BlockController & MultiBlockController

方法	方法描述	调用peer参数	BlockController	MultiBlockController
queryBlockFileSize	获取区块总文件大小 （单位：byte）			
fetchBlockHeight				
fetchTransactionNum				
getLatestBlockInfo				
rangeBlockInfoByBlockHeight				
getBlockData				
getBlockData				
getBlockDataWithTxId				
fetchBlockKeyRange				
getTransactionInfoByBlock				
getChainKeyByTxId				
queryValueByChainKeyAndTxId				
queryValueByChainKeyAndTxId				
queryTxId				
queryTxId				

ChaincodeInvoker

方法	参数对象属性
byte[] **query**(Invoker.Read invoker)	**chainSource** < *id* (string)， *chaincode* (string)， *channel* (string) >, **functionName** (string), **params** (string)
String **write**(Invoker.Write invoker)	**chainSource** (id, chaincode, channel), **functionName** (string), **async** (bool), **params** ([]byte)
String **simpleInvoke**(Invoker.Simple invoker)	**chainId** (string), **functionName** (string), **params** (JSONObject), **chainCode** (string), **channel** (string), **txId** (string)

1. query方法

query方法的参数首先转换成RemoteRequest：

```
RemoteRequest(
  chainId:              Read.chainSource.id
  functionName:         Read.functionName
  params:               {"Args":["query", Read.functionName, Read.params]}
  chainCode:            Read.chainsource.chaincode
  channel:              Read.chainSource.channel
  txId:                 ""
  write:                false
  async:                false
  proposal:             true
)
```

然后转换成最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", Read.functionName, Read.params]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        Read.functionName
  nonce_:               未知
}
```

2. write方法

write方法的参数首先转换成RemoteRequest：

```
RemoteRequest(
  chainId:              Write.chainSource.id
  functionName:         Write.functionName
  params:               {"Args":["invoke",
Write.functionName,base64.encode(Write.params)]}
  chainCode:            Write.chainsource.chaincode
  channel:              Write.chainSource.channel
  txId:                 ""
  write:                true
  async:                Write.async
  proposal:             true
)
```

然后转换成最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            true
  channelName_:         Write.chainSource.channel
  chaincodeName_:       Write.chainsource.chaincode
  chaincodeCtor_:       {"Args":["invoke",
Write.functionName,base64.encode(Write.params)]}
  transactionId_:       TransactionIdUtil.generateTxId(nonce, creator)
  isAsync_:             Write.async
  isProposal_:          true
  functionName_:        Write.functionName
  nonce_:               未知
}
```

3. simpleInvoke方法

simpleInvoke方法的参数首先转换成RemoteRequest：

```
RemoteRequest(
  chainId:              Simple.chainId
  functionName:         Simple.functionName
  params:               Simple.params.toJSONString()
  chainCode:            Simple.chainCode
  channel:              Simple.channel
  txId:                 Simple.txId
  write:                false
  async:                false
  proposal:             false
)
```

然后转换成最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Simple.channel
  chaincodeName_:       Simple.chainCode
  chaincodeCtor_:       Simple.params.toJSONString()
  transactionId_:       Simple.txId
  isAsync_:             false
  isProposal_:          false
  functionName_:        Simple.functionName
  nonce_:               未知
}
```

ChainCRUDService

> 查询相关

1. 根据Key 查询Value

```
ChainValue query(String key);
```

查询功能是通过调用chaincode来完成的，最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", "queryData", key]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        "queryData"
  nonce_:               未知
}
```

2. 根据 chain key 和 txId 锁定唯一的 chain value

```
ChainValue queryHistoryValue(String txId, String key);
```

查询功能是通过调用chaincode来完成的，最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", "queryHistoryValue", "[txId, key]"]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        "queryHistoryValue"
  nonce_:               未知
}
```

3. 根据 chain key 查询最新的 txId

```
String queryTxId(String key);
```

查询功能是通过调用chaincode来完成的，最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", "queryLatestTxId", key]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        "queryLatestTxId"
  nonce_:               未知
}
```

4. 根据 chain key 获取所有的 txId

```
List<String> queryTxIds(String key);
```

查询功能是通过调用chaincode来完成的，最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", "queryTxIds", key]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        "queryTxIds"
  nonce_:               未知
}
```

5. 获取当前的 chain key 所有的历史记录

```
List<ChainValueHistory> queryHistory(String key, SuperviseEnum superType);
```

第二个参数时监管类型，有两种：ordinary、supervise

如果是ordinary 类型，则传递给peer节点的参数为：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", "queryHistory", key]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        "queryHistory"
  nonce_:               未知
}
```

如果是supervise 类型，则传递给peer节点的参数为：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", "superviseQueryHistory", key]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        "superviseQueryHistory"
  nonce_:               未知
}
```

6. 判断当前Key在Chain上是否存在

```
boolean exist(String key);
```

查询功能是通过调用chaincode来完成的，最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", "batchExistKey", "[key]"]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        "batchExistKey"
  nonce_:               未知
}
```

7. 批量检查key是否存在

```
List<String> batchExist(List<String> keys);
```

查询功能是通过调用chaincode来完成的，最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", "batchExistKey", "[key1,key2,...]"]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        "batchExistKey"
  nonce_:               未知
}
```

8. 根据范围查询Chain所有存在的key

```
List<String> rangeQueryKeys(RangeKey rangeKey);
```

最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:            false
  channelName_:         Read.chainSource.channel
  chaincodeName_:       Read.chainsource.chaincode
  chaincodeCtor_:       {"Args":["query", "queryRangeKeys", "
{\"startKey\":startKey,\"endKey\":endKey}"]}
  transactionId_:       ""
  isAsync_:             false
  isProposal_:          true
  functionName_:        "queryRangeKeys"
  nonce_:               未知
}
```

9. 其他

```
List<ChainItem> batchQuery(List<String> keys);
```

最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:              false
  channelName_:           Read.chainSource.channel
  chaincodeName_:         Read.chainsource.chaincode
  chaincodeCtor_:         {"Args":["query", "batchQueryData", "
[key1,key2,...]"]}
  transactionId_:         ""
  isAsync_:               false
  isProposal_:            true
  functionName_:          "batchQueryData"
  nonce_:                 未知
}
```

```
List<ChainDirectItem> batchDirectQuery(List<String> keys);
```

最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:              false
  channelName_:           Read.chainSource.channel
  chaincodeName_:         Read.chainsource.chaincode
  chaincodeCtor_:         {"Args":["query", "batchDirectQuery", "
[key1,key2,...]"]}
  transactionId_:         ""
  isAsync_:               false
  isProposal_:            true
  functionName_:          "batchDirectQuery"
  nonce_:                 未知
}
```

```
List<ChainItem> batchSuperQuery(List<String> keys);
```

最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:              false
  channelName_:           Read.chainSource.channel
  chaincodeName_:         Read.chainsource.chaincode
  chaincodeCtor_:         {"Args":["query", "batchSuperviseQuery", "
[key1,key2,...]"]}
  transactionId_:         ""
  isAsync_:               false
  isProposal_:            true
  functionName_:          "batchSuperviseQuery"
  nonce_:                 未知
}
```

```
List<ChainItem> rangeQuery(RangeKey rangeKey);
```

最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:              false
  channelName_:           Read.chainSource.channel
  chaincodeName_:         Read.chainsource.chaincode
  chaincodeCtor_:         {"Args":["query", "queryRangeData", "
{\"startKey\":startKey,\"endKey\":endKey}"]}
  transactionId_:         ""
  isAsync_:               false
  isProposal_:            true
  functionName_:          "queryRangeData"
  nonce_:                 未知
}
```

```
List<RangeQueryData> batchRangeQuery(List<RangeKey> keys);
```

最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:              false
  channelName_:           Read.chainSource.channel
  chaincodeName_:         Read.chainsource.chaincode
  chaincodeCtor_:         {"Args":["query", "batchQueryRangeData", "
[{\"startKey\":startKey1,\"endKey\":endKey1},
{\"startKey\":startKey2,\"endKey\":endKey2},...]"]}
  transactionId_:         ""
  isAsync_:               false
  isProposal_:            true
  functionName_:          "batchQueryRangeData"
  nonce_:                 未知
}
```

> 数据上链

```
String insert(ChainWriteRequest request);
```

最终传递给peer节点的参数：

```
ProposalRequest{
  isInvoke_:              true
  channelName_:           Write.chainSource.channel
  chaincodeName_:         Write.chainsource.chaincode
  chaincodeCtor_:         {"Args":["invoke",
Write.functionName,base64.encode(Write.params)]}
  transactionId_:         TransactionIdUtil.generateTxId(nonce, creator)
  isAsync_:               Write.async
  isProposal_:            true
  functionName_:          Write.functionName
  nonce_:                 未知
}
```