## Интерфейсы соединений КОПФа

1ый столбик – метод конкретного типа соединения

2ой столбик – вызываемый этим методом метод на Thrift’e

ClusterConnection

|  |  |
| --- | --- |
| IList<Keyspace> RetrieveKeyspaces() | List<KsDef> describe\_keyspaces() |
| void AddKeyspace(Keyspace keyspace) | string system\_add\_keyspace(KsDef ks\_def) |
| void RemoveKeyspace(string keyspace) | string system\_drop\_keyspace(string keyspace) |

KeyspaceConnection

|  |  |
| --- | --- |
| void AddColumnFamily(ColumnFamily columnFamily) | string system\_add\_column\_family(CfDef cf\_def) |
| void AddColumnFamily(string columnFamilyName) |
| Keyspace DescribeKeyspace() | KsDef describe\_keyspace(string keyspace) |
| void UpdateColumnFamily(ColumnFamily columnFamily) | string system\_update\_column\_family(CfDef cf\_def) |
| void RemoveColumnFamily(string columnFamily) | string system\_drop\_column\_family(string column\_family) |

ColumnFamilyConnection

|  |  |
| --- | --- |
| void AddColumn(string key, Column column) | void insert(byte[] key, ColumnParent column\_parent, Column column, ConsistencyLevel consistency\_level) |
| Column GetColumn(string key, string columnName) | ColumnOrSuperColumn get(byte[] key, ColumnPath column\_path, ConsistencyLevel consistency\_level) |
| bool TryGetColumn(string key, string columnName, out Column result) |
| void DeleteBatch(string key, IEnumerable<string> columnNames) | void batch\_mutate(Dictionary<byte[], Dictionary<string, List<Mutation>>> mutation\_map, ConsistencyLevel consistency\_level) |
| void AddBatch(string key, IEnumerable<Column> columns) |
| void BatchInsert(IEnumerable<KeyValuePair<string, IEnumerable<Column>>> data) |
| Column[] GetRow(string key, string greatThanColumnName, int count) | List<ColumnOrSuperColumn> get\_slice(byte[] key, ColumnParent column\_parent, SlicePredicate predicate, ConsistencyLevel consistency\_level) |
| List<KeyValuePair<string, Column[]>> GetRows(IEnumerable<string> keys, string startColumnName, int count) | Dictionary<byte[], List<ColumnOrSuperColumn>> multiget\_slice(List<byte[]> keys, ColumnParent column\_parent, SlicePredicate predicate, ConsistencyLevel consistency\_level) |
| string[] GetRowsWhere(int maximalCount, IndexExpression[] conditions, string[] columns) | List<KeySlice> get\_indexed\_slices(ColumnParent column\_parent, IndexClause index\_clause, SlicePredicate column\_predicate, ConsistencyLevel consistency\_level) |
| void Truncate() | void truncate(string cfname) |

## Описание сущностей Thrift’a

Прочитать об этих сущностях и их атрибутах можно тут:

1. <http://wiki.apache.org/cassandra/API#Structures> ( + поиск по предыдущим версиям документа)
2. <http://wiki.apache.org/cassandra/StorageConfiguration#Config_Overview> (некоторые параметры названы иначе, нежели в Thrift API)
3. ..\conf\cassandra.yaml (файлик конфигурации БД)
4. ..\bin\cassandra-cli.exe, набрать help;
5. Исходники Aquiles
6. Исходники других клиентов
7. Исходники Cassandra
8. <http://habrahabr.ru/qa/5169/> - автоматическое удаление данных
9. <http://www.datastax.com/docs/0.8/index>
10. <http://www.datastax.com/docs/0.8/data_model/column_families>
11. <http://www.datastax.com/docs/0.8/configuration/storage_configuration>

**Обозначения:**

* [R] – required. Поле обязательно указывать при передаче структуры в команду.
* [E] – exclusive. Если в рамках одной сущности несколько полей имеют такую пометку, значит необходимо указывать только одно из этих полей.
* [1],[2],… - ссылаюсь или рекомендую глянуть источник под заданным порядковым номером (список источников чуть выше)

public class KsDef : TBase

Usage: describe\_keyspace(), describe\_keyspaces(), system\_add\_keyspace()

|  |  |
| --- | --- |
| string Name | [R] |
| string Strategy\_class | [R] Defines how replicas are placed on physical hardware. "org.apache.cassandra.locator.*SimpleStrategy*" - places the first replica at the node whose token is closest to the key (as determined by the Partitioner), and additional replicas on subsequent nodes along the ring in increasing Token order.  "org.apache.cassandra.locator.*NetworkTopologyStrategy*" - for each datacenter you can specify how many replicas you want on a per-keyspace basis. Replicas are placed on different racks within each datacenter, if possible. This strategy requires a rack aware snitch, such as RackInferringSnitch or PropertyFileSnitch.  "org.apache.cassandra.locator.*OldNetworkTopologyStrategy*" places one replica in each of two datacenters, and the third on a different rack in the first. Additional datacenters are not guaranteed to get a replica. Additional replicas after three are placed in ring order after the third without regard to rack or datacenter. [11] |
| Dictionary<string, string> Strategy\_options | Values provided for strategy\_options are used with the NetworkTopologyStrategy replica placement strategy for defining how many replicas to place in each datacenter. |
| int Replication\_factor | [R] |
| List<CfDef> Cf\_defs | [R]list of column family definitions. Can be empty, but not null |

public class CfDef : TBase

Usage: KsDef, system\_add\_column\_family(),system\_update\_column\_family()

|  |  |
| --- | --- |
| string Keyspace | [R] Keyspace this **CFDef** belongs to |
| string Name | [R] |
| string Column\_type | One of "Standard" or "Super". Default is “Standard” |
| string Comparator\_type | Name of comparator used for column sorting. Default is “BytesType”  “BytesType” Simple non-validating byte comparison  “AsciiType” Similar to BytesType, but validates that input is US-ASCII  “UTF8Type” UTF-8 encoded string comparison  “LongType” Compares values as 64 bit longs  “LexicalUUIDType” 128 bit UUID compared by byte value  “TimeUUIDType” Timestamp compared 128 bit version 1 UUID |
| string Subcomparator\_type | Name of comparator used for subcolumns (when column\_type="Super" only). Default is “BytesType” |
| string Comment | Human-readable description of column family |
| double Row\_cache\_size | Specify a fraction in [0;1] or an absolute number of rows to cache. Aquiles: “Do not use this on ColumnFamilies with large rows, or ColumnFamilies with high write:read ratios.” Default is 0.0 |
| double Key\_cache\_size | Specify a fraction in [0;1] or an absolute number of keys to cache. This only cache keys, not columns. Default is 200000.0 |
| double Read\_repair\_chance | probability with which read repairs should be invoked on non-quorum reads. Default is 1.0 |
| List<ColumnDef> Column\_metadata |  |
| int Gc\_grace\_seconds | time to wait before garbage collecting tombstones (deletion markers). Default is 864000 (10 days) |
| string Default\_validation\_class | You can specify it to use for validating the column values. Valid values are the same types listed for the comparator. Cassandra also can validate data on a per-column basis (ColumnDef::Validation\_class). Validators at the column level take precedence over the default validator specified at the column family level. |
| int Id |  |
| int Min\_compaction\_threshold | The min and max boundaries are the number of tables to attempt to merge together at once. Raising the minimum will make minor compactions take more memory and run less often, lowering the maximum will have the opposite effect. Default for Min is 4, for Max is 32. |
| int Max\_compaction\_threshold |
| int Row\_cache\_save\_period\_in\_seconds |  |
| int Key\_cache\_save\_period\_in\_seconds |  |
| int Memtable\_flush\_after\_mins | Default is 60 |
| int Memtable\_throughput\_in\_mb | Default is 1/8 the heapsize |
| double Memtable\_operations\_in\_millions | Default is throughput / 64 \* 0.3 |

public class ColumnDef : TBase

Column metadata defines attributes of the column. Note that the optional index\_name and index\_type must be set together to successfully create a secondary index for a column.

Usage: CfDef

|  |  |
| --- | --- |
| byte[] Name | [R] Binds a validation\_class and (optionally) an indexer to a column. |
| string Validation\_class | [R] Abstract type (like comparator) to check the column value. |
| IndexType Index\_type | Type of index. Currently the only valid value is KEYS. |
| string Index\_name | Name for the secondary index. |

public enum IndexType

Usage: ColumnDef

|  |
| --- |
| {  **KEYS**,  } |

public class ColumnParent : TBase

Путь к предку какого-то множества Column’ов, у которых этот предок общий. Это либо имя Column\_family, либо имя Column\_family + имя Super\_column. Как можно заметить, используется только в рамках ColumnFamilyConnection.

Usage: insert(), get\_slice(), multiget\_slice(), get\_indexed\_slices()

|  |  |
| --- | --- |
| string Column\_family | [R] |
| byte[] Super\_column |  |

public class ColumnPath : TBase

По сути это ColumnParent + Column.Name (который, почему-то, не является обязательным атрибутом).

Usage: get()

|  |  |
| --- | --- |
| string Column\_family | [R] |
| byte[] Super\_column |  |
| byte[] Column |  |

public enum ConsistencyLevel

Usage: insert(), get(), batch\_mutate(), get\_slice(), multiget\_slice(), get\_indexed\_slices()

|  |
| --- |
| {  **ONE** = 1,  **QUORUM** = 2,  **LOCAL\_QUORUM** = 3,  **EACH\_QUORUM** = 4,  **ALL** = 5,  **ANY** = 6,  } |

public class Mutation : TBase

A Mutation encapsulates either a column to insert, or a deletion to execute for a key

Usage: batch\_mutate()

|  |  |
| --- | --- |
| ColumnOrSuperColumn Column\_or\_supercolumn | [E] |
| Deletion Deletion | [E] |

public class Deletion : TBase

A Deletion encapsulates an operation that will delete all columns matching the specified timestamp and predicate. If super\_column is specified, the Deletion will operate on columns within the SuperColumn - otherwise it will operate on columns in the top-level of the key.

Usage: Mutation

|  |  |
| --- | --- |
| long Timestamp | [R] <текущее время> - <01.01.1970> (а не Timestamp удаляемого столбца, как сказано в [1]) |
| byte[] Super\_column |  |
| SlicePredicate Predicate |  |

public class SlicePredicate : TBase

Usage: Deletion, get\_slice(), multiget\_slice(), get\_indexed\_slices()

|  |  |
| --- | --- |
| List<byte[]> Column\_names | [E]Имена Column’ов, которые хотим получить (что если какой-то не нашелся?) |
| SliceRange Slice\_range | [E][Ignored if Column\_names is specified] |

public class SliceRange : TBase

Usage: SlicePredicate

|  |  |
| --- | --- |
| byte[] Start | [R] The column name to start the slice with. This attribute is not required, though there is no default value, and can be safely set to '', i.e., an empty byte array, to start with the first column name. Otherwise, it must be a valid value under the rules of the Comparator defined for the given ColumnFamily. |
| byte[] Finish | [R] The column name to stop the slice at. This attribute is not required, though there is no default value, and can be safely set to an empty byte array to not stop until count results are seen. Otherwise, it must also be a valid value to the ColumnFamily Comparator. |
| bool Reversed | [R] Whether the results should be ordered in reversed order. Similar to ORDER BY blah DESC in SQL. |
| int Count | [R] How many columns to return. Similar to LIMIT 100 in SQL. May be arbitrarily large, but Thrift will materialize the whole result into memory before returning it to the client, so be aware that you may be better served by iterating through slices by passing the last value of one call in as the start of the next instead of increasing count arbitrarily large. |

public class ColumnOrSuperColumn : TBase

Due to the lack of inheritance in Thrift, Column and SuperColumn structures are aggregated by the ColumnOrSuperColumn structure. This is used wherever either a Column or SuperColumn would normally be expected.

Usage: Mutation, get(), get\_slice(), multiget\_slice()

|  |  |
| --- | --- |
| Column Column | [E] |
| SuperColumn Super\_column | [E] |

public class SuperColumn : TBase

Usage: ColumnOrSuperColumn

|  |  |
| --- | --- |
| byte[] Name | [R] |
| List<Column> Columns | [R] |

public class Column : TBase

Usage: ColumnOrSuperColumn, SuperColumn, insert()

|  |  |
| --- | --- |
| byte[] Name | [R] |
| byte[] Value | [R] |
| long Timestamp | [R] can be anything you like, but microseconds since 1970 is a convention. Aquiles, если это поле не указано, помещает в него количество миллисекунд от 01.01.1970 |
| int Ttl | “time to live”. the column will expire after the requested amount of time (in seconds) and be deleted auto-magically. [8] |

public class IndexClause : TBase

Usage: get\_indexed\_slices()

|  |  |
| --- | --- |
| List<IndexExpression> Expressions | [R] The list of IndexExpression objects which must contain one EQ IndexOperator among the expressions |
| byte[] Start\_key | [R] Start the index query at the specified key - can be set to '', i.e., an empty byte array, to start with the first key |
| int Count | [R] The number of results to which the index query will be constrained |

public class IndexExpression : TBase

Usage: IndexClause

|  |  |
| --- | --- |
| byte[] Column\_name | [R] The column name to against which the operator and value will be applied |
| IndexOperator Op | [R] The IndexOperator to use. Currently only EQ is supported for direct queries, but other IndexExpression structs may be created and passed to IndexClause |
| byte[] Value | [R] The value for which to the column |

public enum IndexOperator

Usage: IndexExpression

|  |
| --- |
| {  **EQ**,  **GTE**,  **GT**,  **LTE**,  **LT**,  } |

public class KeySlice : TBase

Encapsulates a mapping of a key to the slice of columns for it. Используется только как возвращаемое значение методов get\_\*\_slices().

Usage: get\_indexed\_slices()

|  |  |
| --- | --- |
| byte[] Key |  |
| List<ColumnOrSuperColumn> Columns |  |

**Еще не использованные:**

public class KeyRange : TBase (принимается в get\_range\_slices)

|  |  |
| --- | --- |
| byte[] Start\_key |  |
| byte[] End\_key |  |
| string Start\_token |  |
| string End\_token |  |
| int Count |  |

public class TokenRange : TBase (возвращается из describe\_ring)

|  |  |
| --- | --- |
| string Start\_token |  |
| string End\_token |  |
| List<string> Endpoints |  |

public class AuthenticationRequest : TBase (принимается в login)

|  |  |
| --- | --- |
| Dictionary<string, string> Credentials |  |

## Описание сущностей клиента КОПФа

public class Keyspace

|  |  |
| --- | --- |
| Dictionary<string, ColumnFamily> ColumnFamilies |  |
| string Name |  |
| int ReplicationFactor |  |
| string ReplicaPlacementStrategy |  |

public class ColumnFamily

|  |  |
| --- | --- |
| double? RowCacheSize |  |
| string Name |  |
| int Id |  |
| List<IndexDefinition> Indexes |  |

public class IndexDefinition

|  |  |
| --- | --- |
| string Name |  |
| ValidationClass ValidationClass |  |

public enum ValidationClass

|  |
| --- |
| {  **Undefined**,  **UTF8Type**,  **LongType**,  } |

public class Column

|  |  |
| --- | --- |
| string Name |  |
| byte[] Value |  |
| int? TTL |  |
| long? Timestamp |  |

public class IndexExpression

|  |  |
| --- | --- |
| string ColumnName |  |
| IndexOperator IndexOperator |  |
| byte[] Value |  |

public enum IndexOperator

|  |
| --- |
| {  **EQ**,  **GTE**,  **GT**,  **LTE**,  **LT**  } |