Informix Product Family
Informix
Version 12.10

IBM Informix JSON Compatibility Guide

SC27-5556-00

Informix Product Family
Informix
Version 12.10

IBM Informix JSON Compatibility Guide

SC27-5556-00

Note
Before using this information and the product it supports, read the information in "Notices" on page B-1.

Contents

Introduction

This introduction provides an overview of the information in this publication and describes the conventions that this publication uses.

About This Publication

This publication contains information about using the IBM® Informix® JSON capability.

This section discusses the intended audience for this publication and the associated software products that you must have to use the administrative utilities.

Types of Users

This publication is written for the following users:

- Database administrators
- System administrators
- Performance engineers

This publication is written with the assumption that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with database server administration, operating-system administration, or network administration

You can access the Informix information centers, as well as other technical information such as technotes, white papers, and IBM Redbooks® publications online at http://www.ibm.com/software/data/sw-library/.

Software Dependencies

This publication is written with the assumption that you are using IBM Informix or IBM Informix Dynamic Server with J/Foundation, Version 12.10, as your database server.

Assumptions about your locale

IBM Informix products can support many languages, cultures, and code sets. All the information related to character set, collation and representation of numeric data, currency, date, and time that is used by a language within a given territory and encoding is brought together in a single environment, called a Global Language Support (GLS) locale.

The IBM Informix OLE DB Provider follows the ISO string formats for date, time, and money, as defined by the Microsoft OLE DB standards. You can override that default by setting an Informix environment variable or registry entry, such as **DBDATE**.

If you use Simple Network Management Protocol (SNMP) in your Informix environment, note that the protocols (SNMPv1 and SNMPv2) recognize only English code sets. For more information, see the topic about GLS and SNMP in the *IBM Informix SNMP Subagent Guide*.

The examples in this publication are written with the assumption that you are using one of these locales: en_us.8859-1 (ISO 8859-1) on UNIX platforms or en_us.1252 (Microsoft 1252) in Windows environments. These locales support U.S. English format conventions for displaying and entering date, time, number, and currency values. They also support the ISO 8859-1 code set (on UNIX and Linux) or the Microsoft 1252 code set (on Windows), which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

You can specify another locale if you plan to use characters from other locales in your data or your SQL identifiers, or if you want to conform to other collation rules for character data.

For instructions about how to specify locales, additional syntax, and other considerations related to GLS locales, see the *IBM Informix GLS User's Guide*.

Demonstration databases

The DB-Access utility, which is provided with your IBM Informix database server products, includes one or more of the following demonstration databases:

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in IBM Informix publications are based on the **stores_demo** database.
- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines.

For information about how to create and populate the demonstration databases, see the *IBM Informix DB-Access User's Guide*. For descriptions of the databases and their contents, see the *IBM Informix Guide to SQL: Reference*.

The scripts that you use to install the demonstration databases are in the `$INFORMIXDIR/bin` directory on UNIX platforms and in the `%INFORMIXDIR%\bin` directory in Windows environments.

Example code conventions

Examples of SQL code occur throughout this publication. Except as noted, the code is not specific to any single IBM Informix application development tool.

If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
   WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement. If you are using DB–Access, you must delimit multiple statements with semicolons.

Tip: Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept that is being discussed.

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the documentation for your product.

Additional documentation

Documentation about this release of IBM Informix products is available in various formats.

You can access Informix technical information such as information centers, technotes, white papers, and IBM Redbooks publications online at http://www.ibm.com/software/data/sw-library/.

Compliance with industry standards

IBM Informix products are compliant with various standards.

IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

Syntax diagrams

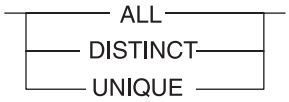
Syntax diagrams use special components to describe the syntax for statements and commands.

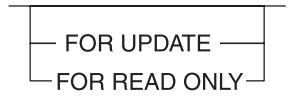
Table 1. Syntax Diagram Components

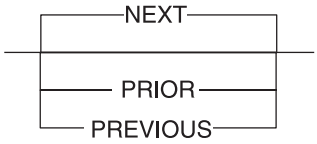
Component represented in PDF	Component represented in HTML	Meaning
	>>----------	Statement begins.
	---------->	Statement continues on next line.
	>----------	Statement continues from previous line.
	----------><	Statement ends.
	--------SELECT----------	Required item.
	--+----------+--- '------LOCAL------'	Optional item.

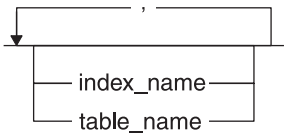
Table 1. Syntax Diagram Components (continued)

Component represented in PDF	Component represented in HTML	Meaning

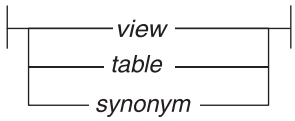
	```---+-----ALL-------+---
	  +--DISTINCT-----+
	  '---UNIQUE------'```
		Required item with choice. Only one item must be present.

	```---+------------------+---
	  +--FOR UPDATE-----+
	  '--FOR READ ONLY--'```
		Optional items with choice are shown below the main line, one of which you might specify.

	``` .---NEXT---------.
	----+----------------+---
	  +---PRIOR--------+
	  '---PREVIOUS-----'```
		The values below the main line are optional, one of which you might specify. If you do not specify an item, the value above the line is used by default.

	``` .-------,-----------.
	  V |
	---+-----------------+---
	  +---index_name---+
	  '---table_name---'```
		Optional items. Several items are allowed; a comma must precede each repetition.

	```>>-| Table Reference |-><```	Reference to a syntax segment.

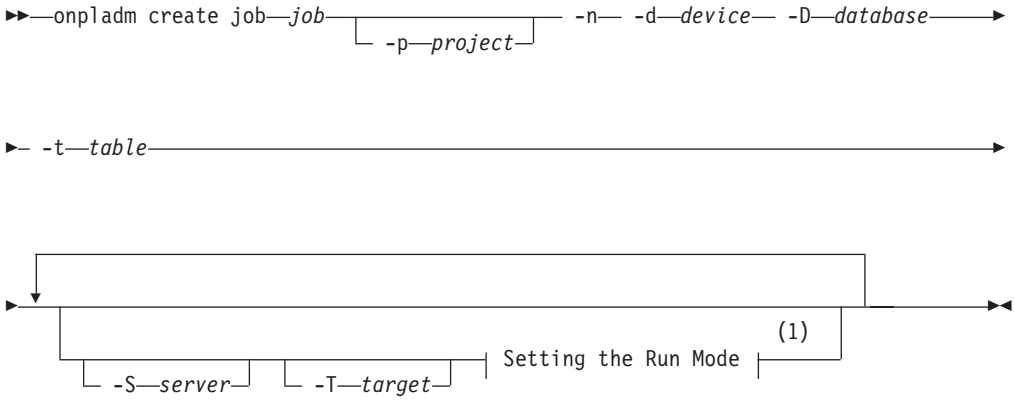
Table Reference
 
	``Table Reference
	|--+-----view--------+--|
	  +------table------+
	  '----synonym------'```
		Syntax segment.

How to read a command-line syntax diagram

Command-line syntax diagrams use similar elements to those of other syntax diagrams.

Some of the elements are listed in the table in Syntax Diagrams.

Creating a no-conversion job

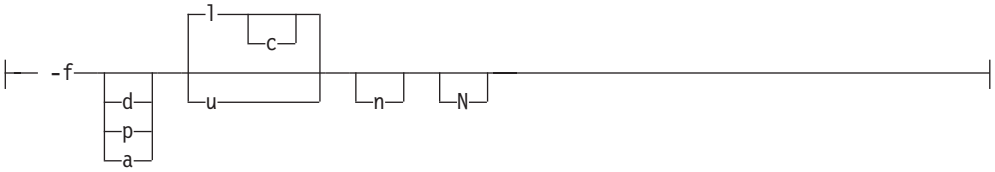


Notes:

1 See page Z-1

This diagram has a segment that is named "Setting the Run Mode," which according to the diagram footnote is on page Z-1. If this was an actual cross-reference, you would find this segment on the first page of Appendix Z. Instead, this segment is shown in the following segment diagram. Notice that the diagram uses segment start and end components.

Setting the run mode:



To see how to construct a command correctly, start at the upper left of the main diagram. Follow the diagram to the right, including the elements that you want. The elements in this diagram are case-sensitive because they illustrate utility syntax. Other types of syntax, such as SQL, are not case-sensitive.

The Creating a No-Conversion Job diagram illustrates the following steps:

1. Include **onpladm create job** and then the name of the job.
2. Optionally, include **-p** and then the name of the project.
3. Include the following required elements:
   - **-n**
   - **-d** and the name of the device
   - **-D** and the name of the database
   - **-t** and the name of the table
4. Optionally, you can include one or more of the following elements and repeat them an arbitrary number of times:
   - **-S** and the server name
   - **-T** and the target server name
   - The run mode. To set the run mode, follow the Setting the Run Mode segment diagram to include **-f**, optionally include **d**, **p**, or **a**, and then optionally include **l** or **u**.
5. Follow the diagram to the terminator.

Keywords and punctuation

Keywords are words that are reserved for statements and all commands except system-level commands.

A keyword in a syntax diagram is shown in uppercase letters. When you use a keyword in a command, you can write it in uppercase or lowercase letters, but you must spell the keyword exactly as it appears in the syntax diagram.

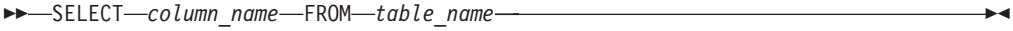
You must also use any punctuation in your statements and commands exactly as shown in the syntax diagrams.

Identifiers and names

Variables serve as placeholders for identifiers and names in the syntax diagrams and examples.

You can replace a variable with an arbitrary name, identifier, or literal, depending on the context. Variables are also used to represent complex syntax elements that are expanded in other syntax diagrams. A variable in a syntax diagram, an example, or text, is shown in *lowercase italic*.

The following syntax diagram uses variables to illustrate the general form of a simple SELECT statement.



When you write a SELECT statement of this form, you replace the variables *column_name* and *table_name* with the name of a specific column and table.

How to provide documentation feedback

You are encouraged to send your comments about IBM Informix user documentation.

Use one of the following methods:

- Send email to docinf@us.ibm.com.
- In the Informix information center, which is available online at http://www.ibm.com/software/data/sw-library/, open the topic that you want to comment on. Click the feedback link at the bottom of the page, complete the form, and submit your feedback.
- Add comments to topics directly in the information center and read comments that were added by other users. Share information about the product documentation, participate in discussions with other users, rate topics, and more!

Feedback from all methods is monitored by the team that maintains the user documentation. The feedback methods are reserved for reporting errors and omissions in the documentation. For immediate help with a technical problem, contact IBM Technical Support at http://www.ibm.com/planetwide/.

We appreciate your suggestions.

Chapter 1. About the Informix JSON compatibility

Applications that use the JSON-oriented query language, created by MongoDB, can interact with data stored in Informix databases. The Informix database server also provides built-in JSON and BSON (binary JSON) data types.

You can use MongoDB community drivers to insert, update, and query JSON documents in Informix.

Informix supports a command line processor, a Java™ API, and a JSON wire protocol listener (wire listener) to work with JSON documents.

You can enable dynamic scaling and high-availability for data-intensive applications by taking the following steps:

- Define a sharded cluster to easily add or remove servers as your requirements change.
- Use shard keys to distribute subsets of data across multiple servers in a sharded cluster.
- Query the correct servers in a sharded cluster and return the consolidated results to the client application.
- Use secondary servers (similar to slaves in MongoDB) in the sharded cluster to maximize availability and throughput. Secondary servers also have update capability.

MongoDB to Informix term mapping

The commonly used MongoDB terminology and concepts are mapped to the equivalent Informix terminology and concepts.

The following table provides a summary of commonly used MongoDB terms and their Informix conceptual equivalents.

Table 1-1. MongoDB concepts mapped to one or more Informix concepts.

MongoDB concept	Informix concept	Description
collection	table	This is the same concept. In Informix this type of collection is sometimes referred to as a JSON collection. A JSON collection is similar to a relational database table, except it does not enforce a schema.
document	record	This is the same concept. In Informix, this type of document is sometimes referred to as a JSON document.
field	column	This is the same concept.

Table 1-1. MongoDB concepts mapped to one or more Informix concepts. (continued)

MongoDB concept	Informix concept	Description
master / slave	primary server / secondary server	This is the same concept. However, an Informix secondary server has additional capabilities. For example, data on a secondary server can be updated and propagated to primary servers.
replica set	high-availability cluster	This is the same concept. However, when the replica set is updated, it then sent to all servers, not only the primary server.
sharded cluster	sharded cluster	This is the same concept. In Informix, a sharded cluster consists of servers that are sometimes referred to as shard servers.
shard key	shard key	This is the same concept.

Chapter 2. Using the wire listener for JSON compatibility

The wire listener is a mid-tier gateway server that enables communication between MongoDB applications and the Informix database server.

The wire listener interprets messages that are based on the MongoDB JSON wire protocol.

When you install Informix, the wire listener is automatically started and connected to the database server with the default operational instance. The wire listener is provided as a single JAR file.

The wire listener configuration file, `jsonListener.properties`, defines every operational characteristic. Default values are established for each property.

If a server instance is created as a part of the installation process:

- The wire listener configuration file `jsonListener.properties` is automatically created.
- The user **IFXJSON**, which has REPLICATION privilege group access, is created and added to the `jsonListener.properties` file. This user ID is used by the wire listener to connect to Informix.

If a server instance is created after the installation process:

- You must manually create the `jsonListener.properties` file with the url parameter specified. For example:

  ```
  url=jdbc:informix-sqli://myhost.ibm.com:9088/
  sysmaster:INFORMIXSERVER=myserver;USER=ifxjson;PASSWORD=mypassword
  ```

- If you plan to use the Informix sharding capability, you must add the user with REPLICATION privilege group access to the url parameter.

Starting the wire listener

You can start the wire listener by using the **start json listener** SQL administration API **task()** or **admin()** function, or the command line argument.

Before you begin

- The wire listener configuration file `jsonListener.properties` must exist. If a server instance is created as a part of the installation process, the `jsonListener.properties` is automatically created with default properties, otherwise you must manually create this file.
- If you are using SQL administration API **task()** or **admin()** function, you must be connected to the **sysadmin** database as user **informix** or another authorized user.
- You must use Java 1.6 or higher. Java 1.6 is delivered with the Informix installation.

Procedure

Use one of the following methods to start the wire listener for the current database server session:

- Run the SQL administration API **task()** or **admin()** function with the **start json listener** argument. For example:

  ```
  EXECUTE FUNCTION task("start json listener");
  ```

- Issue a statement from the command line:

  ```
  java -jar jsonListener.jar -start
  ```

Example

Here is an example of the output from starting the wire listener:

```
starting listener on port 27017
[main] INFO com.ibm.nosql.informix.server.MyJsonListener - JSON
       database server listening on port: 27017, 0.0.0.0/0.0.0.0
```

Related concepts:

Chapter 6, "Troubleshooting Informix JSON compatibility," on page 6-1

Related reference:

start json listener argument: Start the JSON wire listener (Administrator's Reference)

Stopping the wire listener

You can stop the wire listener by using the **stop json listener** argument with the SQL administration API **task()** or **admin()** function, or the command line argument.

Before you begin

- If you are using SQL administration API **task()** or **admin()** function, you must be connected to the **sysadmin** database as user **informix** or another authorized user.
- You must use Java 1.6 or higher. Java 1.6 is delivered with the Informix installation.

Procedure

Use one of the following methods to stop the wire listener for the current database server session:

- Run the SQL administration API **task()** or **admin()** function with the **stop json listener** argument. For example:

  ```
  EXECUTE FUNCTION task("stop json listener");
  ```

- Issue a statement from the command line:
  - If the wire listener is running from the default port 27017:

    ```
    java -jar jsonListener.jar -stop
    ```

    Without any options or configuration file specified, the stop command connects to the local wire listener instance on the default port (27017) and sends it the shutdown command.
  - If the wire listener is running on a non-default port, use either of these methods:
    - Specify the port flag when running the wire listener. For example, if the port flag is 27018:

      ```
      java -jar jsonListener.jar -stop -port 27018
      ```

    - Specify the configuration file used to start the wire listener. For example, if the configuration file is `myproperties.properties`:

```
java -jar jsonListener.jar -stop -config myproperties.properties
```

Related reference:

stop json listener: Stop the JSON wire listener (Administrator's Reference)

Modifying the wire listener properties

You can modify the wire listener properties in the `jsonListener.properties` file.

Before you begin

The wire listener properties are set by default in the `jsonListener.properties` file during installation. The `jsonListener.properties` file is in the `%INFORMIXDIR%\etc\` directory.

Procedure

To modify the wire listener properties:

1. Stop the wire listener.
2. Update the `jsonListener.properties` file.
3. Restart the wire listener.

The jsonListener.properties file

The properties that control the wire listener and the connection between the client and database server are set in the `%INFORMIXDIR%\etc\jsonListener.properties` file.

`listener.port`
This optional parameter specifies the port number to listen on for incoming connections from MongoDB clients. The default value is 27017.

`listener.pool.size.maximum`
This optional parameter specifies the maximum peak size of the thread/socket pool listening for incoming connections from MongoDB clients. The default value is 256.

`listener.pool.size.core`
This optional parameter specifies the maximum sustained size of the thread/socket pool listening for incoming connections from MongoDB clients. The default value is 128.

`url`
This required parameter specifies the URL, user name, and password of the Informix database server to connect to. You must specify the **sysmaster** database in the URL, which is used for administrative purposes by the wire listener. The default value is `jdbc:informix-sqli://localhost:9088/sysmaster:INFORMIXSERVER=ids;USER=informix;PASSWORD=informix`.

When a MongoDB client is connected to the wire listener and requests a connection to another database, the wire listener creates an appropriate internal connection.

Important: If you plan to use the Informix sharding capability, you must add the user with REPLICATION privilege group access to this parameter.

pool.type
This optional parameter specifies the type of pool to use for JDBC connections. The available pool types are:

none This pool type indicates that no thread pooling occurs. Use this type for debugging purposes.

basic This pool type indicates that thread pool maintenance of idle threads is run each time that a connection is returned.

advanced
This pool type indicates that thread pool maintenance is run by a separate thread.

perThread
This pool type indicates that each thread is allocated a connection for its exclusive use.

The default value is basic.

documentIdAlgorithm
This setting determines the algorithm that is used to generate the unique Informix identifier. The _id field of the document is used as the input to the algorithm. The available algorithms are:

ObjectId
The value is computed as follows: If the _id field is of type ObjectId, then the string representation of the ObjectId is used; otherwise, the MD5 algorithm is used to compute the hash of the contents of the _id field. The string representation of an ObjectId is the hexadecimal representation of the 12 bytes that comprise an ObjectId.

The MD5 algorithm provides better performance than the secure hashing algorithms (SHA) and is unique enough for most situations.

This is the default value.

SHA-1
Indicates that the SHA-1 hashing algorithm is used to derive an identifier from the _id field.

SHA-256
Indicates that the SHA-256 hashing algorithm is used to derive an identifier from the _id field.

SHA-512
Indicates that the SHA-512 hashing algorithm is used to derive an identifier from the _id field. This option generates the most unique values, but uses the most processor resources.

sharding.enable
This parameter indicates whether to enable the use of commands and queries on sharded data. The default value is true.

update.one.enable
This parameter indicates whether to enable support for updating a single document.

false
All updates are treated as multiple document updates. This is the default.

With the `update.one.enable=false` setting, the MongoDB **db.collection.update** multi parameter is ignored and all updates are

treated as multiple document updates. The performance of updates is improved with the `update.one.enable=false` setting.

`true`
Allow updates to a single document or multiple documents.

With the `update.one.enable=true` setting, the MongoDB **db.collection.update** multi parameter is accepted. The **db.collection.update** multi parameter controls whether you can update a single document or multiple documents.

Related concepts:

Chapter 6, “Troubleshooting Informix JSON compatibility,” on page 6-1

Related reference:

“Collection commands” on page 4-1

Chapter 3. JSON data sharding

IBM Informix can horizontally partition a collection across multiple database servers. Documents from a collection can be distributed across a cluster of database servers, reducing the number of documents and the size of the index for the database of each server. When you distribute data across database servers, you also distribute performance across hardware, which can result in significant performance improvements. As your database grows in size, you can scale up by adding more database servers.

Horizontal partitioning is also known as *sharding*. The database servers that receive sharded data are *shard servers*, and a cluster of shard servers is a *shard cluster*.

Documents that are inserted on a shard server can be copied to other shard servers in a shard cluster. Queries that are performed on a shard server can select data from other shard servers in a shard cluster. When data is sharded based on a key that specifies certain segmentation characteristics, queries can potentially skip shard servers that do not contain relevant data. This query optimization is another benefit that comes from data sharding.

To use Informix sharding capability, you must complete the following steps:

1. Add existing database servers to a shard cluster. You can create a cluster of shard servers by using MongoDB commands or IBM Informix commands.
2. Define a schema for sharding data. You can create a definition by using MongoDB commands or IBM Informix commands.

Enabling sharding for JSON data

You can enable support for sharding of JSON data.

Before you begin

Verify that the user of the JSON wire listener has REPLICATION privilege group access in the SQL Admin API. If a database server instance is created as a part of the installation process, the user **IFXJSON** is created and added to the `$INFORMIXDIR/etc/jsonListener.properties` file, with REPLICATION privilege group access. If a database server instance is created after the installation process, you must add the user with REPLICATION privilege group access to the **`url`** parameter in the `$INFORMIXDIR/etc/jsonListener.properties` file.

Procedure

To enable sharding for JSON data:

1. Specify trusted hosts for each database server that is added to the shard cluster. Use one of the following methods to add trusted-host names as values to each database server's REMOTE_SERVER_CFG configuration parameter:
   - Use the OpenAdmin Tool (OAT) for Informix. Go to the **Server Administration** > **Configuration** page, and click the **Trusted Hosts** tab.
   - Run the SQL administration API **task()** or **admin()** function with the **cdr add trustedhost** argument and appropriate host values. You must be a Database Server Administrator (DBSA) to run these functions.

2. Set the **sharding.enable** parameter to `true` in each database server's `$INFORMIXDIR/etc/jsonListener.properties` file.
3. Restart the wire listener.

Related reference:

cdr add trustedhost argument: Add trusted hosts (SQL administration API) (Administrator's Reference)

cdr remove trustedhost argument: Remove trusted hosts (SQL administration API) (Administrator's Reference)

cdr list trustedhost argument: List trusted hosts (SQL administration API) (Administrator's Reference)

Creating a shard cluster by running the addShard command in the MongoDB shell

The **sh.addShard** MongoDB shell command adds a database server to a shard cluster.

Before you begin

You must verify the following details:

- All database servers that are participating in a shard cluster are specified as trusted hosts.
- The **sharding.enable** parameter is set to `true` in each database server's `$INFORMIXDIR/etc/jsonListener.properties` file.
- The user of the wire listener has REPLICATION privilege group access in the SQL Admin API.

Procedure

To create a shard cluster by running the **addShard** command in the MongoDB shell:

1. Run the **mongo** command. The command starts the MongoDB shell.
2. Run the **sh.addShard** MongoDB command. For example:

```
prompt> sh.addShard("myhost1.ibm.com:9201")
```

Example

Add a single database server to a shard cluster

The following command adds the database server that is at port **9202** of **myhost2.ibm.com** to a shard cluster. The shard-cluster definition changes to include the new server.

```
prompt> sh.addShard("myhost2.ibm.com:9202")
```

Related reference:

cdr define shardCollection (Enterprise Replication Guide)

cdr add trustedhost argument: Add trusted hosts (SQL administration API) (Administrator's Reference)

cdr remove trustedhost argument: Remove trusted hosts (SQL administration API) (Administrator's Reference)

cdr list trustedhost argument: List trusted hosts (SQL administration API) (Administrator's Reference)

"Database commands" on page 4-4

Creating a shard cluster by running the addShard command through db.runCommand in the MongoDB shell

The **db.runCommand** command with **addShard** command syntax adds database servers to a shard cluster.

Before you begin

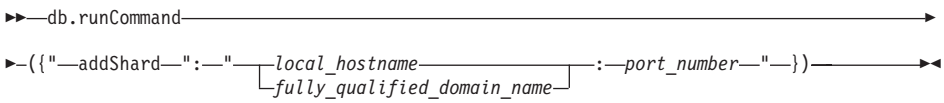
You must verify the following details:

- All database servers that are participating in a shard cluster are specified as trusted hosts.
- The **sharding.enable** parameter is set to `true` in each database server's `$INFORMIXDIR/etc/jsonListener.properties` file.
- The user of the wire listener has REPLICATION privilege group access in the SQL Admin API.

Procedure

To create a shard cluster by running the **addShard** command through **db.runCommand** in the MongoDB shell:

1. Run the **mongo** command. The command starts the MongoDB shell.
2. Run the **db.runCommand** from the MongoDB shell, with **addShard** command syntax.
   - Add a single database server by using the following syntax:

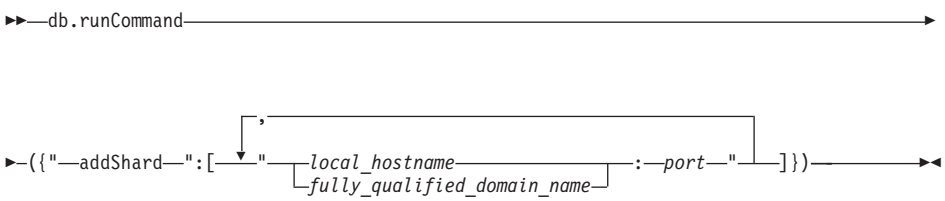


Element	Description	Restrictions
local_hostname	The localhost name for a server.	None.
fully_qualified_domain_name	The full domain name for a server.	None.
port_number	The port that is used for communication with the server.	None.

For example:

```
prompt> db.runCommand({"addShard":"myhost1.ibm.com:9201"})
```

   - Add multiple database servers by using the following syntax:



Element	Description	Restrictions
local_hostname	The localhost name for a server.	None.
fully_qualified_domain_name	The full domain name for a server.	None.
port	The port that is used for communication with the server.	None.

For example:

```
prompt> db.runCommand({["addShard":"myhost2.ibm.com:9202",
   "myhost3.ibm.com:9203"]})
```

Examples

Add a database server to a shard cluster

This example adds the database server that is at port **9204** of **myhost4.ibm.com** to a shard cluster. The shard-cluster definition changes to include the new server.

```
prompt> db.runCommand({"addShard":"myhost4.ibm.com:9204"})
```

Add multiple database servers to a shard cluster

This example adds the database servers that are at port **9205** of **myhost5.ibm.com**, port **9206** of **myhost6.ibm.com**, and port **9207** of **myhost7.ibm.com** to a shard cluster. The shard-cluster definition changes to include the new servers.

```
prompt> db.runCommand({"addShard":["myhost5.ibm.com:9205",
    "myhost6.ibm.com:9206","myhost7.ibm.com:9207"]})
```

Related reference:

cdr define shardCollection (Enterprise Replication Guide)

cdr add trustedhost argument: Add trusted hosts (SQL administration API) (Administrator's Reference)

cdr remove trustedhost argument: Remove trusted hosts (SQL administration API) (Administrator's Reference)

cdr list trustedhost argument: List trusted hosts (SQL administration API) (Administrator's Reference)

"Database commands" on page 4-4

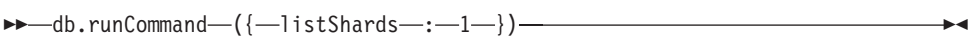
Viewing shard-cluster participants

Run the **db.runCommand** MongoDB shell command with **listShards** syntax to list the Enterprise Replication group names, hosts, and port numbers of all database servers that are participating in a shard cluster.

Procedure

1. Run the **mongo** command. The command starts the MongoDB shell.
2. Run the **listShards** command:

Syntax:



For example, run the following command:

```
db.runCommand({listShards:1})
```

The **listShards** command produces output in the following structure:

```
{
        "serverUsed" : "server_host/IP_address",
        "shards" : [
                {
                        "_id" : "ER_group_name_1",
                        "host" : "host_1:port_1"
                },
                {
                        "_id" : "ER_group_name_2",
                        "host" : "host_2:port_2"
                },
                {
                        "_id" : "ER_group_name_x",
                        "host" : "host_x:port_x"
                }
        ],
        "ok" : 1
}
```

Element	Description	Restrictions
ER_group_name	The Enterprise Replication group name of a database server that is a shard-cluster participant. The Enterprise Replication group name for a database server can be found in the database server's `sqlhosts` file. The default location for the `sqlhosts` file is: • UNIX: `$INFORMIXDIR/etc/sqlhosts` • Windows: `%INFORMIXDIR%\etc\sqlhosts.%INFORMIXSERVER%` The default Enterprise Replication group name for a database server is the database server's name prepended with `g_`. For example, the default Enterprise Replication group name for a database server named **myserver** is **g_myserver**.	None.
host	The host for a shard-cluster participant. The host can be a localhost name or a full domain name.	None.
IP_address	The IP address of the database server that the listener is connected to.	None.
port	The port number that a shard-cluster participant uses to communicate with other shard-cluster participants.	None.
server_host	The host for the database server that the listener is connected to. The host can be a localhost name or a full domain name.	None.

Example

For this example, you have a shard cluster defined by the following command:

```
prompt> db.runCommand({"addShard":["myhost1.ibm.com:9201",
   "myhost2.ibm.com:9202","myhost3.ibm.com:9203",
   "myhost4.ibm.com:9204","myhost5.ibm.com:9205"]})
```

The following example shows output when the **listShards** command is run in the MongoDB shell, and the listener is connected to the database server at myhost1.ibm.com.

```
{
        "serverUsed" : "myhost1.ibm.com/192.0.2.0:9200",
        "shards" : [
                {
                        "_id" : "g_myserver1",
                        "host" : "myhost1.ibm.com:9200"
                },
                {
                        "_id" : "g_myserver2",
                        "host" : "myhost2.ibm.com:9202"
                },
                {
                        "_id" : "g_myserver3",
                        "host" : "myhost3.ibm.com:9203"
                }
                {
                        "_id" : "g_myserver4",
                        "host" : "myhost4.ibm.com:9204"
                }
                {
                        "_id" : "g_myserver5",
                        "host" : "myhost5.ibm.com:9205"
                }
        ],
        "ok" : 1
}
```

Figure 3-1. **listShards** *command output for a shard cluster*

Related concepts:

Installing the OpenAdmin Tool for Informix with the Client SDK (Client Products Installation Guide)

Related reference:

cdr list trustedhost argument: List trusted hosts (SQL administration API) (Administrator's Reference)

cdr list shardCollection (Enterprise Replication Guide)

onstat -g shard command: Print information about the shard cache (Administrator's Reference)

"Database commands" on page 4-4

Shard-cluster definitions for distributing data

A cluster of shard servers uses a definition to distribute data across shard servers.

You must create a shard-cluster definition to distribute data across the shard servers. The definition contains the following information:

- The Informix Enterprise Replication group name of each participating shard server.

- The name of the database and collection that is distributed across the shard servers of a shard cluster.
- The key that is used for sharding data. Key values determine which shard server the document is stored on.
- The sharding method, which is either a hash algorithm or an expression, by which documents are distributed to specific shard servers.

A definition that uses a hash algorithm to shard data is modified when MongoDB commands are used to add a server to the shard cluster.

A definition that uses an expression to shard data can be modified by running the **changeShardCollection** command. If you add a shard server to a definition, you must first add the server to the shard cluster by running the **db.runCommand** command with **addShard** command syntax.

Related reference:

cdr change shardCollection (Enterprise Replication Guide)

cdr delete shardCollection (Enterprise Replication Guide)

Creating a shard-cluster definition that uses a hash algorithm for distributing data across database servers

The **shardCollection** command in the MongoDB shell creates a definition for distributing data across the database servers of a shard cluster.

Procedure

To create a shard-cluster definition that uses a hash algorithm for distributing data across database servers:

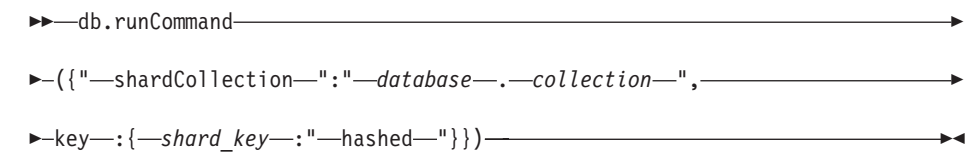
1. Run the **mongo** command. The command starts the MongoDB shell.
2. Run the **shardCollection** command. There are two ways to run the command:
   - Run the **sh.shardCollection** MongoDB command. For example:

     ```
     prompt> sh.shardCollection("database1.collection1",
        {customer_name:"hashed"})
     ```

   - Run the **db.runCommand** from the MongoDB shell, with **shardCollection** command syntax. For example:

     ```
     prompt> db.runCommand({"shardCollection":"database2.collection_2",
        key:{customer_name:"hashed"}})
     ```

     The **shardCollection** command syntax for using a hash algorithm is shown in the following diagram:



Element	Description	Restrictions
database	The name of the database that contains the collection that is distributed across database servers.	The database must exist.
collection	The name of the collection that is distributed across database servers.	The collection must exist.

Element	Description	Restrictions
shard_key	The key that is used to distribute data across the database servers of a shard cluster.	The key must exist. Composite shard keys are not supported.

3. For optimal query performance, connect to the wire listener and run the MongoDB **ensureIndex** command on each of a cluster's shard servers. The **ensureIndex** command ensures that a collection index is created on the shard server.

Results

The name of a shard-cluster definition that is created by a **shardCollection** command that is run through the wire listener is:

`sh_database_collection`

Example

The following command defines a shard cluster that uses a hash algorithm on the key value **year** to distribute data across multiple database servers.

```
prompt> sh.shardCollection("database3.collection3",{year:"hashed"})
```

The name of the created shard-cluster definition is **sh_database3_collection3**.

Related reference:

cdr change shardCollection (Enterprise Replication Guide)

cdr delete shardCollection (Enterprise Replication Guide)

"Database commands" on page 4-4

Creating a shard-cluster definition that uses an expression for distributing data across database servers

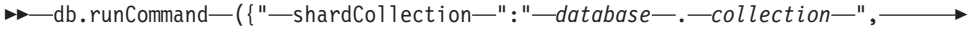
The MongoDB shell **db.runCommand** command with **shardCollection** command syntax creates a definition for distributing data across the database servers of a shard cluster.

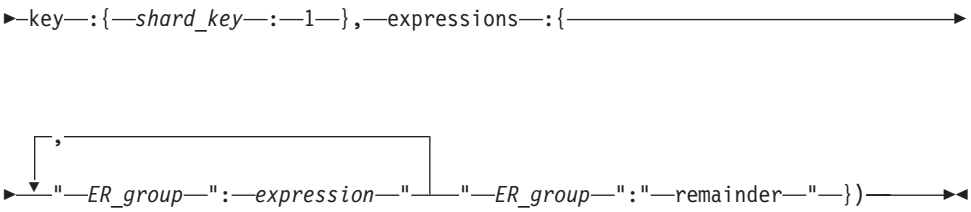
Procedure

To create a shard-cluster definition that uses an expression for distributing data across database servers:

1. Run the **mongo** command. The command starts the MongoDB shell.
2. Run the **db.runCommand** from the MongoDB shell, with **shardCollection** command syntax.

   The **shardCollection** command syntax for using an expression is shown in the following diagram:





Element	Description	Restrictions
collection	The name of the collection that is distributed across database servers.	The collection must exist.
database	The name of the database that contains the collection that is distributed across database servers.	The database must exist.
ER_group	The Enterprise Replication group name of a database server that receives copied data. The default Enterprise Replication group name for a database server is the database server's name prepended with `g_`. For example, the default Enterprise Replication group name for a database server named **myserver** is **g_myserver**.	None.
expression	The expression that is used to select documents by shard key value.	None.
remainder	Specifies the database server that receives documents with shard key values that are not selected by expressions. The remainder expression is required.	The database server must exist.
shard_key	The key that is used to distribute data across the database servers of a shard cluster.	The collection must exist. Composite shard keys are not supported.

3. For optimal query performance, connect to the wire listener and run the MongoDB **ensureIndex** command on each of a cluster's shard servers. The **ensureIndex** command ensures that a collection index is created on the shard server.

Results

The name of a shard-cluster definition that is created by a **shardCollection** command that is run through the wire listener is:

`sh_database_collection`

Examples

Define a shard cluster that uses an expression to distribute collections across multiple database servers

The following command defines a shard cluster that uses an expression on the key value **state** for distributing **collection1** across multiple database servers.

```
prompt> db.runCommand({"shardCollection":"database1.collection1",
   key:{state:1},expressions:{"g_shard_server_1":"in ('KS','MO')",
   "g_shard_server_2":"in ('CA','WA')","g_shard_server_3":"remainder"}})
```

The name of the created shard-cluster definition is **sh_database1_collection1**.

- **KS** and **MO** data are sent to **g_shard_server_1**.
- **CA** and **WA** data are sent to **g_shard_server_2**.
- All data that is not **KS**, **MO**, **CA**, or **WA** data is sent to **g_shard_server_3**.

Define a shard cluster that uses an expression to distribute collections across multiple database servers

The following command defines a shard cluster that uses an expression on the key value **animal** for distributing **collection2** across multiple database servers.

```
prompt> db.runCommand({"shardCollection":"database1.collection2",
   key:{animal:1},expressions:{"g_shard_server_1":"in ('dog','coyote')",
   "g_shard_server_2":"in ('cat')","g_shard_server_3":"in ('rat')",
   "g_shard_server_4":"remainder"}})
```

The name of the created shard-cluster definition is **sh_database2_collection2**.

- **dog** and **coyote** data are sent to **g_shard_server_1**.
- **cat** data is sent to **g_shard_server_2**.
- **rat data** is sent to **g_shard_server_3**.
- All data that is not **dog**, **coyote**, **cat**, or **rat** data is sent to **g_shard_server_4**.

Define a shard cluster that uses an expression to distribute collections across multiple database servers

The following command defines a shard cluster that uses an expression on the key value **year** for distributing **collection3** across multiple database servers.

```
prompt> db.runCommand({"shardCollection":"database1.collection3",
   key:{year:1},expressions:{"g_shard_server_1":"between 1980 and 1989",
   "g_shard_server_2":"between 1990 and 1999",
   "g_shard_server_3":"between 2000 and 2009",
   "g_shard_server_4":"remainder"}})
```

The name of the created shard-cluster definition is **sh_database3_collection3**.

- Documents with a year key value of 1984 are sent to **g_shard_server_1**.
- Documents with a year key value of 1995 are sent to **g_shard_server_2**.
- Documents with a year key value of 2000 are sent to **g_shard_server_3**.
- Documents with a year key value less than 1980 or more than 2009 are sent to **g_shard_server_4**.

Related reference:

cdr add trustedhost argument: Add trusted hosts (SQL administration API) (Administrator's Reference)

cdr remove trustedhost argument: Remove trusted hosts (SQL administration API) (Administrator's Reference)

cdr list trustedhost argument: List trusted hosts (SQL administration API) (Administrator's Reference)

cdr define shardCollection (Enterprise Replication Guide)

cdr change shardCollection (Enterprise Replication Guide)

cdr delete shardCollection (Enterprise Replication Guide)

cdr list shardCollection (Enterprise Replication Guide)

"Database commands" on page 4-4

Changing the definition for a shard cluster

The **db.runCommand** command with **changeShardCollection** command syntax changes the definition for a shard cluster.

Before you begin

If the shard cluster uses an expression for distributing a collection across multiple database servers, you can add database servers to a shard cluster and remove database servers from a shard cluster. If the shard-cluster definition uses a hash algorithm, you can add database servers to the shard cluster by using the **sh.addShard** MongoDB shell command.

If you change a shard-cluster definition to include a new shard server, that server must first be added to a shard cluster by running the **db.runCommand** command with **addShard** command syntax.

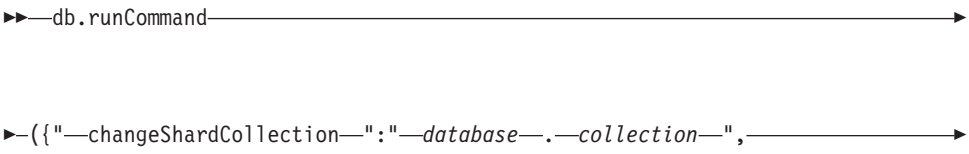
About this task

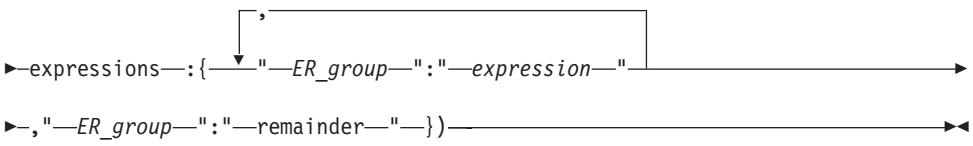
The following steps apply to changing the definition for shard cluster that uses an expression for distributing documents in a collection across multiple database servers.

Procedure

To change the definition for a shard cluster:

1. Run the **mongo** command. The command starts the MongoDB shell.
2. Change the shard-cluster definition by running the **changeShardCollection** command:





Element	Description	Restrictions
collection	The name of the collection that is distributed across database servers.	The collection must exist.
database	The name of the database that contains the collection that is distributed across database servers.	The database must exist.
ER_group	The Enterprise Replication group name of a database server that receives copied data. The default Enterprise Replication group name for a database server is the database server's name prepended with g_. For example, the default Enterprise Replication group name for a database server named **myserver** is **g_myserver**.	None.
expression	The expression that is used to select documents by shard key value.	None.
remainder	The database server that receives documents with shard key values that are not selected by expressions.	

3. For optimal query performance, connect to the wire listener and run the MongoDB **ensureIndex** command on each of a cluster's shard servers. The **ensureIndex** command ensures that a collection index is created on the shard server.

Example

You have a shard cluster that is composed of three database servers, and the shard cluster is defined by the following command:

```
prompt> db.runCommand({"shardCollection":"database1.collection1",
   expressions:{"g_shard_server_1":"in ('KS','MO')",
   "g_shard_server_2":"in ('CA','WA')","g_shard_server_3":"remainder"})
```

To add **g_shard_server_4** and **g_shard_server_5** to the shard cluster and change where data is sent to, run the following command:

```
prompt> db.runCommand({"changeShardCollection":"database1.collection1",
   expressions:{"g_shard_server_1":"in ('KS','MO')",
   "g_shard_server_2":"in ('TX','OK')","g_shard_server_3":"in ('CA','WA')",
   "g_shard_server_4":"in ('OR','ID')","g_shard_server_5":"remainder"})
```

The new shard cluster contains five database servers.

- Documents with a **state** field value of CA or WA are now sent to **g_shard_server_3**.
- Documents with a **state** field value of OR or ID are now sent to **g_shard_server_4**.
- Documents with a **state** field value that is not in the expression are sent to **g_shard_server_5**.

To then remove **g_shard_server_2** and change where the data that was on **g_shard_server_2** is sent to, run the following command:

```
prompt> db.runCommand({"changeShardCollection":"database1.collection1",
   expressions:{"g_shard_server_1":"in ('KS','MO')",
   "g_shard_server_3":"in ('TX','CA','WA')",
   "g_shard_server_4":"in ('OK','OR','ID')",
   "g_shard_server_5":"remainder"})
```

The new shard cluster contains four database servers.

- Documents with a **state** field value of TX are now sent to **g_shard_server_3**.
- Documents with a **state** field value of OK are now sent to **g_shard_server_4**.

Related reference:

cdr define shardCollection (Enterprise Replication Guide)

cdr change shardCollection (Enterprise Replication Guide)

cdr delete shardCollection (Enterprise Replication Guide)

cdr list shardCollection (Enterprise Replication Guide)

Chapter 4. Application programming interfaces and commands

The Informix support for MongoDB application programming interfaces and commands is described here.

Language Drivers

The wire listener parses messages that are based on the MongoDB wire protocol. You can use the MongoDB community drivers to store, update, and query JSON documents with Informix as a JSON data store. These drivers can include Java, C/C++, Ruby, PHP, PyMongo, and so on.

Download the MongoDB drivers for the programming languages at http://docs.mongodb.org/ecosystem/drivers/.

Command utilities and tools

You can use the MongoDB shell and any of the standard MongoDB command utilities and tools.

The supported MongoDB shell is version 2.4.3.

You can run the MongoDB mongodump and mongoexport utilities against MongoDB to export data from MongoDB to Informix.

You can run the MongoDB mongorestore and mongoimport utilities against Informix to import data from MongoDB to Informix.

Collection commands

The **db.collection** commands for the MongoDB shell that are supported by Informix are shown.

Table 4-1. Supported db.collection commands

MongoDB command	Supported	Details
aggregate	No	
count	Yes	

Table 4-1. Supported db.collection commands (continued)

MongoDB command	Supported	Details

createIndex	Yes	You can use the MongoDB createIndex syntax to create an index that works for all data types. For example:
		`db.collection.createIndex( { zipcode: 1 } )`
		`db.collection.createIndex( { state: 1, zipcode: -1} )`
		
		You can use the Informix createIndex syntax to create an index for a specific data type. For example:
		`db.collection.createIndex( { zipcode : [1, "$int"] } )`
		`db.collection.createIndex( { state: [1, "$string"],`
		`zipcode: [-1, "$int"] } )`
		
		Tip: If you are creating an index on a field that has a fixed data type, you can get better query performance by using the Informix createIndex syntax.
		
		For more information, see the "Informix createIndex and ensureIndex syntax support" on page 4-3.

dataSize	Yes	
distinct	No	
drop	Yes	
dropIndex	Yes	
dropIndexes	Yes	

ensureIndex	Yes	You can use the MongoDB ensureIndex syntax to create an index that works for all data types. For example:
		`db.collection.ensureIndex( { zipcode: 1 } )`
		`db.collection.ensureIndex( { state: 1, zipcode: -1} )`
		
		You can use the Informix ensureIndex syntax to create an index for a specific data type. For example:
		`db.collection.ensureIndex( { zipcode : [1, "$int"] } )`
		`db.collection.ensureIndex( { state: [1, "$string"],`
		`zipcode: [-1, "$int"] } )`
		
		Tip: If you are creating an index on a field that has a fixed data type, you can get better query performance by using the Informix ensureIndex syntax.
		
		For more information, see the "Informix createIndex and ensureIndex syntax support" on page 4-3.

find	Yes	
findAndModify	No	
findOne	Yes	
getIndexes	Yes	
getShardDistribution	No	
getShardVersion	No	
getIndexStats	No	
group	No	
indexStats	No	
insert	No	
isCapped	Yes	This command returns false because capped collections are not supported in Informix.

Table 4-1. Supported db.collection commands (continued)

MongoDB command	Supported	Details
mapReduce	No	
reIndex	No	
remove	Yes	The justOne option is not supported. This command deletes all documents that match the query criteria.
renameCollection	Yes	
save	Yes	
stats	Yes	
storageSize	Yes	
totalSize	Yes	
update	Yes	The multi option is supported only if `update.one.enable=true` in the `jsonListener.properties` file. If `update.one.enable=false`, all documents that match the query criteria are updated.
validate	No	

Informix createIndex and ensureIndex syntax support

The following data types are supported:

- $string
- $int
- $integer
- $double
- $number
- $date
- $timestamp
- $binary
- $boolean

Hashed, text, and geospatial indexes are not supported. If any of these MongoDB index types are specified, a regular untyped index is created.

The following options are supported:

- unique
- name

The following options are not supported:

- background
- dropDups
- sparse
- expireAfterSeconds
- v
- weights
- default_language
- language_override

Related reference:

"The jsonListener.properties file" on page 2-3

Database commands

The MongoDB database commands that are supported by Informix are sorted into logical areas.

User commands

Aggregation commands

Table 4-2. Aggregation commands

MongoDB command	Supported	Details
aggregate	No	
count	Yes	
distinct	No	
group	No	
mapReduce	No	

Query and write operation commands

Table 4-3. Query and write operation commands

MongoDB command	Supported	Details
eval	No	
findAndModify	No	
getLastError	Yes	
getPrevError	No	
resetError	No	
text	No	

Database operations

Authentication commands

Table 4-4. Authentication commands

MongoDB command	Supported	Details
logout	No	
authenticate	No	

Diagnostic commands

Table 4-5. Diagnostic commands

MongoDB command	Supported	Details
buildInfo	Yes	Whenever possible, the Informix output fields are identical to MongoDB. There are additional fields that are unique to Informix.
collStats	Yes	The value of any field that is based on the collection size is an estimate, not an exact value. For example, the value of the field 'size' is an estimate.
connPoolStats	No	

Table 4-5. Diagnostic commands (continued)

MongoDB command	Supported	Details
cursorInfo	No	
dbStats	Yes	The value of any field that is based on the collection size is an estimate, not an exact value. For example, the value of the field 'dataSize' is an estimate.
features	No	
getCmdLineOpts	Yes	
getLog	No	
hostInfo	Yes	The `memSizeMB`, `totalMemory`, and `freeMemory` fields indicate the amount of memory that is available to the Java virtual machine (JVM) that is running, not the operating system values.
indexStats	No	
listCommands	No	
listDatabases	Yes	The value of any field that is based on the collection size is an estimate, not an exact value. For example, the value of the field 'sizeOnDisk' is an estimate.
ping	Yes	
serverStatus	Yes	
top	No	
whatsmyuri	Yes	

Instance administration commands

Table 4-6. Instance administration commands

MongoDB command	Supported	Details
clone	No	
cloneCollection	No	
cloneCollectionAsCapped	No	
collMod	No	
compact	No	
convertToCapped	No	
copydb	No	
create	Yes	Informix does not support the following flags: • capped • autoIndexID • size • max
drop	Yes	Informix does not lock the database to block concurrent activity.
dropDatabase	Yes	
dropIndexes	Yes	The MongoDB deleteIndexes command is equivalent.
filemd5	No	
fsync	No	
getParameter	No	
logRotate	No	

Table 4-6. Instance administration commands (continued)

MongoDB command	Supported	Details
reIndex	No	
renameCollection	No	
repairDatabase	No	
setParameter	No	
shutdown	Yes	The timeoutSecs flag is supported. In the Informix, the timeoutSecs flag determines the number of seconds that the wire listener waits for a busy client to stop working before forcibly terminating the session. The force flag is not supported.
touch	No	

Replication commands

Table 4-7. Replication commands

MongoDB command	Supported	Details
isMaster	Yes	
replSetFreeze	No	
replSetGetStatus	No	
replSetInitiate	No	
replSetMaintenance	No	
replSetReconfig	No	
replSetStepDown	No	
replSetSyncFrom	No	
Resync	No	

Sharding commands

Table 4-8. Replication commands

MongoDB command	Supported	Details
addShard	Yes	The MongoDB maxSize and name options are not supported. In addition to the MongoDB command syntax for adding a single shard server, you can use the Informix specific syntax to add multiple shard servers in one command by sending the list of shard servers as an array. For more information, see "Creating a shard cluster by running the addShard command through db.runCommand in the MongoDB shell" on page 3-3.
enableSharding	Yes	This action is not required for Informix and therefore this command has no affect for Informix.
flushRouterConfig	No	
isdbgrid	Yes	
listShards	Yes	The equivalent Informix command is **cdr list server**.
movePrimary	No	
removeShard	No	

Table 4-8. Replication commands (continued)

MongoDB command	Supported	Details
shardCollection	Yes	The equivalent Informix command is **cdr define shardCollection**. The MongoDB unique and numInitialChunks options are not supported.
shardingState	No	
split	No	

Related tasks:

"Creating a shard-cluster definition that uses an expression for distributing data across database servers" on page 3-8

"Viewing shard-cluster participants" on page 3-4

"Creating a shard cluster by running the addShard command in the MongoDB shell" on page 3-2

"Creating a shard cluster by running the addShard command through db.runCommand in the MongoDB shell" on page 3-3

"Creating a shard-cluster definition that uses a hash algorithm for distributing data across database servers" on page 3-7

Query, update, and projection operators

The MongoDB query, update, and projection operators that are supported by Informix are sorted into logical areas.

Query selectors

Array query operators

Unless otherwise specified, queries against array data types are not supported and no rows are returned by Informix.

Table 4-9. Array query operators

MongoDB command	Supported	Details
$elemMatch	No	
$size	Yes	Supported for simple queries only. The operator is only supported when it is the only condition in the query document.

Comparison query operators

Table 4-10. Comparison query operators

MongoDB command	Supported	Details
$all	Yes	Supported for primitive values and simple queries only. The operator is only supported when it is the only condition in the query document.
$gt	Yes	
$gte	Yes	
$in	Yes	
$lt	Yes	
$lte	Yes	
$ne	Yes	

Table 4-10. Comparison query operators (continued)

MongoDB command	Supported	Details
$nin	Yes	
$query	Yes	

Element query operators

Table 4-11. Element query operators

MongoDB command	Supported	Details
$exists	Yes	
$mod	Yes	
$type	Yes	

Geospatial query operators

The geospatial query operators are not supported.

JavaScript query operators

The JavaScript query operators are not supported.

Logical query operators

Table 4-12. Logical query operators

MongoDB command	Supported	Details
$and	Yes	
$or	Yes	
$not	Yes	
$nor	Yes	

Subdocument query elements

Queries that compare non-primitive data types, such as subdocuments, are not supported.

Update operators

Array update operators

Table 4-13. Field update operators

MongoDB command	Supported	Details
$	No	
$addToSet	Yes	Supported for primitive values only. The operator is not supported on arrays and objects.
$pop	Yes	
$pullAll	Yes	Supported for primitive values only. The operator is not supported on arrays and objects.
$pull	Yes	Supported for primitive values only. The operator is not supported on arrays and objects.
$pushAll	Yes	
$push	Yes	

Array update operators modifiers

The array update operators modifiers are not supported.

Bitwise update operators

Table 4-14. Bitwise update operators

MongoDB command	Supported	Details
$bit	Yes	

Field update operators

Table 4-15. Field update operators

MongoDB command	Supported	Details
$inc	Yes	
$rename	Yes	
$setOnInsert	Yes	
$set	Yes	
$unset	Yes	

Isolation update operators

The isolation update operators are not supported.

Projection operators

Comparison query operators

Table 4-16. Comparison query operators

MongoDB command	Supported	Details
$	No	
$elemMatch	No	
$slice	No	
$comment	No	
$explain	Yes	
$hint	Yes	
$maxScan	No	
$max	No	
$min	No	
$orderby	Yes	Not supported for sharded data.
$returnkey	No	
$showdiskLoc	No	
$snapshot	No	

Chapter 5. Monitoring collections

You can use the IBM OpenAdmin Tool (OAT) for Informix to monitor collections in an Informix database.

View collections by using the IBM Informix JSON Plug-in for OpenAdmin Tool (OAT) or by using the IBM Informix Schema Manager Plug-in for OpenAdmin Tool (OAT).

See the OAT help for more information.

Related concepts:

Installing the OpenAdmin Tool for Informix with the Client SDK (Client Products Installation Guide)

Related reference:

cdr list trustedhost argument: List trusted hosts (SQL administration API) (Administrator's Reference)

cdr list shardCollection (Enterprise Replication Guide)

onstat -g shard command: Print information about the shard cache (Administrator's Reference)

Chapter 6. Troubleshooting Informix JSON compatibility

Several troubleshooting techniques, tools, and resources are available for resolving problems that you encounter with Informix JSON compatibility.

Problem	Solution
The wire listener does not start automatically.	If the wire listener does not automatically start: 1. Verify that the user has been created. 2. Manually start wire listener.
I am using a IPv6 connection and the wire listener is unable to connect to the Informix server.	Ensure that the wire listener and Informix server use the same IP address. 1. Change the loopback address in the `sqlhosts` file from `127.0.0.1` to the IP address that is used by your machine for loopback (normally `::1` for IPv6). 2. Restart the Informix server and the wire listener.
Where is the log file?	**UNIX:** The log file is in `$INFORMIXDIR/jsonListener.log`. **Windows:** The log file is named *servername*`_jsonListener.log` and is in your home directory. For example, `C:\Users\ifxjson\ol_informix1210_1_jsonListener.log`.
Can I debug problems with the wire listener?	Yes, from the wire listener command line you can issue `-loglevel` *level*, where *level* is the logging level. Log level options are: • error • warn • info • debug • trace
How do I access the wire listener help?	You can view a list of available command line options by issuing the `-help` command.
How can I view all of the current properties for the jsonListener.properties file?	From the wire listener command line you can issue the `-listProperties` command. This prints all of the supported properties and their default values.

Related tasks:

"Starting the wire listener" on page 2-1

Related reference:

"The jsonListener.properties file" on page 2-3

Appendix. Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

Accessibility features for IBM Informix products

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Informix products. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.
- The attachment of alternative input and output devices.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

Related accessibility information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software.

IBM and accessibility

For more information about the IBM commitment to accessibility, see the *IBM Accessibility Center* at http://www.ibm.com/able.

Dotted decimal syntax diagrams

The syntax diagrams in our publications are available in dotted decimal format, which is an accessible format that is available only if you are using a screen reader.

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), the elements can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read punctuation. All syntax elements that have the same dotted decimal number (for example, all syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines `3.1 USERID` and `3.1 SYSTEMID`, your syntax can include either `USERID` or `SYSTEMID`, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, the word or symbol is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is read as 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol that provides information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, that element is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 refers to a separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

? Specifies an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element (for example, 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

***** Specifies a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be

repeated. For example, if you hear the line `5.1* data-area`, you know that you can include more than one data area or you can include none. If you hear the lines `3*, 3 HOST`, and `3 STATE`, you know that you can include `HOST`, `STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The `*` symbol is equivalent to a loop-back line in a railroad syntax diagram.

`+` Specifies a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times. For example, if you hear the line `6.1+ data-area`, you must include at least one data area. If you hear the lines `2+`, `2 HOST`, and `2 STATE`, you know that you must include `HOST`, `STATE`, or both. As for the * symbol, you can repeat a particular item if it is the only item with that dotted decimal number. The `+` symbol, like the `*` symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy,

modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

N

O

P

Q

R

S

T

U

V

W

Printed in USA

SC27-5556-00