When you create a new Firestore database, you can configure the database instance to run in *Datastore mode* which makes the database backwards-compatible with Datastore. This page helps you understand the difference between the two Firestore database modes: **Native mode** and **Datastore mode**.

Firestore is the next major version of Datastore and a re-branding of the product. Taking the best of Datastore and the Firebase Realtime Database (https://firebase.google.com/docs/database/rtdb-vs-firestore), Firestore is a NoSQL document database built for automatic scaling, high performance, and ease of application development.

Firestore introduces new features such as:

- A new, strongly consistent storage layer
- A collection and document data model
- Real-time updates
- Mobile and Web client libraries

Firestore is backwards compatible with Datastore, but the new data model, real-time updates, and mobile and web client library features are not. To access all of the new Firestore features, you must use Firestore in Native mode.

Firestore in Datastore mode uses Datastore system behavior but accesses Firestore's storage layer, removing the following Datastore limitations:

- Eventual consistency, all Datastore queries become strongly consistent.
- Transactions are no longer limited to 25 entity groups.

- Writes to an entity group are no longer limited to 1 per second.

Datastore mode disables Firestore features that are not compatible with Datastore:

- The project will accept Datastore API requests and deny Firestore API requests.
- The project will use Datastore indexes instead of Firestore indexes.
- You can use Datastore client libraries with this project but not Firestore client libraries.
- Firestore real-time capabilities will not be available.
- In the Cloud Console, the database will use the Datastore viewer.

Existing Datastore databases will be automatically upgraded to Firestore in Datastore mode (/datastore/docs/upgrade-to-firestore). New projects that require a Datastore database should use Firestore in Datastore mode.

Native mode and Datastore mode databases use the same pricing structure and are available in the same locations. Pricing and locations are described in detail in the following pages:

- Firestore pricing (/firestore/pricing)
- Firestore locations (/firestore/docs/locations)

- Firestore in Datastore mode pricing (/datastore/pricing)
- Firestore in Datastore mode locations (/datastore/docs/locations)

When you create a new Firestore database, you must select a database mode. You cannot use both Native mode and Datastore mode in the same project. We recommend the following when choosing a database mode:

- **Use Firestore in Datastore mode for new server projects.**

  Firestore in Datastore mode allows you to use established Datastore server architectures while removing fundamental Datastore limitations. Datastore mode can automatically scale to millions of writes per second.

  ⚠ **Warning:** Once you select Firestore in Datastore mode and execute the first write on the database, you cannot switch to Firestore in Native mode.

- **Use Firestore in Native mode for new mobile and web apps.**

  Firestore offers mobile and web client libraries with real-time and offline features. Native mode can automatically scale to millions of concurrent clients.

  ⚠ **Warning:** Once you select Firestore in Native mode, you cannot switch to Firestore in Datastore mode.

The following table compares the system behavior of the database modes:

	Firestore Native Mode	Firestore Datastore Mode
Data model	Document database organized into documents and collections.	Entities organized into kinds and entity groups.
Storage Layer	New storage layer that is always strongly consistent	New storage layer that is always strongly consistent
Queries and transactions	• Strongly consistent queries across the entire database • Up to 500 documents per transaction across any number of collections.	• Removes the previous consistency limitations of Datastore • Strongly consistent queries across the entire database

	• **Limitation**: No projection queries.	• Transactions can access any number of entity groups
Datastore v1 API (/datastore/docs/reference/data/rest/) **support**	No, requests are denied	Yes
Firestore v1 API (/firestore/docs/reference/rest/) **support**	Yes	No, requests are denied
Real-time updates	Supports the ability to *listen* to a document or a set of documents for real-time updates. (/firestore/docs/query-data/listen) While listening to a document or set of documents, your clients are notified of any data changes and sent the newest set of data.	Not supported
Offline data persistence	The mobile and web client libraries support offline data persistence. (/firestore/docs/manage-data/enable-offline)	Not supported
Client libraries	Firestore client libraries: • Java • Python • PHP • Go • Ruby • C# • Node.js • Android • iOS • Web	Datastore Client libraries: • Java • Python • PHP • Go • Ruby • C# • Node.js
Security	• Cloud Identity and Access Management (IAM) manages	Cloud Identity and Access Management (IAM) manages

	database access • Firestore Security Rules support serverless authentication and authorization for the mobile and web client libraries	database access
Performance	Automatically scales to millions of concurrent clients. Max 10,000 writes per second.	Automatically scales to millions of writes per second.
SLA	Firestore SLA. (/firestore/sla)	Firestore SLA. (/firestore/sla)
Locations	• US (Multi-region) • Europe (Multi-region) • Montréal • Los Angeles • South Carolina • Northern Virginia • São Paulo • London • Frankfurt • Tokyo • Mumbai • Sydney	• US (Multi-region) • Europe (Multi-region) • Montréal • Los Angeles • South Carolina • Northern Virginia • São Paulo • London • Frankfurt • Tokyo • Mumbai • Sydney
Pricing	Same pricing structure	
Console	Firebase Console and Cloud Console Firestore Viewer	Cloud Console Datastore Viewer
Namespaces	Not supported	Namespaces supported
App Engine client library integration	Not supported in the App Engine standard environment Python 2.7 and PHP 5.5 runtimes Supported in the App Engine standard environment Python	Supported in all runtimes

	3.7, PHP 7.2, Java 8, Go, and Node.js runtimes Supported in the App Engine flexible environment, all runtimes