RTI_TRANSACTION Function

Description

RTI_TRANSACTION enables transaction processing (commit and rollback) for those filing systems that support it. The program calls the native transaction mechanisms of the filing system. For example, if RTI_TRANSACTION is used against a TigerLogic D3 volume, then it will use D3's transaction commit and rollback. RTI_Transaction supports Linear Hash (RTP57, "traditional" OpenInsight table) , other multivalue (MVBFS), and SQL (DSBFS) volumes.

Syntax

status = RTI_TRANSACTION(volname, method, errStatus)

Parameters

The function has the following parameters:

Parameter	Description
Volname (in)	name/path of the volume or connection. The volume must be attached prior to calling RTI_TRANSACTION. For linear hash filing systems, this may be null ("").
method (in)	"BEGIN" to begin a transaction "COMMIT" to commit changes " ROLLBACK" to rollback changes to the starting point
errStatus (out)	error or status information passed back from the filing system

Returns

True (1) on success

False (0) on failure. If failed, see ErrStatus for details

Notes

RTI_TRANSACTION supports the MVBFS, DSBFS and RTP57 filing systems. It returns an error if used with a volume bound to any other filing system. RTI_TRANSACTION operates on a per volume basis. If you need to commit and rollback across volumes you should call the function for each volume. RTI_Transaction passes its calls to the native filing system. For a linear hash volume, RTI_TRANSACTION is identical to using the TRANSACT subroutine. You are responsible for configuring the underlying server to support transactions.

Examples

MVBFS

```
errStatus = ''
* Begin the transaction
* volname is the name you used in the connection / attach
* call attach_Table(“MVBFS_TEST”, ‘’, ‘’)
stat = rti_transaction(volname, 'BEGIN', errStatus)
* Read a record in a reversable manner
If stat then
stat = RTI_READU( f_customers, cust_id, cust_rec)
End
* Change it, write it back
If stat Then
cust_rec<1> = ' changed ' : timedate()
stat = RTI_WRITEU( f_customers, cust_id, cust_rec)
End
stat = RTI_READU( f_orders, order_id, orders_rec)
If stat Then
orders_Rec<1> = ' changed ' : timedate()
stat = RTI_WRITEU( f_orders, order_id, orders_rec)
End
* Success? else roll back
If stat then
stat = rti_transaction(volname, 'COMMIT', errStatus)
End Else
stat = rti_transaction(volname, 'ROLLBACK', errStatus)
End
```

SQL

```
* Begin the transaction
stat = rti_transaction(volname, 'BEGIN', errStatus)
If stat then
Read cust_rec from f_customers, cust_id Else
stat = false$
end
End
* Change it, write it back
If stat Then
cust_rec<1> = ' changed ' : timedate()
Write cust_rec On f_customers, cust_id Else
stat = false$
end
End
If stat then
Read orders_rec  from f_orders, order_id Else
stat = false$
end
End
* Change it, write it back
If stat Then
orders_rec<1> = ' changed ' : timedate()
Write orders_rec  On f_orders, order_id Else
stat = false$
end
End
* Success? else roll back
If stat then
stat = rti_transaction(volname, 'COMMIT', errStatus)
End Else
stat = rti_transaction(volname, 'ROLLBACK', errStatus)
End
Linear hash
errStatus = ''
* VOLNAME not required for linear hash use – pass “” instead
* Begin the transaction
stat = rti_transaction('', 'BEGIN', errStatus)
* Read a record in a reversable manner
If stat then
```

```
Lock f_customers, cust_id then
Read cust_rec from f_customers, cust_id Else
stat = false$
end
* Change it, write it back
If stat Then
cust_rec<1> = ' changed ' : timedate()
Write cust_rec On f_customers, cust_id Else
stat = false$
end
End
UnLock f_customers, cust_id Else null
End Else
stat = false$
End
end
If stat Then
Lock f_customers, cust_id then
Read orders_rec from f_orders, order_id Else
stat = false$
end
* Change it, write it back
If stat Then
orders_rec<1> = ' changed ' : timedate()
Write orders_rec On f_orders, order_id Else
stat = false$
end
End
UnLock f_orders, order_id Else null
End Else
stat = false$
End
end
* Success? else roll back
If stat then
stat = rti_transaction('', 'COMMIT', errStatus)
End Else
stat = rti_transaction('', 'ROLLBACK', errStatus)
End
```