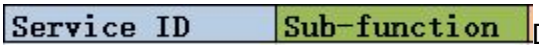
# UDS 诊断教程（二）

UDS 定义的诊断服务从逻辑来说分为以下几类：

1. Diagnostic and Communication Management （诊断和通信管理）
2. Data Transmission （数据传输）
3. Stored Data Transmission （存储数据传输，用于操作 DTC）
4. InputOutput Control （IO 控制）
5. Routine Control （不知如何翻译好，作用是调用 ECU 内部的预置函数）
6. Upload Download （上传下载）

UDS 规定使用 1 个 byte 来表示诊断服务，即所谓的 Service ID,简称 SID。本文介绍一下 Diagnostic and Communication Management 这一类诊断服务中的一部分。

**DiagnosticSessionControl (0x10)**



DiagnosticSessionControl 诊断 request 的格式

DiagnosticSessionControl 这个服务的 SID 是 0x10，request 固定为 2 个 byte，第一个 byte 是 SID，第二个 byte 的低 7bit 是 sub-function，用于指示 ECU 将进入的 session。

UDS 定义的 session 包括：

0x00 ISOSAEReserved（保留）

0x01 defaultSession

0x02 ProgrammingSession

0x03 extendedDiagnosticSession

0x04 safetySystemDiagnosticSession

0x05 – 0x3F ISOSAEReserved（保留）

0x40 – 0x5F vehicleManufacturerSpecific（由整车厂自定义使用）

0x60 – 0x7E systemSupplierSpecific（由 ECU 供应商自定义使用）

0x7F ISOSAEReserved（保留）

DiagnosticSessionControl 用于控制 ECU 在不同的 session 之间进行转换，session 可以看作是 ECU 所处的一种软件状态，在不同的 session 中诊断服务执行的权限不同。 ECU 上电之后，默认处在 defaultSession 中，在这个 session 中很多诊断服务不可以执行，很多诊断相关的数据不能读取或写入。一般的诊断仪启动之后，会给 ECU 发送 10 03，即

让 ECU 进入 extendedDiagnosticSession 中，在这个 session 中可执行的诊断服务就很多了。而如果要让 ECU 保持在 non-defaultSession 中，则需要诊断仪每隔固定的时间发送 0x3E 服务，ECU 才会知道诊断仪有和自己通信的需求，从而保持在 non-defaultSession 中。另一个常用的 session 是 ProgrammingSession，在这个 session 中可以进行软件刷写的一系列诊断服务。0x40 – 0x5F 这个范围中的 session 由整车厂自定义使用，比如，某些诊断服务或诊断数据的操作需要在生产线上执行，即所谓的 End-Of-Line，整车厂可以从这个范围中选择一个值来表示 EOL session；又或者在开发阶段需要某种“超级”session，则也可以从这里选一个值用来使 ECU 进入开发模式的 session。DiagnosticSessionControl 这个服务非常简单，但是它却是 ECU 和诊断通信的第一条诊断命令。

DiagnosticSessionControl 诊断 response 的格式

这个诊断服务的 response 分为三部分，第一部分是 0x50，作为 SID 的 echo；第二部分是进入的 session，作为 sub-function 的 echo；第三部分是 4 个字节，前两个字节代表 P2Server_max，即 ECU 在应用层上对诊断命令的响应时间，后两个字节代表 P2*Server_max

，即 ECU 在暂时无法处理当前诊断命令（具体表现为发送了 NRC 0X78），在应用层上对诊断命令响应的最长时间。

**ECUReset (0x11)**

ECUReset 这条指令的用途是通过诊断请求使 ECU 重启。

ECUReset 诊断 request 的格式

ECUReset 这个服务的 SID 是 0x11，request 固定为 2 个 byte，第一个 byte 是 SID，第二个 byte 的低 7bit 是 sub-function，用于指示 ECU 将模拟哪种方式进行重启。

常用的 sub-function 包括（只举 2 个例子，UDS 还定义了很多其他的值）

**0x01** hardReset 模拟 KL30 的重启

**0x02** keyOffOnReset 模拟 KL15 的重启

当我们通过诊断命令改写了 ECU 的某些数据，或者对 ECU 进行了某些设置，只有将 ECU 重启才能将这些配置生效，所以就有了这个诊断命令。在 ECUReset 执行之后，ECU 会从 Non-defaultsession 回退到 defaultsession 中。

**SecurityAccess (0x27)**

厂家可能会为 ECU 定义某些安全级别稍微高一些的诊断服务，在执行此类服务之前，就需要执行 SecurityAccess 这个诊断命令，进行一个简单的身份验证。

完成 SecurityAccess 有以下步骤：

1. 诊断仪向 ECU 请求“Seed”（通常是一个与时间相关的伪随机数），
2. ECU 向诊断仪发送“Seed”,
3. 诊断仪向 ECU 发送“Key” (根据请求得到的 Seed 和一个本地的密码进行计算得来)
4. ECU 判断诊断仪发来的“Key”是否有效

根据 UDS 的定义，0x03, 0x05, 0x07 – 0x41 这个范围留给用于 requestSeed 的 sub-function；0x04, 0x06, 0x08 – 0x42 这个范围留给用于 sendKey 的 sub-function。具体选择哪对值，由整车厂自己定义。整车厂也可以选择多对 sub-function，用于不同等级的安全访问。

下面我举一个完成 SecurityAccess 的诊断命令的例子，假设 0x05 用于 requestSeed，0x06 用于 sendKey。

诊断仪发送 27 05

ECU 响应 67 05 01 01 01（seed 是 01 01 01）

诊断仪发送 27 06 02 03 04（key 值是 02 03 04，seed 是 01 01 01，假设本地密码为 01 02 03，而算法就是将密码与 seed 相加）

ECU 验证成功 67 06

此时 ECU 就处于 unlocked 的状态了，那些被保护起来的诊断服务和诊断数据可以被操作了。通常来说，如果 ECU 重启，或者回到了 default session，unlocked 状态就失效了，如果要执行相关诊断服务，则需要再次执行上面描述的过程。

时间有限，这篇文章里就介绍这三个诊断服务，即 0x10, 0x11, 0x27，后续有时间我再补充其他的服务。