Document:

Reference Architecture

Part:

Authorisation Reference Architecture

Dated:

April 2022

Company contact:

Chief Technology Officer

cto@downerdefence.com.au

Contents

**No table of contents entries found.**

Figures

**No table of figures entries found.**

Tables

**No table of figures entries found.**



# AUTHORISATION Reference Architecture

References:

## Introduction

Timely access to information underpins our enterprise’s ability to make informed business/operational decisions. Allowing inappropriate access to that same information can completely undermine any business/operation benefit. Protecting information while still making it useful involves achieving a balance between the need to share and the need to know.

1. Traditional ICT mechanisms for assigning privileges to facility access are quite static and do not take into consideration the dynamic nature of Defence business, nor do they account for the fact that the level of protection offered may need to change over time. One of the issues is that authorisation systems do not make use of all the factors available to decide.
2. The factors that can be used to make an authorisation decision are referred to as attributes; utilising them to make such a decision is known as Attribute Based Access Control (ABAC).
3. This Reference Architecture introduces the taxonomies and models to describe the logical application of ABAC and other authorisation models in Defence. These will provide guidance for new systems and a categorisation for existing systems to determine their suitability.

### Definition

In relation to ICT Architecture, *Authorisation* is about making access decisions for resources. The entity requesting access, the resource being accessed, and general environment are all potential sources of attribute that are used to make the access determination. Logically, the factors are business rules that make (potentially real-time) request by request access decisions.

### Scope

This reference architecture applies across all our business, including fixed and deployed environments, interfaces to partners and all security domains.

### Audience

This document is for solution architects and capability planners. It is intended to guide and constrain technical solutions to fit within the overall our enterprise. Planners involved in the capability planning function should apply the language and concepts introduced to ensure a consistent approach.

## Concepts and Terminology

A taxonomy and model are required to understand and apply authorisation to the solutions deployed in our enterprise. IT based authorisation mechanisms need to be applied to control access to both physical and logical resources. The logical resources are typically data in applications. This reference architecture introduces the following concepts:

1. Authorisation model and factors,
2. Anatomy of an application,
3. Provisioning modes.
4. As discussed, authorisation is the determination to allow or deny access to a *resource* by a requesting entity (*requestor*). The resource could be physical, such as a building, room, car, and any other piece of equipment; or it could be logical, such as information or a business system. In addition to the requestor and the resource, the overall *environment* can form part of any access decision as it can introduce attributes that are independent or the requestor and resource, such as threat.
5. The requestor is most often assumed to be a person. However, it can be a non-person entity, such as a software agent.

## Authorisation model and factors

The attributes that are associated with making authorisation decisions fall into the following categories: who, what, where, why, how, when and the overall environment. The diagram at Figure 1 shows where the attributes fit into a logical model. Notice that a set of business rules are applied to the attributes to make the overall authorisation decision. This forms the basis of Attribute Based Access Control (ABAC), which is the ultimate access control model as it can encompass all other forms.

1. 

Figure 1: Logical Access Management Model

## Application Anatomy

While modern applications include authorisation mechanisms to control access to their resources, few allow the decision to be offloaded to a separate PDP and are usually their own PEP. Indeed, some applications manage the full set of functions and identity internally and are therefore completely self-contained.

### Standalone Applications

A logical representation of how a typical, self-contained, application looks is shown in Figure 2.

1. 

Figure 2: Anatomy of a self-contained application

1. In Figure 2, all the user identity information (I) is located completely within the application in some form of data store. The user information is mapped (M) within the application to the functions (F) of the application. This results in code (C) execution to generate some form of output (O). Notice that both the mapping and code areas may use extra attributes from the user store to determine their behaviour.
2. With a fully self-contained application all user administration and authorisation are done in the application. For a single application this is not such a problem, but for thousands it starts to become an issue. Especially, if for each of those applications there may be many functions (for argument, say a hundred) that users are mapped. Across Defence there could be thousands of mapping tables (one per application) containing up to 100,000 users by 100 function entries. So, 1000s of 10 million of mappings to maintain; therefore, potentially ten billion mappings to manage (10,000,000,000).
3. To help manage this, applications can externalise their identity and mapping to a centralised solution and allow business rules to process attributes to determine access to application functions, as shown in Figure 3. In this figure, for each function, an application requests of a central solution whether an identity can execute that function. A set of Business Rules (policies) inform the decisions. This starts to look like the logical representation in Figure 1.
4. While this has the potential to automate access decisions and make them dynamic (as the business rules use attributes that relate to the current state), there are still a lot of business rules (which are effectively mappings) to be maintained. An additional problem is that there are few applications that natively support externalising their PDP/PEP functions.
5. 

Figure 3: Application with ABAC

### Indirect function assignment

One approach to improving the situation is to introduce levels of indirection between the users and the application’s functions. The result is shown in Figure 4.

1. 

Figure 4: An application with indirect mapping

1. In this case Business Roles (B) and Technical Roles (T) are introduced along with extra mapping tables between them (X & Y). They are defined as:
2. **Business Role.** A categorisation of function that is defined in terms that have meaning to business users. Examples would be Payroll Manager, Commanding Officer, or Gun Number. The definition of a Business Role is realised by its mapping (X) to Technical Roles.
3. **Technical Role.** A categorisation of function that relates more to the resources managed by the application. Examples would be Approve Pay Change, Approve Standing Order or Fire Gun. The definition of a Technical Role is defined by its mapping (Y) to application functions (F).
4. Having implemented this extra level of indirection, if the definition of a Business Role changes only its mapping (X) to the Technical Roles must change and so on. So, at the very least, the extra indirection introduced isolates change between distinct parts of the application. This approach is what is referred to Role Based Access Control (RBAC).
5. It turns out that, if this approach was taken for individually for every application, then there is potential to have to maintain 100 times more mappings than discussed above. However, this approach comes into its own if the mapping is external to the application and centralised.

### Centralised management

Centralisation of the indirection between the identities and the application’s functions significantly decreases the size of the problem because the Identities, Business Roles, Technical Roles, and their various mappings only must be managed once, centrally, and not in every application. Depending on the assumptions made about how many Business and Technical roles there are, the overall number of mappings to be maintained can be reduced be two or three orders of magnitude.

1. Examining the models introduced so far, they are three main components:
2. The information store (I) that understands user in the context of the application and their related attributes which is analogous to a PIP.
3. The mapping and/or rules (M) processing that determine access to the application which is the functions of PEP, PDP, and PAP.
4. The functioning component of the application (F).
5. The first two are the candidates for externalising. However, as discussed, many applications are not capable of having these functions externalised. When this is the case, it is still preferable to manage externally. In this case we want the information for the users and mapping/rules engine to be provisioned to the applications.
6. To make the best use of both ABAC and RBAC, and account for many applications’ inability to use ABAC, a hybrid approach is required that has the benefits of central management but accounts for the realities of application architecture. For applications not capable of externalising their authorisation decisions, this involves provisioning the identities and the mappings to the applications directly from a central authority or where possible intercepting application requests with some form of proxy and simulating ABAC for the application.
7. In the application model introduced so far, the mapping of identities to the applications functions (M in Figure 2) represents the PEP, PDP, and PAP as a group, so effectively everything in the model in Annex A except the actual information use to make the authorisation decision. This identity and other attribute information (I) in the application model is the PIP in the Annex A model. This results in there being three major components to consider.
8. Combining the application model with a simplified ABAC functional model from Annex A results in the diagrams at Figure 5 which includes the subject making a request to an object (a function in the application). The diagrams show the variations available in externalising and therefore centralising the various functions.
9. 

Figure 5: Combined application model, showing (a) fully externalised, (b) external identity and (c) no externalisation

1. In Figure 5c, where nothing is done external to the application, the model falls back to being the same as shown in Figure 2 and it does not matter whether the application is internally achieving ABAC or RBAC.
2. These two groupings represent the boundaries that will be able to be externalised and centralised when an application support it.
3. You cannot centralise any further right of an application’s functions (F) as shown in Figure 3 above, as the functions an application performs, the code to do them and the resulting output *are* the application. In the case where an application does expose its functions to a centralised ABAC PDP/PEP point, then all the roles and mappings can occur outside the application centrally. The divided in Figure 5 represents the best scenario for ABAC, where the full model as shown in Annex A is externalised.
4. 

Figure 6: Centralised vs application function

1. In the above figure
2. As discussed above, some Applications can centralise Authorisation entirely, while other types of Applications do not. For applications that do not, either a proxy must be established (if possible), or the central authorisation solution must provision access control information to the application.
3. Reducing the amount of overhead created by provisioning multiple applications in a distributed model may eliminate requests to multiple owners and reduce overall application security management; therefore, centralising Authorisation is imperative.
4. Using Figure 4, centralising Authorisation at positions (Y) or (F) may be possible, depending on the overall intent, nature, and type of Application. However, in some cases Applications may not expose its functions; therefore, centralising an Application from an Identity right through to technical role may be achieved, as shown in Figure 3.
5. 

Figure 7: Centralised Provisioning

Centralised provisioning in Identity and Access Management (IAM) provides:

1. **None.** No information, not even users can be externally provisioned.
2. **Acceptable.** Identity and attributes are provided.
3. **Business Role.** Identity up to and including Business Role is provisioned, therefore partially compliant.
4. **Compliant.** Identity up to and including Technical Role is provisioned, therefore considered as compliant.
5. 0 – Cannot be provisioned to
6. 1 – Only users can be provisioned, mapping uses to app function is still an app specific task
7. 2 – Users and the application groups (business or technical) can be provisioned to the application
8. 3 - ABAC
9. Figure 7: Authorisation Model

## Principles

The following principles are introduced to support this reference architecture:

1. Authorisation must be *Centralised* where-ever possible, except if limited by application or operational necessity.
2. *Decisions* that pertain to Identity are taken by the entity that is assuming the risk.
3. *Identity* and attributes must be transparent.

## Decision Framework

This section introduces the ‘so what.’ Now that we have introduced concepts, terminology, and any relationships, how do we apply the patterns and standards?

# NIST 800-162 ABAC Model

NIST Special Publication 800-162 introduces a function model for ABAC comprising of the following:

1. **Subject.** The requestor (== Entity).
2. **Object.** The protected resource (== Resource).
3. **Digital Policy (DP).** Access control rules that that combine logic with the various attributes that compile directly into machine executable code.
4. **Metapolicy (MP).** A policy about policies, or policy for managing policies, such as assignment of priorities and resolution of conflicts between DPs or other MPs.
5. **Policy Decision Point (PDP).** Computes access decisions by evaluating the applicable DPs, and mediates or deconflicts DPs according to MPs
6. **Policy Enforcement Point (PEP).** Enforces policy decisions in response to a requestor requesting access to a resource; the access control decisions are made by the PDP.
7. **Policy Information Point (PIP).** Serves as the retrieval source of attributes, or the data required for policy evaluation to provide the information needed by the PDP to make the decisions.
8. **Policy Administration Point (PAP).** Provides a user interface for creating, managing, testing, and debugging DPs and MPs, and storing these policies in the appropriate repository.
9. These concepts were originally introduced in RFC2904.
10. This results the following functional model.

