了解NSO服务所有权

目录

简介

本文档介绍思科®网络服务协调器(NSO)中服务所有权的一般概念、常见缺陷和解决方案。

先决条件

要求

本文档适用于所有当前可用的NSO版本，包括NSO 6。所描述的行为仅适用于使用服务和非服务配置组合的NSO设置。虽然本文档示例中使用的特定命令仅适用于所使用的网络元件驱动程序(NED)，但基础逻辑适用于NSO管理的任何设备。

使用的组件

- NSO 6.1.6
- NED：测试版1.0 ，使用此yang模型的自定义构建netconf NED。

```
NED yang model:
module test-ned{
    namespace "http://example.org/ned/service-ownership";
    prefix ownership;
    import ietf-inet-types{ prefix inet;}

    list interface {
        key interface-name;
        leaf interface-name{
            type string;
        }

        leaf ip-address {
            type inet:ipv4-address;
        }

        leaf description {
            type string;
        }
    }
}
```

- 服务：example-service 1.0：使用此yang模型和模板的自定义构建服务

```
module example-service {
  namespace "http://com/example/exampleservice";
  prefix example-service;

  import ietf-inet-types {
    prefix inet;
  }
  import tailf-ncs {
    prefix ncs;
  }

  list example-service {
    key name;

    uses ncs:service-data;
    ncs:servicepoint "example-service";

    leaf name {
      type string;
    }
    leaf-list device {
      type leafref {
        path "/ncs:devices/ncs:device/ncs:name";
      }
    }
    leaf ipaddress {
      type inet:ipv4-address;
    }
```

```
  }
}
```

```
        {/device}
```

```
          FE1
```

```
          {/ipaddress}
```

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

基本信息

服务所有权的目的是允许NSO跟踪与哪种服务相关的配置。删除服务时，NSO需要删除与其相关的配置，它使用服务所有权来确定要删除的配置。当配置由多个服务拥有时，删除其中一个服务只会删除该所有权引用；配置本身仍保留在NSO数据库(CDB)和网络设备上。

所有权通过重新计数和反指针显示。refcount显示拥有该部分配置的图元数。如果配置也归设备所有，则refcount等于服务实例数加上1。反指针显示这些服务实例的路径。没有用于显示“设备所有”的反指针。只有列表和容器的CDB中才显示反指针。单个枝叶不会显示它们的反指针，但它们会继承自其父级。

设备所有

除了配置归服务所有之外，它还可以归设备所有。这有时称为“设备拥有”或“非服务拥有”。虽然本文档使用“设备所有”，但请注意，虽然这一点更易于理解，但非服务所有权不一定包括设备。LSA设置或堆叠服务可以具有无设备的非服务拥有的配置。

将配置添加到CDB而不使用服务部署，而是使用同步自、加载合并或ncs_cli等方法来设置配置时，配置由设备所有。当服务实例获得已拥有设备的配置的所有权时，refcount设置为2以反映共享的所有权。删除服务实例时，不会删除配置，尽管在删除之前只有一个服务实例拥有配置。此外，设备拥有的配置会添加“原始值”标记。如果服务实例覆盖设备拥有的配置并在以后删除该服务，则配置将恢复为原始值。

仅当通过非服务方式添加配置时，CDB中不存在该配置时，才会分配设备所有权。服务拥有的配置在同步源后不会成为设备拥有的配置。但是，如果在顶部部署服务，设备拥有的配置将变为设备拥有和服务拥有。

实际所有权

展示：仅服务拥有

当目标配置为空时部署服务时，服务将创建配置并获取其所有权。用户可以使用show running-config命令检查所有权并附加 | 显示服务元数据。虽然不是必填项，但建议也附加 | 将xml显示为默认CLI样式输出并不总是正确反映数据在CDB中的建模方式。

```
admin@ncs(config)# do show running-config devices device mydevice0 config
% No entries found.
admin@ncs(config)# example-service s1 device mydevice0 ipaddress 192.0.2.1
admin@ncs(config-example-service-s1)# commit dry-run
cli {
    local-node {
        data  devices {
                  device mydevice0 {
                      config {
```

```
             +            interface FE1 {
             +                ip-address 192.0.2.1;
             +            }
                      }
                  }
              }
             +example-service s1 {
             +    device [ mydevice0 ];
             +    ipaddress 192.0.2.1;
             +}
    }
}
admin@ncs(config-example-service-s1)# commit
Commit complete.
admin@ncs(config-example-service-s1)# do show running-config devices device mydevice0 config | display
```

```
          mydevice0
```

```
            FE1
```

```
            192.0.2.1
```

此外，如果添加第二个服务实例以相同配置为目标，则两个服务实例共享所有权。Refcount为2，且有2个反指针。

```
admin@ncs(config-example-service-s1)# example-service s2 device mydevice0 ipaddress 192.0.2.2
admin@ncs(config-example-service-s2)# commit dry-run
cli {
    local-node {
        data  devices {
                  device mydevice0 {
                      config {
                          interface FE1 {
            -                 ip-address 192.0.2.1;
            +                 ip-address 192.0.2.2;
                          }
                      }
                  }
             }
            +example-service s2 {
            +    device [ mydevice0 ];
            +    ipaddress 192.0.2.2;
            +}
    }
}
admin@ncs(config-example-service-s2)# commit
Commit complete.
admin@ncs(config-example-service-s2)# do show running-config devices device mydevice0 config | display
```

```
         mydevice0
```

```
            FE1
```

```
            192.0.2.2
```

展示：仅设备拥有

使用负载合并、ncs_cli或sync-from向CDB添加数据时，如果不使用服务，此数据将成为设备拥有的数据。refcount和backpointer被隐藏。

```
admin@ncs(config)# no example-service
admin@ncs(config)# commit
Commit complete.
admin@ncs(config)# do show running-config devices device mydevice0 config
% No entries found.
admin@ncs(config)# load merge merge-config.xml
Loading.
386 bytes parsed in 0.00 sec (137.22 KiB/sec)
admin@ncs(config)# commit dry-run
cli {
    local-node {
        data  devices {
                  device mydevice0 {
                      config {
             +            interface FE1 {
             +                ip-address 192.0.2.1;
```

```
              +             }
                      }
                  }
               }
    }
}
admin@ncs(config)# commit
Commit complete.
admin@ncs(config)# do show running-config devices device mydevice0 config | display service-meta-data |
```

```
          mydevice0
```

```
            FE1
```

```
            192.0.2.1
```

展示：自有设备+自有服务

此示例演示如何使用服务和同步来源轻松创建合并设备和服务所有权。

您部署该服务，然后使用Commit no-networking仅在CDB中删除该服务。这样，配置仍存在于终端设备上。当您执行同步源时，配置会添加回CDB，但它是设备拥有的，而不是服务拥有的。请记住，在NSO中运行时的show running-config显示CDB数据，而不是设备上当前的数据。

```
admin@ncs(config)# no devices device mydevice0 config
admin@ncs(config)# commit
admin@ncs(config)# do show running-config devices device mydevice0 config
% No entries found.
admin@ncs(config)# example-service s1 device mydevice0 ipaddress 192.0.2.1
admin@ncs(config-example-service-s1)# commit
admin@ncs(config-example-service-s1)# top
admin@ncs(config)# no example-service s1
admin@ncs(config)# commit no-networking
Commit complete.
admin@ncs(config)# devices device mydevice0 sync-from
result true
admin@ncs(config)# do show running-config devices device mydevice0 config | display service-meta-data |
```

```
          mydevice0
```

```
            FE1




            192.0.2.1
```

从同步后，配置仅归设备所有。Refcounter已隐藏。再次部署服务时，refcount变为2，但反向指针仅显示单个服务实例。第二个refcounter代表设备所有权。同样的规则适用于共享服务所有权，如果删除服务，则不会删除配置，因为设备也部分拥有配置。 此外，如果服务数据与服务元数据中存储的“原始值”不匹配，如果删除服务，NSO会将该值恢复为“原始值”。

```
admin@ncs(config)# example-service s1 device mydevice0 ipaddress 192.0.2.2
admin@ncs(config-example-service-s1)# commit dry-run
cli {
    local-node {
        data  devices {
                  device mydevice0 {
                      config {
                          interface FE1 {
            -                 ip-address 192.0.2.1;
            +                 ip-address 192.0.2.2;
                          }
                      }
                  }
             }
            +example-service s1 {
            +    device [ mydevice0 ];
            +    ipaddress 192.0.2.2;
            +}
    }
}
admin@ncs(config-example-service-s1)# commit
Commit complete.
admin@ncs(config-example-service-s1)# do show running-config devices device mydevice0 config | display
```

```
mydevice0


  FE1


  192.0.2.2
```

协调

语法:<path-to-service instance> re-deploy reconcile

可选标志:{ keep-non-service-config } dry-run { outformat native }

目的

协调功能的核心目的是允许用户摆脱不需要的设备所有权，并将所有权完全转移给服务。如果用户已经有一个正常运行的网络，并且他们尝试将所有权转移到NSO，他们通常会首先通过同步来源操作引入配置。一旦CDB拥有所有网络配置，用户就会在现有配置之上部署服务实例。此时，配置仍归设备所有，这会限制服务删除配置的能力。当用户希望让其服务完全拥有配置时，他们可以使用协调功能，该功能可执行3项操作。

1)将所有权转移至服务

2)2)删除“原始值”标志

3)纠正服务所有权

将所有权转移至服务

Reconcile将评估服务拥有的所有配置，如果它发现任何配置属于此服务和设备所有或者属于其他非服务拥有，则会移除此设备所有权，使服务成为独占所有者。将refcount减少1。

注意：如果2个服务拥有部分配置，并且该配置也是非服务拥有的，则参考计数为3。对其中任何一项服务进行协调都会移除该非服务所有权，将参考数减少到2以反映这两种服务。

删除原始值标志

当部署服务并覆盖非服务拥有的数据时，refcount变为2，并且NSO会添加“原始值”标记。如果服务实例被删除，NSO会尝试恢复为服务之前存在的原始值。

在协调过程中，不仅会减少引用计数，而且会删除原始值。现在删除服务会将该值变为空值或者将其更改为由yang模型定义的默认值。

更正服务所有权

在某些情况下，所有权分配可能会不正确。设备拥有配置不正确归服务所有，或者配置不正确归服务和设备所有，而预期只有服务拥有。 协调可以纠正这些不对齐。这对于避免服务删除非服务拥有配置的问题非常重要。

协调为重新部署

协调是重新部署服务的子功能。如果服务与CDB不同步，协调操作除了执行协调功能外，还会执行重新部署。

协调内部工作

虽然只有开发人员才知道协调操作的确切细节，但本文档提供了简化的了解：

1)NSO更正服务所有权

2)NSO从CDB中删除此服务实例所拥有的所有配置，即使它也是其他服务所拥有或设备所拥有

3)NSO重新部署服务实例

4)NSO从其他服务恢复服务反射和回指针

涉及服务所有权的问题

协调失败

如果您发现“重新部署”可以正常工作，但“重新部署协调”失败：这可能表示您的服务设计与协调功能运行方式之间发生了冲突。

问题源于服务代码，该代码尝试从CDB读取配置，然后服务部署了该配置。您只能部署此服务，因为在部署之前，CDB中已存在部分配置。但在协调期间，NSO会临时删除此服务拥有的所有配置，包括在下一步重新部署服务期间服务尝试读取的配置。这通常导致java或python错误报告读取数据失败。

服务正在删除非服务拥有的配置

在此场景中，您会在删除或重新部署服务实例的过程中遇到NSO删除非服务拥有的配置。这是因为服务实例已创建并拥有原始配置，而您稍后会手动（通过ncs_cli、sync-from或其他方法）将部分配置添加到服务拥有的容器或列表中。

此新配置不应由服务拥有，但由于服务拥有容器或列表的完全所有权，因此服务最终间接拥有该容器。

解决此问题的方法是使用re-deploy reconcile { keep-non-service-config }（重新部署协调{ keep-non-service-config }）更正服务所有权。进行此协调时，容器或列表的引用计数增加，以反映此容器或列表同时具有服务所有子节点和非服务所有子节点。在父节点内部，只有服务拥有的节点具有引用计数和反向指针。

展示：服务正在删除非服务拥有的配置

从部署了完全所有权的单个服务实例开始，使用ncs_cli手动向接口添加说明。

```
admin@ncs(config-example-service-s1)# do show running-config devices device mydevice0 config | display
```

```
        mydevice0




          FE1




          192.0.2.1




admin@ncs(config-example-service-s1)# top
admin@ncs(config)# devices device mydevice0 config interface FE1 description "This is an example descri
admin@ncs(config-interface-FE1)# commit
Commit complete.
admin@ncs(config-interface-FE1)# do show running-config devices device mydevice0 config | display servi
```

```
        mydevice0




          FE1




          192.0.2.1




          This is an example description
```

请注意，即使添加了设备拥有的配置，<interface>上的refcount仍然为1。尝试删除服务实例时，即使描述不应该是服务实例的一部分，描述也会被删除。要避免这种情况，您可以运行reconcile命令。

```
admin@ncs(config-interface-FE1)# top
```

```
admin@ncs(config)# no example-service s1
admin@ncs(config)# commit dry-run
cli {
    local-node {
        data  devices {
                  device mydevice0 {
                      config {
             -            interface FE1 {
             -                ip-address 192.0.2.1;
             -                description "This is an example description";
             -            }
                      }
                  }
              }
             -example-service s1 {
             -    device [ mydevice0 ];
             -    ipaddress 192.0.2.1;
             -}
    }
}
admin@ncs(config)# abort
admin@ncs# config
Entering configuration mode terminal
admin@ncs(config)# example-service s1 re-deploy reconcile { keep-non-service-config }
admin@ncs(config)# do show running-config devices device mydevice0 config | display service-meta-data |
```

mydevice0

FE1

```
            192.0.2.1




            This is an example description
```

协调后，list接口的refcount已增加到2。同时，枝叶ip-address的重新计数仍为1。列表条目“interface FE1”包含服务拥有数据和非服务拥有数据。通过使用协调，NSO将重新评估所有权并相应地分配重新计数。现在，删除仅针对服务实例完全拥有的区域。说明或列表条目都不会被删除。

```
admin@ncs(config)# no example-service s1
admin@ncs(config)# commit dry-run
cli {
    local-node {
        data  devices {
                  device mydevice0 {
                      config {
                          interface FE1 {
             -                ip-address 192.0.2.1;
                          }
                      }
                  }
              }
             -example-service s1 {
             -    device [ mydevice0 ];
             -    ipaddress 192.0.2.1;
             -}
    }
}
```

Keep-non-service-config vs Discard-non-service-config

用户有时会误解使用discard-non-service-config。

在协调示例中，使用了“keep-non-service-config”。如果使用discard，则如下所示：

```
admin@ncs(config)# example-service s1 re-deploy reconcile { discard-non-service-config } dry-run
cli {
    local-node {
        data  devices {
                    device mydevice0 {
                        config {
                            interface FE1 {
              -                 description "This is an example description";
                            }
                        }
                    }
                }

    }
}
```

默认值为“keep-non-service-config”。如果两个选项均未定义，则NSO默认为保留。Discard很少使用，因为大多数用户希望保留其网络上的内容，即使它不由NSO管理。Reconcile { discard-non-service-config } dry-run可用于查找CDB中存在哪些不是服务配置的一部分但会在删除或重新部署服务时删除的数据点。

不创建

当与非服务拥有的数据混合使用时，可以使用重新部署协调来更正服务所有权，替代方法是使用nocreate标记来防止冲突。

这是可以在XML服务模板中使用的标记。文档规定“nocreate：合并仅影响模板中已存在的配置项。它从不创建具有此标记的配置。它只修改现有的配置结构。”

使用此标记会产生有趣的副作用：因为服务不创建节点，所以它不拥有节点的所有权。

这通常用于防止NSO尝试删除设备不允许删除的配置的情况。

请注意，此标记由子节点继承，这意味着如果将nocreate标记添加到接口，则此标记也应用于该接口内的任何节点，除非您使用其他标记（例如merge”标记）

向服务模板添加nocreate标记。如果接口FE1不存在，则不配置任何内容。

```
        {/device}


          FE1


          {/ipaddress}
```

重新编译并重新加载软件包，然后测试。

```
admin@ncs(config)# do show running-config devices device mydevice0 config | display service-meta-data |
No entries found.
admin@ncs(config)# example-service s1 device mydevice0 ipaddress 192.0.2.1
admin@ncs(config-example-service-s1)# commit dry-run
cli {
    local-node {
```

```
        data +example-service s1 {
             +    device [ mydevice0 ];
             +    ipaddress 192.0.2.1;
             +}
    }
}
```

即使定义了与之前相同的参数，设备配置中也不会创建接口或任何底层配置。

合并

在接口内部的配置上添加合并标记。请勿向“interface-name”添加标记，因为这是列表接口的密钥。必须始终允许密钥继承列表的行为。重新编译并重新加载软件包。

```
         {/device}
```

```
           FE1
```

```
           {/ipaddress}
```

在部署服务之前手动配置接口FE1。

```
admin@ncs(config)# no example-service
admin@ncs(config)# commit
admin@ncs(config)# do show running-config devices device mydevice0 config | display service-meta-data |
No entries found.
admin@ncs(config)# devices device mydevice0 config interface FE1
admin@ncs(config-interface-FE1)# commit dry-run
cli {
    local-node {
        data  devices {
                  device mydevice0 {
                      config {
             +            interface FE1 {
             +            }
                      }
                  }
              }
    }
}
admin@ncs(config-interface-FE1)# commit
Commit complete.
admin@ncs(config-interface-FE1)# top
admin@ncs(config)# example-service s1 device mydevice0 ipaddress 192.0.2.1
admin@ncs(config-example-service-s1)# commit dry-run
cli {
    local-node {
        data  devices {
                  device mydevice0 {
                      config {
                          interface FE1 {
             +                ip-address 192.0.2.1;
                          }
                      }
                  }
              }
             +example-service s1 {
             +    device [ mydevice0 ];
             +    ipaddress 192.0.2.1;
             +}
    }
}
admin@ncs(config-example-service-s1)# commit
```

```
Commit complete.
admin@ncs(config-example-service-s1)# do show running-config devices device mydevice0 config | display
```

```
        mydevice0
```

```
          FE1
```

```
          192.0.2.1
```

接口具有隐藏的refcount 1：接口是使用ncs_cli部署的，但它在服务包中具有nocreate标记；服务未获得所有权。它是设备拥有的。

主要有refcount 1：它只归服务所有

如果删除服务实例，则只会删除ipaddress，因为它是唯一完全由服务拥有的部分。

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。