AMX Access Management Solutions

AMX includes Identity and Access Management tools. AMX Access Management uses Role Based Access Control and includes tools to analyse and configure RBAC for existing systems with complex responsibilities. Identity and Access Management can be implemented by identitySync once roles are configured.

Access Management tools are:

- responsibilityAnalyzer, a tool to check for inconsistent responsibilities or privileges based on an individual's role.
- roleAnalyzer, a role mining tool for application with existing account responsibilities and privileges.
- SODAnalyzer, a tool to check Separation of Duties of user accounts.

Role Based Access Control

A top down approach to Role Based Access Control (RBAC) implementations manage permissions and responsibilities based on the account owner's role.

- Roles in the organisation are created and assigned to Individuals.
- The responsibilities for each role are established and checked against a Separation of Duties SOD matrix.
- An Identity Management System such as identitySync is configured to manage roles as individuals join, move and leave the organisation.

A bottom up approach is more common, an implementation of RBAC to simplify Access Management as an alternative to using a Request Approve process. Establishing the roles is a Role Mining process:

- A person's role has to be defined from information managed by HR.
- Responsibilities are already assigned, so they may need to be cleaned up. responsibilityAnalyze checks each responsibility to see what role each person assigned it has. This will identify individuals that may have been assigned or not re-assigned responsibilities as their roles change in the organisation. A real example showing an individual having an inappropriate responsibility after a change of location:

```
Responsibility name= sec. RI.Lon
```

```
8,population
5,London,Ceded Reinsurance
1,Boston,Ceded Reinsurance
1,London,Finance
```

- Re-evaluating Separation of Duties can also identify inappropriate responsibilities. SODAnalyze can be used to identify responsibilities that need to be resolved. A real example the output of SODAnalyze:

```
SOD error for 00333750 PI - 1b - Contacts Across Org,PI - 1b - Order Entry Across Org
SOD error for 00555229 PI - 1b - Contacts Across Org,PI - 1b - Order Entry Across Org
SOD error for 00335006 PI - 1b - Contacts Across Org,PI - 1b - Order Entry Across Org
```

- A set of responsibilities for each role can be established using roleAnalyze. roleAnalyze can sort, quantify and create role models containing the equivalent responsibilities. A role is an organisational view of an individual and can be constructed from one or more Identity Attributes for example a combination of Management Unit and Job Description.

roleAnalyze

roleAnalyzer is a Role Mining process that can create roles for use in an Identity Management system. roleAnalyze uses Identity Attributes to establish roles, then the responsibilities of every person with the role are summarised, and if the responsibilities are consistent, the role is established and defined. The role definition describes the responsibilities a person with this role should have. roleAnalyse can create a batch file to create the roles (template users) with the responsibilities (groups) in the ActiveDirectory which can then be used by an Identity Management system such as identitySync.

Features:

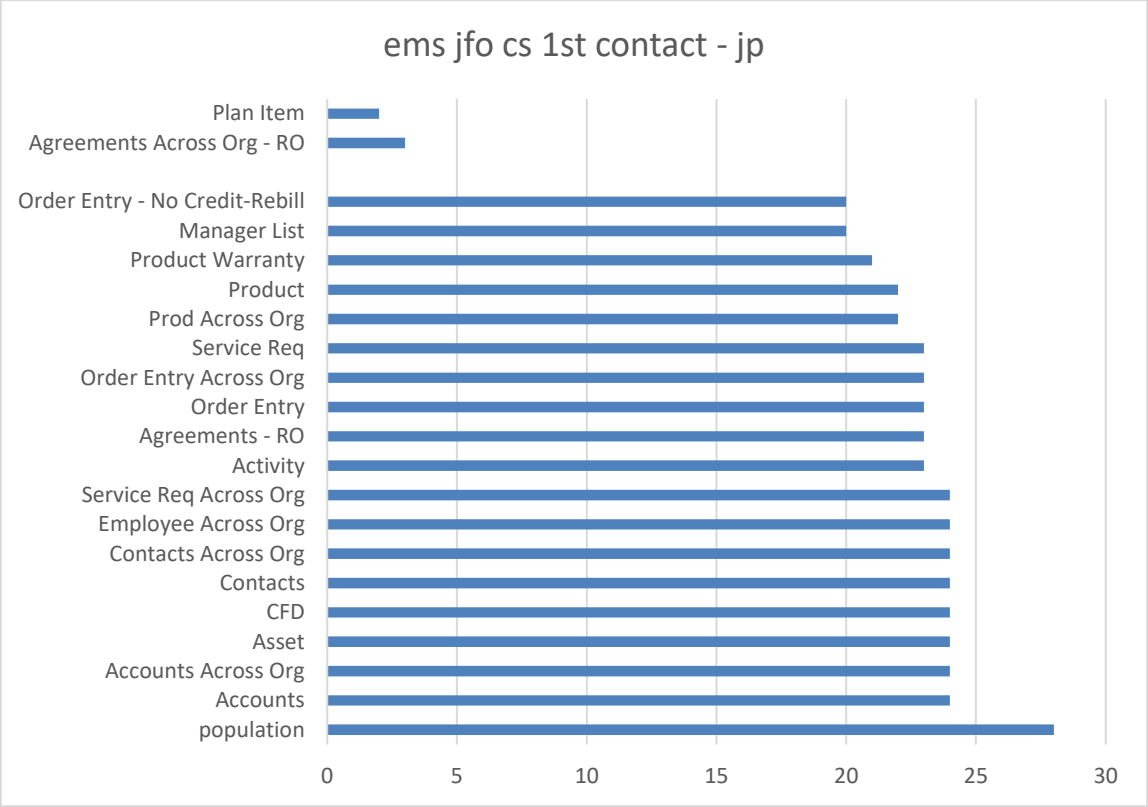
- Allows roles to be made up of one or more identity attributes, such as department AND job role.
- Reports the total number of persons and the number of unique roles.
- For each unique role, the number of persons with each responsibility are reported, sorted by the number of persons.
- Calculates a Quality factor 0 – 1.0 where a perfect score of 1 represents a situation where all the persons in the role have the same set of responsibilities.

- Creates a batch file which can be used to load roles into an Identity Management system. Formatting batch updates uses a template.
- Uses in memory databases, extremely fast to run.

Results

In a bottom-up situation, the results of roleAnalyze need to be interpreted before being used in an Identity Management system to manage roles. A number of issues can arise which must be resolved before Role Based Access Control is turned on.

The role analysis report can be used by Excel to visualise the responsibilities associated with a role. This real example from roleAnalyze shows a well-defined role:



Issue 1 Individuals with no responsibilities.

With a population of 28 there are only 24 individuals with responsibilities. Why do some 4 individuals have no responsibilities? In this situation the administrator has perhaps removed responsibilities rather than disabled the account. It is critical that an Identity Management system does not reverse this until an alternative process is put in place for individuals that leave the organisation.

Issue 2 Individuals with missing responsibilities – Nested Responsibilities

In the example above there are 4 individuals that do not have “PI - 1b - Order Entry - No Credit-Rebill”. This needs to be investigated to see if this is an omission or if this responsibility is a subset of a responsibility they already have, for instance “PI - 1b - Order Entry”. This would have to be resolved by an application specialist, and then “PI - 1b - Order Entry - No Credit-Rebill” might be removed from the role as superfluous.

Situations where a responsibility is included in another responsibility that an individual already has is a Nested Responsibility. In situations where groups can be members of other groups makes the situation obvious. Where it is part of the application's access control, as in the example above, this should be defined in the application's documentation.

Issue 3 Resultant Responsibilities - Nested Roles

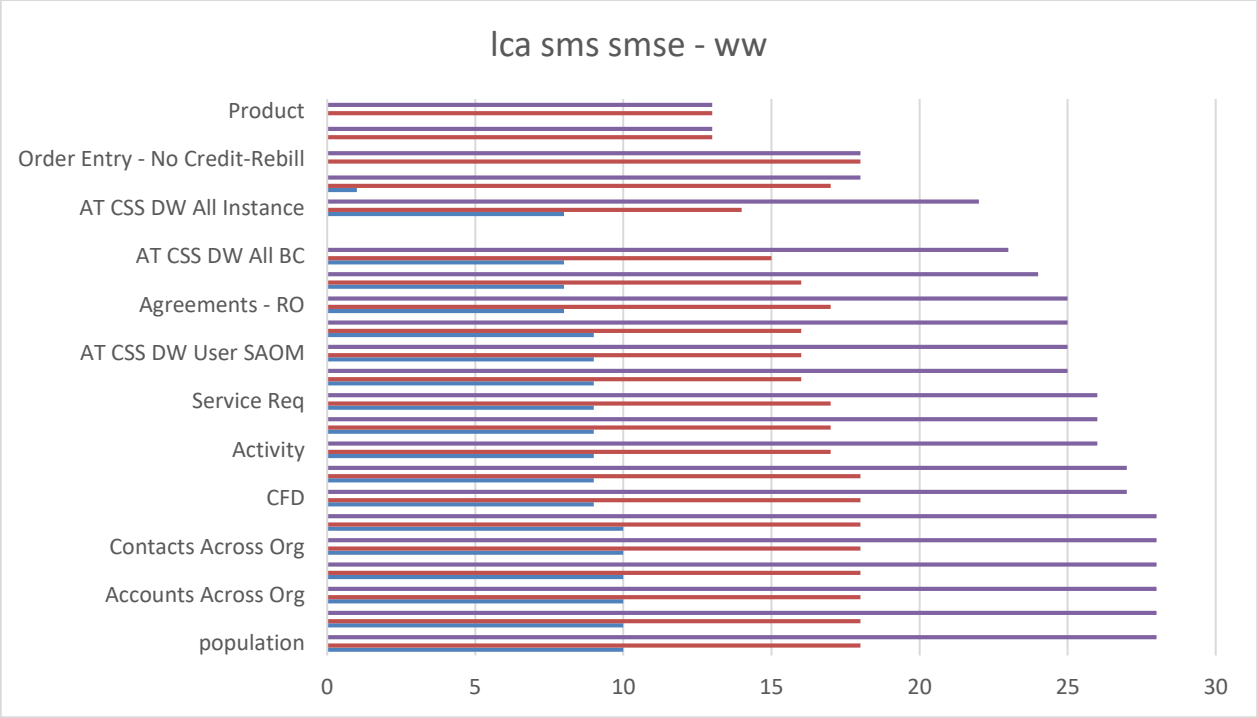
Nested Roles cause a problem for roleAnalyzer. The parent role will simply have all the responsibilities of the child role and some extra ones. The Individuals having the parent role are often Team Leads or Managers, or global staff as opposed to local. The difficulty for roleAnalyzer is when the roles cannot be discriminated, that is there is no information from Identity records concerning the additional responsibilities of some (unknown) number of individuals to determine who should have the parent role.

In some cases the parent role is discretionary, and the extra responsibilities have been given after a Request and Approve process rather than as part of a role. Responsibilities provisioned in this way must be excluded from roleAnalyzer. Often this is simple, the Request and Approve process has a database of responsibilities that it provisions, and an extract, a file containing the responsibilities can be given to roleAnalyzer to ignore in the property roleIgnoreResp.

Issue 4 Resultant Responsibilities – Multiple Roles

When responsibilities are given to a few individuals because they are performing multiple roles, as in this example below, the combined roles, the one with the largest number of individuals has a poor consistency. When better granularity is used and the role is split into two sub-roles, it can be seen that each sub-role has better consistency.

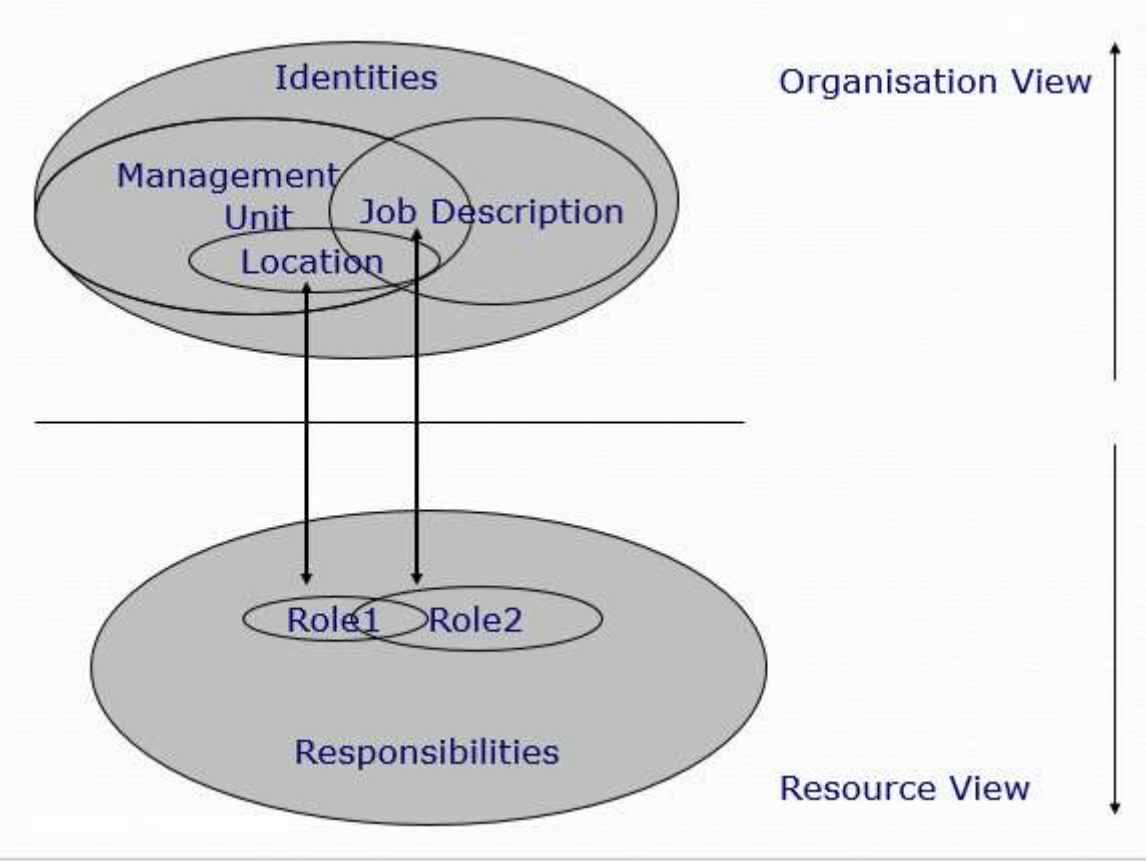
In the example below, responsibilities are given to individuals based on role A, and similar over-lapping responsibilities are given based on role B. roleAnalyze cannot identify roles in this situation. Generally use the finest granularity that is available, and then reduce it to reduce the number of roles as long as the consistency is preserved. Self-evidently when the number of roles approaches the number of Identities the granularity is too fine.



In all these RBAC reviews, the administrators of the application must be the arbiters. Inconsistent responsibilities can be caused by inconsistent administration, roleAnalyze can highlight this but it must be corrected by the administrators.

Issue 5 Incorrect Role Definition

Another issue that has caused problems is when Roles are defined differently when viewed by the Organisation compared to the Application Owner. In the example below, Role1 is based on the Management Unit and Location, while Role2 is based Management Unit and Job Description.



When roleAnalyze reports the Responsibilities associated with each Role, they will be different. Only the Application Administrator can define the Roles. The bigger Issue is when Role characteristics used by the Administrator are not available to roleAnalyzer. In this case it is necessary to find where the Administrator got the information, so that it can be used by roleAnalyze.

Issue 6 Inconsistent Role Names

Experience has found that when Role Names are based on Identity Attributes that are free field they are often inconsistent. The debug file with loggingLevel 2 or greater reports a sorted list of role names. For example:

```
Role Table ============================================================
ASSURANCE SOLUTIONS DIVISION
ATM CT BUSINESS CENTER
ATM CT BUSINESS CNTR
ATM CT BUSINESS CREATION
ATM CT CENTER OF EXPERTISE
ATM CT CUSTOMER TEAM STAFF
...
```

The Best Practice recommendation is to correct the Role Names at source and re-extract the data. Alternatively the Role names can be corrected with a sequence of “replace” Attribute Modifiers. See AMS Reference documentation for more information concerning Attribute Modifiers. For example using replace:

```
Job Title,role;replace/CNTR/CENTER/
```

Input Files

Comma separated files containing:

- Identities, that define the roles from an individual’s attributes such as job, location etc.
- Responsibilities, the privileges or groups that each person has assigned and that will be used to define the roles.

The identity file and the responsibility file can be the same file. There must be a common unique identifier, and this must be defined in the schema. The identity and responsibility file properties have a suffix 1. Currently only 1 file of identities and 1 file of responsibilities can be used.

The responsibilities can be on multiple lines, a format used by applications such as Siebel and Oracle ERP, for example:

```
'personID','responsibility'
'00005211','Accounts'
'00005211','Accounts Across Org'
'00005211','Activity'
'00005211','Agreements - RO'
'00005211','Asset'
```

```
'00005211','Asset Exchange'
'00005211','CFD'
'00005211','Common User'
'00005211','Contacts'
```

Or on a single line using RespListDelim1, the format used by accountSurvey.

Output Files

1. Debug file, always debug.txt

2. Info, always info.txt, a summary report containing:

   The file name of the Identity file
   The date stamp of the Identity file
   The number of identities extracted
   The file name of the responsibilities file
   The date stamp
   The number of individual responsibilities extracted
   The number of roles found, and the number ignored because of too few members.
   Any errors

3. Role Report as specified in the property “roleReport”, a CSV file containing the name of the role and a list of responsibilities on subsequent lines.

4. Role Out file as specified in the property “roleOutFile”, a file containing role definition made with the template and separator that can be used to create roles for identitySync. Formatted using Role Template and Responsibility Separator to create a batch file that can run against the Active Directory.

Role Report File

This example shows the first role as well formed, and the second not so well formed. Since both of these roles are Customer Service Contact Centres in different locations you would also expect them to be fairly similar.

```
Number of unique roles = 304
Number of persons = 8973
```

```
Users per role Ratio 29.51645
Responsibility Cutoff = 80%
….

Role name= ei efo cs 1st contact ctr
36,population,36                    < - test for role min population
35,Accounts Across Org,35
35,Agreements – RO,35
35,Contacts Across Org,35
35,Employee Across Org,35
35,Order Entry,35
35,Order Entry Across Org,35
35,Prod Across Org,35
35,Product Warranty,35
35,Service Req,35
35,Service Req Across Org,35
                                    < - subsequent responsibilities are held by < 80% of population
23,Agreements Across Org – RO,23
9,Manager List,9
4,Order Entry - No Credit-Rebill,4
3,Agreements Across Org,3
1,Order Entry – RO,1
1,Plan Item,1
1,Product Admin,1
1,Asset Exchange,1
Score 1 R1 = 35 Rn = 28             <- Scoring for the role 1 = perfect, 0.5 half good. See Quality Factor.

..
Role name= ei afo cs 1st contact - pw
91,population,91
76,Accounts Across Org,76
76,Contacts Across Org,76
70,Service Req Across Org,70
69,Service Req,69
68,Agreements Across Org – RO,68
```

68,Product Warranty,68
67,Employee Across Org,67
66,Agreements – RO,66
65,Prod Across Org,65

56,Order Entry Across Org,56
55,Order Entry,55
22,Order Entry Across Org – RO,22
19,Order Entry – RO,19
17,Activity – RO,17
13,Service Req – RO,13
9,AT CSS DW Security Model Filters,9
9,AT CSS DW User SAOM,9
9,AT CSS DW User SD,9
7,AT CSS DW Employee BU,7
7,AT CSS DW PII Individual,7
6,Contacts Merge,6
6,Product Admin,6
5,Agreements Across Org,5
5,Manager List,5
5,AT CSS DW APAC Instance,5
5,AT CSS DW All BC,5
4,AT CSS DW All Instance,4
4,AT CSS DW Employee BC,4
4,Order Entry - No Credit-Rebill,4
4,Plan Item,4
4,PDQ Administrator,4
2,AT CSS DW PII All,2
2,Assignment Manager – WLA,2
2,AT CSS DW All BU,2
1,AT- Read Only,1
1,Account Administration,1
1,Asset Exchange,1
1,Assignment Manager - Skills Update,1
1,Employee Export Disable,1
1,Employee List,1

```
1,Product List,1
1,Sales Support Append,1
1,Temp SO Unlock,1
Score 0.9263158 R1 = 76 Rn = 60
```

The person count is repeated on the RHS to make creating a Graph in Excel more efficient.

Role Create File

The Role Create file name is defined in the property roleOutFile. It is intended to be used as a batch file to create the roles in the application. The template allows text and substitution of %role% and %responsibilities%. For example a template:

```
dsadd CN=%role%,OU=Roles,DC=corp,DC=Example,DC=com -samid %role% -desc Role Definition -memberof
%responsibilities%
```

Will create a batch file for the Active Directory to make role templates for identitySync. When identitySync is configured to use roles with the same role definition in the Schema, the roles will be managed as individual's attributes change.

Quality Factor

Quality of roles are calculated from the responsibilities in the resultant role.

For each responsibility the number of persons with the responsibility is accumulated into a running total. The final total is divided by the perfect total, which is when every responsibility has every person assigned to it. A score of 1.0 is perfect, less than 1.0 is less than perfect. For example:

```
10, responsibility1
10, responsibility2
10, responsibility3
10, responsibility4
10, responsibility5
```

Is a perfect role definition. Whereas:

```
10, responsibility1
10, responsibility2
10, responsibility3
6, responsibility4
```

```
4, responsibility5
```

Is not. The running total is 40, compared with a perfect score of 50. The Quality Factor is 40/50 = 0.8. A poor quality such as 0.5 would be:

```
10, responsibility1
4, responsibility2
4, responsibility3
4, responsibility4
3, responsibility5
```

The running total is 25, giving a Quality Factor of 25/50 = 0.5.

The Quality Factor allows well defined roles to be classified, ignoring poorly defined roles.

Properties

1. IdentityDelim1 the delimiter used in the CSV file of identities. Default ","

2. IdentityResource1 the file name of the identities in CSV format

3. IdentitySchema1 the file name of the Schema.

4. ignoreEmptyRole = Y or N. When Y a role with no responsibilities is ignored and not reported

5. LoggingLevel, 1 – 5, 1 = summary, 5 maximum, default 3. The logging methodology is:

1	debug log matches info. Minimal logging.
2	configurations, debug user.
3	details of configuration.
4	identity and resource record names.
5	details of identity and resource attributes and the match process.

6. LoggingType, 1 – 3, default 1:

1	write to info and debug log files only.

2 write to info and debug, with info to console.
3 write to info and debug, with info and debug to console.

7. respDelim1 the delimiter used in the CSV file of responsibilities. Default ",".

8. respListDelim1 the delimiter used in the CSV file of responsibilities when the responsibilities are a list. The format use by accountSurvey. Default ":".

9. respResource1 the file name of the reponsibilities in CSV format.

10. respSchema1 the file name of the schema.

11. roleIgnoreResp the name of a file containing responsibilities that will not be included in the role definitions. Responsibilities that are not provisioned as part of a Role, but by a Request and Approve process. For example "Bike Club" etc

12. roleMinPopulation, roles with < than this value are not reported in roleReport or RoleOutFile. Roles with small populations tend to be uneconomic

13. roleReport the role analysis output file

14. roleReportCutoff when defining a role, responsibilities held by less than this percentage of identities are not part of the role.

15. roleReportFull = Y or N. The full role report has added information concerning identity population having the role

16. roleReportTruncate = Y or N combined with roleReportCutoff, responsibilities held by less than this percentage of the population are not part of the role and are not reported in RoleReport.

17. roleOutFile = file containing role definition made with the template and responsibility separator.

18. roleTemplate template with substitution of %role% and %responsibilities%. Can create a batch file to create the roles specifically for identitySync. For example:

```
dsadd CN=%role%,OU=Roles,DC=corp,DC=Example,DC=com -samid %role% -desc Role Definition -
memberof %responsibilities%
```

19. roleRespSeparator the separator used in the roleTemplate for responsibilities. Default ","

Schema Files

Two Schema files are needed. The Identity Schema must contain an Attribute that can be used as a join with the Response Schema, and the Metaverse Attribute "role". For Example:

```
samAccountName,;DisplayName
title,role
```

The Response Schema must contain an Attribute that can be used as a join with the Identity Schema with the Metaverse flag "join", and the Metaverse Attribute "responsibility". For Example:

```
samAccountName,;DisplayName;join
group,responsibility
```

SODanalyze

Separation of duties or Segregation of Duties analysis creates a report of each account that has responsibilities that violate the organisation's SOD rules. SOD rules define responsibilities that cannot be assigned such that single person has end to end access rights over any function across an organisation that allows them to complete an inappropriate process or commit fraud. Simple examples are:

- Prevent any one individual from modifying code and deploying it in production. This is PCI requirement 6.4.2 Separation of duties between development/test and production environments.
- Prevent any one individual from creating purchase orders and acknowledging receipt of goods or services.
- Prevent any one individual from creating financial reports and updating accounts receivable data or making financial transactions to falsify a company's financial situation. This is important for SOX section 404 compliance.
- Prevent an individual approving their own expense reports.

The critical component of the SOD analysis is the SOD rules. SODanalysis can use two formats, the SOD rule matrix and the SOD rule set.

SOD Rule Matrix

Separation of duties rules for SODanalyze are described using a SOD Matrix in csv form. The matrix can be prepared in Microsoft Excel and it can be described using a mixture of roles and responsibilities. In the example below the SOD matrix uses roles, which are defined in a separate role definition file and used to expand the roles into their component responsibilities to create an internal table of SOD rules based on responsibilities. The SOD rule matrix is symmetric, a non blank entry creates a SOD rule that a person cannot be assigned intersecting roles or responsibilities. For example a developer cannot be assigned a responsibility that allows them access to QA or production release activities. That is a developer cannot be allowed to have the ability approve the release of a modification to production code or to release unapproved code to production. And, someone with the role of Production release cannot have access to any developer responsibility especially the ability to modify and compile code.

Only the left side of the rule matrix is significant for SODanalyze, the right hand side is implied and any entry in the right hand SOD of the rule matrix causes an error message that is intended to prevent any confusion.

	developer	tester	QA	Production release
developer	x		Implied	Implied
tester		x		Implied
QA	Prohibited		x	Implied
Production release	Prohibited	Prohibited	Prohibited	x

This prevents the most common fault of SOD rule definitions, asymmetry. Asymmetric rules cause subtle errors in Access Management such as an account with a role of QA is approved to be given Developer responsibilities, but an account with a role of Developer is declined when applying for a QA role. Asymmetry in large SOD rule matrices is difficult to identify without tools. Consider:

	developer	tester	QA	Production release
developer	x		No	No
tester		x		No

QA			x	No
Production release	No	No	No	x

When reviewing the SOD rules with this approach, the question is “what roles are incompatible with each other”? Answer “developer and QA”. Rather than “what is a QA not allowed to do” which will produce a SOD rule list. This approach reduces the time taken to review the rules by half and ensures symmetry.

It is recommended that when preparing the SOD rules, forms are sent to the subject matter experts in each area that have one line for each pair of roles. So that for instance the development manager replies:

Separation of Duties Survey

Name of Respondent	Bob Brown
Department	Development Manager
Date	11-Nov-15
Responsibility or Role	Incompatible with
Developer	Release to Production
Developer	QA

Use this response to prepare the SOD rule matrix. Where the left hand side of the matrix does not have a cell in the left side of the matrix for “developer”, swap it for “Release to Production”.

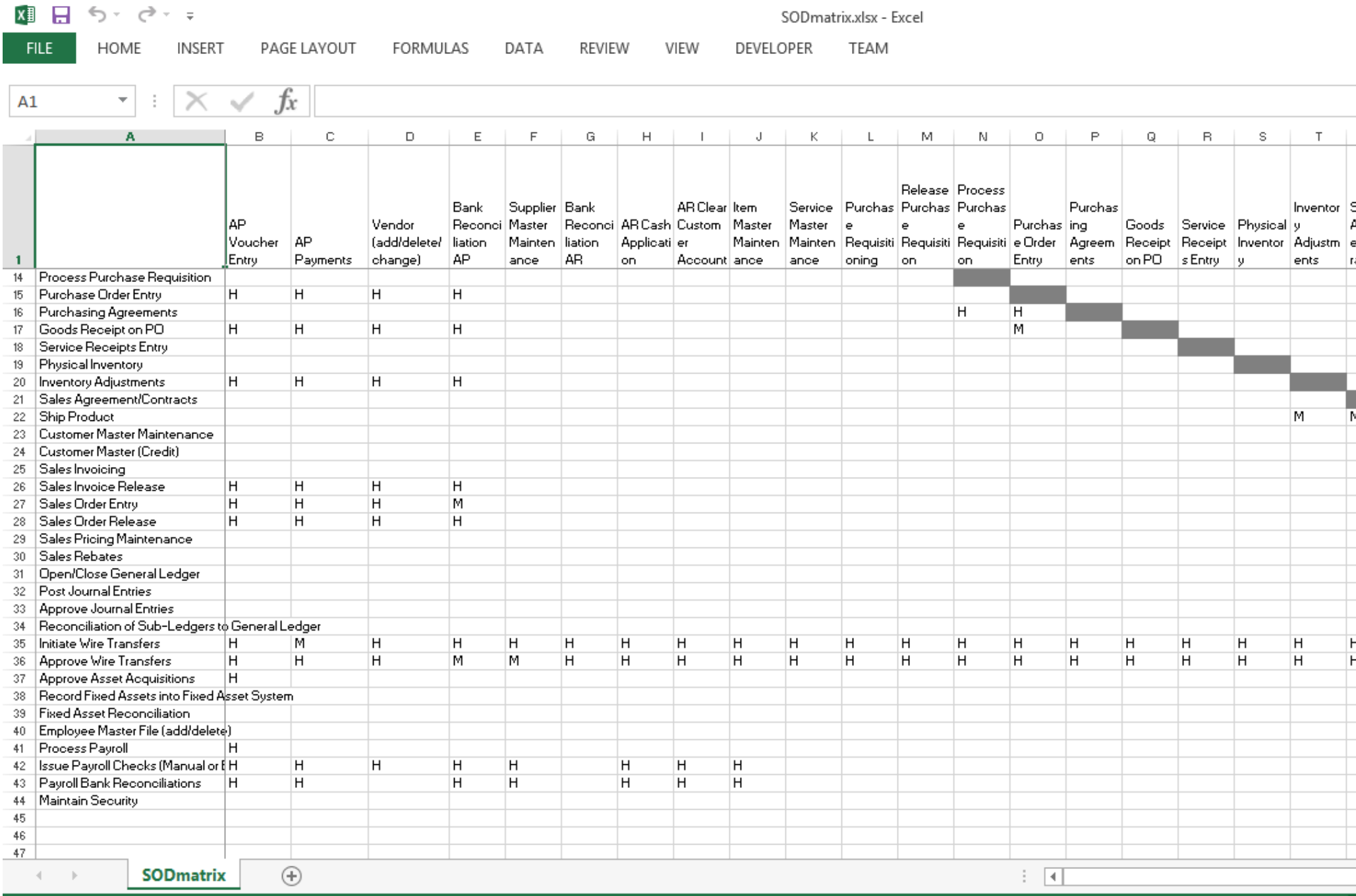
Note that the response from the QA manager is not consistent, with the Development Manager. This is usually because the Subject Matter Experts only look downstream to who they are handing the process off to, and in this example QA forgot about the upstream development process. Unfortunately job progression also sometimes follows the flow of the process. So a developer may become a QA. This makes the downstream only view of SOD rules dangerous because developer responsibilities that are inappropriate for the QA role can be missed out of the SOD rule matrix and when the person changes roles their old responsibilities are not removed.

Separation of Duties Survey

Name of Respondant	Craig Davis
Department	QA Manager
Date	05-Nov-15
Responsibility or Role	Incompatible with
QA	Release to Production

The entry in the SOD rule matrix made from the Development Manager response can be used to confirm with the QA manager that development and QA are incompatible, or used as-is either without comment.

As SOD rule matrices become larger asymmetric errors become more difficult to identify, for example when the SOD rule matrix is defined in terms of responsibilities:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1		AP Voucher Entry	AP Payments	Vendor (add/delete/change)	Bank Reconciliation AP	Supplier Master Maintenance	Bank Reconciliation AR	AR Cash Application	AR Clear Customer Account	Item Master Maintenance	Service Master Maintenance	Purchase Requisitioning	Release Purchase Requisition	Process Purchase Requisition	Purchase Order Entry	Purchasing Agreements	Goods Receipt on PO	Service Receipts Entry	Physical Inventory	Inventory Adjustments
14	Process Purchase Requisition																			
15	Purchase Order Entry	H	H	H	H															
16	Purchasing Agreements													H	H					
17	Goods Receipt on PO	H	H	H	H										M					
18	Service Receipts Entry																			
19	Physical Inventory																			
20	Inventory Adjustments	H	H	H	H															
21	Sales Agreement/Contracts																			
22	Ship Product																			M
23	Customer Master Maintenance																			
24	Customer Master (Credit)																			
25	Sales Invoicing																			
26	Sales Invoice Release	H	H	H	H															
27	Sales Order Entry	H	H	H	M															
28	Sales Order Release	H	H	H	H															
29	Sales Pricing Maintenance																			
30	Sales Rebates																			
31	Open/Close General Ledger																			
32	Post Journal Entries																			
33	Approve Journal Entries																			
34	Reconciliation of Sub-Ledgers to General Ledger																			
35	Initiate Wire Transfers	H	M	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
36	Approve Wire Transfers	H	H	H	M	M	H	H	H	H	H	H	H	H	H	H	H	H	H	H
37	Approve Asset Acquisitions	H																		
38	Record Fixed Assets into Fixed Asset System																			
39	Fixed Asset Reconciliation																			
40	Employee Master File (add/delete)																			
41	Process Payroll	H																		
42	Issue Payroll Checks (Manual or E	H	H	H	H	H		H	H	H										
43	Payroll Bank Reconciliations	H	H		H	H		H	H	H										
44	Maintain Security																			
45																				
46																				
47																				

Using roles reduces the size of the SOD rule matrix, but, the responsibilities for each role must be described in a roles file, the format is the role followed by the responsibilities, for example:

```
dev,devFull1,devFull2,devFull3,gatewayFull,SDLCshare
test,testFull1,testFull2,testFull3,gatewayFull,SDLCshare
QA,QAFull1,QAFull2,gatewayFull,SDLCshare
prod,prodFull1,prodFull2,prodFull3,gatewayFull,firewall60Full,SDLCshare
```

When roles are used some of the responsibilities have to be ignored because they need to be present in both roles. They could be removed from the roles definition or as in this case they can be added to an ignore entitlements file for example:

```
gatewayFull
SDLCshare
```

Which in the example above, will inhibit any SOD collision reports concerning the responsibilities gatewayFull and ,SDLPshare.

An account's or person's responsibilities are described in a responsibilities file in the format accountName, responsibility. Other fields are ignored. For example:

```
glang,devFull1
glang,devFull2
glang,devFull3
glang,gatewayFull
glang,testFull1
glang,SDLCshare
chunter,QAFull1
chunter,QAFull2
chunter,gatewayFull
chunter,SDLCshare
chunter,devFull1
chunter,firewall60Full
```

Resulting in an SOD report such as the example below:

```
SOD error for glang
devFull1,testFull1
devFull2,testFull1
devFull3,testFull1

SOD error for chunter
devFull1,firewall60Full
devFull1,QAFull1
devFull1,QAFull2
firewall60Full,QAFull1
firewall60Full,QAFull2
```

This is a trivial example and a real SOD report using a role based SOD rule matrix can produce a large number of violations. A quick scan can identify the issue. In the example above the person glang has testFull1 in each pair of violations and that is probably the offending responsibility.

Chunter has a lot of dev and QA responsibilities. It is not possible to identify which role should be removed without further investigation.

Sometimes the requirement is not so much to clean up SOD reports, but use them to identify areas where compensating controls are needed.

SOD Rule Set

The alternative to the SOD rule matrix is an SOD rule set. This is a more common, legacy format. SODanalyze checks the rule set for consistency, because unlike the matrix it is easily possible to define an inconsistent SOD rule set. The most common issue is a responsibility contained in a rule does not have its own rule. The rule set can be described using a mixture of roles and responsibilities. The format is:

> Rule_for_roleN, conflicting_role1, conflicting _role2, conflicting _role3, conflicting _role4,……

Roles must be expanded using the roleFile property to match the responsibilities file. An example rule set:

```
dev,test,QA,prod
test,dev,prod
prod,dev,test,QA
```

Is inconsistent because the responsibility QA does not have its own rule

Properties

The analysis uses a properties file:

1. LoggingLevel, 1 – 5, 1 = summary, 5 maximum, default 1. The logging methodology is:
    1. debug log matches info. Minimal logging.
    2. configurations, debug user.
    3. details of configuration.
    4. identity and resource record names.
    5. details of identity and resource attributes and the match process.

2. LoggingType, 1 – 3, default 2:

    1. write to info and debug log files only.
    2. write to info and debug, with info to console.
    3. write to info and debug, with info and debug to console.

3. responsibilitiesFile a CSV file containing an account name and its responsibilities.

4. responsibilitiesDelim the delimiter used in the responsibilities file, default “,”.

5. responsibiliesDelimList, the delimiter where the responsibilities are on one line. Format account name<responsibilitiesDelim>responsibility1<responsibilitiesDelimList>responsibility2. Default “”.

6. roleFile an optional CSV file containing roles and their responsibilities. This is used when the SODruleFile uses roles rather than responsibilities. Using roles makes the SOD rule matrix smaller and easier to use. The roleFile expands the SOD rule matrix roles into the much larger SOD rule matrix of responsibilities. Possibly too large to manage. With logging level of 3 or greater the expanded SOD matrix is written to the debug file.

7. roleDelim the delimiter used in the role file, default “,”.

8. SODignoreFile an optional file containing responsibilities that are defined in the roleFile but must be excluded from the SOD rules. File contains one responsibility per line. For example:

   Roles dev and QA both contain a responsibility SDLCshare. SDLCshare should be added to the ignore file so that the set of responsibilities for dev and QA do not intersect. That is, if any person with a dev responsibility will be reported as a SOD violation if they have any responsibility that is part of the QA role, for instance SDLCshare. When both roles have a common responsibility like SDLCshare this will always be a violation and is a false positive.

9. SODruleDelim the delimiter used in the ruleDelim file, default ",".

10. SODruleFile an optional file containing the SOD rule set in CSV format.

11. SODReport the filename for results of the analysis, a SOD analysis report.

12. SODmatrixDelim the delimiter used in the roleDelim file, default ",".

13. SODmatrixFile an optional file the SOD rule matrix in CSV format

14. SODrisk the risk of the SOD rule defined in the matrix. Default = all rules. If the matrix has H, M and L SODanalyzer can be run with SODrisk H, M or L

Error Messages

Error: bad rule <line>

Line has a null rule responsibility

Error inconsistent SOD rule. Add <rule>,<responsibility> to SOD rules

Error only occurs when a SOD rule file is used. Line defining a SOD rule found a responsibility that does not have a matching rule. The SOD rule set is inconsistent, add the rule for the responsibility <responsibility> to the rule set file.

Error: reading file <file> row# <N> <row label> should be <column label>

The responsibility label on row N must match the column label N, the check is case sensitive. Update the responsibility label on row N.

Error: reading file <file> row# <N> <role or responsibility> non blank right hand side <value>

SODanalyzer uses the left hand side of the SOD rule matrix, this rule matrix has a non blank cell. Transpose and / or delete the cell to the left hand side. This error is to ensure and enforce symmetry in SOD rules.

Error: Segregated role <role> has common responsibility <responsibility> add it to ignore list

The responsibilities for <role> include a responsibility that is part of another role. This makes an inconsistent SOD rule. See property SODignoreFile.