# [MS-DRSR]: Directory Replication Service (DRS) Remote Protocol

Errata below are for Protocol Document Version V34.0 – 2015/10/16.

| **Errata Published*** | **Description** |
|---|---|
| 2016/01/25 | In Section 4.1.27.1.4, DRS_MSG_VERIFYREPLY_V1, updated RequestAttrs to RequiredAttrs in the rpEntInf array definition.<br><br>Changed from:<br>rpEntInf: An array of ENTINF structures that contain the attributes requested in the RequestAttrs field of the input DRS_MSG_VERIFYREQ_V1 structure if the corresponding name is verified.<br><br>Changed to:<br>rpEntInf: An array of ENTINF structures that contain the attributes requested in the RequiredAttrs field of the input DRS_MSG_VERIFYREQ_V1 structure if the corresponding name is verified. |
| 2016/01/25 | In Section 4.1.10.5.12, ProcessFsmoRoleRequest, updated the use of pMsgIn in the procedure implementation to match the procedure's signature (msgIn: DRS_MSG_GETCHGREQ_V10) and the use of NC to match the pNC member of the DRS_MSG_GETCHGREQ_V10 structure.<br><br>Changed from:<br><pre>        if(not FullReplicaExists(GetObjectNC(pMsgIn.NC)) and<br>           not pMsgIn.pPartialAttrSet = null)<br>           msgOut.ulExtendedRet := EXOP_ERR_PARAM_ERR<br>           return<br>        else if not GetFilteredAttributeSet()∩ pMsgIn.pPartialAttrSet = {}<br>    then</pre><br>Changed to:<br><pre>        if(not FullReplicaExists(GetObjectNC(msgIn.pNC^)) and<br>           not msgIn.pPartialAttrSet = null)<br>           msgOut.ulExtendedRet := EXOP_ERR_PARAM_ERR<br>           return<br>        else if not GetFilteredAttributeSet() ∩ msgIn.pPartialAttrSet = {}<br>    then</pre> |
| 2015/11/09 | In Section 4.1.18.2 (Server Behavior of the IDL_DRSRemoveDsServer Method), added pseudo code for method CleanupRODCStates to clean up read-only domain controllers in the IDL_DRSRemoveDsServer implementation.<br><br>Changed from: |

| Errata Published* | Description |
|---|---|
| | (code below) |

```
ULONG
   IDL_DRSRemoveDsServer(
       [in, ref] DRS_HANDLE hDrs,
       [in] DWORD dwInVersion,
       [in, ref, switch_is(dwInVersion)]
           DRS_MSG_RMSVRREQ *pmsgIn,
       [out, ref] DWORD *pdwOutVersion,
       [out, ref, switch_is(*pdwOutVersion)]
           DRS_MSG_RMSVRREPLY *pmsgOut);

   serverDn: unicodestring
   domainDn: unicodestring
   server: DSName
   ntdsdsa: DSName
   otherNtdsdsa: DSName
   spnsToRemove: set of unicodestring
   computerDn: unicodestring
   computer: DSName
   objectsToDelete: set of DSName
   rt: ULONG
   ValidateDRSInput(hDrs, 14)

   serverDn := pmsgIn^.V1.ServerDN
   domainDn := pmsgIn^.V1.DomainDN

   pdwOutVersion^ := 1
   pmsgOut^.V1.fLastDcInDomain = false

   /* Basic parameter validation */
   if dwInVersion ≠ 1 then
     return ERROR_INVALID_PARAMETER
   endif

   if serverDn = null or serverDn = "" then
     return ERROR_INVALID_PARAMETER
   endif

   /* Note that DomainDN may be null, but it cannot be empty. */
   if domainDn = "" then
     return ERROR_INVALID_PARAMETER
   endif

   /* Compute fLastDcInDomain if domainDn is non-null. */
   if domainDn ≠ null then
     otherNtdsdsa := select one o from subtree ConfigNC() where
       (o!objectCategory = nTDSDSA)
       and
       (domainDn in o!hasMasterNCs or domainDn in o!msDS-
hasMasterNCs)
       and
       (o ≠ ntdsdsa)
     if otherNtdsdsa = null then
       pmsgOut^.V1.fLastDcInDomain = true
     else
       pmsgOut^.V1.fLastDcInDomain = false
     endif
   endif

   /* If nothing to commit, processing is complete. */
```

| Errata Published* | Description |
|---|---|
| | |

```
   if not pmsgIn^.V1.fCommit then
     return 0
   endif

   ntdsdsa := DescendantObject([dn: serverDn], "CN=NTDS Settings,")
   if ntdsdsa = null then
     return ERROR_DS_CANT_FIND_DSA_OBJ
   endif

   /* Perform the actual DC metadata removal. */

   /* Locate the computer object for the DC's account. */
   server := ntdsdsa!parent
   computerDn := server!serverReference
   computer := null
   if computerDn ≠ null then
     computer := GetDSNameFromDN(computerDn)
   endif

   /* Remove the subtree of objects rooted at the DC's ntdsDsa
object.*/

   if not AccessCheckObject(ntdsdsa, RIGHT_DS_DELETE_TREE) then
     return ERROR_ACCESS_DENIED
   endif

   rt := RemoveObj(ntdsdsa,true)
   if rt ≠ 0 then
     return rt
   endif

   /* If the DC's computer account exists, remove rIDSet objects
and
    * remove the DRS SPNs from the computer object. */

   if computer ≠ null then
     foreach r in computer!rIDSetReferences
       if (not AccessCheckObject(r, RIGHT_DELETE)) and
          (not AccessCheckObject(r.parent, RIGHT_DS_DELETE_CHILD))
then
         return ERROR_ACCESS_DENIED
       endif

       RemoveObj(r, false)
     endfor

     foreach spn in computer!servicePrincipalName
       if StartsWith(spn, "ldap/") or
          StartsWith(spn, "GC/") or
          StartsWith(spn, "E3514235-4B06-11D1-AB04-00C04FC2DCD2/")
then
         spnsToRemove := spnsToRemove + {spn}
       endif
     endfor

     if not AccessCheckAttr(computer, servicePrincipalName,
         RIGHT_DS_WRITE_PROPERTY) then
       return ERROR_ACCESS_DENIED
     endif
```

| Errata Published* | Description |
|---|---|
|  | ```<br>    computer!servicePrincipalName :=<br>      computer!servicePrincipalName - spnsToRemove<br>  endif<br><br>  return 0<br>```<br>Changed to:<br>```<br>ULONG<br>IDL_DRSRemoveDsServer(<br>    [in, ref] DRS_HANDLE hDrs,<br>    [in] DWORD dwInVersion,<br>    [in, ref, switch_is(dwInVersion)]<br>        DRS_MSG_RMSVRREQ *pmsgIn,<br>    [out, ref] DWORD *pdwOutVersion,<br>    [out, ref, switch_is(*pdwOutVersion)]<br>        DRS_MSG_RMSVRREPLY *pmsgOut);<br><br>serverDn: unicodestring<br>domainDn: unicodestring<br>server: DSName<br>ntdsdsa: DSName<br>otherNtdsdsa: DSName<br>spnsToRemove: set of unicodestring<br>computerDn: unicodestring<br>computer: DSName<br>objectsToDelete: set of DSName<br>rt: ULONG<br>RODCKrbtgtAcct: DSName<br>accountList: set of DSName<br>ValidateDRSInput(hDrs, 14)<br><br>serverDn := pmsgIn^.V1.ServerDN<br>domainDn := pmsgIn^.V1.DomainDN<br><br>pdwOutVersion^ := 1<br>pmsgOut^.V1.fLastDcInDomain = false<br><br>/* Basic parameter validation */<br>if dwInVersion ≠ 1 then<br>  return ERROR_INVALID_PARAMETER<br>endif<br><br>if serverDn = null or serverDn = "" then<br>  return ERROR_INVALID_PARAMETER<br>endif<br><br>/* Note that DomainDN may be null, but it cannot be empty. */<br>if domainDn = "" then<br>  return ERROR_INVALID_PARAMETER<br>endif<br><br>/* Compute fLastDcInDomain if domainDn is non-null. */<br>if domainDn ≠ null then<br>  otherNtdsdsa := select one o from subtree ConfigNC() where<br>    (o!objectCategory = nTDSDSA)<br>``` |

| Errata Published* | Description |
|---|---|
| | (pseudocode below) |

```
        and
        (domainDn in o!hasMasterNCs or domainDn in o!msDS-
hasMasterNCs)
        and
        (o ≠ ntdsdsa)
      if otherNtdsdsa = null then
        pmsgOut^.V1.fLastDcInDomain = true
      else
        pmsgOut^.V1.fLastDcInDomain = false
      endif
    endif

    /* If nothing to commit, processing is complete. */
    if not pmsgIn^.V1.fCommit then
      return 0
    endif

    ntdsdsa := DescendantObject([dn: serverDn], "CN=NTDS Settings,")
    if ntdsdsa = null then
      return ERROR_DS_CANT_FIND_DSA_OBJ
    endif

    /* Perform the actual DC metadata removal. */

    /* Locate the computer object for the DC's account. */
    server := ntdsdsa!parent
    computerDn := server!serverReference
    computer := null
    if computerDn ≠ null then
      computer := GetDSNameFromDN(computerDn)
    endif

    /* Remove the subtree of objects rooted at the DC's ntdsDsa
object.*/

    if not AccessCheckObject(ntdsdsa, RIGHT_DS_DELETE_TREE) then
      return ERROR_ACCESS_DENIED
    endif

    rt := RemoveObj(ntdsdsa,true)
    if rt ≠ 0 then
      return rt
    endif

    /* If the DC's computer account exists, remove rIDSet objects
and
     * remove the DRS SPNs from the computer object. */

    if computer ≠ null then
      foreach r in computer!rIDSetReferences
        if (not AccessCheckObject(r, RIGHT_DELETE)) and
           (not AccessCheckObject(r.parent, RIGHT_DS_DELETE_CHILD))
then
          return ERROR_ACCESS_DENIED
        endif

        RemoveObj(r, false)
      endfor

      foreach spn in computer!servicePrincipalName
```

| Errata Published* | Description |
|---|---|
|  | (code below) |

```
       if StartsWith(spn, "ldap/") or
          StartsWith(spn, "GC/") or
          StartsWith(spn, "E3514235-4B06-11D1-AB04-00C04FC2DCD2/")
or
          StartsWith(spn, "RPC/") then
         spnsToRemove := spnsToRemove + {spn}
       endif
     endfor
     /* Cleanup for read-only domain controllers */

     /* Clear the KrbTgtLink from computer and delete its object */

     /* Get the msDS-KrbTgtLink attribute from the object */
     RODCKrbtgtAcct := computer!msDS-KrbTgtLink

     /* Delet the attribute from the object */
     Computer!msDS-KrbTgtLink := null

     /* Remove the KrbTgtLink */
     RemoveObj(RODCKrbTgtLink, false)

     /* Delete RODC policies */
     computer!msDS-NeverRevealGroup := null
     computer!msDS-RevealOnDemandGroup := null
     computer!msDS-RevealedUsers := null

     /* Delete msDS-AuthenticatedToAccountList links */
     accountList := { computer!msDS-AuthenticatedToAccountList }

     foreach entry in accountList
       entry!msDS-AuthenticatedAtDC := entry!msDS-AuthenticatedAtDC
- computer
     endfor

     if not AccessCheckAttr(computer, servicePrincipalName,
         RIGHT_DS_WRITE_PROPERTY) then
       return ERROR_ACCESS_DENIED
     endif

     computer!servicePrincipalName :=
       computer!servicePrincipalName - spnsToRemove
   endif

   return 0
```

* Date format: YYYY/MM/DD