Decryption of Credit Card Data and Bank Account Data; Risks and Controls

Overview:

Oracle provides its customers the ability to decrypt certain encrypted credit card and bank account data that is likely subject to PCI-DSS compliance and other compliance requirements.

The following is a list of concurrent programs that can decrypt encrypted data in any instance – production and non-production:

- Decrypt Credit Card Data
- Decrypt External Bank Account Data
- Decrypt Transaction Extension Data
- Decrypt Credit Card Transaction Data
- Payments Scheduled Decryption

(NOTE: These are the R12 concurrent program names. The 11i names are: in Appendix E)

These are also contained in a seeded Request Group:

- Decrypt Sensitive Data Request Set

Risks:

These concurrent programs and request set can decrypt credit card and external bank information (i.e. supplier bank accounts). The programs could be run in either a production or non-production environment. Once unencrypted, the data could be easily stolen through a variety of mechanisms – direct database query, through application access (form or HTML page), use of SQL forms, or running a concurrent program.

To understand the current risk in your production and non-production environments, ask the IT department these questions:

1. Do any users have access to these concurrent programs or the request set?
2. What controls are in place to prevent someone from adding these concurrent programs or the request set to another Request Group?
3. What controls are in place to prevent someone from registering the same executable (e.g. IBY_CREDITCARD_DECRYPTION) as another Concurrent Program
4. What controls are in place to prevent someone from entering one of these executables for an existing Concurrent Program to which they have access already?
5. What controls are in place to someone from setting up a new executable calling the same code (i.e. oracle.apps.iby.scheduler)?
6. What controls are in place to someone from changing an existing executable to call the same code – particularly an executable to which they already have access through a concurrent program assigned to them?

7. How are you mitigating the risks related to ad hoc SQL statements in forms that allow SQL injection? Do you know who has access to these forms? Are you monitoring the activity in these forms via log-based or trigger-based software? (See more in MOS Note: 403537.1 and 1334930.1)
8. Are there any unsecured database logins that could provide someone with privileges to any of the underlying tables related to these forms?

To understand how many users have the ability to execute on one of these first six schemes, you'd need to understand who has access to maintain users, change or add request groups, change or add concurrent programs and change or add executables. These forms are typically found in the System Administrator, Application Developer, Applications Administration, and System Administration responsibilities.

NOTE: Internal accounts, for example for payments in payables and payroll are NOT encrypted. Direct deposit accounts used for paying payroll are also NOT encrypted. We are not addressing these risks in this white paper.

Recommendations:

Following are recommendations related to these risks:

- Your provisioning process should consider these risks. Ideally, no one has access to these concurrent programs in any environment. How are you preventing access to these in your non-production environments where more users have access
- Use the query in Appendix A to identify which request groups and request sets, if any, contain these concurrent programs. Remove from ALL request groups
- User the query in Appendix B to can identify who has access to add or change data in the Concurrent Programs, Request Groups, and Executables forms.
- Each of these concurrent programs should be disabled. This would mean they couldn't be used even if they are assigned to a request group. See how to disable in Appendix C.
- Changes to these objects (Concurrent Programs, Request Groups, and Executables) should be monitored via a trigger or log-based solution that provides near-real time visibility to changes. Particular attention should be given to the re-enabling of these programs or removing the end-date of the request group(s) that contain these programs.
- Your cloning process should change or scramble the data when cloning to non-production environments – especially when the security in your non-production environment(s) is less stringent then in production.
- An IT quality or IT audit function should be tracing 100% of the changes made through these forms to the approved changes to test for unapproved changes.

Conclusions

In introducing the ability to decrypt credit card and bank account data Oracle has introduced a series of risks that organizations need to take seriously. If your organization has credit cards stored, the risks are significant and the impact on PCI-DSS compliance could be serious.

Decryption of Credit Card Data and Bank Account Data; Risks and Controls

Acknowledgement

Identification of this issue was by Integrigy Corporation via this article:

"Oracle E-Business Suite PCI DSS Compliance, Requirement 3.4 and Decryption Risk" at:

http://www.integrigy.com/oracle-security-blog/oracle-e-business-suite-pci-dss-compliance-requirement-34-and-decryption-risk

About ERP Risk Advisors

ERP Risk Advisors is a leading provider of Risk Advisory services for organizations using Oracle Applications. We provide consulting and training services related to compliance, security, risk management, and controls. We also assist organizations in implementing GRC-related software from industry-leading companies such as Oracle, Absolute Technologies, CaoSys, ConfigSnapshot, Greenlight Technologies, and MentiSoftware.

About Jeffrey T. Hare, CPA CISA CIA

Jeffrey Hare, CPA CIA CISA is the founder and CEO of ERP Risk Advisors. His extensive background includes public accounting (including Big 4 experience), industry, and Oracle Applications consulting experience. Jeffrey has been working in the Oracle Applications space since 1998 with implementation, upgrade, and support experience. Jeffrey is a Certified Public Accountant (CPA), a Certified Information Systems Auditor (CISA), and a Certified Internal Auditor (CIA). Jeffrey has worked in various countries including Austria, Australia, Brazil, Canada, Germany, Ireland, Mexico, Panama, Saudi Arabia, and United Kingdom. Jeffrey is a graduate of Arizona State University and lives in northern Colorado with his wife and three daughters. You can reach him at jhare@erpra.net or (970) 324-1450.

Jeffrey's first solo book project "Oracle E-Business Suite Controls: Application Security Best Practices" was released in 2009. Jeffrey has written various white papers and other articles, some of which have been published by organizations such as ISACA, the ACFE, and the OAUG. Request these white papers here. Jeffrey is a contributing author for the book "Best Practices in Financial Risk Management" published in 2009.

LinkedIn: linkedin.com/in/jeffreythare
Twitter: twitter.com/jeffreythare
Blog: jeffreythare.blogspot.com

Appendix A – Query to retrieve all Request Group data

Concurrent Requests Contained in Request Groups

Purpose: Identify which request groups have access to sensitive reports and programs

```
-- 27-JUN-06
-- C:\...\QBD Files\Request_Group_Info.sql
-- What request groups have access to what programs.

-- The first one is for "Programs" only.
SELECT a.request_group_name, e.application_name, a.description,
     c.user_concurrent_program_name, d.application_name,
     DECODE(b.request_unit_type,'A','Application',
          decode(b.request_unit_type,'P','Program',
             decode(b.request_unit_type,'S','Req Set', b.request_unit_type))) Type,
     c.concurrent_program_name, b.last_update_date, b.last_updated_by
FROM   fnd_request_groups a,
     FND_REQUEST_GROUP_UNITS b,
     fnd_concurrent_programs_vl c,
     fnd_application_tl d,
     fnd_application_tl e
WHERE  a.request_group_id = b.request_group_id
AND    b.request_unit_id = c.concurrent_program_id
AND    a.application_id = e.application_id
AND    b.unit_application_id = d.application_id
AND    b.request_unit_type = 'P'
-- Variables that can be utilized for tracking.
--AND    a.request_group_name = 'All Reports'
--AND    e.application_name = 'Oracle Payables'
--AND    c.user_concurrent_program_name = 'Mass Additions Create Report'
--AND    e.application_id = 20001  -- Our Payables = 20001
ORDER BY a.request_group_name,
       e.application_name,
       c.user_concurrent_program_name ;
```

-- This script can be used for finding "Set" types only.

```
SELECT a.request_group_name, e.application_name, a.description,
     c.user_request_set_name, d.application_name,
     DECODE(b.request_unit_type,'A','Application',
          decode(b.request_unit_type,'P','Program',
             decode(b.request_unit_type,'S','Req Set', b.request_unit_type))) TYPE,
     c.request_set_name, b.last_update_date, b.last_updated_by
FROM   fnd_request_groups a,
     FND_REQUEST_GROUP_UNITS b,
     fnd_request_sets_vl c,
     fnd_application_tl d,
     fnd_application_tl e
WHERE  a.request_group_id = b.request_group_id
AND    b.request_unit_id = c.request_set_id
AND    a.application_id = e.application_id
AND    b.unit_application_id = d.application_id
```

```
AND    b.request_unit_type = 'S'
-- Variables that can be utilized for tracking.
--AND    b.request_unit_id = 1244
--AND    a.request_group_name = 'All Reports'
--AND    e.application_name = 'Oracle Payables'
--AND    c.user_request_set_name = 'GMS: AP Holds - Set'
--AND    e.application_id = 20001  -- Our Payables = 20001
ORDER BY a.request_group_name, e.application_name, c.user_request_set_name ;
```

-- This script can be used for finding "Application" types only.

```
SELECT a.request_group_name, e.application_name, a.description,
     d.application_name UNIT_APP,
     DECODE(b.request_unit_type,'A','Application',
          decode(b.request_unit_type,'P','Program',
             decode(b.request_unit_type,'S','Req Set', b.request_unit_type))) TYPE,
     b.last_update_date, b.last_updated_by
FROM   fnd_request_groups a,
     FND_REQUEST_GROUP_UNITS b,
     fnd_application_tl d,
     fnd_application_tl e
WHERE  a.request_group_id = b.request_group_id
AND    a.application_id = e.application_id
AND    b.unit_application_id = d.application_id
AND    b.request_unit_type = 'A'
-- Variables that can be utilized for tracking.
--AND    a.request_group_name = 'All Reports'
--AND    d.application_name = 'Oracle Payables'
--AND    e.application_id = 20001  -- Our Payables = 20001
ORDER BY a.request_group_name, e.application_name ;
```

If you are not familiar with how to analyze the data from these reports, contact me at jhare@erpra.net.

Appendix B – Query to Identify who the ability to register Concurrent Programs or make changes to Request Groups

Purpose: This query identifies the users and responsibilities that can access high risk single functions. The limitation of this query is that it does not take into account menu or function exclusions that may be applied at the Responsibility level. NOTE: This query does not take into account whether there is a menu or function exclusion at the Responsibility level. If you aren't aware of what this means, contact us at info@erpra.net.

Concurrent Programs:

- FND_FNDCPMCP_SYS
- FND_FNDCPMCP_DEV
- FND_CP_SEARCH_CONC_PROG

Request Groups:

- FND_FNDRSGRP

Executables:

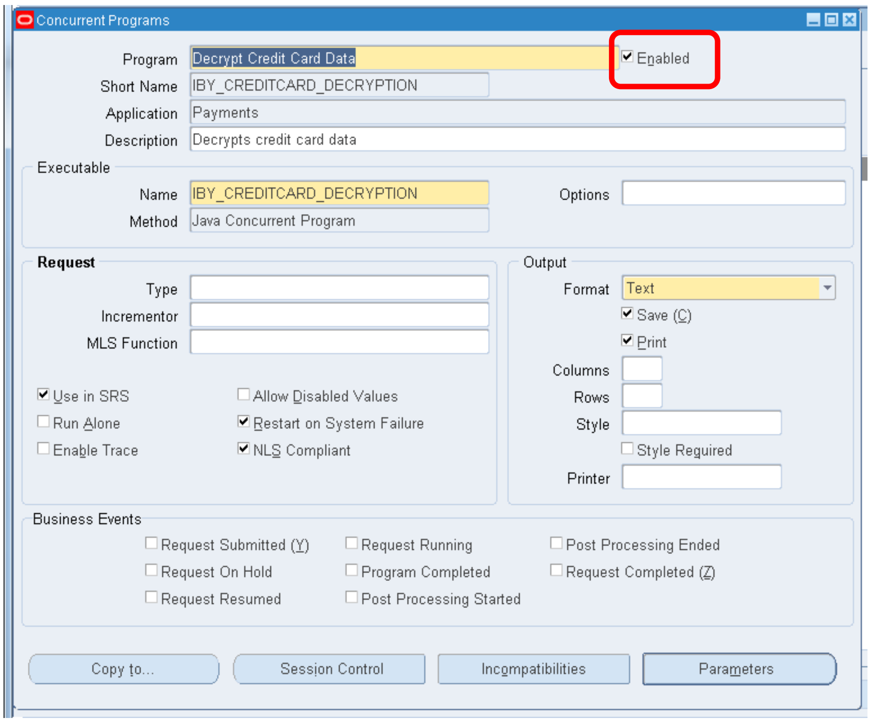
- FND_FNDCPMPE

```
select distinct fu.user_name user_name,fr.responsibility_name resp_name,fff.function_name,fff.user_function_name,
ff.form_name, ff.user_form_name
from applsys.fnd_user fu,
apps.fnd_user_resp_groups furg,
apps.fnd_responsibility_vl fr,
applsys.fnd_compiled_menu_functions fcmf,
apps.fnd_form_functions_vl fff,
apps.fnd_form_vl ff
where fff.form_id=ff.form_id
and furg.responsibility_id = fr.responsibility_id
and furg.responsibility_application_id = fr.application_id
and fr.menu_id = fcmf.menu_id
and fcmf.grant_flag = 'Y'
and fcmf.function_id = fff.function_id
and furg.user_id = fu.user_id
and sysdate between fu.start_date and nvl(fu.end_date, sysdate+1)
and sysdate between fr.start_date and nvl(fr.end_date, sysdate+1)
and fff.function_name in (
select fun.function_name
from apps.fnd_form_functions_vl fun, apps.fnd_form_vl form
where fff.function_name in (
‘FND_FNDCPMCP_SYS’
, ‘FND_FNDCPMCP_DEV’
, ‘FND_CP_SEARCH_CONC_PROG’
, ‘FND_FNDRSGRP’
, ‘FND_FNDCPMPE’)
and fun.form_id=form.form_id)
order by 1,2
```

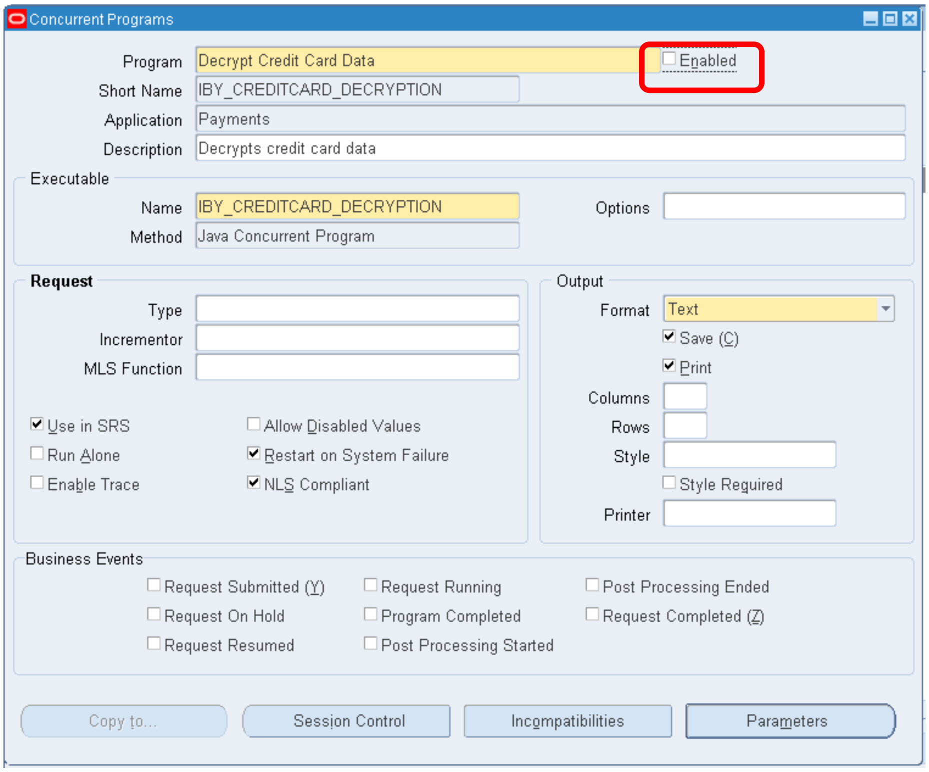
Appendix C – Disabling Concurrent Programs / End-Dating Request Sets

Following is how concurrent programs are disabled:

As installed:

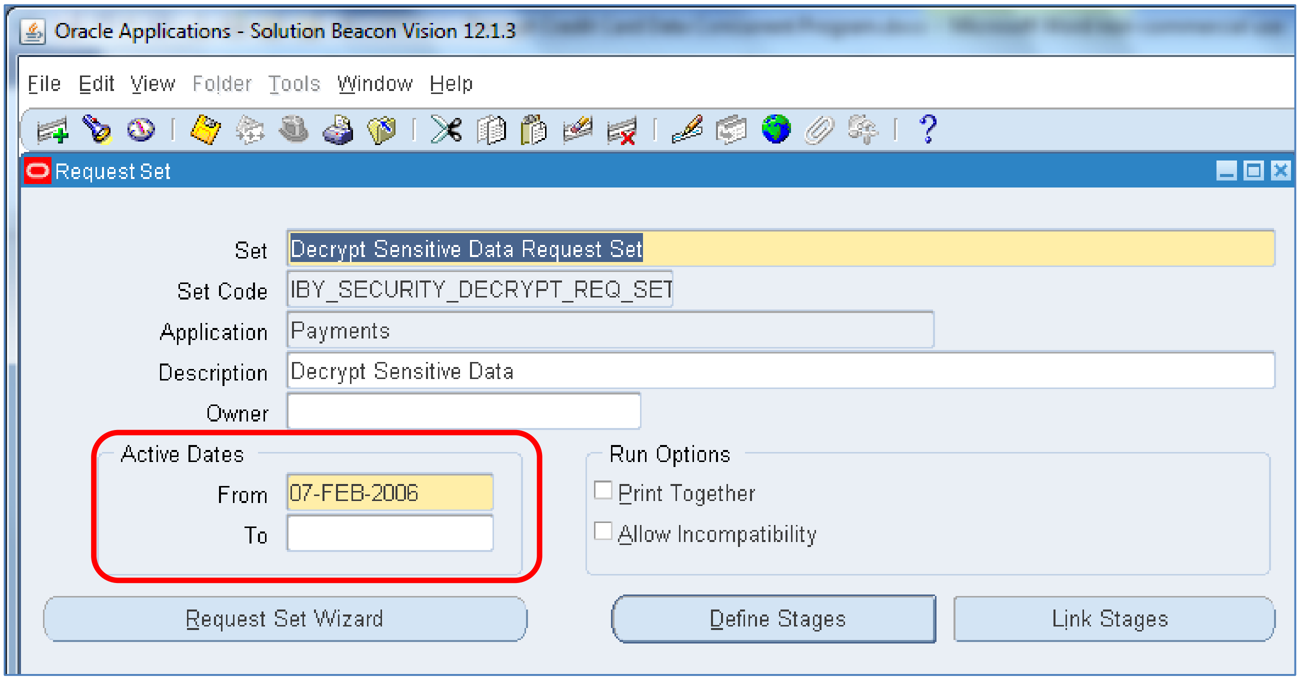


After Disabling:

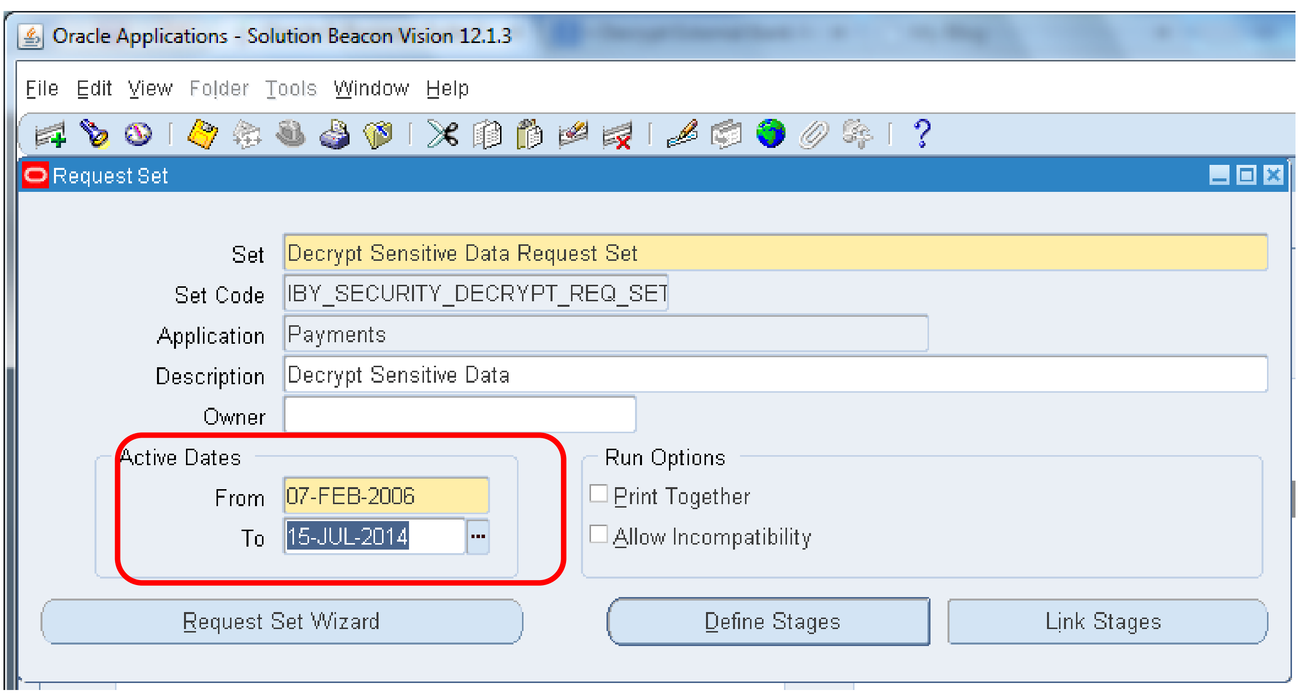


The Request set where these are contained should also be end-dated.

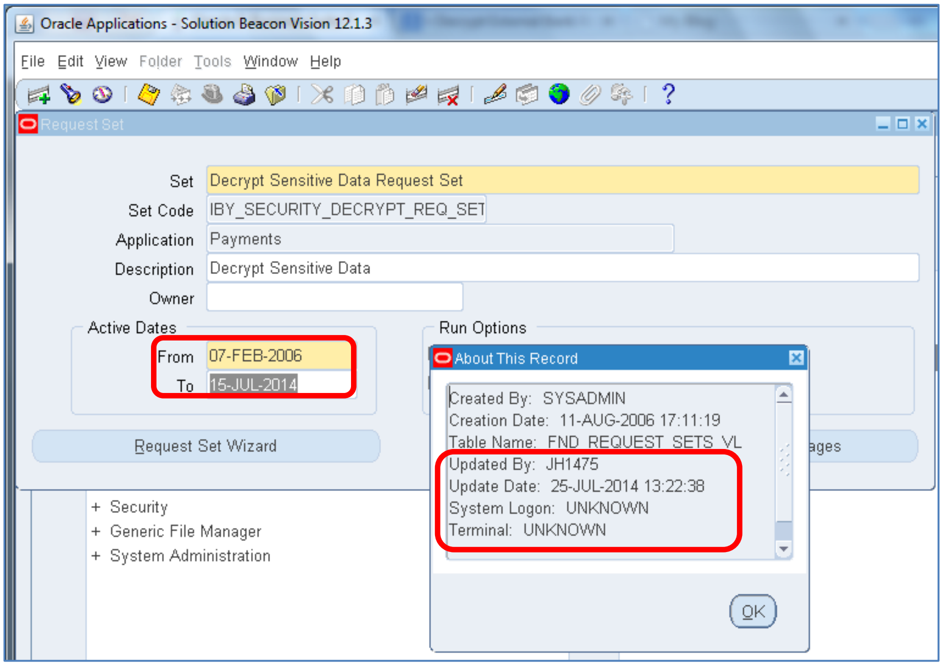
Request set as installed:



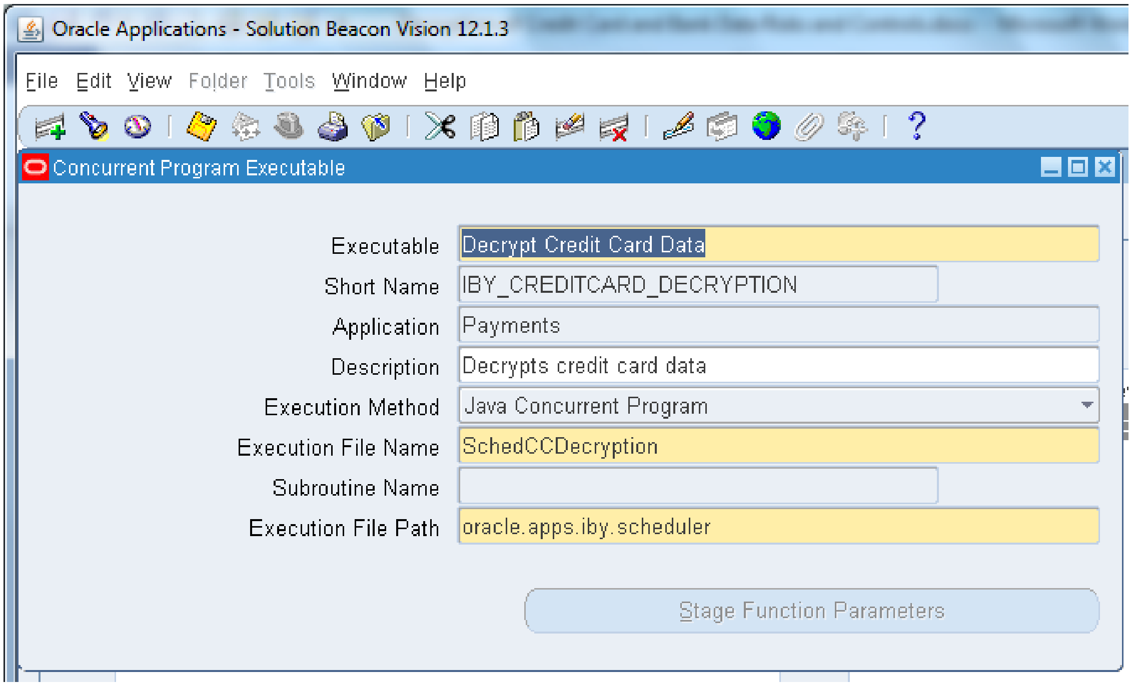
After 'end-dating':



Notice in this form, like most forms where an End Date is entered, you can enter a prior date as an end date. Here is the Row Who record information:



Appendix D – Concurrent Program Executables



Following are the names of the Executables in an R12 environment (note: there are six Concurrent Programs in an 11i environment and the names may be different)

Concurrent Program	**Executable**
Decrypt Credit Card Data	IBY_CREDITCARD_DECRYPTION
Decrypt External Bank Account Data	IBY_EXT_BANKACCT_DECRYPTION
Decrypt Transaction Extension Data	IBY_TRXN_EXTENSION_DECRYPTION
Decrypt Credit Card Transaction Data	IBY_TX_CREDITCARD_DECRYPTION
Payments Scheduled Decryption	IBY_SCHEDULED_ENCRYPTION

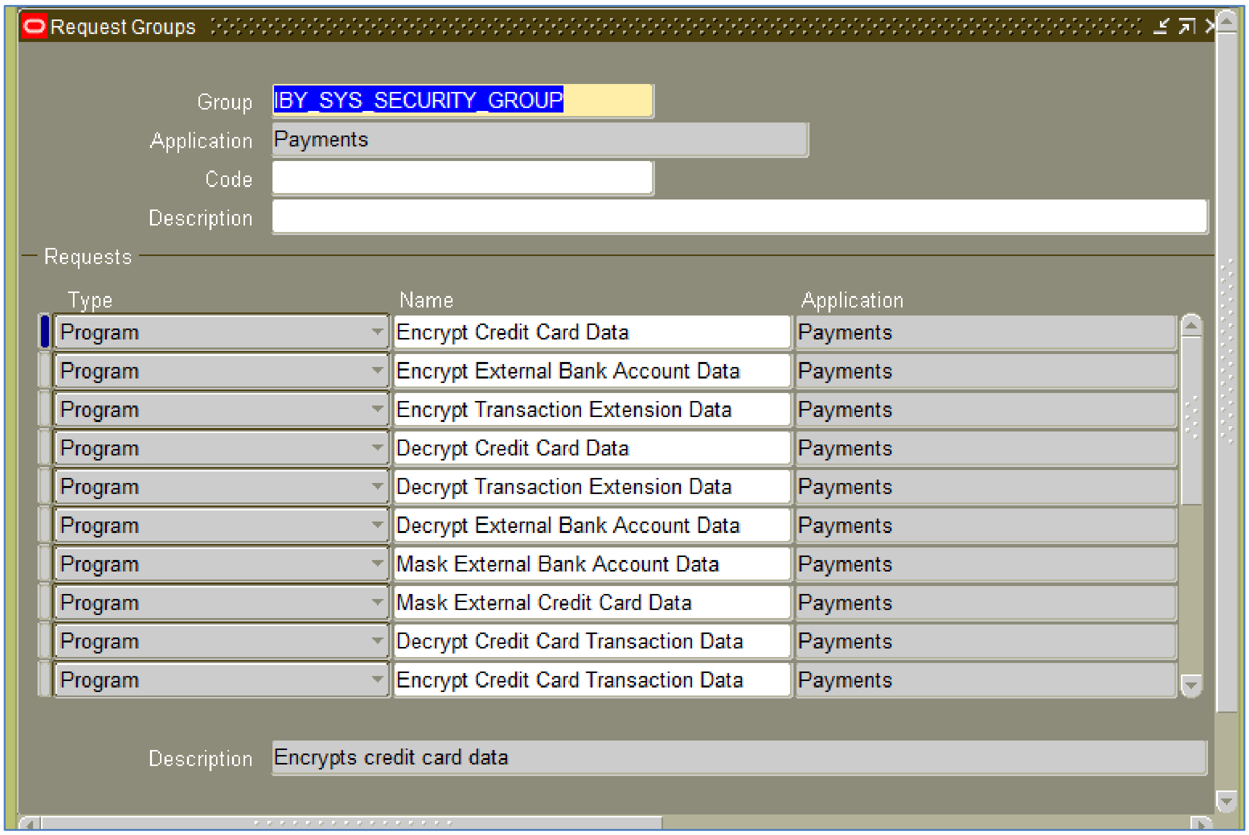
Appendix E: 11 versions of the Concurrent Programs

- Decrypt 11.5.9 iPayment Bankaccount Data
- Decrypt 11.5.9 iPayment Bankaccount Data Worker
- Decrypt 11.5.9 iPayment Creditcard Data
- Decrypt 11.5.9 iPayment Creditcard Data Worker
- Decrypt 11.5.9 iPayment Transaction Data
- Decrypt 11.5.9 iPayment Transaction Data Worker

Appendix F: Seeded Request Group and Responsibilities where these programs are contained

Caveat – above queries should be used to develop a comprehensive list. The information in this Appendix may not be complete.

Request Group: IBY_SYS_SECURITY_GROUP



Request Groups

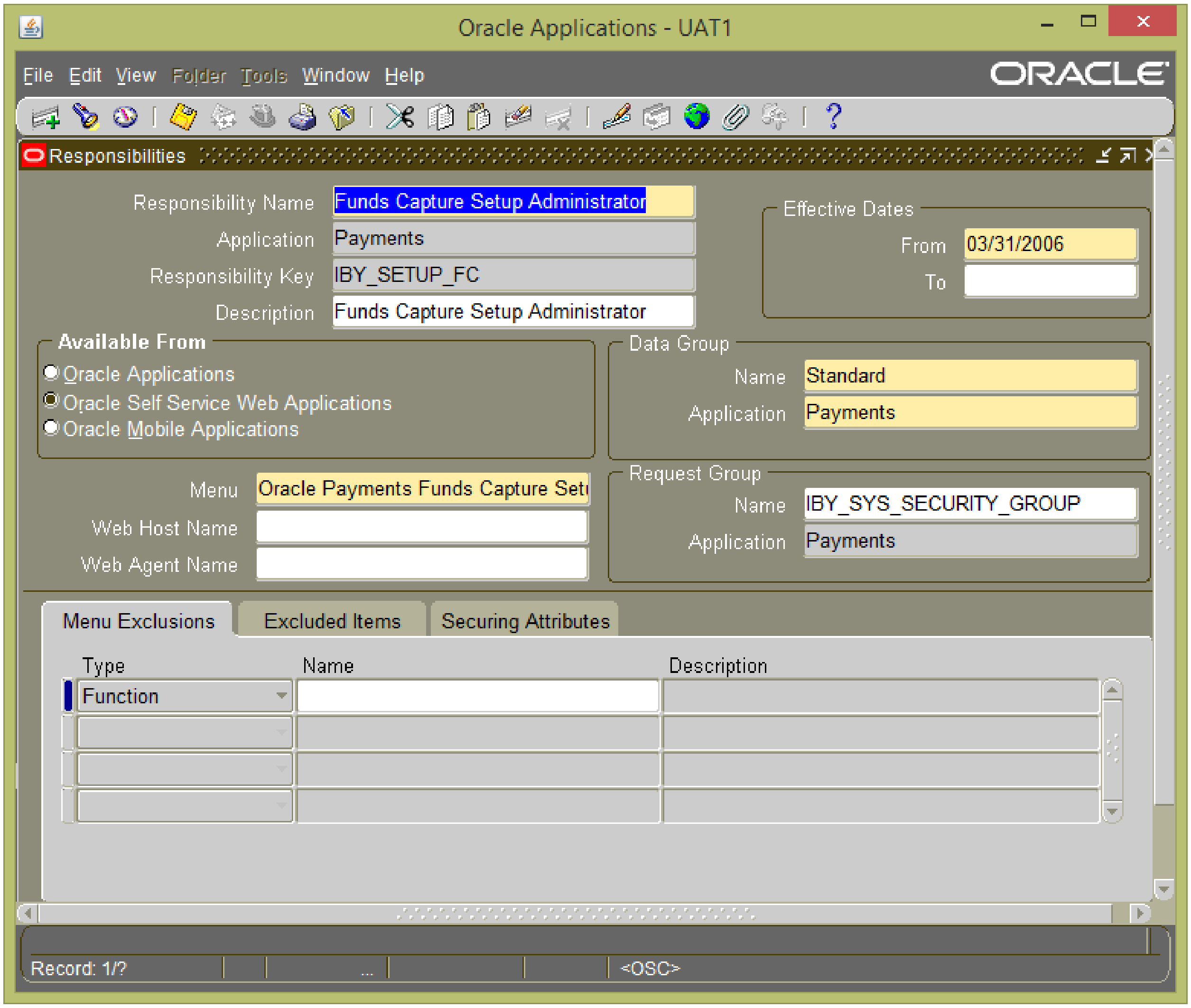
Group: IBY_SYS_SECURITY_GROUP
Application: Payments
Code:
Description:

Requests

Type	Name	Application
Program	Encrypt Credit Card Data	Payments
Program	Encrypt External Bank Account Data	Payments
Program	Encrypt Transaction Extension Data	Payments
Program	Decrypt Credit Card Data	Payments
Program	Decrypt Transaction Extension Data	Payments
Program	Decrypt External Bank Account Data	Payments
Program	Mask External Bank Account Data	Payments
Program	Mask External Credit Card Data	Payments
Program	Decrypt Credit Card Transaction Data	Payments
Program	Encrypt Credit Card Transaction Data	Payments

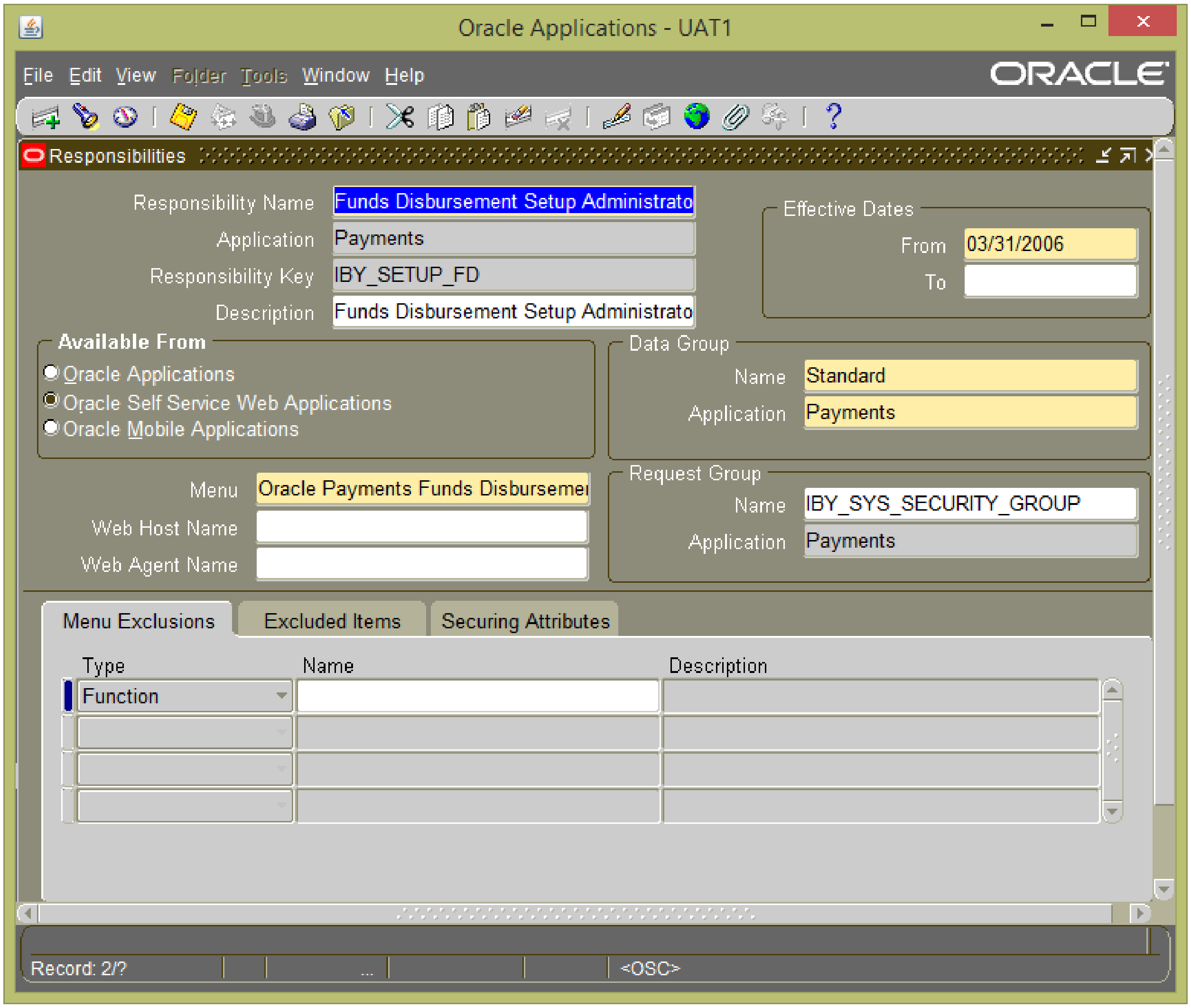
Description: Encrypts credit card data

Responsibility: Funds Capture Setup Administrator



Decryption of Credit Card Data and Bank Account Data; Risks and Controls

Responsibility: Funds Disbursement Setup Administrator



Responsibility: Payments Setup Administrator