# JupiterOne Data Model

The **JupiterOne Data Model** is a reference model used to describe digital resources and the complex interconnections among all the resources in a technology organization as an **entity-relationship graph**.

The data model is defined by a set of Entities and their Relationships. It represents a reference model, not a strict or rigid structure.

## Entity

An Entity is a node/vertex in the graph that represents a resource within your digital infrastructure.

### Class and Type of an Entity

Each Entity has a specific **type** that defines what that entity is, and is assigned one or more higher level **class** that represents a more abstract categorization or labeling of the entity in the perspective of security and technical operations.

#### Type

The **type** property represents the specific type that entity is as defined by the source. For example, an AWS resource may be of type aws\_ec2\_instance or aws\_s3\_bucket or aws\_iam\_user.

#### Class

The **class** of an entity is considered an abstract, super-type that defines what that entity is within the general framework of IT and security operations. In the above example, an aws\_ec2\_instance entity has a class of Host, while an aws\_s3\_bucket is a DataStore, and an aws\_iam\_user a User.

### Common Entity Properties

Most Entities will have the following common properties.

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| name | string | Name of this entity. |
| displayName | string | How this entity is labeled in the UI, e.g. a person’s preferred name or an AWS account alias. Defaults to the same value as name. |
| summary | string | A summary / short description of this entity. |
| description | string | An extended description of this entity. |
| classification | string | The sensitivity of the data; should match an organization’s data classification scheme. For example: “critical”, “confidential”, “internal”, or “public”. |
| criticality | integer | A number that represents the value or criticality of this entity, usually on a scale between 1-10. |
| risk | integer | The risk level of this entity, on a scale between 1-10. |
| trust | integer | The trust level of this entity, on a scale between 1-10. |
| owner | string | The owner of this entity. This could reference the name/email of an individual, or the reference identifier/key to another entity in the graph as the owner. |
| complianceStatus | number | The compliance status of the entity, as a percentage of compliancy. |
| status | string | Status of this entity set by the external source system or by a user, e.g. “Active”, “Inactive”, “Decommissioned”. |
| isActive | boolean | Indicates if this entity is currently active. |
| isPublic | boolean | Indicates if this entity is a public-facing resource (e.g. a public IP or public DNS record) or if the entity is publicly accessible (e.g. a subnet that has a gateway and route to the Internet). Default is false. |
| isValidated | boolean | Indicates if this entity has been validated as known/valid. |
| isTemporary | boolean | Indicates if this entity is a temporary resource, such as an EC2 instance started by ECS, or a temp employee. |
| webLink | string | Hyperlink to the source. For example: https://console.aws.amazon.com/iam/home#/roles/Administrator |
| tag.\* | string | Named tags assigned to the entity (i.e., tag.NameTag, tag.AnotherTag). There could be multiple of these. |
| tags | string[] | An array of hashtags. |
| notes | string[] | User provided notes about this entity. |

### Class Specific Entity Properties

Each specific class of Entity also has its own defined properties. For example, a Person entity will have properties including firstName and lastName, while a Device entity may have properties such as hardwareVendor, hardwareModel, and hardwareSerial.

### Custom Properties

Entities can also have custom properties that are specific to the type of that entity, defined by the source system where the resource belongs to, or defined by the individual or team managing the resource.

### Defined Entities

Here is a list of reference entities defined by the JupiterOne Data Model, each with its own defined set of properties in addition to the shared common properties:

|  |  |
| --- | --- |
| Entity | Description |
| AccessKey | A key used to grant access, such as ssh-key, access-key, api-key/token, mfa-token/device, etc. Extends the Key entity class. |
| AccessPolicy | A policy for access control assigned to a Host, Role, User, UserGroup, or Service. |
| AccessRole | An access control role mapped to a Principal (e.g. user, group, or service). |
| Account | An organizational account for a service or a set of services (e.g. AWS, Okta, Bitbucket Team, Google G-Suite account, Apple Developer Account). Each Account should be connected to a Service. |
| Application | A software product or application. |
| Assessment | An object to represent an assessment, including both compliance assessment such as a HIPAA Risk Assessment or a technical assessment such as a Penetration Testing. Each assessment should have findings (e.g. Vulnerability or Risk) associated. |
| Certificate | A digital Certificate such as an SSL or S/MIME certificate. |
| Cluster | A cluster of compute or database resources/workloads. |
| CodeCommit | A code commit to a repo. The commit id is captured in the \_id property of the Entity. |
| CodeDeploy | A code deploy job. |
| CodeModule | An application code module/library. Such as an npm-module or java-library. |
| CodeRepo | A source code repository. A CodeRepo is also a DataRepository therefore should carry all the required properties of DataRepository. |
| CodeReview | A code review (in git, a pull request). A CodeReview is also an official Record. |
| Control | A security or IT Control. This is most likely an additional Class applied to a Service (e.g. Okta SSO), a Device (e.g. a physical firewall), or a HostAgent (e.g. Carbon Black CbDefense Agent). |
| ControlPolicy | An operational or configuration compliance policy with technical specifications / rules that governs (i.e. enforces, evaluates, or monitors) a security control or IT system. |
| CryptoKey | A key used to perform cryptographic functions, such as an encryption key. Extends the Key entity class. |
| DataObject | An individual data object, such as an aws-s3-object, sharepoint-document, source-code, or a file (on disk). The exact data type is described in the \_type property of the Entity. |
| DataStore | A virtual repository where data is stored, such as aws-s3-bucket, aws-rds-cluster, aws-dynamodb-table, bitbucket-repo, sharepoint-site, docker-registry. The exact type is described in the \_type property of the Entity. |
| Database | A database cluster/instance. |
| Device | A physical device or media, such as a server, laptop, workstation, smartphone, tablet, router, firewall, switch, wifi-access-point, usb-drive, etc. The exact data type is described in the \_type property of the Entity. |
| Firewall | A piece of hardware or software that protects a network/host/application. |
| Framework | An object to represent a standard compliance or technical security framework. |
| Function | A virtual application function. For example, an aws\_lambda\_function, azure\_function, or google\_cloud\_function |
| Gateway | A gateway/proxy that can be a system/appliance or software service, such as a network router or application gateway. |
| Group | A defined, generic group of Entities. This could represent a group of Resources, Users, Workloads, DataRepositories, etc. |
| Host | A system that has an operating system running on it. The exact host type is described in the \_type property of the Entity. The UUID of the host should be captured in the \_id property of the Entity. |
| HostAgent | A software agent or sensor that runs on a host/endpoint. |
| Incident | An operational or security incident. |
| IpAddress | An re-assignable IpAddress resource entity. Do not create an entity for an IP Address *configured* on a Host. Use this only if the IP Address is a reusable resource, such as an Elastic IP Address object in AWS. |
| Key | An ssh-key, access-key, api-key/token, pgp-key, etc. |
| Network | A network, such as an AWS VPC, subnet, or VLAN. |
| NetworkInterface | An re-assignable software defined network interface resource entity. Do not create an entity for a network interface *configured* on a Host. Use this only if the network interface is a reusable resource, such as an Elastic Network Interface object in AWS. |
| Organization | An organization, such as a company (e.g. Amazon) or a business unit (e.g. HR). An organization can be internal or external. Note that there is a more specific Vendor class. |
| PasswordPolicy | A password policy is a specific ControlPolicy. It is separately defined because of its pervasive usage across digital environments and the well known properties (such as length and complexity) unique to a password policy. |
| Person | An entity that represents an actual person, such as an employee of an organization. |
| Policy | A written policy documentation. |
| Procedure | A written procedure and control documentation. A Procedure typically IMPLEMENTS a parent Policy. An actual Control further IMPLEMENTS a Procedure. |
| Project | A software development project. Can be used for other generic projects as well but the defined properties are geared towards software development projects. |
| Record | An official record (e.g. Risk), or a written document (e.g. Policy/Procedure), or a reference entity (e.g. Vulnerability/Weakness). This could be a jira-issue, a bitbucket-pull-request, a jenkins-build, a policy document, a vulnerability finding, a documented risk, or even a paper-record. The exact record type is captured in the \_type property of the Entity. |
| Resource | A generic assignable resource. A resource is typically non-functional by itself unless used by or attached to a host or workload. |
| Risk | An object that represents an identified Risk as the result of an Assessment. The collection of Risk objects in JupiterOne make up the Risk Register. A Control may have a MITIGATES relationship to a Risk. |
| Service | A service provided by a vendor. |
| Site | The physical location of an organization. A Person (i.e. employee) would typically has a relationship to a Site (i.e. located\_at or work\_at). Also used as the abstract reference to AWS Regions. |
| Team | A team consists of multiple member Person entities. For example, the Development team or the Security team. |
| User | A user account/login to access certain systems and/or services. Examples include okta-user, aws-iam-user, ssh-user, local-user (on a host), etc. |
| UserGroup | A user group, typically associated with some type of access control, such as a group in Okta or in Office365. If a UserGroup has an access policy attached, and all member Users of the UserGroup would inherit the policy. |
| Vendor | An external organization that is a vendor or service provider. |
| Vulnerability | A security vulnerability finding (application or system or infrastructure, e.g. a CVE). A single vulnerability may impact multiple resources. The IMPACTS relationship between the Vulnerability and the resource entity that was impacted serves as the record of the finding. The IMPACTS relationship carries properties such as ‘identifiedOn’, ‘remediatedOn’, ‘remediationDueOn’, ‘issueLink’, etc. |
| Weakness | A security weakness (e.g. CWE). |
| Workload | A virtual compute instance, it could be an aws-ec2-instance, a docker-container, an aws-lambda-function, an application-process, or a vmware-instance. The exact workload type is described in the \_type property of the Entity. |

#### Special Entities

There are three special entities defined. These are singleton entities.

|  |  |
| --- | --- |
| Entity | Description |
| Everyone | The global UserGroup that represents “everyone” publicly. |
| Internet | The Internet - ‘0.0.0.0/0’. |
| Root | The entity that represents the top level organization. |

## Relationships

A **Relationship** is the edge between two Entity nodes in the graph. The \_class of the relationship should be, in most cases, a generic descriptive verb, such as HAS or IMPLEMENTS.

Relationships can also carry their own properties. Relationships have the same metadata properties as entities, which are managed by the integration providers.

### Example defined Relationships between abstract Entity Classes

#### HAS / CONTAINS

Account -- HAS -> User  
Account -- HAS -> UserGroup  
Account -- HAS -> AccessRole  
Account -- HAS -> Resource  
CodeRepo -- HAS -> Vulnerability  
Host -- HAS -> Vulnerability  
Organization -- HAS -> Site  
Organization -- HAS -> Organization (e.g. a business unit)  
Application -- HAS -> Vulnerability  
CodeRepo -- HAS -> Vulnerability  
Host -- HAS -> Vulnerability  
Service -- HAS -> Vulnerability  
Site -- HAS -> Network  
Site -- HAS -> Site  
UserGroup -- HAS -> User  
Network -- CONTAINS -> Host  
Network -- CONTAINS -> Database  
Network -- CONTAINS -> Network (e.g. a subnet)

#### IS / OWNS

User -- IS -> Person  
Vulnerability -- IS -> Vulnerability (e.g. a Snyk Vuln IS a CVE)  
Person -- OWNS -> Device

#### EXPLOITS / IMPACTS

Vulnerability -- EXPLOITS -> Weakness  
Vulnerability -- IMPACTS -> CodeRepo | Application

#### USES

Host -- USES -> Resource (e.g. aws\_ec2\_instance USES aws\_ec2\_volume)

#### CONNECTS / TRIGGERS / EXTENDS

Application -- CONNECTS -> Account  
Gateway -- CONNECTS -> Network  
Gateway -- TRIGGERS -> Function  
HOST -- EXTENDS -> Resource

#### IMPLEMENTS / MITIGATES

Procedure -- IMPLEMENTS -> Policy  
Control -- IMPLEMENTS -> Policy  
Control -- MITIGATES -> Risk

#### MANAGES

Person -- MANAGES -> Person  
Person -- MANAGES -> Organization  
Person -- MANAGES -> Team  
User -- MANAGES -> Account  
User -- MANAGES -> UserGroup  
ControlPolicy -- MANAGES -> Control  
AccessPolicy -- MANAGES -> AccessRole

#### EVALUATES / MONITORS / PROTECTS

ControlPolicy -- EVALUATES -> <any entity>  
HostAgent -- MONITORS -> Host  
HostAgent -- PROTECTS -> Host

#### TRUSTS

AccessRole -- TRUSTS -> AccessRole  
AccessRole -- TRUSTS -> Service  
AccessRole -- TRUSTS -> Account

#### ASSIGNED

User -- ASSIGNED -> Application  
User -- ASSIGNED -> AccessRole  
UserGroup -- ASSIGNED -> AccessRole

#### IDENTIFIED / PERFORMED

Person -- PERFORMED -> Assessment  
Assessment -- IDENTIFIED -> Risk  
Assessment -- IDENTIFIED -> Vulnerability

#### PROVIDES

Vendor -- PROVIDES -> Service

#### CONTRIBUTES\_TO

User -- CONTRIBUTES\_TO -> CodeRepo

#### OPENED

User -- OPENED -> CodeReview (i.e. PR)

#### DEPLOYED\_TO

CodeRepo -- DEPLOYED\_TO -> Account  
CodeRepo -- DEPLOYED\_TO -> Host  
CodeRepo -- DEPLOYED\_TO -> Container  
CodeRepo -- DEPLOYED\_TO -> Function