通道capabilities

受众: 通道管理员, 节点管理员

注意：这是一个进阶的Fabric概念， 新用户或应用程序开发人员不一定要理解它。 然而，随着通道和网络的成熟，理解和管理capabilities变得至关重要。 此外，要认识到，尽管更新capabilities与升级节点通常是相关的， 但这两者是不同的过程，这是相当重要的。 我们将在本章节中对此进行详细描述。

因为Fabric是一个通常涉及多个组织的分布式系统， 所以不同版本的Fabric代码可能(而且通常) 存在于网络中的不同节点以及该网络中的通道上。 Fabric允许这种情况—— 即不需要每个peer节点和排序节点都处于相同的版本。 实际上，正因为其支持不同的版本 才使得Fabric节点能够滚动升级。

重要的是，网络和通道以相同的方式来处理事情， 例如为通道配置更新和链代码调用等事情创建确定性结果。 如果没有确定性的结果， 通道上的一个peer节点可能使一笔交易失效， 而另一个peer节点可能确认该交易。

为此，Fabric定义了所谓的“capabilities”。 这些capabilities在每个通道的配置中定义， 通过定义行为会产生一致结果的版本，确保了确定性。 正如您将看到的，这些capabilities的版本与节点二进制版本密切相关。 capabilities使运行在不同版本的节点能够 以兼容和一致的方式在给定特定区块高度的通道配置下运行。 您还将看到，配置树的许多部分都存在capabilities， 这些capabilities是根据特定任务的实施而定义的。

正如您将看到的， 有时需要将您的通道更新到新的capabilities版本以启用新特性。

节点版本和capabilities版本

如果您熟悉Hyperledger Fabric，就会意识到它遵循典型的语义版本模式：v1.1、v1.2.1和v2.0等等。这些版本指的是发行版及其相关的二进制版本。

capabilities遵循相同的语义版本约定。 虽然它也有v1.1 capabilities、v1.2 capabilities和2.0 capabilities等等。 但重要的是注意一些区别。

- **每次发布版本不一定都有一个新的capability level**。 建立新 capabilities的需求取决于具体情况本身， 主要依赖于新特性和旧二进制版本的向后兼容性。 例如，在v1.4.1中添加Raft排序服务 并没有改变交易或排序服务 capabilities的处理方式， 因此不需要设定任何新 capabilities。另一方面， 私有数据 在v1.2之前无法被peer节点处理，因此需要设定v1.2 capability level。 因为不是每次版本发布都包含 改变交易处理方式的新特性(或bug修复)， 所以某些发布的版本不需要任何新 capabilities(例如，v1.4)， 而其他版本只需要特定的新capabilities(例如v1.2和v1.3)。 稍后我们将讨论capabilities的“level”以及它们位于配置树中的位置。
- **在通道中，节点版本必须不能低于特定的capability版本**。 当一个peer节点加入一个通道时，它将顺序读取账本中的所有区块，从通道的创世块开始，持续经过交易区块和所有后续配置区块。如果节点(例如peer节点)试图读取一个包含其不理解的capabilities的更新的区块(例如，v1.4.x中peer节点试图读取包含v2.0应用程序capabilities的区块)，那么**peer节点将崩溃**。这种崩溃行为是故意如此设计的，因为v1.4.x的peer节点不应该试图验证或提交任何超前于该版本的交易。在加入通道之前，**请确**

保该节点不低于与该节点相关的通道配置中指定的capabilities的Fabric版本(二进制)。稍后我们将讨论哪些capabilities与哪些节点相关。然而，由于没有用户希望他们的节点崩溃，所以强烈建议在尝试更新capabilities之前将所有节点更新到所需的版本(最好是更新到最新版本)。这符合Fabric的默认建议，即**始终**使用最新的二进制和capabilities版本。

如果用户不能升级他们的二进制文件，那么capabilities必须保留在较低的版本。 较低版本的二进制文件和capabilities仍然会像它们所期望的那样一起工作。 然而，请记住，即使用户选择不更新它们的capabilities， 始终更新到新的二进制文件也仍是一个最佳做法。 因为capabilities本身也包括一些缺陷修复， 所以总是建议一旦网络中的二进制文件支持capabilities后就更新它们。

capabilities配置分类

正如我们前面讨论的，不存在包含整个通道的单一capability level。相反，有三种capabilities，每一种都代表一个管理区域。

- **排序节点**: 这些capabilities管理排序服务独有的任务和处理。 由于这些capabilities不涉及影响交易或peer节点的操作， 因此更新它们仅能由排序服务管理员完成(peer节点不需要了解排序节点capabilities， 因此无论排序节点capabilities更新了什么，都不会发生崩溃)。 注意，这些capabilities在v1.1和v1.4.2之间没有改变。 然而，正如我们将在**通道**一节中看到的， 这并不意味着v1.1排序节点能在capability level低于v1.4.2的所有通道上运作。
- **应用程序**: 这些capabilities管理peer节点独占的任务和处理。 因为排序服务管理员在决定peer节点组织之间的交易性质方面没有任何作用， 因此更改此capability level只能由peer节点组织来完成。例如， 只能在启用了v1.2(或更高版本)应用程序组capabilities的通道上启用私有数据。 在有私有数据的情况下，这是必须被启用的唯一capabilities， 因为私有数据的工作方式不需要更改通道管理或排序服务处理交易的方式。
- **通道**: 此分类包含由peer节点组织和排序服务**共同管理**的任务。 例如，这个capabilities定义了进行通道配置更新的level， 这些更新由peer节点组织发起并由排序服务排序。 在实际层面上，**这个分类定义了通道中所有二进制文件的最小level，因为排序节点和peer节点都必须不能低于与此capabilities相对应的二进制版本，才能运行该capabilities**。

通道的**排序节点**和**通道** capabilities默认情况下是从排序系统通道继承的， 其中修改它们是排序服务管理员的专属权限。因此， peer节点组织应该在将其peer节点加入该通道之前检查通道的创世块。 虽然通道capabilities是由排序系统通道的排序节点(正如联盟成员那样)管理的, 但是排序管理员将与联盟管理员协同确保仅在联盟准备好后才升级通道capabilities， 这是典型的和符合预期的。

因为排序系统通道没有定义一个**应用程序**capabilities， 所以在为通道创建创世块时，必须在通道配置文件中指定此capabilities。

当指定或修改应用程序capabilities时，**请谨慎**。因为排序服务不验证capability level是否存在，所以它将允许创建(或修改)通道来包含例如v1.8这样的应用程序capabilities，即使不存在这种capabilities。任何peer节点在试图读取一个含此capabilities的配置块将会崩溃，即使可以再次修改该通道为合法的capability level也没有意义，因为没有peer节点能够通过含无效的v1.8 capabilities的区块。

要全面了解当前有效的排序节点、应用程序和通道capabilities，请查看示例`configtx.yaml`文件，这些在“通道capabilities”一节中列出。

有关capabilities的更多具体信息以及它们处于通道配置中的位置， 请查看定义capabilities要求。