Technical Manual v-2 for Integration of the Department Services with Integrated Service Portal (ISP)

(For the use of Departmental IT Team)

Prepared By:

ISP team, National Informatics Centre, UPSU, Lucknow

Table of Contents

1- **Introduction-** ISP is a single window platform for citizen centric services. Citizen can avail any department services at this portal. He also has facility to check status of each application at the portal.

2- **Process Flow –**process flow is as follows

a- Citizen registers himself at the ISP Portal after aadhar verification.
b- Citizen login into ISP portal and fills remaining basic information.
c- To apply for service, citizen select department and service and click on apply button.
d- Citizen is redirected to concerned department portal, where citizen does not need to login.
e- Many data fields get auto filled with citizen personal details received from ISP portal(on consuming web service). Now he fills remaining fields and submit application form.

Note- Department need to use applicant's common details received from web service (autofill personal details fields of user in web form) and make these fields non-editable

f- When application is submitted successfully at department portal, department sends status to ISP portal by consuming web service. Department also sends submitted application form download link and other details as required to ISP portal.
g- When department process application at his end, he sends status at each stage. When service is delivered to citizen, certificate download link and other details are also sent by department**.**

3- **Registration of Service by Department :** Department needs to register service at ISP Portal by providing all relevant details such as service name, service web url(Landing page), nodal officer etc. A unique 5 digit service_code will be generated on successful registration of each service which need to be kept by department for service mapping at isp portal.

4- **Data Security :**For security aspect, all transactions between department and ISPapiwill be encrypted with shared secret key.Key will be made available by ISP to department for the same.Key must be stored in secure manner by the department.**Program for Encryption /Decryption providedinannexure-B**

5- **API Access-** To access API, each department will be provided with username(dept_id) and password which will be needed to generate JWT(JSON Web Token).
URL to generate token –**http://164.100.181.91/ispws/authenticate**

method –post
payload - {

```
“username” : dept_id,
“password”: token_key
}
```

On success, following response with http status code 200will be generated

```
{
    “error”: "",
    "statusMessage" :  "SUCCESSFUL",
    "timestamp": "09-10-2022 12:36:02",
    “token” : token
}
```

On Unsuccessful, following response with http status code 400 will be generated

```
{
    “error”: "To generate token valid user Name and password is required",
    "statusMessage" :  "FAILED",
    "timestamp": "09-10-2022 12:36:02",
    “data”: null
}
```

Department will use this token (If generated successfully) in header of http request in all subsequent API requestin following way..

Header	**value**
Authorization	Bearer token

This token will expire in 60 minutes

6- **Applying for Service :**Each user(Applicant) will create an account on ISP Portal. To avail any service provided by department, user will login to ISP Portal, select required service and click on apply button. With this click, user will be redirected to department landing page with following parameters.

Sr. No.	*Field Name*	*Format*	*Example*
1.	sessionkey	String	Alphanumeric String

7- **ISP WEB API:** ISP api will have following services and departments have to consume them in the web project.

A) Get RequestValidationStatus and get Request ID, Applicant_id, Service_code in response

URL- http://164.100.181.91/ispws/isp/v1/getRequestValidatedAndRequestID

Method: Post

Sr. No.	*Input Parameter Name*	*Format*	*Example*
1	sessionkey	String	Alphanumeric String

When user is redirected to department Landing page, Department will get **sessionkey** in request patameter. Now Department will consume service (**http://164.100.181.91/ispws/isp/v1/getRequestValidatedAndRequestID**)
In following steps-

Step -1 Create a JSON Object with following key value pair

Key	**value**
sessionkey	Sessionkey(as Received in request parameter)

Encrypt thisJSON Object with provided shared secret key.

Step -2 Nowcreate a new JSON Object with following key value pair

Key	**Value**
dept_id	Department id as provided
e_data	Encrypted JSON Object (Created in step1)

Now send this JSON Object while consuming thisweb service.

Note- Dept_id will be provided to each Department by ISP Portal that is to be used in consuming each web service;

Response - Response of above service will be received in JSON Object in following format.

```
{
“ error” : Error message (if operation is unsuccessful  otherwise blank)  ;
“statusMessage” : SUCCESSFUL ,if operation is successful or
                   FAILED,   if  unsuccessful ;
 “data” :Data encrypted with department secret key if successful
          Null  if  failed
 “timestamp” : timestamp
}
```

“data” containsfollowing details in encrypted form. To access data, dept needs to decrypt it.

```
  {
        “request_id” : “xxxxxxxxxxxxxx”,
        “applicant_id” : “xxxxxxxxxxxxx”,
        “service_code” : “xxxxxx”,
}
```

Note- Department needs to save request_id , applicant_id and service_code at their end for future reference.

B) GetCommonDetails Of Applicant Method–

URL -http://164.100.181.91/ispws/isp/v1/getApplicantCommonDetails

Method: Post

Sr. No.	***Input Parameter Name***	***Format***	***Example***
1	request_id	String	14digit string

Description of Input Parameters:

a) **Request_id**– 14 digit unique number for uniquely identifying each request made by citizen to avail service. It will be received in response of first web service

To consume this service follow below steps-

Step-1 Create a JSON Object with following key value pair.

Key	**value**
request_id	request_id(Received in responseobject)

Encrypt thisJSON Object with provided shared secret key.

Step-2 Now create a new JSON Object with following key value pair

Key	**Value**
dept_id	Department id as provided
e_data	Encrypted JSON Object(Created in step1)

Now send this JSON Object while consuming this web service.

Note- Dept_id will be provided to each Department by ISP Portal that is to be used in consuming each web service;

Response - Response of above service will be received in JSON Object in following format.

{

" error" : Error message (if operation is unsuccessful otherwise blank) ;

“statusMessage” : SUCCESSFUL,if operation is successful or
FAILED,if unsuccessful ;
“data” : Data encrypted with department secret key **if successful**
Null **if failed**
“timestamp” : timestamp
}

Data containsfollowing details in encrypted form. To access data dept need to decrypt it.

```
{
      “applicant_id”: “test000000”,

            //Service Details

                  “service_name_eng” : “Registration”,
                  “service_name_hindi” : “xxxxxxx”,
                  “service_code” : “sc109”,

            //Family ID, Member ID and Aadhar Reference no

                  “family_id” :”118812569782”,
                  “member_id” : “11881256978201”,
                  “aadhar_reference_no” : “xxxxxxxxxxxxxxxxxxxxx”

            //Common details

                  “first_name_eng”:”TEST”,
                  “middle_name_eng": “TEST”,
                  "last_name_eng"   :  “TEST”,

                  “first_name_hindi” : “TEST”,
                  “middle_name_hindi ": “TEST”,
                  “last_name_hindi” : ”TEST”,

                  "gender"  :  “M/F/T”, (code M-Male, F-Female, T- Transgender)

                  “father_or_husband_or_guardian” : “father/husband/guardian”

                  “father_or_husband_or_guardian_name_eng” : “TEST”,
                  “father_or_husband_or_guardian_name_hindi”: “TEST”,

                  “mother_name_eng” : “TEST”,
                  “mother_name_hindi” : “TEST”,

                  “category” :  1/2/3/4  (Code 1-GENERAL, 2-OBC,  3-SC,   4-ST)

                  “dob” : “yyyy-mm-dd”,

                  “pan_no” : “CBRPH7823H”,

                  “mobile” : “9700087655”,

                  “email” : “myemail@gmail.com”,
             “marital_status” : 1/2/3/4/5

             (Code 1-Single, 2-Married, 3-Divorced, 4- Widow, 5-Widower)

             “income” : income in digit

             “occupation” : occupation code

//Occupation will be received in code only,
//Occupation Code table may be downloaded from department login on ISP
Portal.



      //Residential Address Fields

             “residential_area_type” :1 or 2    (1-rural 2-urban)
             “residential_house_no” : “F100”,
             “residential_mohalla” : “Vikasnagar”,
             “residential_pin” : “125896”,



      //Below 11 fields of residential address will be received in code only,
      //Code table may be downloaded from department login on ISP Portal.


             “residential_district” : district code,
             “residential_tehsil” : tehsil code,
             “residential_vikaskhand” :vikaskhand code ,
             “residential_grampanchayat” :grampanchayat code ,
             “residential_rajasvagram” :rajasva gram code,
             “residential_ward” : ward code,
             “residential_nagarpalika” :nagarpalika code ,
             “residential_nagarnigam” :nagarnigam code ,
             “residential_nagarpanchayat” :nagarpanchayat code ,
             “residential_police_station” : police station code,
             “residential_state” : state code



      //Permanent Address Fields

             “permanent_area_type” :1 or 2    (1-rural 2-urban)
             “permanent_house_no” : “B/105”,
             “permanent_mohalla” : “vikasnagar”,
             “permanent_pin” : “111111”,




      //Below 11 fields of permanent address will be received in code only,
      //Code table may be downloaded from department login on ISP Portal.
```

```
               “permanent_district” : district code,
               “permanent_tehsil” : tehsil code,
               “permanent_vikaskhand” :vikaskhand code ,
               “permanent_grampanchayat” :grampanchayat code ,
               “permanent_rajasvagram” :rajasva gram code,
               “permanent_ward” : ward code,
               “permanent_nagarpalika” :nagarpalika code ,
               “permanent_nagarnigam” :nagarnigam code ,
               “permanent_nagarpanchayat” :nagarpanchayat code ,
               “permanent_police_station” : police station code,
               “permanent_state” : state code


  //Dummy Fields(20 dummy fields from d1 to d20 will be received)
               “d1” : null,
               “d2” : null,
               “d3” : null,
               “d4” : null,
               “d5” : null,
               “d6” : null,
               “d7” : null,
               “d8” : null,
               “d9” : null,
               “d10” : null,
               “d11” : null,
               “d12” : null,
               “d13” : null,
               “d14” : null,
               “d15” : null,
               “d16” : null,
               “d17” : null,
               “d18” : null,
               “d19” : null,
               “d20”: null

}
```

C)-Return Application successful submission acknowledgement- This service must be consumed only once when citizen successfully submit his application to avail service from department.

(Only for status code - S104)

URL-http://164.100.181.91/ispws/isp/v1/returnApplicationAcknowledgement

Method: post

Sr. No.	***Field Name***	***Format***	***Example***	***Required / Optional***
1.	request_id	String	14 digit string	**Required**
2.	applicant_id	String	alphanumeric string of length-10	**Required**
3.	service_code	String	alphanumeric string of length- 5	**Required**
4.	application_id	String	To be provided by Department max length 20	**Required**
5.	application_url	String	Application uri to download and take print of submitted application(if not available leave blank)	**Required**
6.	status_code	String	4digit code As provided **(Status code may be downloaded from department login on ISP portal)**	**Required**
7.	remarks	String	Remarks max length 2000 (If not available leave blank)	**Required**
8.	pendency_level	String	Pendency level code **(may be downloaded from department login on ISP portal)**	**Required**
9.	pending_with_officer	String	Officer-name and designation Send “**NA**” if not available.	**Required**
10.	action_taken_time	String	Timestamp in format- yyyy-mm-ddhh:mm:ss	**Required**
11.	selected_district_for_processing_application	String	District_code **(District code table may be downloaded from department login on ISP portal)**	**Required**
12.	designated_code	String	Designation-level mapping code for “**PadabhihitAdhikari**”. Code can be get from isp department login in code table menu by clicking on **Department Level Post Mapping Code**	**Required**
13.	designated_location_code	String	Location code for “**PadabhihitAdhikari**” Code can be get from isp department login in code table menu	**Required**
14.	designated_target_da	String	Target date of delivery of service in yyyy-mm-dd format	**Required**

	te			
15.	first_appellate_code	String	Designation-level mapping code for "**First appellate officer**". code can be get from isp department login in code table menu by clicking on **Department Level Post Mapping Code**	**Required**
16.	first_appellate_location_code	String	Location code for "**First appellate officer**".code can be get from isp department login in code table menu	**Required**
17.	first_appellate_days	String	Maximum no of days to resolve first appeal by **First appellate officer** Example – "25"	**Required**
18.	second_appellate_code	String	Designation-level mapping code for "**Secondappellate officer**". code can be get from isp department login in code table menu by clicking on **Department Level Post Mapping Code**	**Required**
19.	second_appellate_location_code	String	Location code for "**Secondappellate officer**". Code can be get from isp department login in code table menu	**Required**
20.	second_appellate_days	String	Maximum no of days to resolve second appeal by **Secondappellate officer** Example – "25"	**Required**
21.	d1	String	Contact no of the officer application is pending with . Use "**NA**" if not available.	
22.	d2	String	Not Required(Blank)	…..
23.	d3	String	Not Required(Blank)	
24.	d4	String	Not Required(Blank)	
25.	…….	……	…..	
26.	……..	……	…..	
27.	…. ..	……	……	
28.	…….	……	……..	
29	d16	String	Not Required(Blank)	
30	d17	String	Not Required(Blank)	
31.	d18	String	Not Required(Blank)	

32.	d19	String	Not Required(Blank)	
33.	d20	String	Not Required(Blank)	

a) **request_id**– 14 digit unique number for uniquely identifying each request made by citizen to avail service.

b) **applicant_id**- 10 literals long alphanumeric unique string to uniquely identify each citizen registered as user to ISP portal.

c) **service_code**- 5 digit unique number for uniquely identify each services of each department mapped on ISP portal.(Received in response of "/isp/v1/getApplicantCommonDetails" l)

d) **application_id**- Application id provided bydepartment.

e) **application_url** – Application URI

f) **status_code**– Status code as provided(may be downloaded from department login on ISP portal)

g) **remarks** – remarks

h) **pendency_level**- Pendency level code(may be downloaded from department login on ISP portal)

i) **Pending_with_officer-** provide Officer name and Designation, Send "**NA**" if not available.

j) **action_taken_time** –Timestamp in string format yyyy-mm-ddhh:mm:ss when user successfully submitted application for service.

k) **selected_district_for_processing_application-** Enter District code (**District code table may be downloaded from department login on ISP portal**)

l) **designated_code:** Designation-level mapping code for "**First appellate officer**". code can be get from isp department login in code table menu by clicking on **Department Level Post Mapping Code**

m) **designated_location_code:**Location code for "**AbhihitAdhikari**".code can be get from isp department login in code table menu

n) **designated_target_date:**Target date of delivery of service in yyyy-mm-dd format

o) **first_appellate_code:**Designation-level mapping code for "**First appellate officer**". code can be get from ispdepartment login in code table menu by clicking on **Department Level Post Mapping Code**

p) **first_appellate_location_code:** Location code for "**First appellate officer**". code can be get from isp department login in code table menu

q) **first_appellate_days:**Maximum no of days to resolve first appeal by **First appellate officer.**

r) **second_appellate_code:**Designation-level mapping code for "**Second appellate officer**". code can be get from isp department login in code table menu by clicking on **Department Level Post Mapping Code**

s) **second_appellate_location_code**: Location code for "**Second appellate officer**". Code can be get from isp department login in code table menu.

t) **second_appellate_days**: Maximum no of days to resolve second appeal by **Second appellate officer.**

u) **d1:** Contact no of the officer application is pending with. Send "NA" if not available

v) **d2:** Not Required(Blank)

……

s) **d20:** Not Required(Blank)

Department will consume this service by following 2 steps-

Step 1 – Put Above Required/Optional Input parameter as key and their respective value as value in JSON Object and Encrypt this JSON Object with given shared secret key.

Example-

Key	**Value**
request_id	14 digit string as received above
applicant_id	Alphanumeric string as received above
……	…...
……	…...

Step-2 Now create a new JSON Object with following key value pair

Key	**Value**
dept_id	Department id as provided
e_data	Encrypted JSON Object (Created in step - 1)

Now send this JSON Object while consuming this web service.

Response - Response of above service will be received in JSON Object in following format.

{
" error" : Error message (if operation is unsuccessful otherwise blank) ;
"statusMessage" : SUCCESSFUL ,if operation is successful or
FAILED,if unsuccessful ;
"data" :null;
"timestamp" : timestamp
}

D- Return ServiceStatusMethod: This service must be consumed to push update(status) to ISP server whenever there is any change in status of application

(For all status code excluding S104 and S112)

URL - http://164.100.181.91/ispws/isp/v1/returnServiceStatus

Method: post

Sr. No.	***Input Parameter Name***	***Format***	***Example***	***Required / Optional***
1	request_id	String	14 digit alphanumeric string	**Required**
2	service_code	String	alphanumeric string of length 5	**Required**
3.	applicant_id	String	alphanumeric string of length 10	**Required**
4.	application_id	String	Provided by department max length 20	**Required**
5.	status_code	String	Satus code (Status code may be downloaded from department login on ISP portal)	**Required**
6.	remarks	String	Remarks describing statusmax length 2000 (If not available leave blank)	**Required**
7	pendency_level	String	Pendency level code (may be downloaded from department login on ISP portal)	**Required**
8	pending_with_officer	String	Officer-name and designation.Use "**NA**" if not available.	**Required**
8	action_taken_time	String	Time in format- **yyyy-mm-ddhh:mm:ss**	**Required**
9	d1	String	Contact no of the officer application is pending with . Use "**NA**" if not available.	
10	d2	String	Not Required(Blank)	
11.	d3	String	Not Required(Blank)	
12.	d4	String	Not Required(Blank)	
13.	d5	String	Not Required(Blank)	
14.	d6	String	Not Required(Blank)	

15.	d7 “	String	Not Required(Blank)	
16.	d8	String	Not Required(Blank)	
17.	d9	String	Not Required(Blank)	
18.	d10	String	Not Required(Blank)	
19	d11	String	Not Required(Blank)	
20	d12	String	Not Required(Blank)	
21.	d13	String	Not Required(Blank)	
22.	d14	String	Not Required(Blank)	
23.	d15	String	Not Required(Blank)	
24.	d16	String	Not Required(Blank)	
25.	d17	String	Not Required(Blank)	
26.	d18	String	Not Required(Blank)	
27.	d19	String	Not Required(Blank)	
28.	d20	String	Not Required(Blank)	

Description of Input Parameters:

a. **Request_id–** 14 digit unique number for uniquely identifying each request made by citizen to avail service.(As received in second web service call)

b. **service_code**- 5 digit unique number for uniquely identify each services of each department mapped on ISP portal.(Received in response of **“/isp/v1/getApplicantCommonDetails”** api call)

c. **Applicant_id**- 10 literals long alphanumeric unique string to uniquely identify each citizen registered as user to ISP portal.

d. **Application_id**- Application id will be provided bydepartment.

e. **Status_Code**- Status code of the application of current processing stage.(status code may be downloaded from department login on ISP portal)

f. **Remarks** – Send remarks if any.

g. **Pendency_level**- Pendency level code(may be downloaded from department login on ISP portal)

h. **Pending_with_offcer**- Officer name and designation, send "**NA**" if not available.

i. **Action_taken_time-** String containing timestamp in format (**yyyy-mm-ddhh:mm:ss**) when action was taken.

j. **d1:** Contact no of the officer application is pending with. Send "NA" if not available

k. **d2:**Not Required(Blank)

l. ….

m. ….

n. ….

o. **d20:** Not Required(Blank)

Department will consume this service by following 2 steps-

Step 1 – Put Above Required/Optional Input parameter as key and their respective value as value in JSON Object and Encrypt this JSON Object with given shared secret key.

Example-

Key	**Value**
request_id	14 digit string as received above
applicant_id	Alphanumeric string as received above
……	…...
……	…...

Step-2 Now create a new JSON Object with following key value pair

Key	**Value**
dept_id	Department id as provided
e_data	Encrypted JSON Object (Created in step - 1)

Now send this JSON Object while consuming this web service.

Response - Response of above service will be received in JSON Object in following format.

```
{
" error" : Error message (if operation is unsuccessful  otherwise blank)  ;
"statusMessage" : SUCCESSFUL ,if operation is successful or
                  FAILED , if  unsuccessful ;
 "data" :null;
 "timestamp" : timestamp
}
```

E)-Return service Delivery Status-This service only be consumed when Service is finally delivered **(For Status code S112)**

URL- http://164.100.181.91/ispws/isp/v1/returnDeliveredServiceStatus

Method: post

Sr. No.	***Field Name***	***Format***	***Example***	***Required / Optional***
l.	request_id	String	14 digit string	**Required**
2.	applicant_id	String	alphanumeric string of length-10	**Required**
3.	service_code	String	alphanumeric string of length- 5	**Required**
4.	application_id	String	To be provided by Department max length 20	**Required**
5.	certificate_url	String	Certificateurlto download and take print of Certificate(if not available leave blank)	**Required**
6	certificate_no	String	Certificate_No (if not available leave blank)	**Required**
7.	status_code	String	4digit code As provided **(Status code may be downloaded from department login on ISP portal)**	**Required**
8.	action_taken_time	String	Timestamp in format- yyyy-mm-ddhh:mm:ss	**Required**
9	dbt_amount	String	If Service is DBT then it is must to provide DBT amount(in digit)	**Optional**
10.	remarks	String	Remarks max length 2000(If not available leave blank)	**Required**
11.	certificate_expiry_date	String	If certificate or NOC is issued under this service then pass certificate/NOC expiry **date(yyyy-mm-dd)** if certificate is life time valid then pass **999999**	Required
12.	d1	String	Not Required(Blank)	
13.	d2	String	Not Required(Blank)	
14.	d3	String	Not Required(Blank)	
15.	d4	String	Not Required(Blank)	
16.	d5	String	Not Required(Blank)	

17.	d6	String	Not Required(Blank)	
18	d7	String	Not Required(Blank)	
19.	d8	String	Not Required(Blank)	
20	d9	String	Not Required(Blank)	
21.	d10	String	Not Required(Blank)	
22.	d11	String	Not Required(Blank)	
23.	d12	String	Not Required(Blank)	
24.	d13	String	Not Required(Blank)	
25.	d14	String	Not Required(Blank)	
26.	d15	String	Not Required(Blank)	
27.	d16	String	Not Required(Blank)	
28	d17	String	Not Required(Blank)	
29.	d18	String	Not Required(Blank)	
30.	d19	String	Not Required(Blank)	
31.	d20	String	Not Required(Blank)	

- **Request_id**– 14 digit unique number for uniquely identifying each request made by citizen to avail service.
- **Applicant_id**- 10 literals long alphanumeric unique string to uniquely identify each citizen registered as user to ISP portal.
- **service_code**- 5 digit unique number for uniquely identify each services of each department mapped on ISP portal.
- **Application_id**- Application id providedbydepartment.
- **certificate_url** – Certificate URL
- **certificate_no –** Certificate No generated by department
- **Status_code**– Status code as provided(may be downloaded from department login on ISP portal)
- **action_taken_time** –Timestamp in string format yyyy-mm-ddhh:mm:ss when user successfully submitted application for service.
- **dbt_amount-**If Service is DBT then provide DBT amount(in Rs.)

- **remarks** – remarks
- **certificate_expiry_date:**If certificate or NOC is issued under this service then pass certificate/NOC expiry **date(yyyy-mm-dd)** if certificate is life time valid then pass **999999**
- d1: Not Required(Blank)
- d2: Not Required(Blank)
- ….
- **d20:** Not Required(Blank)

Department will consume this service by following 2 steps-

Step 1 – Put Above Required/Optional Input parameter as key and their respective value as value in JSON Object and Encrypt this JSON object with given shared secret key.

Example-

Key	**Value**
request_id	14 digit string as received above
applicant_id	Alphanumeric string as received above
……	…...
……	…...

Step-2 Now create a new JSON Object with following key value pair

Key	**Value**
dept_id	Department id as provided
e_data	Encrypted JSON Object (Created in step - 1)

Now send this JSON Object while consuming this web service.

Response - Response of above service will be received in JSON Object in following format.

{

" error" : Error message (if operation is unsuccessful otherwise blank) ;

"statusMessage" : SUCCESSFUL ,if operation is successful or **FAILED,**if unsuccessful ;

"data" :null;

"timestamp" : timestamp

}

How to send data to ISP portal(In Post Method)-

Department will need to put all required parameters in a JSON Object and will encrypt this jsonobject with provided secret key.

This encrypted object(Encrypted_JSON_Object) along with dept_id(provided) will be put in new json object asfollowing key value pair-

key	**value**
dept_id	dept_id
e_data	Encrypted_JSON_Object

This JSON Object will be sent to ISP in web service;

Receive response from ISP portal-

Response will be received in JSON Object in following format.

{

“ error” : Error message (if operation is unsuccessful otherwise blank) ;

“statusMessage” : SUCCESSFUL ,if operation is successful or **FAILED,**if unsuccessful ;

“data” : Data encrypted with department secret key;

“timestamp” : timestamp

}

8- Scheduler for Failed Status Delivey-

While sending status code to ISP server using provided web service in real time there may by some issues like network unavailability, service down or some other technical issues or http bad request(http status code -400). In this case all the data of failed web service transaction should by stored at your end and **department needs to create a scheduler at it's end to reschedule delivery of failed status till it is finally delivered**(till receiving of status code 200 and statusMessage successful against called web service)

9- Redirection link to ISP portal

User can be redirected to ISP portal by following link-

```
http://164.100.181.91/getMessageFromDepartment?sessionkey=SESSION_KEY_RECEIVED&
return_success_or_failed=SUCCESS_OR_FAILED&MESSAGETEXT=MESSAGE_TO_BE_SENT_
TO_ISP
```

```
SESSION_KEY_RECEIVED – sessionkey received in web service call
SUCCESS_OR_FAILED – SUCCESS/FAILED
MESSAGE_TO_BE_SENT_TO_ISP – Message
```

ANNEXURE-A

1-Status Code Table-

Note- Updated staus code table is available on ISP portal in department login

Status code	**Status Description**
S101	Application Saved As Draft
S102	Fee Pending
S103	Fee Paid
S104	Application Submitted
S105	Forwarded
S106	Pending For Approval
S107	Pending for verification
S108	Pending for clarification
S109	Objection Raised
S110	Rejected
S111	Application Accepted
S112	Service Delivered
S113	Certificate Issued
S114	Form Resubmitted
S115	Approved
S116	Partially closed (Open for Feedback)
S117	**Fit for Approval**
S118	**Submit to officer**
S119	Prior Certificate Issue PriorSanction
S120	Secondary CertificateIssue

Order of some mandatory status code

1- **Status code S104 and S110/S112 are mandatory and will be received only once and S104 must be sent before S110/S112.**

2- **After S110/S112 is received, no other status code will be accepted.**

3- **If query is raised then S109 must be sent. And when user answer query and resubmit application S114 must be sent. No status code other than S114 will be accepted just after S109.**

4- **When fee is paid status code S103 should be sent(After S102 only S103 will be accepted)**

ANNEXURE-B

===================ENCRYPTION/DECRYPTION ==================

.NET (C#)

Namespace required

```
Using System;
Using System.Text;
Using System.Security.Cryptography;
Using System.IO;
Using System.Net.Mime.MediaTypeNames;
```

Calling Method-

```
public static string encryptString(string plainText, string pass)
   {

byte[] salt = newbyte[] { 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30 ,
                          0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31};

byte[] encrypted;
byte[] key = new Rfc2898DeriveBytes(pass, salt, 1000, HashAlgorithmName.SHA512).GetBytes(32);
using (RijndaelManaged rijAlg = newRijndaelManaged())
     {

rijAlg.Key = key;
rijAlg.IV = new byte[] {0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1};

rijAlg.Padding = PaddingMode.PKCS7;
rijAlg.Mode = CipherMode.CBC;
ICryptoTransform encryptor = rijAlg.CreateEncryptor(rijAlg.Key, rijAlg.IV);
using (MemoryStream msEncrypt = new MemoryStream())
using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
        {
```

```
swEncrypt.Write(plainText);
swEncrypt.Flush();
csEncrypt.FlushFinalBlock();
encrypted = msEncrypt.ToArray();
//  Console.WriteLine(encrypted);
            }
        }
return Convert.ToBase64String(encrypted);

    }

public static string decryptString(string _cipherText, string pass)
    {

            if (_cipherText == null || _cipherText.Length<= 0)
throw new ArgumentNullException("cipherText");

byte[] cipherText = Convert.FromBase64String(_cipherText);

byte[] IV = new byte[] { 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1 };

byte[] salt = new byte[] { 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30 ,
                                    0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31};
byte[] Key = new Rfc2898DeriveBytes(pass, salt, 1000, HashAlgorithmName.SHA512).GetBytes(32);
// Check arguments.
if (Key == null || Key.Length<= 0)
throw new ArgumentNullException("Key");
if (IV == null || IV.Length<= 0)
throw new ArgumentNullException("IV");

string plaintext = null;

using (RijndaelManaged rijAlg = new RijndaelManaged())
        {
rijAlg.Key = Key;
rijAlg.IV = IV;

rijAlg.Padding = PaddingMode.PKCS7;
rijAlg.Mode = CipherMode.CBC;

// Create a decryptor to perform the stream transform.
ICryptoTransform decryptor = rijAlg.CreateDecryptor(rijAlg.Key, rijAlg.IV);

// Create the streams used for decryption.
using (MemoryStream msDecrypt = new MemoryStream(cipherText))
            {
```

```
using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
                {
using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                    {
// Read the decrypted bytes from the decrypting stream
// and place them in a string.
plaintext = srDecrypt.ReadToEnd();
                    }
                }
            }
        }

return plaintext;

    }
```

//For JAVA Platform

--Used Libraries

```
Import java.nio.charset.StandardCharsets;

Import java.security.spec.KeySpec;

import java.util.Base64;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.SecretKeySpec;
```

//Calling Methods

```
public static String encryptString(String strToEncrypt, String key)
    {
try {

                String salt = "0000000011111111";
```

```
byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1 };

IvParameterSpec ivspec    = new IvParameterSpec(iv);

SecretKeyFactory factory   = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");

KeySpec spec = new PBEKeySpec(key.toCharArray(), salt.getBytes(),   1000, 256);

SecretKey tmp = factory.generateSecret(spec);

SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");

        Cipher cipher = Cipher.getInstance( "AES/CBC/PKCS5Padding");

cipher.init(Cipher.ENCRYPT_MODE, secretKey,  ivspec);

return Base64.getEncoder().encodeToString( cipher.doFinal(strToEncrypt.getBytes(

             StandardCharsets.UTF_8)));

      }

catch (Exception e) {

System.out.println("Error while encrypting: "   + e.toString());

      }

return null;



   }


public static String decryptString(String strToDecrypt, String key)
 {
                try {

                String salt = "0000000011111111";



        byte[] iv ={ 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1 };

        IvParameterSpec ivspec    = new IvParameterSpec(iv);

SecretKeyFactory factory     = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
```

```
        KeySpec spec = new PBEKeySpec(key.toCharArray(), salt.getBytes(), 1000, 256);

        SecretKey tmp = factory.generateSecret(spec);

        SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");

                Cipher cipher = Cipher.getInstance( "AES/CBC/PKCS5PADDING");

        cipher.init(Cipher.DECRYPT_MODE, secretKey,   ivspec);

        return new String(cipher.doFinal( Base64.getDecoder().decode(strToDecrypt)));

            }

        catch (Exception e) {

        System.out.println("Error while decrypting: "

                                + e.toString());

            }

        return null;

}
```

//For PHP Platform

```
<?php


$plaintext ;

$password ;

$salt = "0000000011111111";

$bytes = array('0','0','0','0','0','0','0','0','1', '1','1', '1','1', '1','1', '1');

$iv = implode(array_map("chr", $bytes));
```

```
$iterations = 1000;

$keyLength = 32;


$prepared_key = openssl_pbkdf2($password, $salt, $keyLength, $iterations, "sha512");


$ciphertext_b64 = base64_encode(openssl_encrypt($plaintext,"AES-256-
CBC",$prepared_key,OPENSSL_RAW_DATA, $iv));

echo $ciphertext_b64 . "<br/>";


$pt = openssl_decrypt(base64_decode($ciphertext_b64),"AES-256-CBC",
$prepared_key,OPENSSL_RAW_DATA, $iv);

echo $pt . "<br/>";


?>
```

End OfDocumment