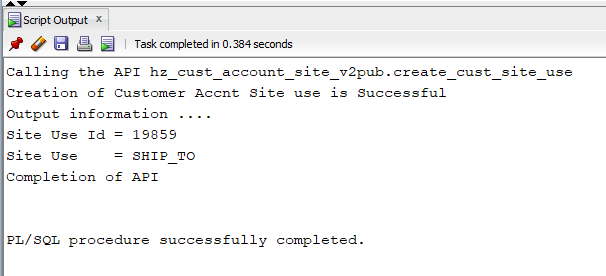
***DBMS\_OUTPUT.PUT\_LINE***('Completion of API');

END;

In this program we use HZ\_CUST\_ACCOUNT\_SITE\_V2PUB.CREATE\_CUST\_SITE\_USE to create an Account Site Use. In this program first we need to initialize the user using

FND\_GLOBAL.APPS\_INITIALIZE. We need to pass Cust\_acct\_site\_id (Account site id) and Site use code and created by module, to create a Customer Account Site use.

Output:



* Check whether the customer site use is created or not by using **HZ\_CUST\_SITE\_USES** Table.

