freshworks

Levelling up app storage with **Entity Storage**

How to store complex, queryable data in apps
without managing external infrastructure.

Harish Janjam *(Product Manager, Freshworks)*
Kaustav Das Modak *(Developer Advocate, Freshworks)*

freshworks

Introduction

What is Entity Storage for Freshworks apps?

Work with custom business objects

```
// entity: "crew"
{
  "name": "Geordi La Forge",
  "affiliation": "Federation",
  "organization": "Starfleet",
  "is_active": true,
  "designation": "Chief Engineer",
  "ship_registry": "NCC-1701-E"
}
```

```
// entity: "ships"
{
  "registry": "NCC-1701-E",
  "class": "Sovereign",
  "owner": "Federation",
  "operator": "Starfleet",
  "launch_date": 49027.5
}
```

…directly in Freshworks apps without managing a database.

Schema-based, queryable object store

Entity Schema

- Define entity in JSON
- Multiple entities per app
- Data-types for fields

./config/entities.json

Runtime capabilities

- Get entity schema
- Create records
- Query records
- Update records
- Delete records

No external database infrastructure needed

Runtime API

- .schema()
- .create()
- .get()
- .getAll()
- .update()
- .delete()

Available for frontend and serverless apps

Common use cases

KYC management

Store KYC document details in the app.[+]

[+] Pair with a file-uploading service

Data synchronization

Full or half-duplex data sync with external systems.

Reduce number of API calls required.

Catalog management

Store and retrieve catalog & inventory information for ITSM product.

Reduce API usage cost.

freshworks

Comparison with storage solutions

How does Entity Storage compare to other data storage options that you can use in a Freshworks app?

Comparison with key-value storage

Choosing between data-storage options offered by the Freshworks Platform

Key-Value Storage

- Unstructured data store
- String keys with JSON values
- Only key-based lookup
- Max key + value size: 8 KB
- Available on all plans
- TTL, `setIf`, increment values
- Available for all plans and products

Entity Storage

- Structured data store
- Entities can have fields of different types
- Rich queries
- Max record size: 100 KB*
- Currently available on limited plans*
- For Freshdesk, Freshservice, Freshsales+

** More on these later in the Limitations section*
+ Currently supported products

Comparison with SQL databases

When your app needs to store complex data

SQL databases*

- Hosted externally
- Additional setup & maintenance
- Table and rows
- SQL as the command and query language
- Relational data

** Non-exhaustive comparison*

Entity Storage✧

- Available as a Platform feature
- No external infrastructure required
- Entities and records
- Runtime CRUD APIs
- Not relational yet

✧ *Entity limits and other rate-limits apply*

freshworks

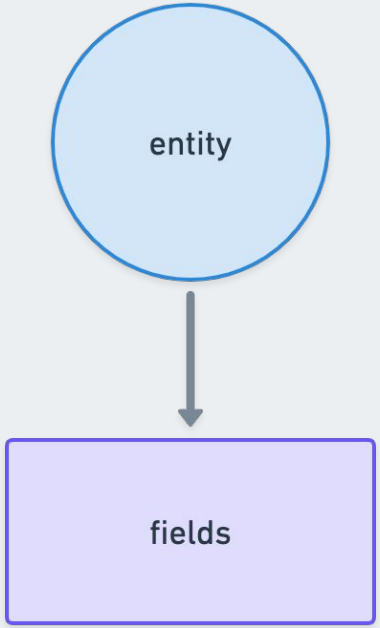
How to use it

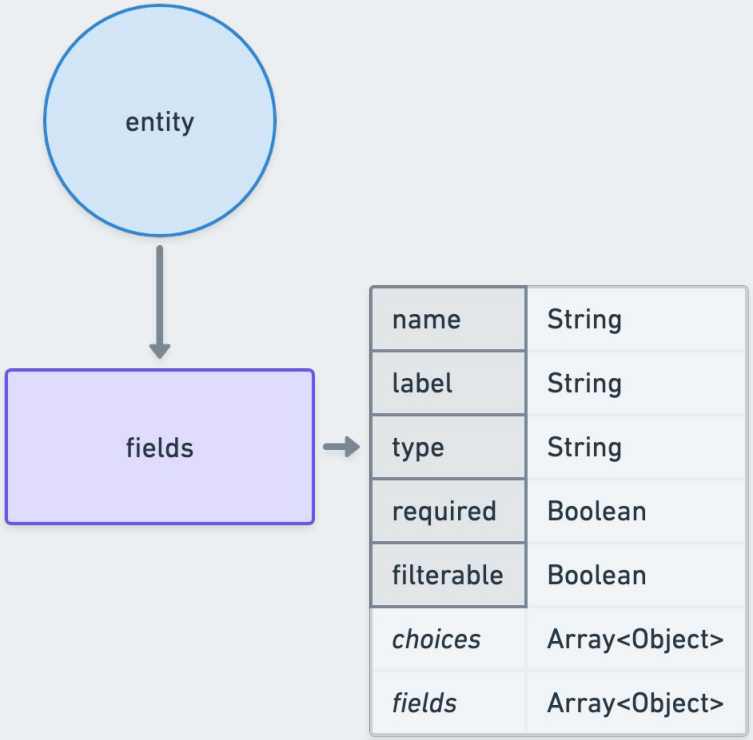
Define entity schema and use them from app code

freshworks

Entity schema

Defined in ./config/entities.json





Anatomy of an entity schema

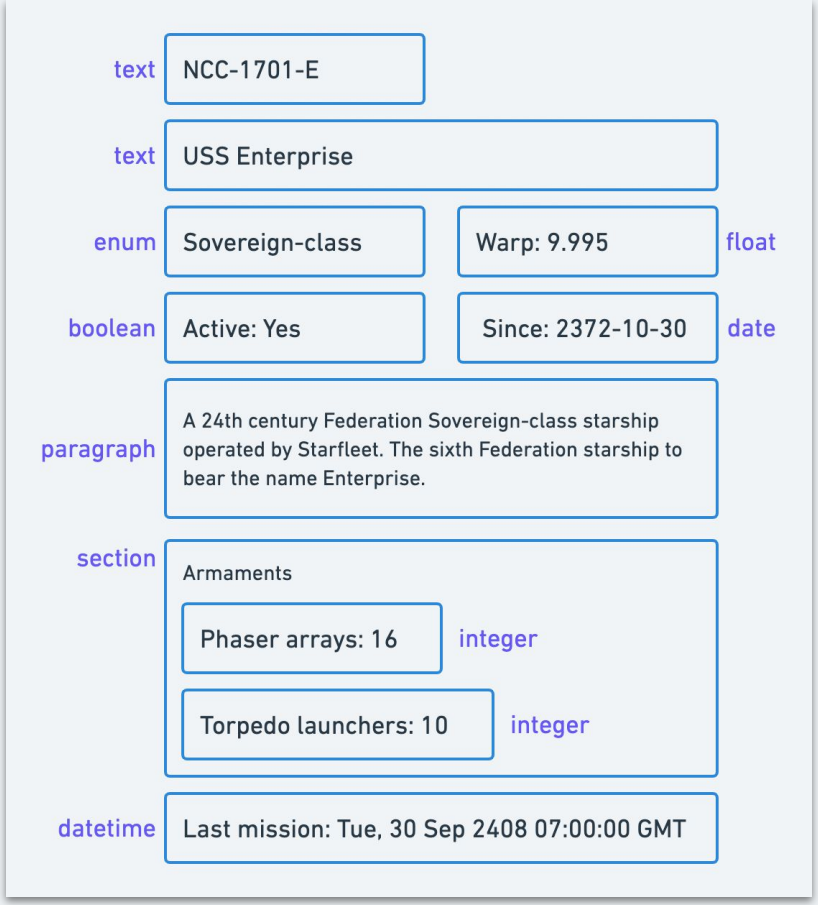
```
{
  "ships": {
    "fields": [
      {
        "name": "registry",
        "label": "Registry",
        "type": "text",
        "required": true,
        "filterable": true
      },
      {…
      },
      {
        "name": "class",
        "label": "Starship class",
        "type": "enum",
        "required": true,
        "filterable": true,
        "choices": [
          { "id": 1, "value": "Galaxy" },
          { "id": 2, "value": "Sovereign" },
          { "id": 3, "value": "Cruiser" },
          { "id": 4, "value": "Freighter" },
          { "id": 5, "value": "Warbird" },
          { "id": 404, "value": "Unknown"}
        ]
      }
    ]
  }
}
```

An example ./config/entities.json file

freshworks

Types of fields

```
{
  "ships": {
    "fields": [
      {
        "name": "registry",
        "label": "Registry",
        "type": "text",
        "required": true,
        "filterable": true
      }
    ]
  }
}
```



text	<= 64 characters
paragraph	<= 2048 characters
integer	whole numbers
float	decimal numbers
enum	multiple options
boolean	true or false
date	YYYY-MM-DD
datetime	ISO-8601 date format

section	group of fields

List of field types

text: NCC-1701-E

text: USS Enterprise

enum: Sovereign-class

Warp: 9.995 — float

boolean: Active: Yes

Since: 2372-10-30 — date

paragraph: A 24th century Federation Sovereign-class starship operated by Starfleet. The sixth Federation starship to bear the name Enterprise.

section: Armaments

- Phaser arrays: 16 — integer
- Torpedo launchers: 10 — integer

datetime: Last mission: Tue, 30 Sep 2408 07:00:00 GMT

text	<= 64 characters
paragraph	<= 2048 characters
integer	whole numbers
float	decimal numbers
enum	multiple options
boolean	true or false
date	YYYY-MM-DD
datetime	ISO-8601 date format

section	group of fields

filterable

List of field types that are filterable

text: NCC-1701-E

text: USS Enterprise

enum: Sovereign-class

Warp: 9.995 — float

boolean: Active: Yes

Since: 2372-10-30 — date

paragraph: A 24th century Federation Sovereign-class starship operated by Starfleet. The sixth Federation starship to bear the name Enterprise.

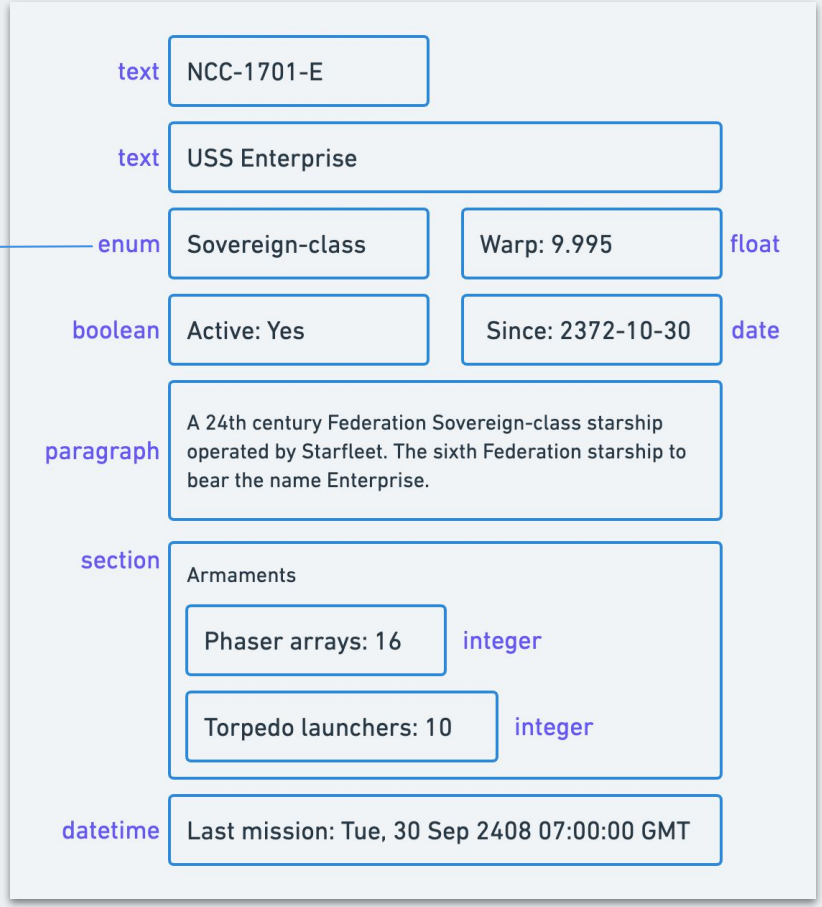
section: Armaments

Phaser arrays: 16 — integer

Torpedo launchers: 10 — integer

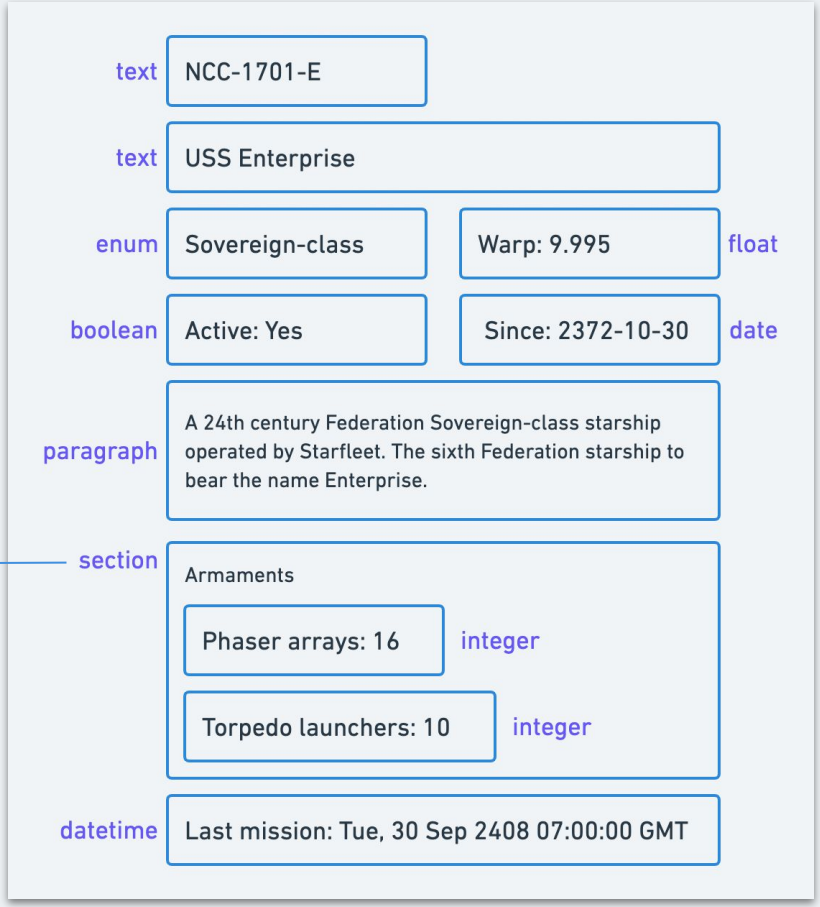
datetime: Last mission: Tue, 30 Sep 2408 07:00:00 GMT

```
{
  "name": "class",
  "label": "Starship class",
  "type": "enum",
  "required": true,
  "filterable": true,
  "choices": [
    { "id": 1, "value": "Galaxy" },
    { "id": 2, "value": "Sovereign" },
    { "id": 3, "value": "Cruiser" },
    { "id": 4, "value": "Freighter" },
    { "id": 5, "value": "Warbird" },
    { "id": 404, "value": "Unknown"}
  ]
},
```



“type”= “enum”

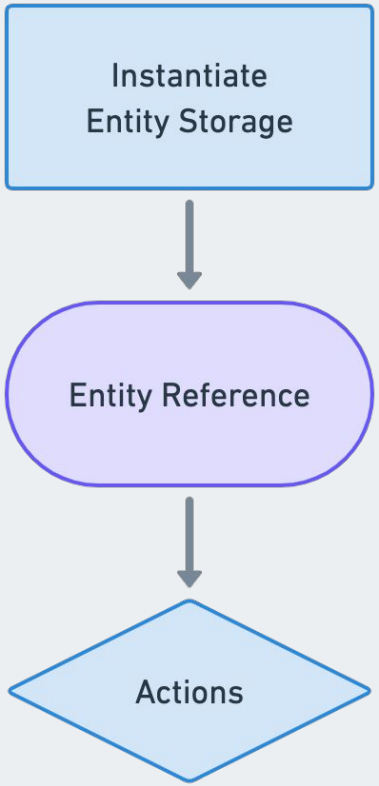
```
{
  "name": "armaments",
  "label": "Armaments",
  "type": "section",
  "fields": [
    {
      "name": "num_phaser_arrays",
      "label": "Phaser arrays",
      "type": "integer"
    },
    {
      "name": "num_torpedo_launchers",
      "label": "Torpedo launchers",
      "type": "integer"
    }
  ]
},
```



"type"= "section"

freshworks

Entity Storage API



Instantiate Entity Storage and then perform actions on an entity reference

Get entity reference

Frontend

```
function prepareEntity(client) {
  const entity = client.db.entity({ version: "v1" });
  return entity.get("ships");
}
```

Serverless

```
function prepareEntity() {
  const entity = $db.entity({ version: "v1" });
  return entity.get("ships");
}
```

Get entity schema

```
// Get entity reference
const $entity = prepareEntity();

// Get entity schema
await $entity.schema();
```

Call .schema() on an entity reference

```
{
  "entity": {
    "id": 1,
    "name": "ships",
    "prefix": "ships",
    "fields": [
      {
        "name": "registry",
        "label": "Registry",
        "type": "text",
        "required": true,
        "filterable": true,
        "choices": []
      },
      {/* ... */ }
    ]
  }
}
```

Response

Create a record

```
const payload = {
  registry: "NCC-1701-E",
  name: "USS Enterprise",
  class: "Sovereign",
  warp: 9.995,
  is_active: true,
  launch_date: "2372-10-30",
  about: "A 24th Century[ ... ]",
  num_phaser_arrays: 16,
  num_torpedo_launchers: 10,
  last_mission_datetime: "2408-09-30T07:00:00.000Z",
};

await $entity.create(payload)
```


```
{
  "record": {
    "display_id": "ships-1",
    "created_time": "2022-05-22T16:45:04.599Z",
    "updated_time": "2022-05-22T16:45:04.599Z",
    "data": {
      "registry": "NCC-1701-E",
      "name": "USS Enterprise",
      "class": "Sovereign",
      "warp": 9.995,
      "is_active": true,
      "launch_date": "2372-10-30",
      "about": "A 24th Century[ ... ]",
      "num_phaser_arrays": 16,
      "num_torpedo_launchers": 10,
      "last_mission_datetime": "2408-09-30T07:00:00.000Z",
      "armaments": null
    }
  }
}
```


Response

Fetch a single record

```
·await $entity.get(record.display_id);
```

Pass record's display_id to .get()


```
{
··"record": {
····"display_id": "ships-1",
····"created_time": "2022-05-22T16:45:04.599Z",
····"updated_time": "2022-05-22T16:45:04.599Z",
····"data": {
·····"registry": "NCC-1701-E",
·····"name": "USS Enterprise",
·····"class": "Sovereign",
·····"warp": 9.995,
·····"is_active": true,
·····"launch_date": "2372-10-30",
·····"about": "A 24th Century[ ... ]",
·····"num_phaser_arrays": 16,
·····"num_torpedo_launchers": 10,
·····"last_mission_datetime": "2408-09-30T07:00:00.000Z",
·····"armaments": null
····}
··}
}
```


Response

Update a record

```
// Fetch a record
let { record } = await $entity.get(display_id);

// Change record fields
record.data.warp = 9.999;
record.data.num_phaser_arrays = 18;

// Update the record
await $entity.update(display_id, record.data);
```


```
{
  "record": {
    "display_id": "ships-1",
    "created_time": "2022-05-23T09:41:57.668Z",
    "updated_time": "2022-05-23T09:41:57.748Z",
    "data": {
      "registry": "NCC-1701-E",
      "name": "USS Enterprise",
      "class": "Sovereign",
      "warp": 9.999,
      "is_active": true,
      "launch_date": "2372-10-30",
      "about": "A 24th Century[ … ]",
      "armaments": null,
      "num_phaser_arrays": 18,
      "num_torpedo_launchers": 10,
      "last_mission_datetime": "2408-09-30T07:00:00.000Z"
    }
  }
}
```


Pass record's display_id and updated field values to .update()

Response

Query records

Search all records

```
// Returns first page of 100 records
let page1 = await $entity.getAll({});

// Fetch next page of records
let page2 = await $entity.getAll({
  next: page1.links.next
});
```

Pass an empty object to .getAll() for serverless
Don't pass an empty object for frontend


```
{
  "records": [
    {
      "display_id": "ships-2",
      "created_time": "2022-05-23T11:31:02.355Z",
      "updated_time": "2022-05-23T11:31:02.355Z",
      "data": {
        "registry": "NCC-1701-D",
        "name": "USS Enterprise",
        "class": "Galaxy",
        "warp": 9.5,
        "is_active": false,
        "launch_date": "2363-10-04",
        "about": "Starfleet's flagship [ … ]",
        "armaments": null,
        "num_phaser_arrays": 12,
        "num_torpedo_launchers": 2,
        "last_mission_datetime": "2463-10-01T07:00:00.000Z"
      }
    },
    // ← snip (99 other records) →
  ],
  "links": {
    "next": {
      "marker": "L1v3l0ng4ndPR0sp3crFj7NiigDlittVkGc7RmPjKF3"
    }
  }
}
```


Response

Query records by field value

```
// Fetch all starships of Galaxy class
let res = await $entity.getAll({
  query: {
    "class": "Galaxy"
  }
});
```

The `query` property of the options object takes in filters

Queries that satisfy either condition

```
// Query starships that are Sovereign or Cruisers
let res = await $entity.getAll({
  query: {
    $or: [
      { class: "Sovereign" },
      { class: "Cruiser" }
    ]
  }
});
```

`$or` takes an array of objects for each field and its value to look for

Queries that satisfy all conditions

```
// Constitution-class starships named "USS Enterprise"
let res = await $entity.getAll({
  query: {
    $and: [
      { name: "USS Enterprise" },
      { class: "Constitution" }
    ]
  }
});
```

`$and` takes an array of objects for each field and its value to look for, similar to the `$or` operator

freshworks

Limitations

Technical limitations and pricing plan-based limitations

Technical Limitations

Number of entities per app	5
Number of fields per entity	20
Number of filterable fields per entity	5
Number of records per entity	Unlimited
Usage Rate Limit	50 concurrent invocations per app
Max Size of each record	100kb

Marketplace Pricing Plan Limitations

Developers

- To build and publish apps, you need to be on Pro Trial account or above.
- This is the default trial plan when you sign up

Customers

- Public Apps: Installable only by customers in Enterprise (Erstwhile Forest) Plan and above
- Private Apps: Can be uploaded only by customers in Enterprise plan or Pro Trial plans

Resources

Learn about Entity Storage in detail

Documentation

- Entity Storage docs for Freshdesk
- Entity Storage docs for Freshservice
- Entity Storage docs for Freshsales

Code samples

- Custom Todos (Freshdesk)
- Freshfoods (Freshdesk)
- Employee Sync (Freshservice)
- Contact KYC (Freshsales)

freshworks

Open discussion

Share your thoughts about Entity Storage and how you use plan to use the feature.

Or write in **community.freshworks.dev**