国际标准

ISO 14229-1

第二版2013-03-15

道路车辆 – 统一诊断服务（UDS） –

第1部分：规格和要求

Véhiculesroutiers – 诊断统一服务（SDU） – Partie 1: Spécificationet exigences

参考号ISO 14229-1：2013（E）

内容

页

前言

ISO（国际标准化组织）是国际标准组织（ISO成员机构）的全球联合会。 制定国际标准的工作通常通过ISO技术委员会进行。 对建立技术委员会的主题感兴趣的每个成员团体均有权参加该委员会的代表。 与ISO有联系的国际组织，政府和非政府组织也参与了这项工作。 ISO与国际电工委员会（IEC）就电工标准化的所有事宜密切合作。

国际标准是根据ISO / IEC指令第2部分给出的规则起草的。

技术委员会的主要任务是制定国际标准。 技术委员会通过的国际标准草案分发给各成员机构进行投票。 作为国际标准出版需要至少75％的成员机构投票批准。

请注意本文件的某些内容可能成为专利权的主题。 ISO不负责识别任何或所有此类专利权。

ISO 14229-1由技术委员会ISO / TC 22，道路车辆，小组委员会SC 3，
电气和电子设备。

本第二版取消并取代了经过技术修订的第一版（ISO 14229-1：2006）。

ISO 14229由以下部分组成，总标题为道路车辆 - 统一诊断服务（UDS）：

第1部分：规范和要求

第2部分：会话层服务

第3部分：CAN实施的统一诊断服务（UDSonCAN）

第4部分：FlexRay实现的统一诊断服务（UDSonFR）

第5部分：Internet协议实施的统一诊断服务（UDSonIP）

第6部分：K线实施的统一诊断服务（UDSonK-Line）

以下部分正在准备中：

第7部分：本地互联网络实施（UDSonLIN）上的统一诊断服务

未来部分的标题将起草如下：

— *第n部分：统一诊断服务在...实施（UDSon ...）*

介绍

无论串行数据链路是什么，ISO14229都是为了定义诊断系统的通用要求而建立的。

为此，ISO 14229基于开放系统互连（OSI）基本参考模型，符合ISO 7498-1和ISO / IEC 10731标准，该标准将通信系统构建为七层。 当映射到该模型时，诊断测试仪（客户端）和电子控制单元（ECU，服务器）使用的服务按照表1分为以下几个层次：

— 应用层（第7层），ISO 14229-1，ISO 14229-3 UDSonCAN，ISO 14229-4 UDSonFR，ISO 14229-5 UDSonIP，ISO 14229-6 UDSonK-Line，ISO 14229-7中规定的统一诊断服务 UDSonLIN，进一步标准和ISO 27145-3 WWH-OBD。

— 表示层（第6层），特定车辆制造商，ISO 27145-2 WWH-OBD。

— 会话层服务（第5层）在ISO 14229-2中规定。

— ISO 15765-2 DoCAN中规定的传输层服务（第4层），FlexRay上的ISO 10681-2通信，ISO 13400-2 DoIP，ISO 17987-2 LIN，ISO 27145-4 WWH-OBD。

— 在ISO 15765-2 DoCAN中规定的网络层服务（第3层），在FlexRay上的ISO 10681-2通信，ISO 13400-2 DoIP，ISO 17987-2 LIN，ISO 27145-4 WWH-OBD。

— 数据链路层（层2），在ISO 11898-1，ISO 11898-2，ISO 17458-2，ISO 13400-3，IEEE 802.3，ISO 14230-2，ISO 17987-3 LIN和其他标准ISO 27145- 4 WWH-OBD。

— 在ISO 11898-1，ISO 11898-2，ISO 17458-4，ISO 13400-3，IEEE 802.3，ISO 14230-1，ISO 17987-4 LIN和其他标准ISO 27145-4中规定的物理层（层1） WWH-OBD。

注：本标准中的诊断服务适用于各种应用，例如道路车辆 – 行驶记录仪系统，道路车辆 – 牵引车和牵引车之间电气连接的数字信息交换，道路车辆 – 诊断系统等。该标准提供了与上述实现标准的长期向后兼容性。

表1 – 适用于OSI层的诊断/编程规范示例

<table>
<tr><th>适用性</th><th>OSI七层</th><th colspan="6">增强的诊断服务</th><th>WWH-OBD</th></tr>
<tr><td rowspan="8"></td><td>应用程序（第7层）</td><td colspan="6">ISO 14229-1，ISO 14229-3 UDSonCAN，ISO 14229-4 UDSonFR，ISO 14229-5 UDSonIP，ISO 14229-6 UDSonK-Line，ISO 14229-7 UDSonLIN，进一步标准</td><td>ISO 27145-3</td></tr>
<tr><td>演示文稿（第6层）</td><td colspan="6">车辆制造商具体</td><td>ISO 27145-2</td></tr>
<tr><td>会话（第5层）</td><td colspan="7">ISO 14229-2</td></tr>
<tr><td>运输（第4层）</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td>进一步标准</td><td rowspan="4"></td></tr>
<tr><td>网络（第3层）</td><td>进一步标准</td></tr>
<tr><td>数据链接（第2层）</td><td rowspan="2"></td><td>ISO 17458-2</td><td rowspan="2"></td><td>ISO 14230-2</td><td>ISO 17987-3</td><td>进一步标准</td></tr>
<tr><td>物理（第1层）</td><td>ISO 17458-4</td><td>ISO 14230-1</td><td>ISO 17987-4</td><td>进一步标准</td></tr>
</table>

道路车辆 - 统一诊断服务（UDS） -

第1部分：
规格和要求

1 范围

ISO 14229的这一部分规定了诊断服务的数据链路独立要求，诊断服务允许诊断测试仪（客户端）控制车载电子控制单元（ECU，服务器）中的诊断功能，例如电子燃油喷射，自动变速箱，防抱死制动系统等连接到嵌入在公路车辆中的串行数据链路。

它指定了通用服务，它允许诊断测试程序（客户端）停止或恢复数据链路上的非诊断消息传输。

ISO 14229的本部分不适用于两个电子控制单元之间的车辆通信数据链路上的非诊断消息传输。 然而，ISO 14229的这部分内容并未限制ECU中的车载车载测试仪（客户端）实施，以便利用车辆通信数据链路上的诊断服务来执行双向诊断数据交换。

本部分ISO 14229没有规定任何实施要求。

2 规范性参考文献

以下参考文件对于本文件的应用是必不可少的。 凡是注日期的引用文件，仅注明引用的版本。 凡是不注日期的引用文件，其最新版本（包括所有修改单）适用于本文件。

ISO 14229-2，道路车辆 - 统一诊断服务（UDS） - 第2部分：会话层服务

3 术语，定义，符号和缩略语

3.1 术语和定义

就本文件而言，下列术语和定义适用。

3.1.1
启动管理器
部分启动软件在ECU上电或复位后立即执行，其主要目的是检查是否有有效的应用程序可用于执行，而不是将控制权交给重新编程软件

注意 引导管理器也可以考虑将控制转换到重编程软件的其他条件。

3.1.2
启动内存分区
启动软件所在的服务器内存区域

3.1.3
启动软件
该软件在服务器内存的特殊部分执行，主要用于启动ECU并执行服务器编程

注1　　该内存区域在正常编程序列期间不会被擦除，并且必须在服务器应用程序丢失或以其他方式视为无效时始终执行，以确保重新编程服务器的能力。

笔记2　　另见3.1.1和3.1.17。

3.1.4
客户
功能是测试仪的一部分，可以使用诊断服务

注意　　测试人员通常使用其他功能，如数据库管理，特定解释，人机界面。

3.1.5
诊断数据
位于电子控制单元的存储器中的数据，该数据可以由检测器检查和/或可能修改

注1　　诊断数据包括模拟输入和输出，数字输入和输出，中间值和各种状态信息。

笔记2　　诊断数据的例子有车速，节气门角度，镜像位置，系统状态等。为诊断数据定义了三种类型的值：

当前值：电子控制单元的正常操作当前使用（或由此产生）的值；

存储的值：在特定时刻（例如，当发生故障或定期发生）当前值的内部副本；该副本是在电子控制单元的控制下完成的；

静态值：例如VIN。

服务器没有义务为了诊断目的而保留其数据的内部副本，在这种情况下，测试人员只能请求当前值。

注3　　定义维修店或开发测试会话会选择不同的服务器功能（例如，只能在开发测试会话中访问所有内存位置）。

3.1.6
诊断程序
例程嵌入在电子控制单元中，并且可以由服务器在来自客户端的请求时启动

注意它既可以运行而不是正常的运行程序，也可以在此模式下启用并使用正常的运行程序执行。 在第一种情况下，服务器的正常操作是不可能的。 在第二种情况下，在电子控制单元的所有其他部分正常工作的同时，可以启用多个诊断程序。

3.1.7
诊断服务
信息交换由客户发起以便要求来自服务器的诊断信息和/或为了诊断目的而修改其行为

3.1.8
诊断会话
在启用了一组特定的诊断服务和功能的服务器中的状态

3.1.9
诊断故障码DTC
用于由车载诊断系统识别的故障状态的数字通用标识符

3.1.10
ECU
电子控制单元，至少包含一台服务器

注意　　被认为是电子控制单元的系统包括防抱死制动系统（ABS）和发动机管理系统。

3.1.11
功能单元
一套功能密切或互补的诊断服务

3.1.12
整数类型
具有区分值的简单类型，其是正整数和负整数，包括零

注意　　整数类型的范围未在ISO 14229的本部分中规定。

3.1.13
本地客户端
客户端连接到与服务器相同的本地网络，并且是与服务器相同的地址空间的一部分

3.1.14
本地服务器
服务器连接到与客户端相同的本地网络，并且是与客户端相同的地址空间的一部分

3.1.15
OSI
开放系统互连

3.1.16
永久性DTC
即使在清除DTC请求之后仍保留在非易失性存储器中的诊断故障码（DTC），直到满足其他标准（通常是监管）（例如，每个DTC的适当监视器已成功通过）

注意　　有关所有必要要求，请参阅相关法规。

3.1.17
记录
一个或多个诊断数据元素通过单一识别手段一起被引用

注意　　包括各种输入/输出数据和故障代码的快照是记录的例子。

3.1.18
远程服务器
服务器不直接连接到主诊断网络

注1　　远程服务器通过远程地址进行标识。 远程地址表示一个独立于主网络地址的独立地址空间。

注2：通过主网络上的本地服务器到达远程服务器。 主网络上的每个本地服务器都可以作为一个独立远程服务器的门户。因此，一对地址必须始终标识一个远程服务器：一个标识远程网络的门的本地地址和一个标识远程服务器本身的远程地址。

3.1.19
远程客户端
客户端不直接连接到主诊断网络

注1　　远程客户端通过远程地址进行标识。

笔记2　　远程地址表示一个独立于主网络地址的独立地址空间。

3.1.20
重新编程软件
部分启动软件允许重新编程电子控制单元

3.1.21
安全
通过车辆诊断数据链路保护车辆模块免受“未经授权”侵入的机制

3.1.22
服务器
功能是电子控制单元的一部分，并提供诊断服务

注意　　该国际标准区分服务器（即功能）和电子控制单元，因此该标准与实施保持独立。

3.1.23
支持DTC
诊断故障代码，其当前被配置/校准并且能够在预定义的车辆条件下执行

3.1.24
测试仪
系统，用于控制车载电子控制单元的测试，检查，监控或诊断等功能，并且可以专用于特定类型的操作员（例如专用于车库机械的车外扫描工具，车外测试专用于装配工厂的工具或车载测试仪）

注意　　测试人员也被称为客户。

3.2　缩略语

.con　　服务原语。确认

.ind　　服务原语。指示

.req　　服务原语。请求

A_PCI　　应用层协议控制信息ECU电子控制单元

EDR　　事件数据记录器

N/A　　不适用

NR_SI　　否定响应服务标识符NRC负面响应代

码

OSI　　开放系统互连

RA 远程地址

SA 源地址

SI 服务标识符

TA 目标地址TA_type

目标地址类型

4 约定

ISO 14229的这一部分是基于OSI服务公约（ISO / IEC 10731: 1994）中讨论的公约，因为它们适用于诊断服务。

这些约定指定了服务用户和服务提供者之间的交互。 信息通过服务原语在服务用户和服务提供者之间传递，服务原语可以传递参数。

服务和协议之间的区别总结在图1中。

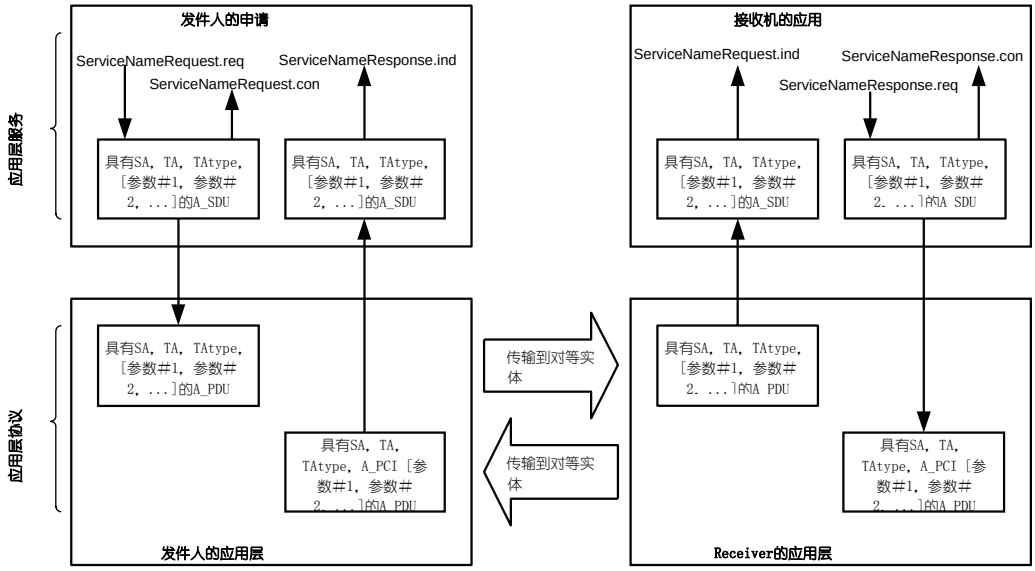


图1 – 服务和协议

这部分ISO 14229定义了确认和未确认的服务。

已确认的服务使用六个服务原语request, req_confirm, indication, response, rsp_confirm和confirmation。

未经确认的服务仅使用请求, req_confirm和indication服务原语。

对于ISO 14229本部分定义的所有服务，请求和指示服务原语总是具有相同的格式和参数。 因此，对于所有服务，响应和确认服务原语（req_confirm和rsp_confirm除外）始终具有相同的格式和参数。 在本国际标准中定义服务原语时，只列出请求和响应服务原语。

5 文档概述

图2描述了根据OSI模型实现的UDS文档参考。

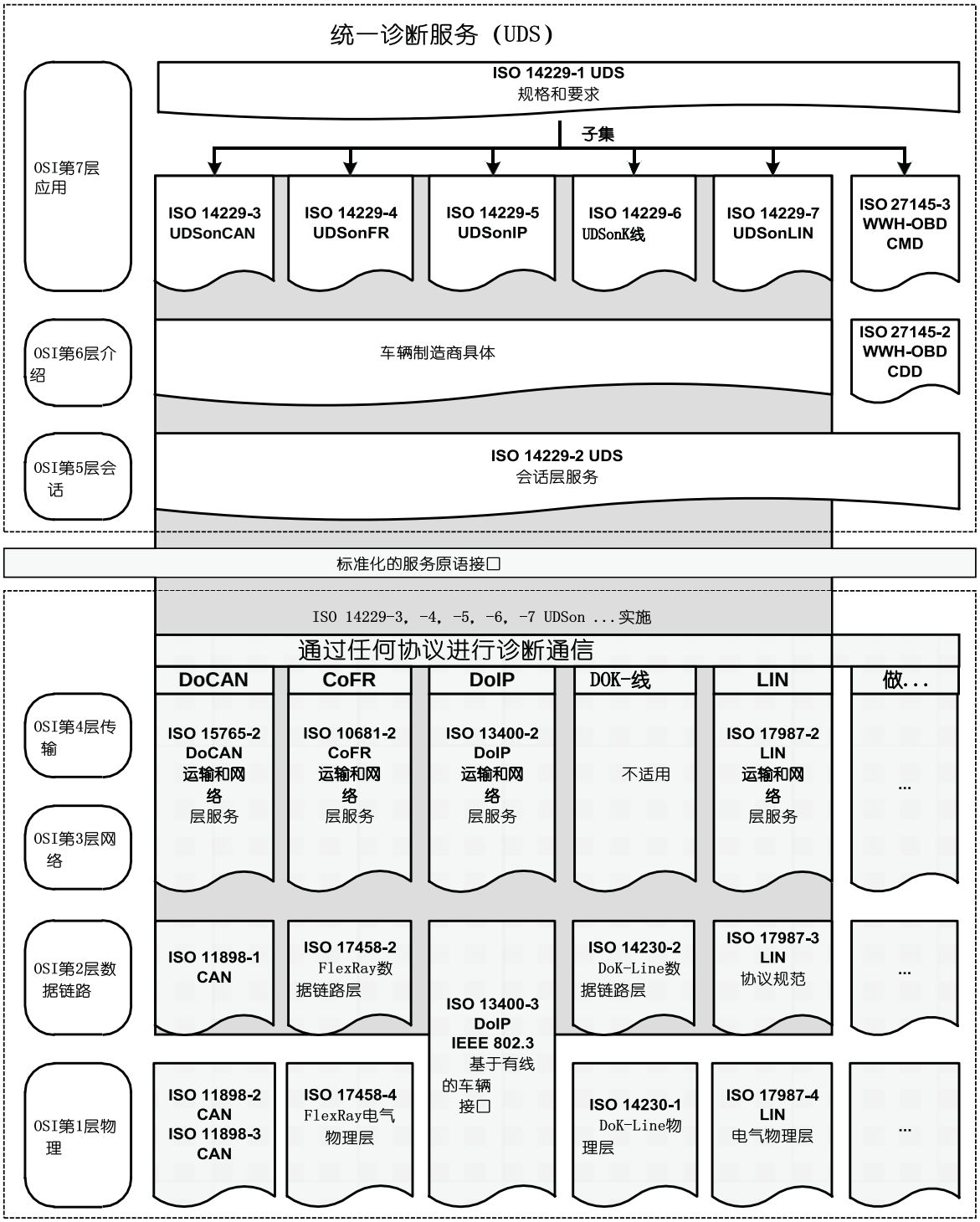


图2 - 根据OSI模型实现UDS文档参考

6 应用层服务

6.1 一般

应用层服务通常被称为诊断服务。 应用层服务用于基于客户端 – 服务器的系统，以执行诸如测试，检查，监控或诊断车载车辆服务器等功能。 通常被称为外部测试设备的客户端使用应用层服务来请求在一个或多个服务器中执行诊断功能。 服务器通常是ECU的一部分，它使用应用层服务将请求的诊断服务提供的响应数据发送回客户端。 客户端通常是板外测试器，但在某些系统中也可以是板上测试器。 应用层服务的使用与作为板外或板上测试器的客户端无关。 在同一车辆系统中可以有多个客户。

诊断应用程序层的服务访问点提供了许多服务，这些服务都具有相同的通用结构。 对于每个服务，指定了六个服务原语：由诊断测试器应用程序中的客户端功能使用的服务请求原语将关于所请求的诊断服务的数据传递给诊断应用程序层；

— 由诊断测试器应用中的客户端功能使用的服务请求原语将关于所请求的诊断服务的数据传递给诊断应用层；

— 由诊断测试仪应用程序中的客户端功能使用的服务请求确认原语指示在服务请求原语中传递的数据在诊断测试仪连接到的车辆通信总线上成功发送

— 由诊断应用层使用的服务指示原语将数据传递给ECU诊断应用程序的服务器功能；

— 由ECU诊断应用中的服务器功能使用的服务响应原语将由所请求的诊断服务提供的响应数据传递给诊断应用层；

— 服务器功能在ECU诊断应用中使用的服务响应确认原语指示在服务响应原语中传递的数据在车辆通信总线上成功发送，ECU接收到诊断请求；

— 一个服务确认原语，由诊断应用程序层用来将数据传递给诊断测试仪应用程序中的客户端功能。

图3描述了应用层服务原语 – 确认服务。

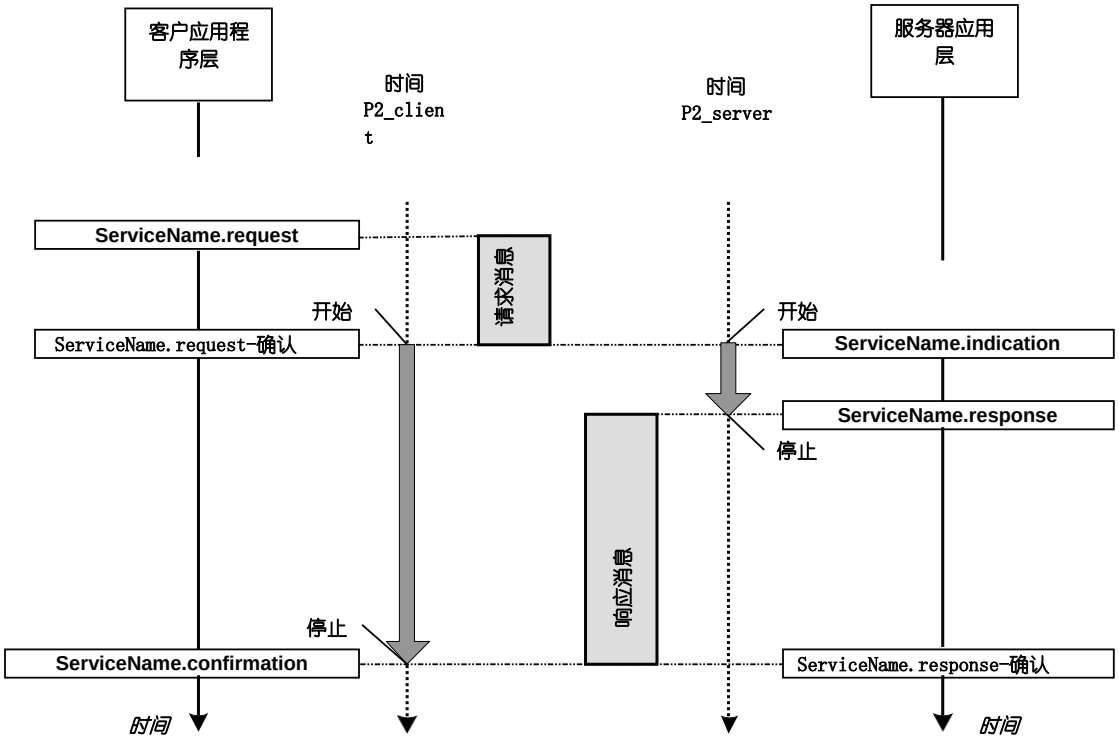


图3 – 应用层服务原语 – 确认的服务

图4描述了应用层服务原语 – 未确认的服务。

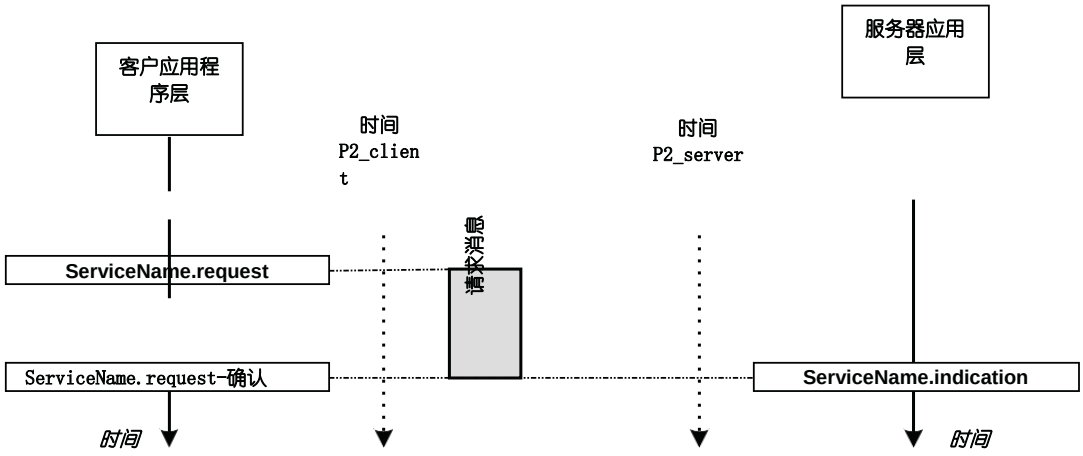


图4 – 应用层服务原语 – 未经确认的服务

对于给定的服务，请求确认原语和响应确认原语总是具有相同的服务数据单元。 这些服务原语的目的是指示先前的请求或响应服务原语调用的完成。 本国际标准中的服务描述不会使用这些服务原语，但数据链接特定的实现文档可能会使用它们来定义服务执行参考点（例如，ECUReset服务会在响应完全传输到客户端通过服务响应确认原语在服务器中指示）。

6.2 应用程序层服务的格式说明

根据车辆诊断系统的配置方式，应用层服务可以有两种不同的格式。 应用层服务的格式由参数A_Mtype控制。

如果车辆系统配置为客户端可以通过使用A_SA和A_TA地址参数来访问所有服务器，则应使用应用层服务的默认格式。 这意味着A_Mtype =诊断。

如果车辆系统被配置为使得客户端除了允许寻址某些服务器的A_SA和A_TA地址参数之外还需要地址信息，则应该使用远程格式的应用层服务。 这意味着A_Mtype =远程诊断。

应用层服务的不同格式在6.3中指定。

6.3 服务原语的格式描述

6.3.1 一般定义

所有的应用层服务具有相同的通用格式。 服务原语以下面的形式写入:

```
service_name.type (
                    参数A, 参数B, 参数C.
                    [, 参数1, ...]
                    )
```

哪里:

— “service_name”是诊断服务的名称（例如DiagnosticSessionControl），

— “type”表示服务原语的类型（例如请求），

— “参数A, ...”是作为由服务原语传递的值的列表（寻址信息）的A_SDU（应用层服务数据单元），

— “参数A, 参数B, 参数C”是必须包含在所有服务呼叫中的强制性参数，

— “ , 参数1, ... ”是取决于特定服务的参数（例如参数1可以是DiagnosticSessionControl服务的diagnosticSession）。 括号表示这部分参数列表可能为空。

6.3.2 服务请求和服务指示原语

对于每个应用层服务，根据以下通用格式指定服务请求和服务指示原语：

```
service_name.request   (
                       A_MType,
                       A_SA,
                       A_TA,
                       A_TA_type,
                       [A_AE],
                       A_Length, A_Data , 参数
                       1, ... ,
                       )
```

请求原语由诊断测试器应用程序中的客户端功能使用，以启动服务并将有关所请求的诊断服务的数据传递给应用程序层。

```
service_name.indication   (
                       A_MType,
                       A_SA,
                       A_TA,
                       A_TA_type,
                       [A_AE],
                       A_LengthA_Data , 参数
                       1, ... ,
                       )
```

指示原语被应用层用来指示对ECU诊断应用有意义的内部事件，并将关于所请求的诊断服务的数据传递给ECU诊断应用的服务器功能。

特定应用层服务的请求和指示原语总是具有相同的参数和参数值。 这意味着当数据从客户端传输到服务器时，各个参数的值不应该被应用层的通信对等协议实体改变。 客户端应用程序中的客户端函数在服务请求调用中传递给应用程序层的相同值应由诊断应用程序的服务器功能从对等应用程序层的服务指示中接收。

6.3.3 服务响应和服务确认原语

对于每个确认的应用层服务，根据以下一般格式指定服务响应和服务确认原语：

```
service_name.response (
                        A_Mtype,
                        A_SA,
                        A_TA,
                        A_TA_type,
                        [A_AE],
                        A_LengthA_Data , 参数
                        1, ... ,
                        )
```

响应原语由ECU诊断应用程序中的服务器功能使用，以启动服务并将所请求的诊断服务提供的响应数据传递给应用程序层。

```
service_name.confirm   (
                        A_Mtype,
                        A_SA,
                        A_TA,
                        A_TA_type,
                        [A_AE],
                        A_LengthA_Data , 参数
                        1, ... ,
                        )
```

确认原语被应用层用来指示对客户端应用有意义的内部事件，并将关联的先前服务请求的结果传递给诊断测试器应用中的客户端功能。 它不一定表示远程对等接口上有任何活动，例如，如果所请求的服务不受服务器支持或通信中断。

特定应用程序层服务的响应和确认原语始终具有相同的参数和参数值。 这意味着当数据从服务器传输到客户端时，应用层的通信对等协议实体不应改变单个参数的值。 ECU诊断应用程序的服务器功能在服务响应调用中传递给应用程序层的相同值应由诊断测试仪应用程序中的客户端功能从对等应用程序层的服务确认中接收。

对于每个响应和确认原语，将指定两个不同的服务数据单元（两组参数）。

— 如果所请求的诊断服务可以由ECU中的服务器功能成功执行，则肯定响应和肯定确认原语应与第一服务数据单元一起使用。

— 如果请求的诊断服务失败或ECU中的服务器功能无法及时完成，则应使用否定响应和确认原语与第二个服务数据单元一起使用。

6.3.4 服务请求 - 确认和服务响应 - 确认原语

对于每个应用层服务，根据以下通用格式指定服务请求确认和服务响应确认原语：

```
service_name.req_confirm (
                            A_Mtype,
                            A_SA,
                            A_TA,
                            A_TA_type,
                            [A_AE],
                            A_Result
                            )
```

请求 - 确认原语被应用层用来指示对客户端应用很重要的内部事件，并将关联的先前服务请求的通信结果传递给诊断测试器应用中的客户端功能。

```
service_name.rsp_confirm  (
                            A_Mtype,
                            A_SA,
                            A_TA,
                            A_TA_type,
                            [A_AE],
                            A_Result
                            )
```

响应确认原语被应用层用来指示对服务器应用程序有意义的内部事件，并将关联的先前服务响应的通信结果传递给ECU应用程序中的服务器功能。

6.4 服务数据单元规范

6.4.1 强制性参数

6.4.1.1 一般定义

应用程序层服务包含三个必需参数。 以下参数定义适用于本国际标准（标准和远程格式）中指定的所有应用层服务。

6.4.1.2 A_Mtype，应用层消息类型

类型： 列举

范围：诊断，远程诊断描述：

参数Mtype应用于识别6.2中规定的车辆诊断系统的格式。 ISO 14229的这部分规定了该参数的两个值的范围：

如果A_Mtype = diagnostics，则service_name原语应由参数A_SA，A_TA和A_TAtype组成。

如果A_Mtype =远程诊断，则service_name原语应由参数A_SA，A_TA，A_TAtype和A_AE组成。

6.4.1.3 A_SA，应用层源地址

类型： 2字节无符号整数值范围：

0x0000 – 0xFFFF描述：

参数SA应用于编码客户端和服务器标识符。

对于服务请求（和服务指示），A_SA表示请求诊断服务的客户端功能的地址。 每个请求诊断服务的客户端功能应以一个A_SA值表示。 如果在同一诊断测试器中实现多个客户端功能，则每个客户端功能都应有其自己的客户端标识符和相应的A_SA值。

对于服务响应（和服务确认），A_SA表示已执行所请求的诊断服务的服务器功能的地址。 服务器功能可以仅在一个ECU中实现，或者可以在多个ECU中分布和实现。 如果一个服务器功能只在一个ECU中实现，那么它应该只被编码一个A_SA值。 如果一个服务器功能在多个ECU中分配和实现，那么相应的服务器功能地址应为每个单独的服务器功能编码一个A_SA值。

如果远程客户端或服务器是消息的原始来源，则A_SA代表从远程网络到主网络的门户的本地服务器。

注意 如果对请求消息使用物理寻址，则响应消息中的A_SA值将与相应请求消息中的A_TA值相同。

6.4.1.4 A_TA，应用层目标地址

类型： 2字节无符号整数值范围：

0x0000 – 0xFFFF描述：

参数A_TA应用于编码客户端和服务器标识符。 两种不同的寻址方法称为：

— 物理寻址和

— 功能寻址

被指定用于诊断。 因此，可以为车辆系统定义两组独立的目标地址（每个寻址方法一个）。

物理寻址应始终是一个专用的消息给在一个ECU中实现的服务器。 当使用物理寻址时，通信是客户端和服务器之间的点对点通信。

如果客户端不知道应对诊断服务请求作出响应的服务器功能的物理地址，或者服务器功能在多个ECU中作为分布式功能实现，则客户端使用功能地址。 当使用功能性寻址时，通信是从客户端到在一个或多个ECU中实现的服务器的广播通信。

对于服务请求（和服务指示），A_TA表示应执行所请求的诊断服务的服务器的服务器标识符。 如果正在寻址远程服务器，则A_TA代表从主网络到远程网络的本地服务器。

对于服务响应（和服务确认），A_TA表示最初请求诊断服务的客户端功能的地址，并且将接收所请求的数据（即请求的A_SA）。 服务响应（和服务确认）应始终使用物理寻址。 如果远程客户端正在寻址，则A_TA代表从主网络到远程网络的本地服务器。

注意 响应消息的A_TA值将始终与相应请求消息的A_SA值相同。

6.4.1.5 A_TA_Type，应用层目标地址类型

类型： 枚举范围：物理，

功能描述：

参数A_TA_type是A_TA参数的扩展名。 它用于表示为消息传输选择的寻址方法。

6.4.1.6 A_Result类

型： 枚举范围：

ok，error说明：

req_confirm和rsp_confirm原语使用参数'A_Result'指示消息是否已正确传输（ok）或消息传输是否成功（错误）。

6.4.1.7 A_Length

类型： 4字节无符号整数值范围：0_d

– $(2^{32}-1)_d$

描述：

该参数包括要发送/接收的数据的长度。

6.4.1.8 A_Data类型：

字符串范围：

不适用说明：

该参数包括所有由上层实体交换的数据。

6.4.2 车辆系统要求

车辆制造商应确保系统中的每台服务器都有唯一的服务器标识。 车辆制造商还应确保系统中的每个客户都有唯一的客户标识。

车辆系统中诊断网络的所有客户端和服务器地址应编码到相同的源地址范围内。 这意味着客户和服务器在给定的车辆系统中不应该以相同的A_SA值表示。

服务器的物理目标地址应始终与服务器的源地址相同。

远程服务器标识符可以独立于主网络上的客户端和服务器标识符进行分配。

通常，只有寻址的服务器才能响应客户端请求消息。

6.4.3 可选参数 – A_AE，应用层远程地址

类型： 2字节无符号整数值范围：

0x0000 – 0xFFFF描述：

A_AE用于扩展可用地址范围以编码客户端和服务器标识符。 A_AE只能用于实现本地服务器和远程服务器概念的车辆。 远程地址代表自己的地址范围，并且独立于主网络上的地址。

参数A_AE应用于编码远程客户端和服务器标识符。 A_AE可以表示远程目标地址或远程源地址，具体取决于携带A_AE的消息的方向。

对于主网络上的客户端发送的服务请求（和服务指示），A_AE代表将执行所请求的诊断服务的服务器的远程服务器标识符（远程目标地址）。

A_AE可以用作物理地址和功能地址。 对于A_AE的每个值，系统构建者应指定该值是否代表物理地址或功能地址。

注意A_TA_type指定A_TA的寻址方法的方式没有特殊参数来表示物理或功能远程地址。 物理和功能远程地址共享1个字节范围内的值，每个值的含义由系统构建者定义。

对于由远程服务器发送的服务响应（和服务确认），A_AE表示已执行所请求的诊断服务的远程服务器的物理位置（远程源地址）。

远程服务器可能只能在一个ECU中实现，或者可以在多个ECU中分配和实现。 如果一个远程服务器仅在一个ECU中实现，那么它应该只被编码一个A_AE值。 如果远程服务器在多个ECU中分布和实现，则远程服务器标识符应为远程服务器的每个物理位置编码一个A_AE值。

7 应用层协议

7.1 一般定义

应用层协议应始终是确认的消息传输，这意味着对于从客户端发送的每个服务请求，应该有一个或多个来自服务器的相应响应。

本规则的唯一例外情况是使用功能性寻址或请求/指示规定不产生响应/确认的情况。 为了不给系统增加许多不必要的信息，即使服务器未能完成所请求的诊断服务，也有一些情况下不应发送否定响应信息。 这些异常情况在本规范中的相关子条款中进行了描述（例如，参见7.5）。

应用层协议应与会话层协议并行处理。 这意味着，即使客户端正在等待对先前请求的响应，它也应该保持适当的会话层时间（例如，如果需要发送TesterPresent请求以保持诊断会话进入其他服务器，实现取决于数据链接层使用）。

7.2 协议数据单元规范

A_PDU（应用层协议数据单元）由A_SDU（应用层服务数据单元）和层特定控制信息A_PCI（应用层协议控制信息）直接构成。 A_PDU应具有以下通用格式：

```
A_PDU   (
        Mtype,
        SA,
        TA,
        TA_type, [RA, ]
        A_Data = A_PCI + 参数1, ... ,
        长度
        )
```

哪里：

“Mtype，SA，TA，TA_type，RA，Length”与在A_SDU中使用的参数相同；

“A_Data”是为每个单独的应用程序层服务定义的一串字节数据。 A_Data字符串应以A_PCI开头，后面跟随每个服务指定的来自A_SDU的所有服务特定参数。 括号表示这部分参数列表可能为空；

“长度”决定A_Data的字节数；

7.3 应用协议控制信息

7.3.1 PCI，协议控制信息

A_PCI由两种格式组成。 由A_PCI参数的第一个字节的值标识的格式。 对于第一字节不等于0x7F的所有服务请求和服务响应，应适用下列定义：

```
A_PCI

         (
        SI
        )
```

哪里：

“SI”是参数服务标识符；

对于第一个字节等于0x7F的服务响应，应使用以下定义：

```
A_PCI
            (NR
        _SI, SI
        )
```

哪里：

“NR_SI”是识别负面服务响应/确认的特殊参数；

“SI”是参数服务标识符；

注意 对于服务ReadDataByPeriodicIdentifier（0x2A，见10.5）中定义的周期性数据响应消息的传输，在应用层协议数据单元（A_PDU）中不存在A_PCI。

7.3.2 SI，服务标识符

类型：1字节无符号整数值

根据表2中的定义，范围：0x00 – 0xFF。

表2 – 服务标识符值

服务标识符（SI）	服务类型（位6）	在哪里定义
0x10 – 0x3E	ISO 14229-1服务请求	ISO 14229-1
0x3F	不适用	由文件保留
0x50 – 0x7E	ISO 14229-1积极的服务回应	ISO 14229-1
0x7F	负面响应服务标识符	ISO 14229-1
0x80 – 0x82	不适用	由ISO 14229-1保留
0x83 – 0x88	ISO 14229-1服务请求	ISO 14229-1
0x89 – 0xB9	不适用	由ISO 14229-1保留
0xBA – 0xBE	服务请求	由系统供应商定义
0xBF – 0xC2	不适用	由ISO 14229-1保留
0xC3 – 0xC8	ISO 14229-1积极的服务回应	ISO 14229-1
0xC9 – 0xF9	不适用	由ISO 14229-1保留
0xFA – 0xFE	积极的服务回应	由系统供应商定义
为 0xFF	不适用	由文件保留

注意请求消息的服务标识符与肯定响应消息的服务标识符之间存在一一对应关系，SI字节值的第6位表示服务类型。 所有请求消息的SI位6 = 0。除ReadDataByPeriodicIdentifier（0x2A，参见10.5）服务的周期性数据响应消息外，所有正响应消息的SI位6 = 1。

描述：

SI应被用于对服务原语中已被调用的特定服务进行编码。 每个请求服务应分配一个唯一的SI值。 每个积极响应服务应分配一个相应的唯一SI值。

服务标识符用于表示从应用程序层传递到较低层（从较低层返回）的A_Data数据字符串中的服务。

7.3.3 NR_SI，否定响应服务标识符

类型： 1字节无符号整数值固定值：

0x7F

描述：

参数NR_SI是识别负面服务响应/确认的特殊参数。 它应成为负面答复/确认消息的A_PCI的一部分。

注意 NR_SI值与SI值一致。 为了使A_Data编码和解码更容易，NR_SI值不被用作SI值。

7.4 负面响应/确认服务原语

根据表3，每个诊断服务都有消息A_Data字节指定的否定响应/否定确认消息。第一个A_Data字节（A_PCI.NR_SI）始终是特定的否定响应服务标识符。 第二个A_Data字节（A_PCI.SI）应该是否定响应消息所对应的来自服务请求/指示消息的服务标识符值的副本。

表3 – 负面响应A_PDU

A_PDU参数	参数名称	Cvt	字节值	助记符
SA	源地址	M	0xXXXX	SA
TA	目标地址	M	0xXXXX	TA
TAtype	目标地址类型	M	0xXX	TAT
RA	远程地址（可选）	C	0xXXXX	RA
A_Data.A_PCI.NR_SI	负面响应SID	M	0x7F	SIDNR
A_Data.A_PCI。 SI	<服务名称>请求SID	M	0xXX	SIDRQ
A_Data.Parameter 1	responseCode	M	0xXX	NRC_
M（强制性）： 在发出否定响应A_PDU的情况下，那些A_PDU参数应该存在。 C（有条件）： RA（远程地址）PDU参数仅在远程寻址的情况下才存在。				

注意 A_Data表示否定响应消息的消息数据字节。

参数responseCode用于否定响应消息中，以指示诊断服务失败或无法及时完成的原因。 值在A.1中定义。

7.5 服务器响应实施规则

7.5.1 一般定义

以下子条款指定服务器在执行服务时的行为。 服务器和客户端应遵循这些实施规则。

传说

缩写	描述
suppressPosRspMsgIndicationBit	TRUE =服务器不应发送肯定的响应消息（例外见NRC 0x78的定义中的附件A.1） FALSE =服务器应发送正面或负面的回应讯息
PosRsp	积极回应消息的缩写
NegRsp	负面响应消息的缩写
NoRsp	不发送积极或消极反应意思的缩写
NRC	负面响应代码的缩写
所有	服务器支持所有请求的客户端请求消息的数据参数
至少1	服务器必须支持至少1个客户端请求消息的数据参数
没有	服务器不支持所请求的客户端请求消息的数据参数

无论寻址模式如何（物理地址，功能地址类型），服务器应支持其诊断服务列表。

重要 – 根据以下各小节中的表格的要求，负号响应消息的SNS（serviceNotSupported），SNSIAS（serviceNotSupportedIn-ActiveSession），SFNS（sub-functionNotSupported），SFNSIAS（sub-functionNotSupportedInActive-Session）和ROOR（requestOutOfRange）在功能寻址用于请求消息时不应发送（例外见NRC 0x78定义中的附录A.1）。

7.5.2 一般服务器响应行为

本小节中规定的一般服务器响应行为对于所有请求消息都是必需的。 验证步骤从接收请求消息开始。 该图分为三个小节

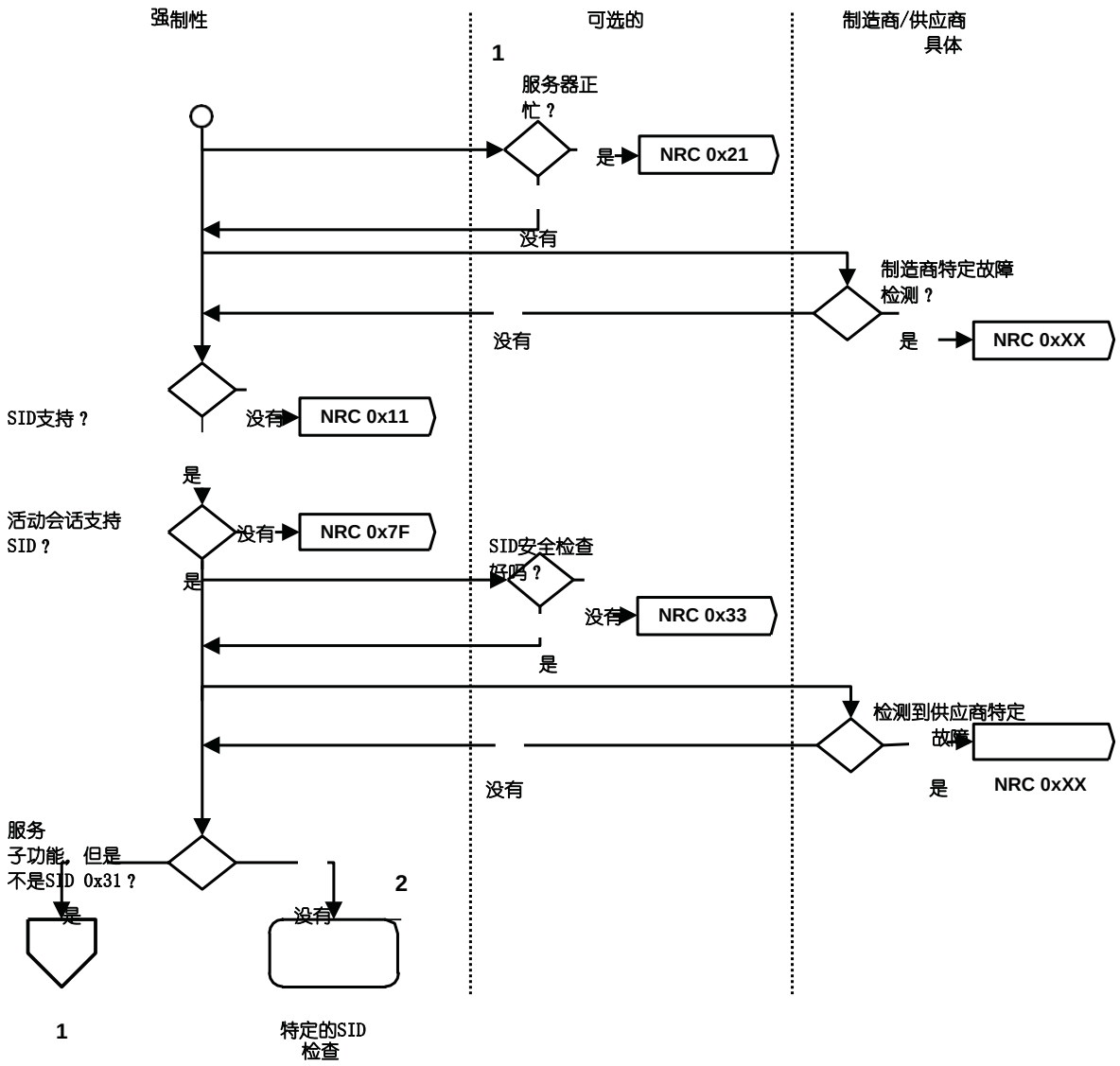
— 强制性的：由每个请求消息评估，

— 可选：可以根据每个请求消息进行选择性评估，

— 制造商/供应商特定：该程序可以通过额外的制造商/供应商特定检查进行扩展。

注意 根据指定NRC处理的所有数字中实施的选择，对于所有可能的测试模式序列，不保证特定的NRC。

图5描述了一般服务器响应行为。



键

1 诊断请求不能被接受，因为另一个诊断任务已经被另一个客户端请求和正在进行。

2 请参阅每项服务的响应行为（支持的否定响应代码）

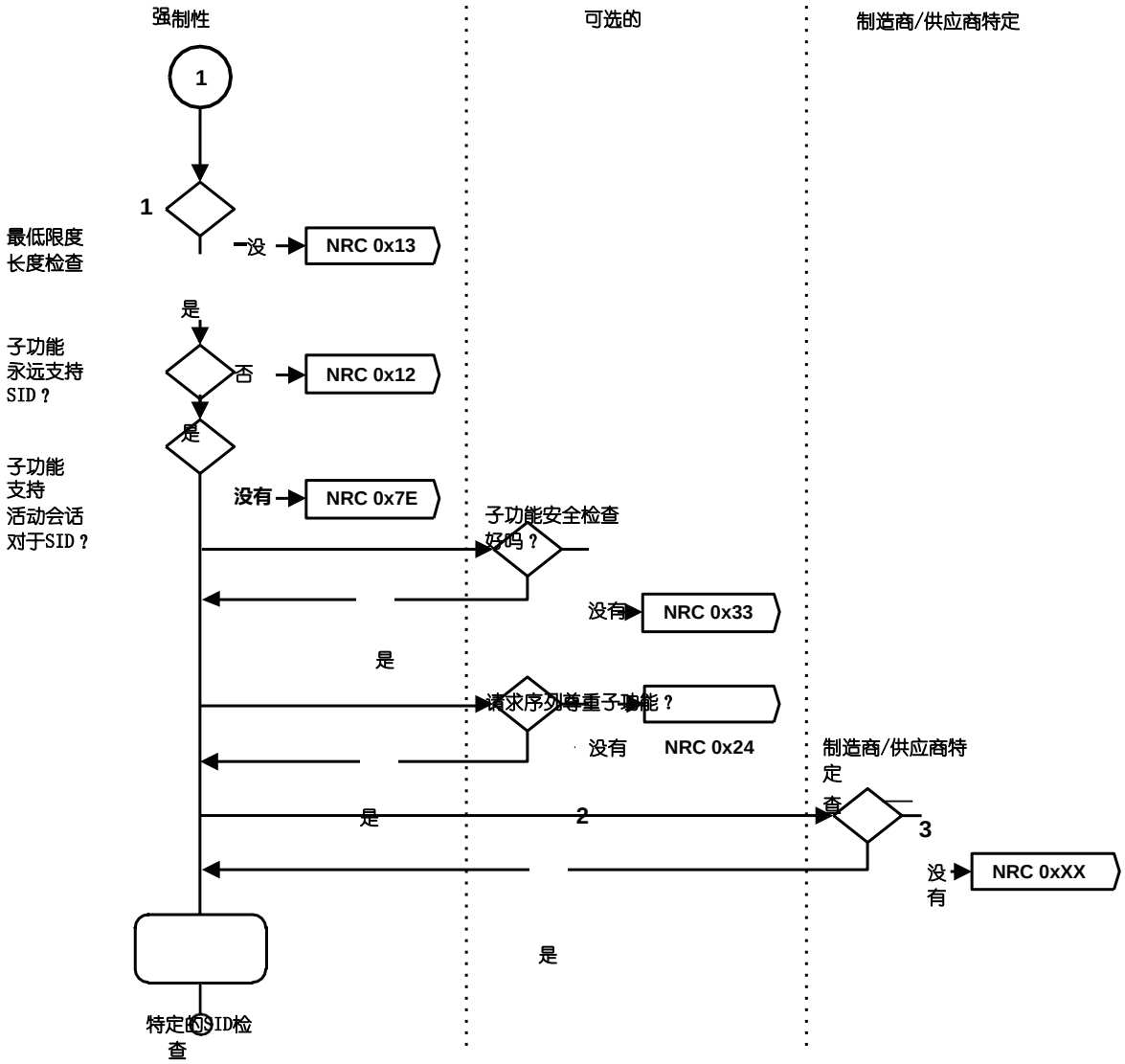
图5 – 一般服务器响应行为

7.5.3 具有子功能参数和服务器响应行为的请求消息

7.5.3.1 具有子函数参数的请求消息的一般服务器响应行为

本子条款中规定的一般服务器响应行为对于具有子功能参数的所有请求消息都是必需的。 本节内容中的请求消息被定义为遵循ISO 14229本部分定义的格式要求的服务请求消息。

图6描述了具有子功能参数的请求消息的一般服务器响应行为。



键

1 至少2个（SID +子功能参数）。

2 如果子功能受序列检查，例如LinkControl或SecurityAccess。

3 请参阅每项服务的响应行为（支持的否定响应代码）。

图6 – 具有子功能参数的请求消息的一般服务器响应行为

7.5.3.2 物理寻址的客户端请求消息

本子条款中规定的服务器响应行为在每个服务的服务描述中被引用，该服务描述支持从客户端接收到的物理寻址请求消息中的子函数参数。

表4显示了可能的物理寻址通信方案。

表4 – 具有子功能参数和服务器响应行为的物理寻址请求消息

服务器案例#	客户端请求消息		服务器功能			服务器响应		注释
	解决方案	子功能（suppress-PosRspMsg-Indication-Bit）	SI支持	子函数支持的	数据参数支持（只适用）	信息	NRC	到服务器响应
a)	物理	（位= 0）	是	是	至少1	PosRsp	---	服务器发送正面回应
b)					至少1	NegRsp	NRC= 0xXX	服务器发送否定答复，因为错误发生在读取数据参数请求消息
c)							NRC= ROOR	负与NRC 0x31响应
d)			没有	--	--		NRC= SNS或SNSIAS	NRC 0x11或NRC 0x7F为负值
e)			是	没有	--		NRC= SFNS或SFNSIAS	NRC 0x12或NRC 0x7E为负值
f)					至少1	NoRsp	---	服务器不不发送回复
g)					至少1		NRC= 0xXX	服务器发送否定答复，因为读取请求消息的数据参数时发生错误
h)					没有		NRC= ROOR	负与NRC 0x31响应
i)			没有	--	--		NRC= SNS	负与NRC 0x11的响应
j)			是	没有	--		NRC= SFNS	负与NRC 0x12的响应

具有子功能的物理寻址的客户端请求消息上的服务器响应情况的描述：

a) 服务器发送肯定响应消息，因为客户端的请求支持服务标识符和子函数参数，并显示响应消息。

b) 服务器发送否定响应消息（例如IMLOIF: incorrectMessageLengthOrIncorrectFormat），因为服务标识符和子函数参数支持客户端请求，但出现了一些其他错误（例如根据服务标识符和子函数参数请求消息）在处理子功能期间。

c) 服务器发送带有否定响应代码ROOR（requestOutOfRange）的否定响应消息，因为支持服务标识符和子函数参数，但客户端的请求消息不支持所请求的数据参数。

d) 服务器发送否定响应消息与否定响应代码SNS（serviceNotSupported）或SNSIAS（serviceNotSupportedInActiveSession），因为服务标识符不支持客户端的请求并显示响应消息。

e) 服务器发送否定响应消息，其中包含负面响应代码SFNS（sub-functionNotSupported）或SFNSIAS（sub-FunctionNotSupportedInActiveSession），因为服务标识符是受支持的，并且子函数参数不支持数据参数（如果适用）客户端的请求并显示响应消息。

f) 服务器不发送响应消息，因为客户端的请求支持服务标识符和子函数参数，并指示没有响应消息。

注意 如果使用否定响应代码RCRRP（requestCorrectlyReceivedResponsePending），则应该给出最终响应，与suppressPosRspMsgIndicationBit值无关。

g) 与b）中的效果相同（即，发送否定响应消息），因为对于需要在物理寻址的请求消息中发送的任何否定响应，忽略suppressPosRspMsgIndicationBit。

h) 与c）中的效果相同（即，发送否定响应消息），因为在收到物理寻址的请求消息时需要发送的任何否定响应将忽略suppressPosRspMsgIndicationBit。

i) 与d）中的效果相同（即，发送否定响应消息），因为对于需要在物理寻址的请求消息中发送的任何否定响应，suppressPosRspMsgIndicationBit被忽略。

j) 与e）中的效果相同（即，发送否定响应消息），因为对于需要在物理寻址的请求消息中发送的任何否定响应，suppressPosRspMsgIndicationBit被忽略。

7.5.3.3 功能上处理的客户端请求消息

本子条款中规定的服务器响应行为在每个服务的服务描述中被引用，该服务描述支持从客户端接收到的功能编址请求消息中的子函数参数。

表5显示了可能的功能寻址通信方案。

表5 – 具有子功能参数和服务器响应行为的功能编址的请求消息

服务器案例#	客户端请求消息		服务器功能			服务器响应		对服务器响应的评论
	解决方案	子函数（suppressPosRspMsg-IndicationBit）	SI支持	支持子功能	数据参数支持（只适用）	信息	NRC	
a)					至少1	PosRsp	---	服务器发送正面回应
b)					至少1	NegRsp	NRC= 0xXX	服务器发送否定答复，因为读取请求消息的数据参数时发生错误
c)					没有		--	服务器不发送回复
d)			没有	--	--		---	服务器不发送回复
e)			是	没有	--		---	服务器不发送回复
f)					至少1	NoRsp	---	服务器不发送回复
g)					至少1	NegRsp	NRC= 0xXX	服务器发送否定答复，因为读取请求消息的数据参数时发生错误
h)					没有		--	服务器不发送回复
i)			没有	--	--		---	服务器不发送回复
j)			是	没有	--		---	服务器不发送回复

具有子功能的功能性寻址客户端请求消息上的服务器响应情况的描述：

a) 服务器发送肯定响应消息，因为客户端的请求支持服务标识符和子函数参数，并显示响应消息。

b) 服务器发送否定响应消息（例如IMLOIF：incorrectMessageLengthOrIncorrectFormat），因为服务标识符和子函数参数支持客户端请求，但出现了一些其他错误（例如根据服务标识符和子函数参数请求消息）在处理子功能期间。

c) 服务器不发送任何响应消息，因为负面响应代码ROOR（requestOutOfRange，由于服务标识符和子函数参数受支持而由服务器标识，但客户端请求不支持所需的数据参数）始终被抑制的功能性地寻址的请求消息。在这种情况下，suppressPosRspMsgIndicationBit无关紧要。

d) 服务器不发送任何响应消息，因为服务器识别的否定响应代码SNS（serviceNotSupported）和SNSIAS（serviceNotSupportedInActiveSession）是因为服务标识符不支持客户端的请求，所以在功能编址的请求消息的情况下始终会被抑制。 在这种情况下，suppressPosRspMsgIndicationBit无关紧要。

e) 服务器不发送响应消息，因为服务器识别的否定响应代码SFNS（sub-functionNotSupported）和SFNSIAS（sub-functionNotSupportedInActiveSession）由于支持服务标识符而且子函数参数不支持数据参数（如果适用）客户请求的情况下，在功能性地处理请求的情况下总是被抑制。 在这种情况下，suppressPosRspMsgIndicationBit无关紧要。

f) 服务器不发送响应消息，因为客户端的请求支持服务标识符和子函数参数，并指示没有响应消息。

注意 如果使用否定响应代码RCRRP（requestCorrectlyReceivedResponsePending），则应该给出最终响应，与suppressPosRspMsgIndicationBit值无关。

g) 对于任何否定响应，suppressPosRspMsgIndicationBit将被忽略，因此与b）中的效果相同（即，发送否定响应消息）。 在功能上解决请求消息的情况下也是如此。

h) 由于负面响应码ROOR（requestOutOfRange，由服务器识别，因为支持服务标识符和子函数参数，但不支持所需的数据参数），所以与c）中的效果相同（即，不发送响应消息）客户端的请求）总是被抑制以防功能性地寻址请求消息。 在这种情况下，suppressPosRspMsgIndicationBit无关紧要。

i) 因为由于服务标识符不受客户端请求支持而由服务器识别的否定响应代码SNS（serviceNotSupported）和SNSIAS（serviceNotSupportedInActiveSession）总是始终如此（即，没有响应消息被发送）在功能上处理的请求消息的情况下被抑制。 在这种情况下，suppressPosRspMsgIndicationBit无关紧要。

j) 因为由于服务标识符被支持而被服务器识别的否定响应码SFNS（sub-functionNotSupported）和SFNSIAS（sub-functionNotSupportedInActiveSession）函数参数不受客户端请求支持，在功能编址请求消息的情况下总是被抑制。 在这种情况下，suppressPosRspMsgIndicationBit无关紧要。

7.5.4 请求消息没有子功能参数和服务器响应行为

7.5.4.1 没有子功能参数的请求消息的一般服务器响应行为

没有子功能参数的请求消息没有可用的通用服务器响应行为。 本节内容中的请求消息被定义为遵循ISO 14229本部分定义的格式要求的服务请求消息。

7.5.4.2 物理地处理客户请求消息

本子条款中规定的服务器响应行为在每个服务的服务描述中被引用，它不支持子函数参数，而是从客户端收到的物理地址请求消息中的数据参数。

表6显示了可能的物理寻址通信方案。

表6 – 没有子功能参数和服务器响应行为的物理寻址请求消息

服务器案例#	客户端请求消息	服务器功能		服务器响应		对服务器响应的评论
	解决方案	SI支持的	参数支持	信息	NRC	
a)			所有		---	服务器发送正面回应
b)			至少1		---	服务器发送正面回应
c)			至少1		NRC=0xXX	服务器发送否定响应，因为错误发生时读取请求消息的数据参数
d)			没有		NRC=ROOR	负面响应0x31 同 NRC
e)		没有	---		NRC = SNS或SNSIAS	否定响应0x11或NRC 0x7F 同 NRC

没有子功能的物理寻址客户端请求消息（数据参数跟随服务标识符）的服务器响应情况的描述：

a) 服务器发送肯定响应消息，因为服务标识符和所有数据参数都支持客户端的请求消息。

b) 服务器发送肯定响应消息，因为客户端的请求消息支持服务标识符和至少一个数据参数。

c) 服务器发送一个否定响应消息（例如，IMLOIF：incorrectMessageLengthOrIncorrectFormat），因为服务标识符是受支持的，并且客户端请求消息至少支持一个数据参数，但在处理期间发生了一些其他错误（例如请求消息的长度错误）的服务。

d) 服务器发送带有否定响应代码ROOR（requestOutOfRange）的否定响应消息，因为服务标识符是受支持的，并且所请求的数据参数都不支持客户端的请求消息。

e) 由于服务标识符不支持客户端的请求消息，因此服务器会发送否定响应消息和否定响应代码SNS（serviceNotSupported）或SNSIAS（serviceNotSupportedInActiveSession）。

7.5.4.3 功能上处理的客户端请求消息

本子条款中规定的服务器响应行为是在每个服务的服务描述中引用的，它不支持子函数参数，而是从客户端接收到的功能性寻址请求消息中的数据参数。

表7显示了具有功能寻址的可能通信方案。

表7 – 没有子功能参数和服务器响应行为的功能编址的请求消息

服务器案例#	客户端请求消息	服务器功能		服务器响应		对服务器响应的评论
	解决方案	SI 支持的	参数支持	信息	NRC	
a)			所有		---	服务器发送正面回应
b)			至少1		---	服务器发送正面回应
c)			至少1	NegRsp	NRC= 0xXX	服务器发送否定响应，因为错误发生时读取请求消息的数据参数
d)			没有		---	服务器做出响应 不 发送 a
e)		没有	---		---	服务器做出响应 不 发送 a

没有子功能的功能性寻址客户端请求消息（数据参数跟随服务标识符）的服务器响应情况的描述：

a) 服务器发送肯定响应消息，因为服务标识符和所有数据参数都支持客户端的请求消息。

b) 服务器发送肯定响应消息，因为客户端的请求消息支持服务标识符和至少一个数据参数。

c) 服务器发送否定响应消息（例如IMLOIF: incorrectMessageLengthOrIncorrectFormat），因为服务标识符是受支持的，并且客户端的请求消息至少支持一个，多于一个或所有数据参数，但会发生其他一些错误（例如错误的长度请求消息）在服务的处理期间。

d) 服务器不发送任何响应消息，因为负面响应代码ROOR（requestOutOfRange;由于支持服务标识符而发生，但客户端的请求不支持所请求的数据参数）始终在功能性地寻址请求时被抑制。

e) 服务器不发送响应消息，因为服务器识别的否定响应代码SNS（serviceNotSupported）和SNSIAS（serviceNotSupportedInActiveSession）是因为服务标识符不受客户端请求支持，所以在功能性地寻址请求的情况下始终会被禁止。

7.5.5 服务器响应行为的伪代码示例

以下是服务器伪代码示例，用于描述服务器从客户端接收请求时应执行的逻辑步骤。

```
SWITCH (A_PDU.A_Data.A_PCI.SI)
    {
    CASE Service_with_sub-function:                          / *测试是否支持带子功能的服务* /
        IF (message_length> = 2) THEN                        / *使用子函数* / SWITCH检查消息的最小长度
            (A_PDU.A_Data.A_Data.Parameter1&0x7F)

                                                             / *得到没有位7的子功能参数值* /
               {
               CASE子函数_00:                                / *测试是否支持子函数参数值* / IF (message_length == expected_sub-
                   function_message_length) THEN
                       :                                     / *准备回复消息* /
```

```
                    responseCode = positiveResponse;          / *积极回应讯息; 设置内部NRC = 0x00
*/
                其他
                    responseCode = IMLOIF;                    / * NRC 0x13: incorrectMessageLengthOrInvalidFormat * / ENDIF
                打破;
            CASE子函数_01:                                    / *测试是否支持子功能参数值* /
                :                                             / *准备回复消息* /
                responseCode = positiveResponse;              / *积极回应讯息; 设置内部NRC = 0x00
*/
                :
            CASE子函数_127:                                   / *测试是否支持子功能参数值* /
                : / *准备回复消息* /
                responseCode = positiveResponse;              / *积极回应讯息; 设置内部NRC = 0x00
*/
                打破;
            默认:
                responseCode = SFNS;                          / * NRC 0x12: 子功能不支持* /
            }
        其他
            responseCode = IMLOIF;                            / * NRC 0x13: incorrectMessageLengthOrInvalidFormat * / ENDIF
        suppressPosRspMsgIndicationBit = (A_PDU.A_Data.Parameter1&0x80) ;
                                                              / *结果为0x00或0x80 * / IF
         ( (suppressPosRspMsgIndicationBit) && (responseCode == positiveResponse) &&
            ("尚未发送NRC 0x78响应" ) ) 然后                  / *测试是否需要正响应, 如果是responseCode
                                                                    是正值0x00 * /
            suppressResponse = TRUE;                          / *标志不发送肯定回复信息* / ELSE
            suppressResponse = FALSE;                         / *标志发送响应消息* / ENDIF
        打破;
    CASE Service_without_sub-function:                        / *测试是否支持没有子功能的服务* / suppressResponse = FALSE;    /
        *标志发送响应消息* /
        IF (message_length == expected_message_length) THEN IF
            (A_PDU.A_Data.Parameter1 == supported) THEN
                                                              / *测试SID之后的数据参数是否为
支持的*/
                :                                             / *读取数据并准备响应消息* /
                responseCode = positiveResponse;              / *积极回应讯息; 设置内部NRC = 0x00
*/


            其他
                responseCode = ROOR;                          / * NRC 0x31: requestOutOfRange * /
            ENDIF
        其他
            responseCode = IMLOIF;                            / * NRC 0x13: incorrectMessageLengthOrInvalidFormat * / ENDIF
        打破;
    默认:
        responseCode = SNS;                                   / * NRC 0x11: serviceNotSupported * /
    }
IF (A_PDU.TA_type == functional && ( (responseCode == SNS) || (responseCode == SFNS) || (responseCode == SNSIAS) || (responseCode ==
SFNSIAS) || (responseCode == ROOR) ) &&
 ("尚未发送NRC 0x78响应" ) ) 然后
                                                              / *禁止否定回复消息* /
其他
    IF (suppressResponse == TRUE) THEN                        / *压制正面回应消息* /
    其他                                                      / *发送消极或积极的回应* /
    ENDIF
ENDIF
```

当对请求消息使用功能性寻址，并且需要发送具有NRC = RCRRP (requestCorrectlyReceivedResponsePending)的否定响应消息时，则使用NRC = SNS (serviceNotSupported)，NRC = SNSIAS (serviceNotSupportedIn-ActiveSession)的最终否定响应消息， NRC = SFNS (sub-functionNotSupported)，NRC = SFNSIAS (sub-functionNotSupportedIn-ActiveSession)或NRC = ROOR (requestOutOfRange)，如果它是接收到的请求消息的PDU分析的结果，也应该被发送。

7.5.6 具有物理和功能寻址的多个并发请求消息

常见的服务器实现在服务器中只有一个诊断协议实例可用。 一个诊断协议实例一次只能处理一个请求。 规则是任何收到的消息（不管寻址模式是物理的还是功能的）都会占用这个资源，直到处理请求消息为止（发送最终响应或没有响应的应用程序调用）。

只有两个例外需要分别处理：

— 客户端使用保持活动逻辑将先前启用的会话保持在一个或多个服务器中处于活动状态。 保活 – 逻辑定义为SPRMIB = true的功能性寻址的有效TesterPresent消息，并且必须由旁路逻辑处理。 服务器要确保该特定消息不能“阻塞”服务器的应用层，并且可以处理紧随其后的消息。

— 如果服务器支持一个或多个法定诊断请求，并且在非法定服务（例如，增强型诊断）处于活动状态时收到其中一个请求，则应中止现用服务，启动默认会话并启动法定诊断服务将被处理。 如果编程会话处于活动状态，则此要求不适用。

有关如何处理多个客户端的更多信息，请参见附录J.

8 服务描述约定

8.1 服务说明

本小节定义了本规范中每个诊断服务的描述。 它定义了每个诊断服务的一般服务描述格式。

本小节简要介绍了该服务的功能。 每个诊断服务规范都以客户端和服务器执行的操作的描述开始，这些操作是特定于每个服务的。 每个服务的描述包括一个表格，其中列出了其基元的参数：请求/指示，对正面或负面结果的响应/确认。 全部具有相同的结构：

对于给定的请求/指示和响应/确认A_PDU定义，每个参数的存在由以下约定（Cvt）值之一描述：

表8定义了A_PDU参数约定。

表8 – A_PDU参数约定

类型	名称	描述
M	强制性	该参数必须存在于A_PDU中。
C	条件	该参数可以基于特定标准（例如，A_PDU内的子功能/参数）出现在A_PDU中。
S	选择	表示该参数是必需的（除非另有说明），并且是参数列表中的一项选择。
U	用户选项	该参数可能存在也可能不存在，具体取决于用户的动态使用情况。
注意标记为“M”（强制性）的“<服务名称>请求SID”不得暗示该服务必须得到服务器的支持。 如果服务器支持该服务，’M’仅指示在请求A_PDU中强制存在此参数。		

8.2 请求消息

8.2.1 请求消息定义

本小节包括一个或多个表，它们定义了服务请求/指示的A_PDU（应用层协议数据单元，参见7）参数。 如果不同子功能参数（$ Level）的请求消息在A_Data参数的结构上不同，并且无法在单个表中明确指定，则每个子功能参数（$ Level）可能会有一个单独的表。

表9定义了具有子功能的请求A_PDU定义

表9 – 具有子功能的请求A_PDU定义

A_PDU参数	参数名称	Cvt	字节值	助记符
MType	消息类型	M	xx	公吨
SA	源地址	M	xxxx	SA
TA	目标地址	M	xxxx	TA
TAtype	目标地址类型	M	xx	TAT
RA	远程地址	C	xxxx	RA
A_Data.A_PCI.SI	<服务名称>请求SID	M	xx	SIDRQ
A_Data.Parameter 1	子功能=［参数］	S	xx	LEV_PARAM
A_Data.Parameter 2 : A_Data.Parameter k	数据参数#1 : 数据参数#k-1个	U : U	xx : xx	DP_ ...#1 : DP_ ...#K-1
长度	A_Data的长度	M	xxxxxxxx	LGT
C: RA（远程地址）PDU参数仅在远程寻址的情况下才存在。				

表10定义了没有子功能的请求A_PDU定义。

表10 – 没有子功能的请求A_PDU定义

A_PDU参数	参数名称	Cvt	字节值	助记符
MType	消息类型	M	xx	公吨
SA	源地址	M	xxxx	SA
TA	目标地址	M	xxxx	TA
TAtype	目标地址类型	M	xx	TAT
RA	远程地址	C	xxxx	RA
A_Data.A_PCI.SI	<服务名称>请求SID	M	xx	SIDRQ
A_Data.Parameter 1 : A_Data.Parameter k	数据参数#1 : 数据参数#k中	U : U	xx : xx	DP_ ...#1 : DP_ ...#k中
长度	A_Data的长度	M	xxxxxxxx	LGT
C: RA（远程地址）PDU参数仅在远程寻址的情况下才存在。				

在所有请求/指示中，寻址信息MType，TA，SA，TAtype和Length是强制性的。 寻址信息RA可选地存在。

注意为了定义目的，地址信息显示在上表中。 进一步的服务请求/指示定义仅指定A_Data A_PDU参数，因为A_Data A_PDU参数表示服务请求/指示的消息数据字节。

8.2.2 请求消息子函数参数$ Level (LEV_) 定义

本小节定义了为服务<服务名称>的请求/指示定义的子函数$ levels (LEV_) 参数。

如果所描述的服务不使用子函数参数值并且没有使用suppressPosRspMsgIndicationBit (这隐含地指示需要响应)，则本小节不包含任何定义。

子功能参数字节分为表11中定义的两部分（位级）。

表11 - 子功能参数结构

位的位置	描述
7	**suppressPosRspMsgIndicationBit** 该位指示服务器是否应抑制肯定响应消息。 '0'= FALSE，不要抑制肯定的答复信息（需要肯定的答复信息）。 '1'=真，禁止响应消息（不应发送肯定响应消息;被寻址的服务器不应发送肯定响应消息）。 独立于suppressPosRspMsgIndicationBit，负面响应消息由服务器根据7.5中规定的限制发送。 即使不需要积极响应（即，SPRMIB = true），也必须完全传递服务的执行，以保持实现一致，而不管SPRMIB的值如何。 对于任何给定的服务，服务器支持的所有子功能参数值（即，子功能结构的比特6-0）都应支持'0'和'1'的suppressPosRspMsgIndicationBit值。
6-0	**子功能参数值** 子功能参数的位0-6包含服务的子功能参数值（0x00 - 0x7F）。

子功能参数值是7位值（子功能参数字节的位6-0），可以有多个值来进一步指定服务行为。

除了suppressPosRspMsgIndicationBit之外，支持子函数参数值的服务应支持在子函数参数值表中定义的子函数参数值。

每个服务都包含一个表格，用于定义子功能参数值的值，仅考虑位0-6。

注意如果SPRMIB对于大量数据的响应为TRUE，需要使用分页缓冲区处理，则可能导致第一批数据的传输仍可能在响应时间窗口内启动，但服务执行的终止超出了响应时间窗口的限制。 如果在这种情况下响应受到抑制，则无法通知客户端延迟，但服务器仍处于忙碌状态，尚未准备好接收其他请求。 对于客户端，建议不要求大量数据，并在同一请求中设置SPRMIB（例如，SID 0x19 SF 0x0A），因为这会破坏SPRMIB的目的。 对于服务器实现，建议发送NRC 0x78 (RCRRP)，并随后发送肯定响应，以防在SPRMIB为TRUE时使用分页缓冲区处理。

表12定义了请求消息子功能参数定义。

表12 – 请求消息子功能参数定义

位6 – 0	描述	Cvt	助记符
xx	**子功能#1** 子功能参数#1的描述	宙	SUBFUNC1
:	:	:	:
xx	**子功能#m的** 子功能参数#m的描述	宙	SUBFUNCm

上表12中的约定（Cvt）栏应按表13中的定义进行解释。

表13 – 子功能参数约定

类型	名称	描述
M	强制性	服务器支持时，子功能参数必须被服务器支持。
U	用户选项	根据服务的使用情况，服务器可能支持或不支持子功能参数。

完整的子功能参数字节值是根据suppressPosRspMsgIndicationBit的值和所选的子功能参数值计算得出的。

表14定义了子功能字节值的计算。

表14 – 子函数字节值的计算

位7	位6	位5	位4	位3	位2	位1	位0
SuppressPosRspMsg IndicationBit	SubFunction参数值在服务的子功能参数值表中指定						
得到的子函数参数字节值（位7-0）							

8.2.3 请求消息数据参数定义

本小节为服务的请求/指示定义了数据参数$ DataParam (DP_)
<服务名称>。 如果所描述的服务不使用任何数据参数，本小节不包含任何定义。 数据参数部分可以包含多个字节。 本小节提供了每个数据参数的一般描述。 详细的定义可以在本文件的附件中找到。 具体附件取决于服务。 附件还规定了在服务器支持服务的情况下是否应该支持数据参数或者是否支持用户可选。

表15定义了请求消息数据参数。

表15 – 请求消息数据参数定义

定义
数据参数#1 数据参数#1的描述
:
数据参数#N 数据参数#n的描述

8.3 积极的回应消息

8.3.1 积极响应消息的定义

本小节定义了服务响应/确认的A_PDU参数（参见7.2关于应用层协议数据单元A_PDU的详细描述）。 当不同子功能参数$ Level的响应消息在A_Data参数的结构上不同时，每个子功能参数$ Level可能会有一个单独的表。

注意诊断服务（如果需要）的肯定响应消息应在执行诊断服务后发送。 如果诊断服务需要不同的处理（例如ECUReset服务），则可以在诊断服务的服务描述中找到何时发送肯定响应消息的适当描述。

表16定义了肯定响应A_PDU。

表16 – 正面响应A_PDU

A_PDU参数	参数名称	Cvt	字节值	助记符
SA	源地址	M	xxxx	SA
TA	目标地址	M	xxxx	TA
TAtype	目标地址类型	M	xx	TAT
RA	远程地址	C	xxxx	RA
A_Data.A_PCI.SI	<服务名称>响应SID	S	xx	SIDPR
A_Data.Parameter 1 : A_Data.Parameter k	数据参数#1 : 数据参数#k中	U : U	xx : xx	DP_ ... #1 : DP_ ... #k中
C: RA（远程地址）PDU参数仅在远程寻址的情况下才存在。				

在所有响应/确认中，寻址信息TA，SA和TAtype是强制性的。 寻址信息RA可选地存在。

注意为了定义目的，地址信息显示在上表中。 进一步的服务响应/确认定义仅指定A_Data A_PDU参数，因为A_Data A_PDU参数表示服务响应/确认的消息数据字节。

8.3.2 肯定回复消息数据参数定义

本小节定义了服务<服务名称>的响应/确认的数据参数。 如果所描述的服务不使用任何数据参数，本小节不包含任何定义。 数据参数部分可以包含多个字节。

本小节提供了每个数据参数的一般描述。 详细的定义可以在本文件的附件中找到。 具体附件取决于服务。 附件还规定了在服务器支持服务的情况下是否应该支持数据参数或者是否支持用户可选。

表17定义了响应数据参数。

表17 – 响应数据参数定义

定义
数据参数#1 数据参数#1的描述。 如果请求支持子功能参数字节，则该参数是来自请求消息的子功能参数字节中包含的7位子功能参数值的回波，其中位7设置为零。 来自子函数参数字节的suppressPosRspMsgIndicationBit不会被回显。
:
数据参数#m的 数据参数#m的描述

8.4 支持的否定响应代码（NRC_）

本小节定义了为这项服务实施的否定响应代码。 下面给出了每个响应代码发生的情况。 否定响应消息的定义可以在7.4中找到。 服务器应使用否定响应A_PDU来指示已识别的错误情况。

除了每个服务描述中规定的否定答复代码（如果适用）之外，还应使用附录A.1中列出的否定答复代码。 详情可在附件A.1中找到。

表18定义了支持的否定响应代码。

表18 – 支持的否定响应代码

NRC	描述	助记符
0xXX	NegativeResponseCode#1 1.条件#1 : 米 条件#m的	NRC_
:	:	NRC_
0xXX	NegativeResponseCode#N 1.条件#1 : k 。 条件#k中	NRC_

8.5 消息流示例

本小节包含服务<服务名称>的消息流示例。 所有示例都显示在消息级别（不包含寻址信息）。

表19定义了请求消息流程示例。

表19 – 请求消息流示例

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1 (A_PCI)	<服务名称>请求SID	0xXX	SIDRQ
#2	子功能/数据参数#1	0xXX	LEV_/DP_
:	:	0xXX	DP_
#n	数据参数#m的	0xXX	DP_

表20定义了肯定响应消息流程示例。

表20 – 肯定应答消息流示例

消息方向	服务器 客户端		
消息类型	响应		
A_Data	说明（所有值均为十六进制）	字节值	助记符
#1 (A_PCI)	<服务名称>响应SID	0xXX	SIDPR
#2	数据参数#1	0xXX	DP_
:	:	:	:
#n	数据参数#N-1	0xXX	DP_

如果适用于服务<Service Name>，可能会有多个示例（例如，每个子功能参数$ Level一个）。

表21显示了否定响应消息的消息流示例。

表21 – 否定响应消息流示例

消息方向	服务器 客户端		
消息类型	响应		
A_Data	说明（所有值均为十六进制）	字节值	助记符
#1 (A_PCI.NR_SI)	负面响应SID	0x7F	SIDRSIDNRQ
#2 (A_PCI.SI)	<服务名称>请求SID	0xXX	SIDRQ
#3	responseCode	0xXX	NRC_

9 诊断和通信管理功能单元

9.1 概观

表22定义了诊断和通信管理功能单元。

表22 – 诊断和通信管理功能单元

服务	描述
DiagnosticSessionControl	客户端请求控制与服务器的诊断会话。
ECUReset	客户端强制服务器执行重置。
SecurityAccess	客户端请求解锁安全服务器。
CommunicationControl	客户端控制服务器中通信参数的设置（例如，通信波特率）。
TesterPresent	客户端向服务器指示它仍然存在。
AccessTimingParameter	客户端使用此服务来读取/修改活动通信的时间参数。
SecuredDataTransmission	客户端使用此服务以扩展数据链接安全性执行数据传输。
ControlDTCSetting	客户端控制服务器中DTC的设置。
ResponseOnEvent	客户端请求设置和/或控制服务器中的事件机制。
连接控制	客户端请求控制通信波特率。

9.2 DiagnosticSessionControl (0x10) 服务

9.2.1 服务说明

DiagnosticSessionControl服务用于在服务器中启用不同的诊断会话。

诊断会话在服务器中启用一组特定的诊断服务和/或功能。 该服务提供了服务器可以报告对于启用的诊断会话有效的数据链路层特定参数值（例如，定时参数值）的功能。 本国际标准的用户应定义每个诊断会话中启用的服务和/或功能的确切组合。

应始终只有一个诊断会话在服务器中处于活动状态。 服务器应在启动时始终启动默认的诊断会话。 如果没有启动其他诊断会话，则只要服务器通电，默认诊断会话就应该运行。

服务器应能够在正常操作条件下和车辆制造商规定的其他操作条件下（例如，跛行回家操作条件）提供诊断功能。

如果客户端已经请求了一个已经运行的诊断会话，那么服务器应该发送一个肯定的响应消息，并且如图7所示那样描述在会话之间转换时的服务器内部行为。

无论客户何时请求新的诊断会话，服务器都应在新会话的时间在服务器中处于活动状态之前发送DiagnosticSessionControl肯定响应消息。 有些情况可能要求在发送肯定响应之前必须输入新的会话，同时保持发送响应的旧协议定时。 如果服务器无法启动所请求的新诊断会话，则它应使用DiagnosticSessionControl否定响应消息进行响应，并且当前会话应继续（请参阅diagnosticSession参数定义，以获取有关服务器和客户端应如何操作的更多信息）。 非默认诊断会话（不包括programmingSession）中的一组诊断服务和诊断功能是defaultSession中提供的功能的超集，这意味着切换到任何非默认诊断会话时，defaultSession的诊断功能也可用。 会话可以使车辆制造商特定的服务和功能，这不是本文档的一部分。

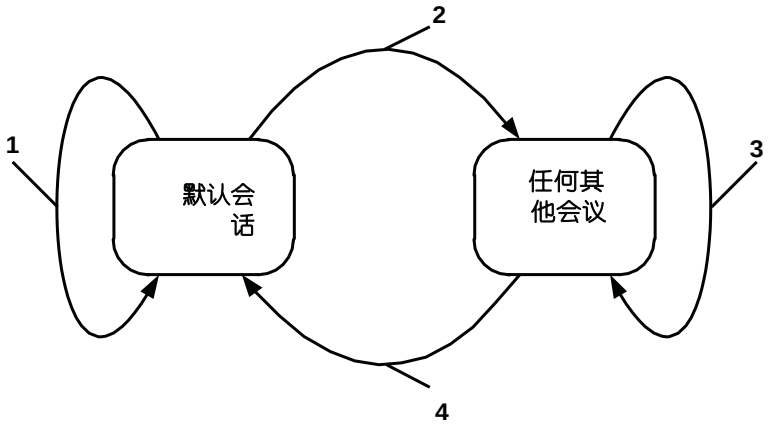
要启动新的诊断会话，服务器可能会要求满足某些条件。 所有这些条件都是用户定义的。 这种情况的例子是：

— 服务器可能仅允许具有特定客户端标识符（客户端诊断地址）的客户端启动特定的新诊断会话（例如服务器可能要求只有具有客户端标识符0xF4的客户端才能启动extendedDiagnosticSession）。

— 可能需要满足某些安全条件（例如，车辆不应移动或发动机不能运转）。

在一些系统中，当新的诊断会话开始时，希望改变通信定时参数。 DiagnosticSessionControl服务实体可以使用适当的服务原语来改变为底层指定的定时参数，以改变本地节点中的通信定时，并可能在客户端想要与之通信的节点中改变通信定时。

图7提供了有关诊断会话转换的概述以及服务器在转换到其他会话时应执行的操作。



键

1 默认会话：当服务器在defaultSession中并且客户端请求启动defaultSession时，服务器应该完全重新初始化defaultSession。 在激活的会话期间，服务器应重置所有激活/启动/更改的设置/控制。 这不包括编入非易失性存储器的长期变化。

2 其他会话：当服务器从defaultSession转换到除defaultSession之外的任何其他会话时，服务器只应停止在defaultSession期间通过ResponseOnEvent（0x86）服务在服务器中配置的事件（类似于stopResponseOnEvent）。

3 相同或其他会话：当服务器从除defaultSession以外的任何诊断会话转换到defaultSession以外的其他会话（包括当前活动的诊断会话）时，服务器应（重新）初始化诊断会话，这意味着：

i) 应该停止通过ResponseOnEvent（0x86）服务在服务器中配置的每个事件。

ii) 安全性应重新锁定。 请注意，安全访问的锁定应将任何依赖于安全访问的活动诊断功能重置为未锁定状态（例如，DID的活动inputOutputControl）。

iii) 应保持新会话中支持并且不依赖安全访问的所有其他活动诊断功能。 例如，任何已配置的周期性调度程序在从一个非defaultSession转换到另一个或同一个非DefaultSession时应保持活动状态，并且不会影响CommunicationControl和ControlDTCSetting服务的状态，这意味着正常通信应保持禁用状态在会话切换的时间点禁用。

4 默认会话：当服务器从默认会话以外的任何诊断会话转换到defaultSession时，服务器应通过ResponseOnEvent（0x86）服务停止服务器中配置的每个事件，并启用安全性。 任何其他在defaultSession中不支持的活动诊断功能都将被终止。例如，任何已配置的周期性调度程序或输出控制应被禁用，并且CommunicationControl和ControlDTCSetting服务的状态应被重置，这意味着正常通信在会话切换到时被禁用时应重新启用defaultSession。 在激活的会话期间，服务器应重置所有激活/启动/更改的设置/控制。 这不包括编入非易失性存储器的长期变化。

图7 – 服务器诊断会话状态图

表23定义了在defaultSession和非defaultSession（定时服务）期间允许的服务。 任何非defaultSession都与必须由客户端保持活动状态的诊断会话计时器相关联。

表23 – 默认和非默认诊断会话期间允许的服务

服务	defaultSession	非defaultSession
DiagnosticSessionControl – 0x10	x	x
ECUReset – 0x11	x	x
SecurityAccess – 0x27	不适用	x
CommunicationControl – 0x28	不适用	x
TesterPresent – 0x3E	x	x
AccessTimingParameter – 0x83	不适用	x
SecuredDataTransmission – 0x84	不适用	x
ControlDTCSetting – 0x85	不适用	x
ResponseOnEvent – 0x86	x[a]	x
LinkControl – 0x87	不适用	x
ReadDataByIdentifier – 0x22	x[b]	x
ReadMemoryByAddress – 0x23	x[c]	x
ReadScalingDataByIdentifier – 0x24	x[b]	x
ReadDataByPeriodicIdentifier – 0x2A	不适用	x
DynamicallyDefineDataIdentifier – 0x2C	x[d]	x
WriteDataByIdentifier – 0x2E	x[b]	x
WriteMemoryByAddress – 0x3D	x[c]	x
ClearDiagnosticInformation – 0x14	x	x
ReadDTCInformation – 0x19	x	x
InputOutputControlByIdentifier – 0x2F	不适用	x
RoutineControl – 0x31	x[e]	x
请求下载 – 0x34	不适用	x
请求上传 – 0x35	不适用	x
TransferData – 0x36	不适用	x
RequestTransferExit – 0x37	不适用	X
RequestFileTransfer – 0x38	不适用	X

[a] 具体实现是否在defaultSession期间允许ResponseOnEvent服务。

[b] [c]安全内存区域需要SecurityAccess服务，因此需要非默认诊断会话[d]dataIdentifier可以在默认和非默认情况下动态定义诊断会话。

[e] 安全例程需要一个SecurityAccess服务，因此需要一个非默认的诊断会话。 需要客户端主动停止的例程也需要非默认会话。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

9.2.2 请求消息

9.2.2.1 请求消息定义

表24定义了请求消息。

表24 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	DiagnosticSessionControl请求SID	M	0x10	DSC
#2	子函数= [diagnosticSessionType]	M	0x00 – 0xFF	LEV_DS_

9.2.2.2 请求消息子函数参数$ Level (LEV_) 定义

DiagnosticSessionControl服务使用子函数参数diagnosticSessionType来选择服务器的特定行为。 表25详细说明了可能的诊断会话的解释和用法。

指定了以下子函数值 (suppressPosRspMsgIndicationBit (bit 7) 未显示) 。

表25 – 请求消息子功能参数定义

比特6-0	描述	Cvt	助记符
0x00	**ISOSAEReserved** 该值由本文档保留。	M	ISOSAERESRVD
0x01	**defaultSession** 此诊断会话启用服务器中的默认诊断会话，并且不支持任何诊断应用程序超时处理规定 (例如，不需要TesterPresent服务来保持会话处于活动状态) 。 如果除defaultSession以外的任何其他会话在服务器中处于活动状态，并且defaultSession再次启动，则应遵循以下实施规则 (另请参阅上面给出的服务器诊断会话状态图) ： 服务器在发送DiagnosticSessionControl肯定响应消息时应停止当前的诊断会话，并应在之后启动新请求的诊断会话。 如果服务器发送了DiagnosticSessionControl肯定响应消息，则在诊断会话期间，如果客户端解锁了服务器，则它应该重新锁定服务器。 如果服务器发送带有DiagnosticSessionControl请求服务标识符的否定响应消息，则应继续活动会话。 注意如果使用的数据链路需要初始化步骤，则初始化的服务器应默认启动默认的诊断会话。 在初始化步骤之后，不需要将diagnosticSession设置为defaultSession的DiagnosticSessionControl。	M	DS

表25 – （续）

比特6-0	描述	Cvt	助记符
0x02	**ProgrammingSession** 此诊断会话启用支持服务器内存编程所需的所有诊断服务。 如果服务器在引导软件中运行programmingSession，则只能通过客户端启动的ECUReset（0x11）服务，sessionType等于defaultSession的DiagnosticSessionControl（0x10）服务或服务器中的会话层超时保留programmingSession。 如果服务器在启动软件中运行，并且它接收到sessionType等于defaultSession的DiagnosticSessionControl（0x10）服务，或者会话层发生超时并且两种情况都存在有效的应用程序软件，则服务器应重新启动应用程序软件。 本文档没有具体说明如何实现有效应用软件重启的各种实现方法（例如，可以在启动软件中直接确定有效的应用软件，执行ECU复位时的ECU启动阶段等）。	U	PRGS
0x03	**extendedDiagnosticSession** 该诊断会话可用于启用支持“怠速，CO值等”功能调整所需的所有诊断服务。在服务器的内存中。 它也可以用于启用诊断服务，这些诊断服务与功能调整无关（例如，参见表23中的定时服务）。	U	EXTDS
0x04	**safetySystemDiagnosticSession** 该诊断会话支持所有需要的诊断服务，以支持与安全系统相关的功能（例如安全气囊部署）。	U	SSDS
0x05 – 0x3F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
0x40 – 0x5F	**vehicleManufacturerSpecific** 这个值的范围保留给车辆制造商特定用途。	U	VMS
0x60 – 0x7E	**systemSupplierSpecific** 这个值的范围保留给系统供应商特定用途。	U	SSS
0x7F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

9.2.2.3 请求消息数据参数定义

该服务不支持请求消息中的数据参数。

9.2.3 积极的回应消息

9.2.3.1 积极响应消息的定义

表26定义了肯定响应消息的定义。

表26 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	DiagnosticSessionControl响应SID	M	0x50	DSCPR
#2	子函数= [diagnosticSessionType]	M	0x00 – 0xFF	LEV_DS_
#3 : #6	sessionParameterRecord []#1 = [数据#1 : 数据#4]	M : M	0x00 – 0xFF : 0x00 – 0xFF	SPREC_DATA_1 : DATA_m

9.2.3.2 肯定回复消息数据参数定义

表27定义了响应消息数据参数定义。

表27 – 响应消息数据参数定义

定义
diagnosticSessionType 该参数是来自请求消息的子功能参数的比特6-0的回声。
sessionParameterRecord 此参数记录包含服务器报告的会话特定参数值。 sessionParameterRecord的内容在表28和表29中定义。

表28 和 表29 确定 该 结构体 的 该 响应 信息 数据参数 sessionParameterRecord，适用于在支持的数据链路上实施此服务。

表28 – sessionParameterRecord定义

字节 POS 机。 在记录中	描述	Cvt	字节值	助记符
#1		M	0x00 – 0xFF	SPREC_P2SMH
#2		M	0x00 – 0xFF	P2SML
#3		M	0x00 – 0xFF	P2ESMH
#4		M	0x00 – 0xFF	P2ESML

表29 – sessionParameterRecord内容定义

参数	描述	字节数	解析度	最小值	最大值
$P2_{Server\_max}$	服务器支持激活的诊断会话的默认$P2_{Server_max}$计时。	2	1毫秒	0毫秒	65 535毫秒
$P2^*_{Server\_max}$	增强型（NRC 0x78）$P2_{Server_max}$服务器支持激活的诊断会话。	2	10毫秒	0毫秒	655 350毫秒

有关$P2_{服务器}$和$P2^*_{服务器}$的更多详细信息，请参阅ISO 14229-2。

9.2.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表30中。如果错误情况适用于服务器，则应使用列出的否定响应。

表30 – 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果不满足请求DiagnosticSessionControl的标准，则应返回此NRC。	CNC

9.2.5 消息流示例 (s) DiagnosticSessionControl

9.2.5.1 示例#1 – 开始编程会话

此消息流显示如何在服务器中启用诊断会话"programmingSession"。 通过将suppressPosRspMsgIndicationBit (子函数参数的位7) 设置为 "FALSE" ('0'), 客户端请求获得响应消息。 对于给定的例子，假定$P2_{Server_max}$等于50ms并且$P2^*_{Server_max}$等于5 000ms。

表31定义了DiagnosticSessionControl请求消息流程示例#1。

表31 – DiagnosticSessionControl请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	DiagnosticSessionControl请求SID	0x10	DSC
#2	diagnosticSessionType = programmingSession, suppressPosRspMsgIndicationBit = FALSE	0x02	DS_ECUPRGS

表32定义了DiagnosticSessionControl肯定响应消息流程示例#1。

表32 – DiagnosticSessionControl肯定响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	DiagnosticSessionControl响应SID	0x50	DSCPR
#2	diagnosticSessionType = programmingSession	0x02	DS_ECUPRGS
#3	sessionParameterRecord [$P2_{Server\_max}$ (高字节)]	0x00	SPREC_1
#4	sessionParameterRecord [$P2_{Server\_max}$ (低字节)]	0x32	SPREC_2
#5	sessionParameterRecord [$P2^*_{Server\_max}$ (高字节)]	0x01	SPREC_3
#6	sessionParameterRecord [$P2^*_{Server\_max}$ (低字节)]	0xF4	SPREC_4

9.3 ECUReset (0x11) 服务

9.3.1 服务说明

客户端使用ECUReset服务请求重置服务器。

该服务请求服务器根据ECUReset请求消息中嵌入的resetType参数值的内容有效地执行服务器重置。 ECUReset肯定响应消息（如果需要）应在服务器中执行复位之前发送。 服务器重置成功后，服务器应激活defaultSession。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

ISO 14229的这部分内容没有定义ECU从正面响应消息到ECU复位请求的时间，直到复位成功完成。 建议在此期间ECU不接受任何请求消息并发送任何响应消息。

9.3.2 请求消息

9.3.2.1 请求消息定义

表33定义了请求消息定义。

表33 - 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ECUReset请求SID	M	0x11	ER
#2	子功能= [resetType]	M	0x00 - 0xFF	LEV_RT_

9.3.2.2 请求消息子函数参数$ Level (LEV_) 定义

ECUReset请求消息使用子函数参数resetType来描述服务器如何执行重置（suppressPosRspMsgIndicationBit（bit 7）未显示）。

表34定义了请求消息子功能参数定义。

表34 - 请求消息子功能参数定义

位6 - 0	描述	Cvt	助记符
0x00	**ISOSAEReserved** 该值由本文档保留。	M	ISOSAERESRVD
0x01	**在硬格** 该值标识了模拟通常在服务器先前已从其电源（即电池）断开连接之后执行的开机/启动序列的“硬重置”条件。 执行的操作是特定于实现的，并未由标准定义。 这可能导致易失性存储器和非易失性存储器位置重新初始化为预定值。	U	HR

表34 – （续）

位6 – 0	描述	Cvt	助记符
0x02	**keyOffOnReset** 该值标识与司机将点火钥匙关闭并重新打开相似的状况。 这个复位条件应该模拟一个关断序列（即中断开关电源）。 执行的操作是特定于实现的，并未由标准定义。 通常，非易失性存储器位置的值被保留；易失性存储器将被初始化。	U	KOFFONR
0x03	**softReset** 此值标识“软重置”条件，如果适用，这会导致服务器立即重新启动应用程序。执行的操作是特定于实现的，并未由标准定义。 典型的操作是重新启动应用程序，而不重新初始化以前学习的配置数据，自适应因素和其他长期调整。	U	SR
0x04	**enableRapidPowerShutDown** 该子功能适用于不由点火提供动力但仅由电池供电的ECU。 因此，关机强制进入睡眠模式，而不是关闭电源。 睡眠意味着断电，但仍然可以唤醒（电池供电）。 子功能的目的是在点火转到关闭位置后减少ECU的待机时间。 该值请求服务器启用并执行“快速关闭电源”功能。 一旦“钥匙/点火”关闭，服务器应立即执行该功能，当服务器执行断电功能时，应直接或在定义的准备时间之后转换到睡眠模式，如果客户需要响应消息，并且服务器已经准备好执行“快速关机”功能，则服务器应在“快速关机”功能开始之前发送肯定响应消息，下一次出现“开机键”或“点火开启”信号终止“快速关闭电源”功能。 注意此子功能仅适用于支持待机模式的服务器！	U	ERPSD
0x05	**disableRapidPowerShutDown** 该值请求服务器禁用先前启用的“快速关闭电源”功能。	U	DRPSD
0x06 – 0x3F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD
0x40 – 0x5F	**vehicleManufacturerSpecific** 这个值的范围保留给车辆制造商特定用途。	U	VMS
0x60 – 0x7E	**systemSupplierSpecific** 这个值的范围保留给系统供应商特定用途。	U	SSS
0x7F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

9.3.2.3 请求消息数据参数定义

该服务不支持请求消息中的数据参数。

9.3.3 积极的回应消息

9.3.3.1 积极响应消息的定义

表35定义了肯定响应消息。

表35 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ECUReset响应SID	M	0x51	ERPR
#2	子功能= [resetType]	M	0x00 – 0x7F	LEV_RT_
#3	powerDownTime	C	0x00 – 0xFF	太平洋夏令时
C: 如果子函数参数设置为enableRapidPowerShutDown值 (0x04)，则此参数存在;				

9.3.3.2 肯定回复消息数据参数定义

表36定义了响应消息的数据参数。

表36 – 响应消息数据参数定义

定义
resetType 该参数是来自请求消息的子功能参数的比特6-0的回声。
powerDownTime 该参数向客户端指示服务器将保持在断电顺序中的待机顺序的最小时间。 该参数的分辨率是每次计数一 (1) 秒。 以下值是有效的: — 0x00 – 0xFE: 0 – 254秒powerDownTime, — 0xFF: 表示失败或时间不可用。

9.3.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 表37中记录了每个响应代码发生的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表37 – 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果未满足ECUReset请求的标准，则将返回该NRC。	CNC
0x33	**securityAccessDenied** 如果请求的重置受到保护且服务器未处于解锁状态，则应发送此NRC。	伤心

9.3.5 消息流示例ECUReset

本小节指定了满足在服务器中成功执行ECUReset服务的例子的条件。

服务器状况：点火=开，系统不应处于运行模式（例如，如果系统是发动机管理，发动机应关闭）。

通过将suppressPosRspMsgIndicationBit（子函数参数的第7位）设置为'FALSE'，客户端请求获得响应消息。

在服务器执行resetType之前，服务器应发送ECUReset肯定响应消息。 表38定义了ECUReset请求消息流程示例#1。

表38 – ECUReset请求消息流程示例#1

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ECUReset请求SID	0x11	ER
#2	ResetType = hardReset, suppressPosRspMsgIndicationBit = FALSE	0x01	RT_HR

表39定义了ECUReset积极响应消息流程示例#1。

表39 – ECUReset正面响应消息流程示例#1

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ECUReset响应SID	0x51	ERPR
#2	resetType = hardReset	0x01	RT_HR

9.4 SecurityAccess (0x27) 服务

9.4.1 服务说明

此服务的目的是提供访问数据和/或诊断服务的手段，这些服务因安全，排放或安全原因而受到限制。 用于将例程或数据下载/上传到服务器和从服务器读取特定存储器位置的诊断服务是可能需要安全访问的情况。 下载到服务器的不正常程序或数据可能会损坏电子设备或其他车辆部件，或者导致车辆遵守排放，安全或安全标准。安全概念使用种子和密钥关系。

使用此服务的典型示例如下所示:

— 客户端请求“种子”,

— 服务器发送“种子”,

— 客户端发送“密钥”(适用于接收的种子),

— 服务器响应“密钥”有效，并且它将自行解锁。

'requestSeed'子功能参数值应始终为奇数，并且相同安全级别的相应'sendKey'子功能参数值应等于'requestSeed'子功能参数值加1。

在任何时刻只有一个安全级别处于活动状态。 例如，如果与requestSeed 0x03关联的安全级别处于活动状态，并且测试者请求已成功解锁与requestSeed 0x01关联的安全级别，则只有与requestSeed 0x01关联的安全级别支持的安全功能才能在此时解锁。 之前由与requestSeed 0x03关联的安全级别解锁的任何其他安全功能将不再处于活动状态。 安全级别编号是任意的，并不意味着级别之间的任何关系。

客户端应通过发送服务SecurityAccess'requestSeed'消息来请求服务器“解锁”。 服务器应通过使用服务SecurityAccess'requestSeed'肯定响应消息发送“种子”来响应。 然后，客户端应该通过使用适当的服务SecurityAccess'sendKey'请求消息返回一个“密钥”号码回应服务器。 服务器应将此“密钥”与内部存储/计算的密钥进行比较。 如果两个号码匹配，则服务器应启用(“解锁”)客户端对特定服务/数据的访问，并通过服务SecurityAccess'sendKey'正面响应消息指示。 如果两个数字不匹配，则应视为错误的访问尝试。 无效密钥要求客户端从SecurityAccess'requestSeed'消息的开头重新开始(有关安全访问处理详细信息的更多详细信息，请参阅附录I)。

如果服务器支持安全性，但在接收到SecurityAccess'requestSeed'消息时所请求的安全级别已解锁，则该服务器应使用种子值等于零(0)的SecurityAccess'requestSeed'肯定响应消息服务进行响应。 服务器永远不会为当前锁定的给定安全级别发送全零种子。 客户端应使用此方法通过检查非零种子来确定服务器是否被锁定为特定的安全级别。

在服务器能够在服务器上电/重置之后以及在一定次数的错误访问尝试(参见下面的进一步描述)之后，积极响应来自客户端的服务SecurityAccess'requestSeed'消息之前，可能需要车辆制造商特定的时间延迟。 如果支持此延迟计时器，则应在车辆制造商指定的错误访问次数达到或服务器启动/重置以及之前执行的SecurityAccess服务由于单次错误访问尝试而失败后启用延迟。 如果服务器支持这个延迟计时器，那么在执行成功的SecurityAccess服务'sendKey'之后，服务器会在服务器清除上电/复位时延迟计时器调用的服务器内部指示信息。 如果服务器支持这个延时定时器，并且无法确定之前在加电/复位之前执行的SecurityAccess服务是否失败，那么延时定时器应在加电/复位后始终处于活动状态。 只有当服务器在加电/复位时被锁定时才需要延迟。 车辆制造商应选择是否支持延时定时器。

尝试访问安全性不应妨碍正常的车辆通信或其他诊断通信。

如果服务器被锁定时请求安全服务，则提供安全性的服务器应支持拒绝消息。

在特定诊断会话期间请求的某些诊断功能/服务可能需要成功的安全访问序列。 在这种情况下，需要以下一系列服务：

DiagnosticSessionControl服务，

SecurityAccess服务，

安全的诊断服务。

服务器中启用的diagnosticSession（会话已启动）允许使用不同的accessModes。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

9.4.2 请求消息

9.4.2.1 请求消息定义

表40定义了请求消息定义 – 子功能= requestSeed。

表40 – 请求消息定义 – 子功能= requestSeed

A_Data字节	参数名称	Cvt	字节值	助记符
#1	SecurityAcces请求SID	M	0x27	SA
#2	子功能= [securityAccessType = requestSeed]	M	0x01, 0x03, 0x05, 0x07 – 0x7D	LEV_ SAT_RSD
#3 : #n	securityAccessDataRecord[] = [参数#1 : 参数#m]	U : U	0x00 – 0xFF : 0x00 – 0xFF	SECACCDR_ PARA1 : PARAm

表41定义了请求消息定义 – 子功能= sendKey。

表41 – 请求消息定义 – 子功能= sendKey

A_Data字节	参数名称	Cvt	字节值	助记符
#1	SecurityAcces请求SID	M	0x27	SA
#2	子函数= [securityAccessType = sendKey]	M	0x02, 0x04, 0x06, 0x08 – 0x7E	LEV_SAT_SK
#3 : #n	securityKey[] = [键#1 (高字节) : 键#m (低字节)]	M : U	0x00 – 0xFF : 0x00 – 0xFF	SECKEY_ KEY1HB : KEYmLB

9.4.2.2 请求消息子函数参数$ Level (LEV_) 定义

子函数参数securityAccessType向服务器指示该服务正在进行的步骤，客户端要访问的安全级别以及种子和密钥的格式。 如果服务器支持不同级别的安全性，则每个级别应由requestSeed值标识，该值与sendKey值具有固定关系：

— “requestSeed = 0x01”标识“requestSeed = 0x01”和“sendKey = 0x02”之间的固定关系

— “requestSeed = 0x03”标识“requestSeed = 0x03”和“sendKey = 0x04”之间的固定关系

表42中定义了requestSeed和sendKey (suppressPosRspMsgIndicationBit (bit 7) 未显示) 的值。

表42 - 请求消息子功能参数定义

位6 - 0	描述	Cvt	助记符
0x00	**ISOSAEReserved** 该值由本文档保留。	M	ISOSAERESRVD
0x01	**requestSeed** RequestSeed具有车辆制造商定义的安全级别。	U	RSD
0x02	**sendKey** SendKey与车辆制造商定义的安全等级。	U	SK
0x03, 0x05, 0x07 – 0x41	**requestSeed** RequestSeed制 与不同的水平 的安全 定义 通 该 车辆造商。 过	U	RSD
0x04, 0x06, 0x08 – 0x42	**sendKey** SendKey与制造 不同层次的 安全 定义 通 该 车辆商。 过	U	SK
0x43 – 0x5E	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
0x5F	**ISO26021-2值** RequestSeed具有不同级别的安全性，用于生命末期激活如ISO 26021-2中定义的车载烟火装置。	U	RSD
0x60	**ISO26021-2 sendKey值** SendKey具有不同级别的安全性，用于生命末期激活如ISO 26021-2中定义的车载烟火装置。	U	SK
0x61 – 0x7E	**systemSupplierSpecific** 这个值的范围保留给系统供应商特定用途。	U	SSS
0x7F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

9.4.2.3 请求消息数据参数定义

表43定义了请求消息的数据参数。

表43 – 请求消息数据参数定义

定义
securityKey（高字节和低字节） 请求消息中的"Key"参数是由与特定"种子"值相对应的安全算法生成的值。
securityAccessDataRecord 此参数记录是用户可选的，用于在请求种子信息时将数据传输到服务器。 它可以包含服务器中验证的客户端标识。

9.4.3 积极的回应消息

9.4.3.1 积极响应消息的定义

表44定义了肯定响应消息。

表44 – 肯定应答消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	SecurityAccess响应SID	M	67	SAPR
#2	子函数=［securityAccessType］	M	00-7F	LEV_SAT_SK
#3 ： #n	securitySeed[] = [种子#1（高字节） ： 种子#m（低字节）］	C ： C	0x00 – 0xFF ： 0x00 – 0xFF	SECSEED_ SEED1HB ：SEEDmLB
C: 此参数的存在取决于securityAccessType参数。 如果securityAccessType参数指示客户端想要从服务器检索种子，则必须存在。				

9.4.3.2 肯定回复消息数据参数定义

表45定义了响应消息的数据参数。

表45 – 响应消息数据参数定义

定义
securityAccessType 该参数是来自请求消息的子功能参数的比特6-0的回声。
securitySeed（高字节和低字节） 种子参数是服务器发送的数据值，客户端在计算访问安全性所需的密钥时将使用该值。 如果请求消息是通过将子函数设置为请求服务器种子的值发送的，那么securitySeed数据字节仅出现在响应消息中。

9.4.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表46中。如果错误情况适用于服务器，则应使用列出的否定响应。

表46 – 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果未满足请求安全访问标准，则应返回此NRC。	CNC
0x24	**requestSequenceError** 如果在未收到'requestSeed'请求消息的情况下收到'sendKey'子功能，则发送。	RSE
0x31	**requestOutOfRange** 如果用户可选securityAccessDataRecord包含无效数据，则应发送此NRC。	ROOR
0x35	**无效的密钥** 如果收到预期的'sendKey'子函数值并且密钥的值与服务器的内部存储/计算密钥不匹配，则发送。	IK
0x36	**exceededNumberOfAttempts** 如果延迟计时器由于超出允许的错误访问尝试的最大次数而处于活动状态，则发送。	ENOA
0x37	**requiredTimeDelayNotExpired** 如果延迟计时器处于活动状态并发送请求，则发送。	RTDNE

9.4.5 消息流示例 (s) SecurityAccess

9.4.5.1 假设

对于以下给出的消息流示例，如果服务器处于"锁定"状态，则必须满足以下条件才能成功解锁服务器:

— 子功能请求种子: 0x01 (requestSeed)

— 子功能发送密钥: 0x02 (sendKey)

— 服务器种子 (2字节) : 0x3657

— 服务器密钥 (2字节) : 0xC9A9 (例如种子值的2的补码)

通过将suppressPosRspMsgIndicationBit (子函数参数的第7位) 设置为 "FALSE" ('0') , 客户端请求获得响应消息。

9.4.5.2 示例#1 – 服务器处于“锁定”状态

9.4.5.2.1 步骤#1：请求种子

表47定义了安全访问请求消息流程示例#1-步骤#1。

表47 – SecurityAccess请求消息流程示例#1 – 步骤#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	SecurityAccess请求SID	0x27	SA
#2	SecurityAccessType = requestSeed, suppressPosRspMsgIndicationBit = FALSE	0x01	SAT_RSD

表48定义了安全访问肯定响应消息流程示例#1-步骤#1。

表48 – SecurityAccess肯定响应消息流程示例#1 – 步骤#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	SecurityAccess响应SID	0x67	SAPR
#2	securityAccessType = requestSeed	0x01	SAT_RSD
#3	securitySeed [byte#1] =种子#1（高字节）	0x36	SECHB
#4	securitySeed [byte#2] =种子#2（低字节）	0x57	SECLB

9.4.5.2.2 步骤#2：发送密钥

表49定义了SecurityAccess请求消息流程示例#1 – 步骤#2。

表49 – SecurityAccess请求消息流程示例#1 – 步骤#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	SecurityAccess请求SID	0x27	SA
#2	securityAccessType = sendKey, suppressPosRspMsgIndicationBit = FALSE	0x02	SAT_SK
#3	securityKey [byte#1] = key#1（高字节）	0xC9	SECKEY_HB
#4	securityKey [byte#2] = key#2（低字节）	0xA9	SECKEY_LB

表50定义了SecurityAccess肯定响应消息流程示例#1-步骤#2。

表50 - SecurityAccess肯定响应消息流程示例#1 - 步骤#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	SecurityAccess响应SID	0x67	SAPR
#2	securityAccessType = sendKey	0x02	SAT_SK

9.4.5.3 示例#2 - 服务器处于“解锁”状态

9.4.5.3.1 步骤#1：请求种子

表51定义了SecurityAccess请求消息流程示例#2 - 步骤#1。

表51 - SecurityAccess请求消息流程示例#2 - 步骤#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	SecurityAccess请求SID	0x27	SA
#2	securityAccessType = requestSeed, suppressPosRspMsgIndicationBit = FALSE	0x01	SAT_RSD

表52定义了安全访问肯定响应消息流程示例#2 - 步骤#2。

表52 - SecurityAccess肯定响应消息流程示例#2 - 步骤#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	SecurityAccess响应SID	0x67	SAPR
#2	securityAccessType = requestSeed	0x01	SAT_RSD
#3	securitySeed [byte#1] =种子#1（高字节）	0x00	SECHB
#4	securitySeed [byte#2] =种子#2（低字节）	0x00	SECLB

9.5 CommunicationControl (0x28) 服务

9.5.1 服务说明

此服务的目的是打开/关闭（a）服务器（如应用程序通信消息）的某些消息的传输和/或接收。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

9.5.2 请求消息

9.5.2.1 请求消息定义

表53定义了请求消息。

表53 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	CommunicationControl请求SID	M	0x28	CC
#2	子函数= [controlType]	M	0x00 – 0xFF	LEV_CTRLTP
#3	communicationType	M	0x00 – 0xFF	CTP
#4	nodeIdentificationNumber (高字节)	C[a]	0x00 – 0xFF	NIN
#5	nodeIdentificationNumber (低字节)	C[a]	0x00 – 0xFF	NIN
[a] C参数的存在需要controlType为0x04或0x05。				

9.5.2.2 请求消息子函数参数$ Level (LEV_) 定义

子函数参数controlType包含有关服务器应如何修改communicationType参数中引用的通信类型(suppressPosRspMsgIndicationBit (bit 7) 未在表54中显示) 的信息。

表54 – 请求消息子功能参数定义

位6 – 0	描述	Cvt	助记符
0x00	**enableRxAndTx** 该值表示应该为指定的通信类型启用消息的接收和传输。	U	ERXTX
0x01	**enableRxAndDisableTx** 该值表示应启用消息接收，并且对于指定的通信类型应禁用传输。	U	ERXDTX
0x02	**disableRxAndEnableTx** 此值表示应禁用消息接收，并且应为指定的通信类型启用传输。	U	DRXETX
0x03	**disableRxAndTx** 该值表示对于指定的通信类型应禁用消息的接收和传输。	U	DRXTX
0x04	**enableRxAndDisableTxWithEnhancedAddressInformation** 该值表示寻址的总线主机应将相关的子总线段切换到仅诊断调度模式。	U	ERXDTXWEAI
0x05	**enableRxAndTxWithEnhancedAddressInformation** 该值表示寻址的总线主机应将相关的子总线段切换到应用程序调度模式。	U	ERXTXWEAI
0x06 – 0x3F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD
0x40 – 0x5F	**vehicleManufacturerSpecific** 这个值的范围保留给车辆制造商特定用途。	U	VMS

表54 – （续）

位6 – 0	描述	Cvt	助记符
0x60 – 0x7E	**systemSupplierSpecific** 这个值的范围保留给系统供应商特定用途。	U	SSS
0x7F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

9.5.2.3 请求消息数据参数定义

表55定义了请求消息的数据参数。

表55 – 请求消息数据参数定义

定义
communicationType 该参数用于引用要控制的通信类型。 communicationType参数是一个位代码值，它允许同时控制多种通信类型。 （关于通信类型数据参数的编码，见附录B.1）
nodeIdentificationNumber 该2字节参数用于识别车辆某处的子网络上的节点，无法使用较低OSI层1至6的寻址方法寻址该参数。仅当子功能参数controlType为设置为0x04或0x05（关于nodeIdentificationNumber数据参数的编码，请参见附录B.4）

9.5.3 积极的回应消息

9.5.3.1 积极响应消息的定义

表56定义了肯定响应消息。

表56 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	CommunicationControl响应SID	M	0x68	CCPR
#2	子函数= [controlType]	M	0x00 – 0x7F	LEV_CTRLTP

9.5.3.2 肯定回复消息数据参数定义

表57定义了肯定响应消息的数据参数。

表57 – 响应消息数据参数定义

定义
controlType
该参数是来自请求消息的子功能参数的比特6-0的回声。

9.5.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表58中。如果错误情况适用于服务器，则应使用列出的否定响应。

表58 - 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 在服务器处于关键正常模式活动时使用，因此无法禁用/启用请求的通信类型。	CNC
0x31	**requestOutOfRange** 如果服务器在通信类型或nodeIdentificationNumber参数中检测到错误，则应使用此响应代码。	ROOR

9.5.5 消息流示例CommunicationControl (禁用传输网络管理消息)

通过将suppressPosRspMsgIndicationBit (子函数参数的第7位) 设置为“FALSE”('0')，客户端请求获得响应消息。

表59定义了CommunicationControl请求消息流示例。

表59 - CommunicationControl请求消息流示例

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	CommunicationControl请求SID	0x28	CC
#2	controlType = enableRxAndDisableTx, suppressPosRspMsgIndicationBit = FALSE	0x01	ERXTX
#3	communicationType =网络管理	0x02	NWMCP

表60定义了CommunicationControl肯定响应消息流程示例。

表60 - CommunicationControl正面响应消息流程示例

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	CommunicationControl响应SID	0x68	CCPR
#2	ControlType	0x01	CTRLTP

9.5.6 消息流示例CommunicationControl（将远程网络切换到地址为0x000A的节点连接到的仅诊断调度模式）

通过将suppressPosRspMsgIndicationBit（子函数参数的第7位）设置为“FALSE”（'0'），客户端请求获得响应消息。

表61定义了CommunicationControl请求消息流程示例。

表61 – CommunicationControl请求消息流示例

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	CommunicationControl请求SID	0x28	CC
#2	controlType = enableRxAndDisableTxWithEnhancedAddressInformation, suppressPosRspMsgIndicationBit = FALSE	0x04	ERXTX
#3	communicationType =普通消息	0x01	NMCP
#4	nodeIdentificationNumber（高字节）	0x00	NIN
#5	nodeIdentificationNumber（低字节）	0x0A	NIN

表62定义了CommunicationControl肯定响应消息流程示例。

表62 – CommunicationControl肯定响应消息流示例

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	CommunicationControl响应SID	0x68	CCPR
#2	controlType = enableRxAndDisableTxWithEnhancedAddressInformation, suppressPosRspMsgIndicationBit = FALSE	0x04	ERXTX

9.5.7 消息流示例CommunicationControl（切换到具有增强地址信息的应用程序调度模式，连接到子网的节点0x000A被寻址）

通过将suppressPosRspMsgIndicationBit（子函数参数的第7位）设置为“FALSE”（'0'），客户端请求获得响应消息。

表63定义了CommunicationControl请求消息流示例。

表63 - CommunicationControl请求消息流示例

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	CommunicationControl请求SID	0x28	CC
#2	controlType = enableRxAndTxWithEnhancedAddressInformation, suppressPosRspMsgIndicationBit = FALSE	0x05	ERXTX
#3	communicationType =普通消息	0x01	NMCP
#4	nodeIdentificationNumber (高字节)	0x00	NIN
#5	nodeIdentificationNumber (低字节)	0x0A	NIN

表64定义了CommunicationControl肯定响应消息流程示例。

表64 - CommunicationControl正面响应消息流示例

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	CommunicationControl响应SID	0x68	CCPR
#2	controlType = enableRxAndTxWithEnhancedAddressInformation, suppressPosRspMsgIndicationBit = FALSE	0x05	ERXTX

9.6 TesterPresent (0x3E) 服务

9.6.1 服务说明

该服务用于向服务器（或多个服务器）指示客户端仍然连接到车辆，并且先前已激活的某些诊断服务和/或通信将保持活动。

该服务用于将一个或多个服务器保存在defaultSession之外的诊断会话中。 这可以通过定期发送TesterPresent请求消息来完成，也可以在没有其他诊断服务的情况下防止服务器自动返回到defaultSession。在ISO14229的实施规范中可以找到在保持单个服务器或多个服务器处于defaultSession以外的诊断会话时使用此服务的详细会话要求。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

9.6.2 请求消息

9.6.2.1 请求消息定义

表65定义了请求消息。

表65 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	TesterPresent请求SID	M	0x3E	TP
#2	子函数= [zeroSubFunction]	M	0x00 / 0x80	LEV_ZSUBF

9.6.2.2 请求消息子函数参数$ Level (LEV_) 定义

表66 指定 该 子功能 参数 值 定义 对于 这个 服务 (suppressPosRspMsgIndicationBit (bit 7) 未显示) 。

表66 – 请求消息子功能参数定义

位6 – 0	描述	Cvt	助记符
0x00	**zeroSubFunction** 此参数值用于指示此服务不支持suppressPosRspMsgIndicationBit旁边的子函数值。	M	ZSUBF
0x01 – 0x7F	**ISOSAEReserved** 这个值的范围是由这个文件保留的。	M	ISOSAERESRVD

9.6.2.3 请求消息数据参数定义

该服务不支持请求消息中的数据参数。

9.6.3 积极的回应消息

9.6.3.1 积极响应消息的定义

表67定义了肯定响应消息。

表67 – 肯定回应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	TesterPresent响应SID	M	0x7E	TPPR
#2	子函数= [zeroSubFunction]	M	0x00	LEV_ZSUBF

9.6.3.2 肯定回复消息数据参数定义

表68定义了肯定响应消息的数据参数。

表68 – 响应消息数据参数定义

定义
zeroSubFunction 该参数是来自请求消息的子功能参数的比特6-0的回声。

9.6.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表69中。如果错误情况适用于服务器，则应使用列出的否定响应。

表69 - 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF

9.6.5 消息流示例 (s) TesterPresent

9.6.5.1 示例#1 - TesterPresent (suppressPosRspMsgIndicationBit = FALSE)

表70定义了TesterPresent请求消息流程示例#1。

表70 - TesterPresent请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	TesterPresent请求SID	0x3E	TP
#2	zeroSubFunction, suppressPosRspMsgIndicationBit = FALSE	0x00	ZSUBF

表71定义了TesterPresent肯定响应消息流程示例#1。

表71 - TesterPresent肯定响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	TesterPresent响应SID	0x7E	TPPR
#2	zeroSubFunction, suppressPosRspMsgIndicationBit = FALSE	0x00	ZSUBF

9.6.5.2 示例#2 - TesterPresent (suppressPosRspMsgIndicationBit = TRUE)

表72定义了TesterPresent请求消息流程示例#2。

表72 – TesterPresent请求消息流程示例#2

消息方向		客户端　服务器	
消息类型		请求	
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	TesterPresent请求SID	0x3E	TP
#2	zeroSubFunction, suppressPosRspMsgIndicationBit = TRUE	0x80	ZSUBF

没有响应由服务器发送。

9.7　AccessTimingParameter（0x83）服务

9.7.1　服务说明

AccessTimingParameter服务用于在通信链路处于活动状态期间读取和更改通信链路的默认时间参数。

此服务的使用很复杂，取决于服务器的能力和数据链路拓扑。 每个诊断会话只支持一个扩展时序参数集。 由于服务器支持不同的扩展时序参数集，建议仅将此服务用于物理寻址。

建议使用以下一系列服务：

— DiagnosticSessionControl（diagnosticSessionType）服务；

— AccessTimingParameter（readExtendedTimingParameterSet）服务；

— AccessTimingParameter（setTimingParametersToGivenValues）服务；

对于需要服务器发送响应的情况，客户端和服务器应在服务器发送AccessTimingParameter肯定响应消息后激活新的计时参数设置。 如果不允许响应消息，则客户端和服务器应在发送/接收请求消息之后激活新的定时参数。

成功切换到另一个或相同的诊断会话后（例如通过DiagnosticSessionControl，ECUReset服务或会话计时超时），服务器和客户端应将其计时参数重置为默认值。

AccessTimingParameter服务为访问服务器时间参数提供了四种不同的模式：

— readExtendedTimingParameterSet;

— setTimingParametersToDefaultValues;

— readCurrentlyActiveTimingParameters;

— setTimingParametersToGivenValues;

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

9.7.2 请求消息

9.7.2.1 请求消息定义

表73定义了请求消息。

表73 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	AccessTimingParameter请求SID	M	0x83	ATP
#2	子功能=[timingParameterAccessType]	M	0x00 – 0xFF	LEV_TPAT_
#3 : #n	TimingParameterRequestRecord [字节#1 : 字节#m]	C : C	0x00 – 0xFF : 0x00 – 0xFF	TPREQR_ B1 : Bm
C: TimingParameterRequestRecord仅在timingParameterAccessType = setTimingParametersToGivenValues时出现。TimingParameterRequestRecord的结构和内容是数据链路层依赖，因此在ISO 14229的实现规范中定义。				

9.7.2.2 请求消息子函数参数$ Level (LEV_) 定义

AccessTimingParameter服务使用子函数参数timingParameterAccessType来选择服务器的特定行为。 表74中详细说明了可能的timingParameterIdentifiers的说明和用法。指定了以下子函数值(suppressPosRspMsgIndicationBit (bit 7) 未显示):

表74 – 请求消息子功能参数定义

位6 – 0	描述	Cvt	助记符
0x00	**ISOSAEReserved** 该值由本文档保留。	M	ISOSAERESRVD
0x01	**readExtendedTimingParameterSet** 在收到带有timingParameterAccessType = readExtendedTimingParameterSet的AccessTimingParameter指示原语时，服务器应读取扩展时序参数集，即服务器能够支持的值。 如果对定时参数集的读访问成功，则服务器应发送带有肯定响应参数的AccessTimingParameter响应原语。 如果对定时参数集的读取访问不成功，则服务器应发送一个否定响应消息和适当的否定响应代码。 该子功能用于为当前激活的诊断会话提供一组额外的定时参数。 使用timingParameterAccessType = setTimingParametersToGivenValues 只要这个组(读通过timingParameterAccessType = readExtendedTimingParameterSet)可以设置时序参数。	U	RETPS

表74 – （续）

位6 – 0	描述	Cvt	助记符
0x02	**setTimingParametersToDefaultValues** 在收到带有timingParameterAccessType = setTimingParametersToDefaultValues的AccessTimingParameter indication原语后，服务器应将所有定时参数更改为默认值，并在默认定时参数变为活动状态之前发送带有肯定响应参数的AccessTimingParameter响应原语（如果suppressPosRspMsgIndicationBit设置为'FALSE'，否则定时参数将在成功评估请求消息后变为活动）。 如果出于任何原因定时参数不能被更改为默认值，则服务器应维持当前活动的定时参数并发送具有适当否定响应代码的否定响应消息。 默认定时值的定义取决于所使用的数据链接，并在ISO 14229的实施规范中进行了规定。	U	STPTDV
0x03	**readCurrentlyActiveTimingParameters** 一旦接收到具有timingParameterAccessType = readCurrentlyActiveTimingParameters的AccessTimingParameter indication原语，服务器应读取当前使用的定时参数。 如果对定时参数的读取访问成功，则服务器应发送带有肯定响应参数的AccessTimingParameter响应原语。 如果出于任何原因对当前使用的定时参数的读取访问是不可能的，则服务器将发送具有适当的否定响应代码的否定响应消息。	U	RCATP
0x04	**setTimingParametersToGivenValues** 在接收到具有timingParameterAccessType = setTimingParametersToGivenValues的AccessTimingParameter indication原语后，服务器应当检查在当前条件下是否可以改变定时参数。 如果条件有效，则服务器应执行所有必要的操作以更改定时参数，并在新定时参数值变为活动状态之前发送带有肯定响应参数的AccessTimingParameter响应原语（suppressPosRspMsgIndicationBit设置为'FALSE'，否则定时参数应在成功评估请求消息后生效）。 如果定时参数不能由于任何原因而改变，则服务器应维持当前有效的定时参数并发送具有适当否定响应码的否定响应消息。 无法将服务器的计时参数设置为通过timingParameterAccessType = readExtendedTimingParameterSet读取的最小值和最大值之间的任何一组值。服务器的时间参数只能设置为正确的时间参数读via timingParameterAccessType = readExtendedTimingParameterSet。这样做的请求应该被服务器拒绝。	U	STPTGV
0x05 – 0xFF	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

9.7.2.3 请求消息数据参数定义

表75定义了请求消息的数据参数。

表75 – 请求消息数据参数定义

定义
TimingParameterRequestRecord 此参数记录包含要通过timingParameterAccessType = setTimingParametersToGivenValues在服务器中设置的时间参数值。 该参数记录的内容和结构是数据链路层特定的，可以在ISO 14229的实施规范（例如ISO 14229-3）中找到。

9.7.3 积极的回应消息

9.7.3.1 积极响应消息的定义

表76定义了肯定响应消息。

表76 – 肯定回应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	AccessTimingParameter响应SID	M	0xC3	ATPPR
#2	timingParameterAccessType	M	0x00 – 0x7F	TPAT_
#3 : #n	TimingParameterResponseRecord [字节#1 : 字节#m]	C : C	0x00 – 0xFF : 0x00 – 0xFF	TPRSPR_ B1 : Bm
C: 只有在timingParameterAccessType = readExtendedTimingParameterSet或readCurrentlyActiveTimingParameters时才会出现TimingParameterResponseRecord。 TimingParameterResponseRecord的结构和内容是数据链路层依赖，因此在ISO 14229的实现规范中定义。				

9.7.3.2 肯定回复消息数据参数定义

表77定义了肯定响应消息的数据参数。

表77 – 响应消息数据参数定义

定义
timingParameterAccessType 该参数是来自请求消息的子功能参数的比特6-0的回声。
TimingParameterResponseRecord 此参数记录包含通过timingParameterAccessType = readExtendedTimingParameterSet或readCurrentlyActiveTimingParameters从服务器读取的计时参数值。 该参数记录的内容和结构是特定于数据链路层的，可以在ISO 14229的实施规范中找到。

9.7.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 表78列出了每个响应代码出现的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表78 – 支持的否定响应代码

NRC	描述	助记符
0x12	子functionNotSupported 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 消息的长度或格式错误。	IMLOIF
0x22	**conditionsNotCorrect** 如果未满足请求AccessTimingParameter的条件，则应返回此NRC。	CNC
0x31	**requestOutOfRange** 如果TimingParameterRequestRecord包含无效的定时参数值，则应发送该NRC。	ROOR

9.7.5 消息流示例AccessTimingParameter

9.7.5.1 示例#1 – 将时序参数设置为默认值

此消息流显示如何在服务器中设置默认计时参数。 通过将suppressPosRspMsgIndicationBit (子函数参数的第7位) 设置为 “FALSE” ('0') , 客户端请求获得响应消息。

表79定义了AccessTimingParameter请求消息流程示例#1。

表79 – AccessTimingParameter请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	AccessTimingParameter请求SID	0x83	ATP
#2	timingParameterAccessType = setTimingParametersToDefaultValues; suppressPosRspMsgIndicationBit = FALSE	0x02	TPAT_STPTDV

表80定义了AccessTimingParameter肯定响应消息流程示例#1。

表80-AccessTimingParameter肯定响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	AccessTimingParameter响应SID	0xC3	ATPPR
#2	timingParameterAccessType = setTimingParametersToDefaultValues	0x02	TPAT_STPTDV

9.8 SecuredDataTransmission (0x84) 服务

9.8.1 服务说明

9.8.1.1 目的

此服务的目的是传输受到第三方攻击保护的数据 – 这可能会危及数据安全。

如果客户打算在安全模式下使用本文档中定义的诊断服务，则SecuredDataTransmission服务适用。 它也可以用于在客户端和服务器之间的安全模式下传输符合其他应用协议的外部数据。 在这种情况下的安全模式意味着传输的数据受到加密方法的保护。

9.8.1.2 安全子层

图8说明了安全子层。 为了在安全模式下执行诊断服务，必须在服务器和客户端应用程序中添加安全子层。

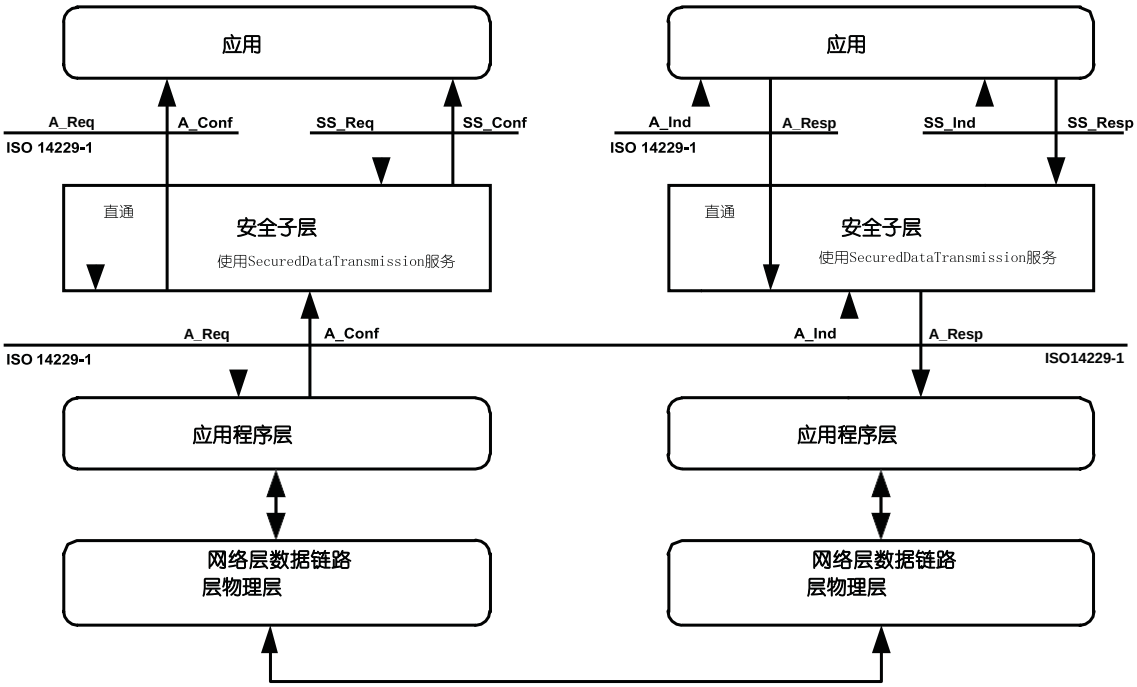


图8 – 安全子层的实现

有两种方法可以在客户端和服务器之间执行诊断服务数据传输:

— 不安全的数据传输模式

该应用程序使用本文档中定义的诊断服务和应用程序层服务原语在客户端和服务器之间交换数据。 安全子层在客户端和服务器的“应用程序”和“应用程序层”之间执行数据的“传递”。

— 安全的数据传输模式

应用程序使用诊断服务或外部服务以及安全子层服务基元在客户端和服务器之间交换数据。 安全子层使用SecuredDataTransmission服务发送/接收安全数据。 安全链接必须是点对点通信。 因此只允许物理寻址，这意味着只涉及一台服务器。

安全子层与应用程序的接口符合ISO / OSI模型惯例，因此提供以下四个安全子层（SS_）服务原语

— SS_SecuredMode.req: 安全子层请求

— SS_SecuredMode.ind: 安全子层指示

— SS_SecuredMode.resp: 安全子层响应

— SS_SecuredMode.conf: 安全子层确认

ISO 14229的这一部分定义了确认和未确认的服务。 在安全模式下，只允许确认的服务（suppressPosRspMsgIndicationBit = FALSE）。 根据这一要求，下列服务不允许在安全模式下执行:

— ResponseOnEvent (0x86) ;

— ReadDataByPeriodicIdentifier (0x2A) ;

— TesterPresent (0x3E) ;

已确认的服务（suppressPosRspMsgIndicationBit = FALSE）使用四个应用层服务原语请求，指示，响应和确认。 在安全模式下执行已确认的诊断服务时，这些映射到四个安全子层服务原语上，反之亦然。

当以安全模式执行诊断服务时，安全子层的任务是加密由“应用程序”提供的数据，以解密由“应用层”提供的数据并添加，检查和移除安全特定的数据元素。 安全子层使用应用层的SecuredDataTransmission (0x84) 服务根据外部协议（请求和响应）发送和接收整个诊断消息或消息，这些消息或消息应在安全模式下交换。

安全子层为安全执行诊断服务提供服务“SecuredServiceExecution”给应用程序。

“SecuredServiceExecution”服务的安全子层请求和指示原语根据以下一般格式指定：

```
SS_SecuredMode.request    (
                          SA,
                          TA,
                          TA_type
                          [, RA]
                          [, 参数1, ...]
                          )

SS_SecuredMode.indication (
                          SA,
                          TA,
                          TA_type
                          [, RA]
                          [, 参数1, ...]
                          )
```

SecuredServiceExecution服务的安全子层层响应和确认原语根据以下通用格式指定：

```
SS_SecuredMode.response   (
                          SA,
                          TA,
                          TA_type
                          [, RA, ]
                          结果
                          [, 参数1, ...]
                          )

SS_SecuredMode.confirm    (
                          SA,
                          TA,
                          TA_type,
                          [RA, ]
                          结果
                          [, 参数1, ...]
                          )
```

安全子层服务原语中所示的寻址信息直接映射到应用层的寻址信息上，反之亦然。

9.8.1.3 安全子层访问

访问用于安全服务执行的安全子层的概念与本文档中描述的应用层接口类似。 安全子层使用应用层服务原语。

以下描述了在安全模式下执行已确认的诊断服务:

— 客户端应用程序使用安全子层SecuredServiceExecution服务请求以安全模式执行诊断服务。 安全子层执行所需的动作以建立与服务器的链接，添加特定的安全相关参数，如果需要，加密要在安全模式下执行的诊断服务的服务数据，并使用应用层SecuredDataTransmission服务请求将安全数据传输到服务器。

— 服务器接收由服务器的安全子层处理的应用层SecuredDataTransmission服务指示。 服务器的安全子层检查安全特定参数解密加密数据，并通过安全子层SecuredServiceExecution服务指示将要以安全模式执行的服务的数据呈现给应用程序。 应用程序执行服务并使用安全子层SecuredServiceExecution服务响应以安全模式响应服务。 服务器的安全子层添加特定的安全相关参数，根据需要加密响应消息数据，并使用应用层SecuredDataTransmission服务响应将响应数据发送给客户端。

— 客户端接收应用层SecuredDataTransmission服务确认原语，该原语由客户端的安全子层处理。 客户端的安全子层检查安全特定参数，解密加密的响应数据并通过安全子层SecuredServiceExecution确认将数据呈现给应用程序。

图9图示了在安全模式下执行确认的诊断服务时安全子层，应用层和应用程序的交互。

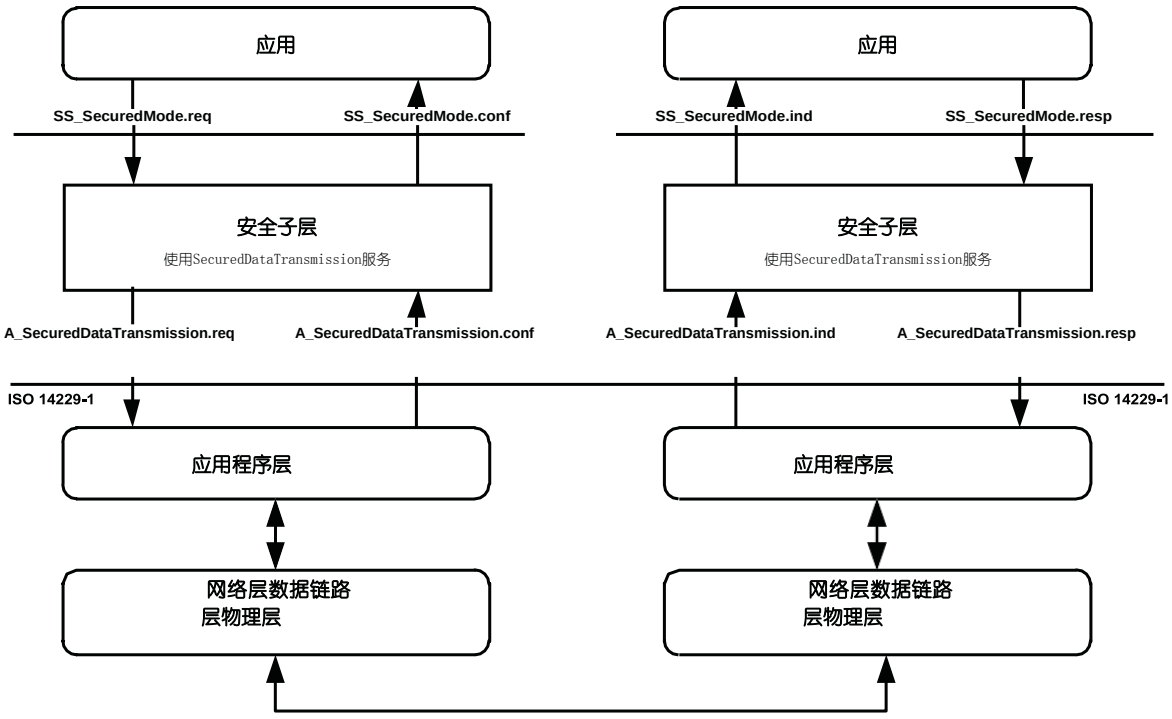


图9 – 安全子层，应用层和应用程序交互

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

9.8.2 请求消息

9.8.2.1 请求消息定义

安全子层生成应用层SecuredDataTransmission请求消息参数。

表81定义了请求消息。

表81 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	SecuredDataTransmission请求SID	M	0x84	SDT
#2 : #n	securityDataRequestRecord[] = [securityDataParameter#1 : securityDataParameter#m]	M : M	0x00 – 0xFF : 0x00 – 0xFF	SECDRQR_ SDP_ : SDP_

9.8.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

9.8.2.3 请求消息数据参数定义

表82定义了请求消息的数据参数。

表82 – 请求消息数据参数定义

定义
securityDataRequestRecord 此参数包含由安全子层处理的数据。

9.8.3 积极的回应消息

9.8.3.1 积极响应消息的定义

表83定义了肯定响应消息。

表83 – 肯定回应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
1	SecuredDataTransmission响应SID	M	0xC4	SDTPR
2 : n	securityDataResponseRecord[] = [securityDataParameter#1 : securityDataParameter#m]	M : M	0x00 – 0xFF : 0x00 – 0xFF	SECDRQR_ SDP_ : SDP_

9.8.3.2 肯定回复消息数据参数定义

表84定义了肯定响应消息的数据参数：

表84 – 响应消息数据参数定义

定义
securityDataResponseRecord 此参数包含由安全子层处理的数据。

9.8.4支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表85中。响应代码总是在没有加密的情况下发送，即使根据请求A_PDU中的configurationProfile响应A_PDU必须加密。 如果错误情景适用于服务器，则应使用列出的否定响应。

表85 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果请求A_PDU的长度不正确，服务器应使用此响应码。	IMLOIF
0x38 – 0x4F	**reservedByExtendedDataLinkSecurityDocument** 扩展数据链接安全性保留了该值的范围。	RBEDLSD

注意上面列出的响应代码适用于SecuredDataTransmission (0x84) 服务。 如果在安全模式下执行的诊断服务需要否定响应，则该否定响应通过SecuredDataTransmission积极响应消息以安全模式发送到客户端。

9.9 ControlDTCSetting (0x85) 服务

9.9.1 服务说明

ControlDTCSetting服务应由客户端用来停止或恢复服务器中DTC状态位的更新。 在ReadDTCInformation的某些子函数的正响应的statusOfDTC参数中报告DTC状态位 (有关位的定义，请参阅D.2) 。

ControlDTCSetting请求消息可用于停止更新个别服务器或一组服务器中的DTC状态位。 如果被寻址的服务器不能停止DTC状态位的更新，则它应以ControlDTCSetting否定响应消息作出响应，指出拒绝的原因。

当服务器接受子函数值为DTCSettingType = off的ControlDTCSetting请求时，服务器应暂停对DTC状态位的任何更新 (即冻结当前值) ，直到功能再次启用。 在一个ControlDTCSetting请求被执行并且子功能被设置为“on”或转换到一个不支持ControlDTCSetting的会话时 (例如会话层超时到defaultSession，ECU复位等) ，DTC状态位信息的更新应该继续) 。 即使请求的DTC设置状态已经激活，如果服务在活动会话中被支持并且所请求的子功能设置为“开”或“关”，服务器仍应发送肯定响应。

如果客户端发送ClearDiagnosticInformation (0x14) 服务，则ControlDTCSetting不应禁止重置服务器的DTC状态位。 各个DTC状态位的行为应根据D.2，图D.1 – 图D.8中的定义执行。

DTC状态位记录相对于代表特定故障状况的数字标识符 (DTC) 的某些信息。 ControlDTCSetting只打开/关闭DTC状态位更新。 ControlDTCSetting服务不是为了导致故障监控被关闭，也不是为了引起故障软件策略

被禁用。 不建议故障软件或故障安全策略与DTC状态位直接关联或耦合（例如，接受的ClearDiagnosticInformation请求不会直接删除任何活动的故障软件）。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

9.9.2 请求消息

9.9.2.1 请求消息定义

表86定义了请求消息。

表86 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ControlDTCSetting请求SID	M	0x85	CDTCS
#2	子函数= [DTCSettingType]	M	0x00 – 0xFF	LEV_DTCSTP_
#3 : #n	DTCSettingControlOptionRecord [] = [参数#1 : 参数#m的	U : U	0x00 – 0xFF : 0x00 – 0xFF	DTCSCOR_ PARA1 : PARAm

9.9.2.2 请求消息子函数参数$ Level (LEV_) 定义

ControlDTCSetting请求消息使用子函数参数DTCSettingType来向服务器指示诊断故障代码状态位更新是停止还是重新开始（suppressPosRspMsgIndicationBit (bit 7) 未在表87中示出）。

表87 – 请求消息子功能参数定义

位6 – 0	描述	Cvt	助记符
0x00	**ISOSAEReserved** 该值由本文档保留。	M	ISOSAERESRVD
0x01	**上** 根据正常的操作条件，服务器应恢复更新诊断故障码状态位	M	上
0x02	**离** 服务器应停止更新诊断故障代码状态位。	M	关闭
0x03 – 0x3F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD
0x40 – 0x5F	**vehicleManufacturerSpecific** 这个值的范围保留给车辆制造商特定用途。	U	VMS
0x60 – 0x7E	**systemSupplierSpecific** 这个值的范围保留给系统供应商特定用途。	U	SSS
0x7F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

9.9.2.3 请求消息数据参数定义

表88定义了请求消息的数据参数。

表88 – 请求消息数据参数定义

定义
DTCSettingControlOptionRecord 当控制DTC状态位的更新（例如，它可以包含要打开或关闭的DTC列表）时，该参数记录是用户可选的，用于向服务器传输数据。

9.9.3 积极的回应消息

9.9.3.1 积极响应消息的定义

表89定义了肯定响应消息。

表89 – 肯定回应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ControlDTCSetting响应SID	M	0xC5	CDTCSPR
#2	DTCSettingType	M	00-7F	DTCSTP

9.9.3.2 肯定回复消息数据参数定义

表90定义了肯定响应消息的数据参数。

表90 – 响应消息数据参数定义

定义
DTCSettingType 该参数是来自请求消息的子功能参数的比特6-0的回声。

9.9.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 表91中记录了每个响应代码发生的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表91 – 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 在服务器处于关键正常模式活动时使用，因此无法执行请求的DTC控制功能。	CNC

表91 – （续）

NRC	描述	助记符
0x31	**requestOutOfRange** 服务器应该使用这个 响应 码, if it 检测 一 错误 in 该 个 DTCSettingControlOptionRecord 。	ROOR

9.9.5 消息流示例ControlDTCSetting

9.9.5.1 示例#1 – ControlDTCSetting (DTCSettingType = off)

请注意，此示例不使用服务的功能将附加数据传输到服务器。 通过将suppressPosRspMsgIndicationBit（子函数参数的位7）设置为“FALSE”（'0'），客户端请求获得响应消息。

表92定义了ControlDTCSetting请求消息流程示例#1。

表92- ControlDTCSetting请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ControlDTCSetting请求SID	0x85	RDTCS
#2	DTCSettingType = off, suppressPosRspMsgIndicationBit = FALSE	0x02	DTCSTP_OFF

表93定义ControlDTCSetting肯定响应消息流程示例#1。

表93- ControlDTCSetting肯定响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ControlDTCSetting响应SID	0xC5	RDTCSPR
#2	DTCSettingType =关闭	0x02	DTCSTP_OFF

9.9.5.2 示例#2 – ControlDTCSetting (DTCSettingType = on)

此示例不使用服务的功能将附加数据传输到服务器。 通过将suppressPosRspMsgIndicationBit（子函数参数的位7）设置为“FALSE”（'0'），客户端请求获得响应消息。

表94定义了ControlDTCSetting请求消息流程示例#2。

表94 - ControlDTCSetting请求消息流程示例#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ControlDTCSetting请求SID	0x85	ENC
#2	DTCSettingType = on, suppressPosRspMsgIndicationBit = FALSE	0x01	DTCSTP_ON

表95定义了ControlDTCSetting肯定响应消息流程示例#2。

表95- ControlDTCSetting肯定响应消息流程示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ControlDTCSetting响应SID	0xC5	RDTCSPR
#2	DTCSettingType = on	0x01	DTCSTP_ON

9.10 ResponseOnEvent（0x86）服务

9.10.1 服务说明

ResponseOnEvent服务请求服务器启动或停止在指定事件上传输响应。

该服务提供了在服务器中发生指定事件的情况下自动执行诊断服务的可能性。 客户端指定事件（包括可选事件参数）以及事件发生时要执行的服务（包括服务参数）。 请参阅图10以简要了解客户端和服务器行为。

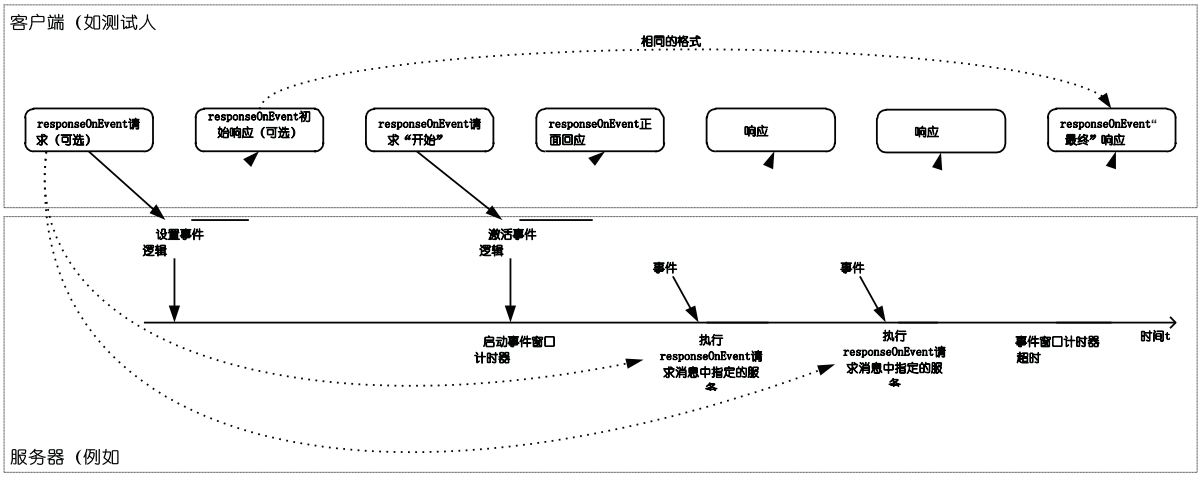


图10 - ResponseOnEvent服务 - 客户端和服务器行为

注意　　上图假定事件窗口定时器配置为在服务器断电之前超时，因此最终的ResponseOnEvent肯定响应消息显示在事件定时窗口的末尾。

服务器应在接收时评估ResponseOnEvent请求消息的子功能和数据内容。 这包括以下子功能和参数：

事件类型，

eventWindowTime，

eventTypeRecord（eventTypeParameter#1-#m）。

如果ResponseOnEvent请求消息中的数据无效，则发送否定响应代码0x31的否定响应。 serviceToRespondToRecord不是此评估的一部分。 当指定事件发生时，将评估serviceToRespondToRecord参数，这会触发serviceToRespondToRecord中包含的服务的执行。 在事件发生时，应执行serviceToRespondToRecord（诊断服务请求消息）。 在条件不正确的情况下，应发送带有适当否定响应码的否定响应消息。 应按其发生顺序发出多个事件的信号。

以下实施细则适用：

a) ResponseOnEvent服务可以在任何会话中设置和激活，包括defaultSession。 TesterPresent服务不一定需要保持ResponseOnEvent服务处于活动状态。

b) 如果在诊断服务正在进行时发生指定的事件，这意味着要么正在接收请求消息，要么正在执行请求，或者正在进行响应消息（这包括负面响应消息处理以及响应代码0x78）被传送（如果suppressPosRspMsgIndicationBit = FALSE），那么包含在serviceToRespondToRecord中的请求消息的执行将被推迟直到完成正在进行的诊断服务。

注意　　在某些情况下，由于ServiceToRespondTo的推迟，ServiceToRespondTo Records中包含的数据可能与导致该事件的数据值不一致。

c) 如果多个事件发生而另一个事件正在进行，多个事件的处理（例如忽略除第一个/最后一个事件或堆积事件以外的所有事件）应为vehicleManufacturerSpecific。

d) 当事件逻辑得到满足并且事件在事件时间窗口内生成时，服务器应执行serviceToRespondToRecord中包含的服务。

e) 一旦ResponseOnEvent服务由ResponseOnEvent请求“启动”启动，服务器就会响应已建立事件逻辑的客户端，并启动ROE事件直到事件窗口时间到期。

f) 移动到任何非默认会话的DiagnosticSessionControl请求应停止ResponseOnEvent服务，而不管是否激活与当前会话或同一非默认会话不同的非默认会话。 在返回到DefaultSession中之前在DefaultSession中处于活动状态的所有ResponseOnEvent服务应重新激活。

g) 多个ResponseOnEvent服务可以与不同的需求（不同的EventTypes，serviceToRespondToRecords等）同时运行以启动和停止诊断服务。 启动和停止子功能应始终控制所有初始化的ResponseOnEvent服务。

h) 如果已设置ResponseOnEvent服务，则应适用以下内容：

1) 如果eventType子功能参数的第6位设置为0（不存储事件），则当服务器断电时，事件应终止服务器在复位或开机后不应继续ResponseOnEvent诊断服务（即ResponseOnEvent服务为终止）。

2) 如果eventType子功能参数的第6位设置为1（存储事件），则应根据服务器重新启动后设置的ResponseOnEvent继续发送serviceToRespondTo-responses。 StoreEvent因此只允许与无限eventWindowTime结合使用。

i) “suppressPosResponseMessageIndicattionBit”=“yes”只应由客户端用于eventType = stopResponseOnEvent，startResponseOnEvent或clearResponseOnEvent。 当检测到指定事件时，服务器应始终返回对事件触发响应的响应。

j) 只有当设置了有限窗口时间并且有限窗口时间已过时，服务器才会返回ResponseOnEvent“final”响应（参见图15）以指示ResponseOnEvent（0x86）服务。 如果在有限窗口时间结束之前ROE已被任何方式停止（例如“stopResponseOnEvent”子功能或会话更改），则不会发送最终响应。

k) 为了避免干扰正常的诊断操作，建议仅实施ResponseOnEvent服务以应用于瞬态事件和条件。 每发生一次事件，服务器都会返回一次响应。 对于持续一段时间的情况，响应服务应在最初发生时只执行一次。 如果eventType被定义为使得serviceToRespondTo-response可能以高频率发生，则必须采取适当措施以防止背靠背serviceToRespondTo-response。 serviceToRespondTo-responses之间的最小间隔时间可以是eventTypeRecord（车辆制造商专用）的一部分。

建议仅使用表96中列出的服务，以便在发生指定事件时执行服务。 （serviceToRespondToRecord请求服务标识符）。

表96 – 与ResponseOnEvent服务一起使用的推荐服务

推荐的服务（ServiceToRespondTo）	请求SID	响应SID
ReadDataByIdentifier	0x22	0x62
ReadDTCInformation	0x19	0x59
RoutineControl	0x31	0x71
InputOutputControlByIdentifier	0x2F	0x6F

出于性能原因（例如，避免错过执行serviceToRespondToRecord请求服务标识符），建议遵守以下建议：

— DID可能包含可测量的数据（例如，避免定义事件逻辑读取常量“校准”标签）

— serviceToRespondToRecord肯定响应的大小可能受到车辆制造商特定值的限制

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

9.10.2 请求消息

9.10.2.1 请求消息定义

表97定义了请求消息。

表97 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ResponseOnEvent请求SID	M	0x86	鱼子
#2	子函数= [eventType]	M	0x00 – 0xFF	LEV_ETP
#3	eventWindowTime	M	0x00 – 0xFF	EWT
#4 : #(m-1)+4	eventTypeRecord[] = [eventTypeParameter 1 : eventTypeParameter m]	C1 : C1	0x00 – 0xFF : 0x00 – 0xFF	ETR_ ETP1 : ETPm
#N- (R-1) -1 #N- (R-1) : #n	serviceToRespondToRecord[] = [serviceId服务参数1 : 服务参数r]	C2 C3 : C3	0x00 – 0xFF 0x00 – 0xFF : 0x00 – 0xFF	STRTR_ SI SP1 : SPr
C1：如果eventType需要指定事件响应的附加参数，则表示存在。 C2：如果子函数参数不等于reportActivatedEvents，stopResponseOnEvent，startResponseOnEvent，ClearResponseOnEvent，则必须存在 C3：如果服务响应的服务请求需要额外的服务参数则呈现				

9.10.2.2 请求消息子函数参数$ Level (LEV_) 定义

9.10.2.2.1 ResponseOnEvent请求消息子函数参数定义

ResponseOnEvent请求消息使用子函数参数eventType来指定要在服务器中配置的事件并控制ResponseOnEvent设置。 表98中给出的每个子函数参数值还指定了适用的eventTypeRecord (suppressPosRspMsgIndicationBit (bit 7)) 的长度，未在下表中显示）。

eventType子函数参数的第6位用于指示事件是否应存储在服务器中的非易失性存储器中，并在服务器的下一次启动时重新激活，或者一旦服务器掉电时它将终止 (storageState参数）。

表98 – eventType子功能位6定义 – storageState

位6值	描述	Cvt	助记符
0x00	**doNotStoreEvent** 该值表示当服务器掉电并且服务器在复位或上电后（即ResponseOnEvent服务终止）不应继续ResponseOnEvent诊断服务时，该事件应终止。	M	的DNSe
0x01	**storeEvent** 该值表示 1) 在默认会话中ROE启动或停止请求时，事件将根据在重置或上电（即，恢复ResponseOnEvent服务）后设置的ResponseOnEvent恢复或停止发送serviceToRespondTo-response。 2) 在任何ROE建立事件逻辑请求中，该请求的事件逻辑必须持久存储，直到明确清除事件逻辑（通过clearResponseOnEvent）或事件逻辑被相同类别的新ROE建立事件逻辑请求覆盖。	U	SE

表99定义了请求消息子功能参数。

表99 – 请求消息子功能参数定义

位5 – 0值	描述	Cvt	ROE子功能类型	助记符
0x00	**stopResponseOnEvent** 该值用于停止服务器发送事件响应。 已设置的事件逻辑未清除，但可以使用startResponseOnEvent子函数参数重新启动。 eventTypeRecord的长度：0字节	U	控制	STPROE
0x01	**onDTCStatusChange** 该值将该事件标识为检测到的与为该事件指定的DTCStatusMask相匹配的新DTC。 eventTypeRecord的长度：1个字节 UI补充提示：UA服务器常驻DTC计数算法应以特定周期率（例如约1秒）计数满足客户机定义的DTCStatusMask的DTC数量。 如果计数与先前执行的计数不同，则客户端应生成导致执行serviceToRespondTo的事件。 最后的计数应作为下次计算的参考。 此eventType需要在请求消息中指定DTCStatusMask（eventTypeParameter#1）。	U	建立	ONDTCS
0x02	**onTimerInterrupt** 该值将事件标识为定时器中断，但定时器及其值不是ResponseOnEvent服务的一部分。 这个eventType需要在请求消息（eventTypeRecord）中指定更多细节。 eventTypeRecord的长度：1个字节	U	建立	OTI

表99 – （续）

位5 – 0值	描述	Cvt	ROE子功能类型	助记符
0x03	**onChangeOfDataIdentifier** 该值将事件标识为由dataIdentifier标识的新内部数据记录。 数据值是车辆制造商特定的。 这个eventType需要在请求消息（eventTypeRecord）中指定更多细节。 eventTypeRecord的长度：2个字节	U	建立	OCODID
0x04	**reportActivatedEvents** 此值用于指示在肯定响应中，所有事件都已报告在服务器中使用ResponseOnEvent服务激活（并且当前处于活动状态）。 eventTypeRecord的长度：0字节	U	控制	RAE
0x05	**startResponseOnEvent** 该值用于指示服务器激活已设置的事件逻辑（包括事件窗口计时器）并开始发送事件响应。 eventTypeRecord的长度：0字节。	M	控制	STRTROE
0x06	**clearResponseOnEvent** 此值用于清除已在服务器中设置的事件逻辑（这也会停止服务器发送事件响应。） eventTypeRecord的长度：0字节。	U	控制	CLRROE
0x07	**onComparisonOfValues** 定义的数据值的变化由dataIdentifier标识的数据值事件标识的特定记录中定义。 通过这个子功能，用户可以在发生从定义的测量值比较中收集到的特定结果时定义一个事件。 将包含在分配给定义的数据标识符的数据记录中的特定测量值与给定的比较值进行比较。 指定的运算符定义了比较的类型。 如果比较结果为肯定，则发生该事件。 eventTypeRecord的长度：10个字节	U	建立	OCOV
0x08 – 0x1F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	-	ISOSAERESRVD
0x20 – 0x2F	**VehicleManufacturerSpecific** 这个值的范围保留给车辆制造商特定用途。	U	建立	VMS
0x30 – 0x3E	**SystemSupplierSpecific** 这个值的范围保留给系统供应商特定用途。	U	建立	SSS
0x3F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	-	ISOSAERESRVD

9.10.2.2.2 详细的请求消息子函数onTimerInterrupt参数说明

通过子功能onTimerInterrupt，服务器可以在定时器可配置的时间段内触发事件。

eventTypeRecord使用以下计时器计划定义计时器值:

— 缓慢的速度，

— 中等价格，

— 快速。

定义与每个计时器计划选项关联的时间速率是制造商的具体任务。 见表100。

表100 – 比较逻辑参数定义

eventTypeRecord	事件将被生成	定时器类型
0x01	每次慢速计时器值过去。	慢速率计时器。 例如，计时器在1秒后过去
0x02	每次中等速率计时器过去。	中等速率计时器。 例如，计时器在300 ms后过去
0x03	每次快速速率计时器过去。	快速率定时器。 例如，计时器在25 ms后过去

9.10.2.2.3 详细请求消息子函数onChangeOfDataIdentifier参数说明

通过子函数onChangeOfDataIdentifier，服务器可以在固定的时间内轮询测量结果并比较内容，因此服务器可能会丢失一些更改并触发少于预期的事件是可以接受的。

eventTypeRecord设置必须为任何更改进行监视的两个字节DID值。

9.10.2.2.4 详细的请求消息子函数onComparisonOfValues参数说明

表101- 表103规定了在指定读取DID之间的比较的serviceToRespondToRecord的情况下具有子函数onComparisonOfValues参数的请求消息的参数。

表101 – onComparisonOfValues参数定义的子函数

数据字节	参数名称	字节值	评论	详细要求
1	服务ID	0x86	请求SID	---
2	事件类型	0x07	子函数 onComparisonOfValues	---
3	eventWindowTime	0x02	InfiniteTimeWindow规范	---
4	eventTypeRecord byte1	0x01	数据标识符（DID）高字节	可以是与serviceToRespondToRecord中使用的DID不同的DID。 可以是动态定义的DID。
5	eventTypeRecord字节2	0x04	DataIdentifier（DID）低字节	---

表101 – （续）

数据字节	参数名称	字节值	评论	详细要求
6	eventTypeRecord字节3	0x01	比较逻辑，见表102	eventTypeRecord字节3设置比较方法的逻辑，
7	eventTypeRecord字节4	0x00	原始参考比较值MSB	eventTypeRecord字节4，5，6，7设置参考比较值。
8	eventTypeRecord字节5	0x00	原始参考比较值	---
9	eventTypeRecord字节6	0x01	原始参考比较值	---
10	eventTypeRecord字节7	0x00	原始参考比较值LSB	---
11	eventTypeRecord字节8	0x0A	滞后值	eventTypeRecord字节8定义了从0％（0x00）到100％（0x64）的百分比值。
12	eventTypeRecord字节9	0xBC	本地化字节1 MSB，见表103	eventTypeRecord字节9,10定义了数据标识符中的值的本地化，参见表103。
13	eventTypeRecord字节10	0x58	本地化字节2 LSB，见表103	---
14	serviceToRespondTo字节1	0x22	SID	serviceToRespondToRecord设置要读取和比较的服务和DID。 在第一个积极的回应信息numberOfIdentifiedEvents字段始终设置为0x00。
15	serviceToRespondTo字节2	0xA1	DID1	---
16	serviceToRespondTo字节3	0x00	DID2	---

表102定义了比较逻辑参数定义。

表102 – 比较逻辑参数定义

比较逻辑ID	事件将在生成时生成
0x01	比较参数<测量值
0x02	比较参数>测量值
0x03	比较参数=测量值
0x04	比较参数<>测量值

表103定义了值16位位域参数定义的定位。

表103 – 值16位位域参数定义的本地化

位字段的位置	描述
15	(MSB)，位= 0表示无符号比较，位= 1与符号比较
14 - 10	位#10 (LSB) – 位#14 (MSB)包含要比较的数据标识符值的长度。 值0x00应用于比较所有4个字节。 所有其他值应以位为单位设置大小。 用5位，长度的最大尺寸是31位。
9 - 0	从哪里提取数据标识符值的位置响应消息上的偏移量。 位#0 (LSB) – 位#9 (MSB)包含起始位编号偏移量。 使用10位，偏移的最大大小为1023位。

9.10.2.3 请求消息数据参数定义

表104定义了请求消息的数据参数。

表104 – 请求消息数据参数定义

定义
eventWindowTime 参数eventWindowTime用于指定事件逻辑在服务器中处于活动状态的窗口。 如果eventWindowTime的参数值设置为0x02，则响应时间是无限的。 在无限事件窗口和storageState等于doNotStoreEvent的情况下，建议通过某个信号关闭事件窗口（例如关闭电源）。 有关指定的eventWindowTimes，请参阅B.2。 不得使用有限事件窗口和storeState等于storeEvent的组合。 注意 如果eventType等于ROE控制子功能，则此参数不适用于服务器评估。
eventTypeRecord 此参数记录包含指定eventType的其他参数。
serviceToRespondToRecord 此参数记录包含每次在eventTypeRecord中定义的指定事件发生时要在服务器中执行的服务的服务参数（服务Id和服务参数）。

9.10.3 积极的回应消息

9.10.3.1 积极响应消息的定义

表105定义了所有子函数的肯定响应消息，但reportActivatedEvents。

表105 – 所有子功能但reportActivatedEvents的肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ResponseOnEvent响应SID	M	0xC6	ROEPR
#2	事件类型	M	0x00 – 0x7F	ETP
#3	numberOfIdentifiedEvents	M	0x00 – 0xFF	NOIE
#4	eventWindowTime	M	0x00 – 0xFF	EWT

表105 – （续）

A_Data字节	参数名称	Cvt	字节值	助记符
#5 : #(m-1)+5		C1 : C1	0x00 – 0xFF : 0x00 – 0xFF	ETR_ ETP1 : ETPm
#N-（R-1）-1 #N-（R-1） : #n		M C2 : C2	0x00 – 0xFF 0x00 – 0xFF : 0x00 – 0xFF	STRTR_ SI SP1 : SPr
C1：如果eventType需要指定事件响应的附加参数，则表示存在。 C2：提供服务的服务请求以响应所需的附加服务参数				

表106定义了子函数= reportActivatedEvents的肯定响应消息。

表106 – 子功能的肯定响应消息定义= reportActivatedEvents

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ResponseOnEvent响应SID	M	0xC6	ROEPR
#2	eventType = reportActivatedEvents	M	0x04	ETP_RAE
#3	numberOfActivatedEvents	M	0x00 – 0xFF	NOIE
#4	eventTypeOfActiveEvent#1	C1	0x00 – 0xFF	EVOAE
#5	eventWindowTime#1	C1	0x00 – 0xFF	EWT
#6 : #(m-1)+6	eventTypeRecord#1 [] = [eventTypeParameter 1 : eventTypeParameter m]	C2 : C2	0x00 – 0xFF : 0x00 – 0xFF	ETR_ ETP1 : ETPm
#对 – （O-1）-1 #对 – （O-1） : #p	serviceToRespondToRecord#1 [] = [serviceId服务参数1 : 服务参数o]	C3 C4 : C4	0x00 – 0xFF 0x00 – 0xFF : 0x00 – 0xFF	STRTR_ SI SP1 : SPo的
:	:	:	:	:
#N-（R-1）-4-（Q-1）	eventTypeOfActiveEvent#k中	C1	0x00 – 0xFF	EVOAE
#N-（R-1）-3-（Q-1）	eventWindowTime#k中	C1	0x00 – 0xFF	EWT
#N-（R-1）-2-（Q-1） : #N-（R-1）-2	eventTypeRecord#k [] = [eventTypeParameter 1 : eventTypeParameter q]	C2 : C2	0x00 – 0xFF : 0x00 – 0xFF	ETR_ ETP1 : ETPm
#N-（R-1）-1 #N-（R-1） : #n	serviceToRespondToRecord#k [] = [serviceId服务参数1 : 服务参数r]	C3 C4 : C4	0x00 – 0xFF 0x00 – 0xFF : 0x00 – 0xFF	STRTR_ SI SP1 : SPr

C1：报告活动事件。

C2：显示活动事件（eventTypeOfActiveEvent）的报告事件类型是否需要指定事件响应的附加参数。

C3：报告活动事件时必须存在。

C4：显示服务响应的服务请求是否需要额外的服务参数。

9.10.3.2 肯定回复消息数据参数定义

表107定义了肯定响应消息的数据参数。

表107 – 响应消息数据参数定义

定义
事件类型 该参数是请求消息的子函数参数的位6 – 0的回显。
eventTypeOfActiveEvent 此参数是为设置活动事件而发出的请求消息的子函数参数的回显。 适用的值是为eventType子函数参数指定的值。
numberOfActivatedEvents 此参数包含客户端请求报告活动事件数时活动事件的数量。 该数字反映了响应消息中报告的事件数量。
numberOfIdentifiedEvents 此参数包含活动事件窗口期间标识事件的数量，并且仅适用于在事件窗口结束时发送的响应消息（在有限事件窗口的情况下）。 对请求消息的初始响应应在此参数中包含一个零（0）。
eventWindowTime 该参数是来自请求消息的eventWindowTime参数的回显。 报告活动事件时，此参数包含事件处于活动状态的剩余时间。
eventTypeRecord 该参数是请求消息中的eventTypeRecord参数的回显。 在报告活动事件时，此参数是为了设置活动事件而发出的请求的eventTypeRecord的回显。
serviceToRespondToRecord 此参数是请求消息中的serviceToRespondToRecord参数的回显。 在报告活动事件时，此参数是为了设置活动事件而发出的请求的serviceToRespondToRecord的回声。

9.10.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表108中。如果错误情况适用于服务器，则应使用列出的否定响应。

表108 – 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果不支持子功能参数，则应发送该NRC	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果请求消息的长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 当服务器处于关键的正常模式活动并因此无法执行所请求的功能时使用。	CNC
0x31	**requestOutOfRange** 服务器应使用此响应代码 如果它在eventTypeRecord参数中检测到错误；如果指定的eventWindowTime无效； 如果请求的DID不被支持； 如果请求有限事件窗口和storeState等于storeEvent的组合	ROOR

9.10.5 消息流示例（s）ResponseOnEvent

9.10.5.1 假设

对于消息流示例，假定eventWindowTime等于0x08定义了一个80秒的事件窗口（eventWindowTime * 10秒）。 通过将suppressPosRspMsgIndicationBit（子函数参数的第7位）设置为“FALSE”（'0'），客户端请求获得响应消息。

注意 eventWindowTime的定义是车辆制造商特定的，除了B.2中规定的某些值。

以下条件适用于显示的消息流示例和流程图：

触发信号：
车辆制造商需要定义一个特定的触发信号，使客户端（外部测试设备，OBD-Unit，诊断主站等）启动ResponseOnEvent请求消息。 这个触发信号可以由一个事件以及一个像心跳时间（应该大于eventWindowTime）的固定时间表来启用。 此外，在用作触发信号的数据链路上可能存在同步消息（例如，同步信号）。

打开活动窗口：
接收到ResponseOnEvent请求消息后，服务器应该避开请求。 如果评估结果是肯定的，服务器应设置事件逻辑并发送ResponseOnEvent服务的初始肯定响应消息。 要激活事件逻辑，客户端必须请求ResponseOnEvent子函数startResponseOnEvent。 在积极响应之后，事件逻辑被激活并且事件窗口计时器正在运行。 车辆制造商应该使用参数eventWindowTime（例如时间窗口，点火开/关窗口）来详细定义事件窗口。 在检测到指定的eventType（EART_）的情况下，服务器必须立即响应ResponseOnEvent请求消息中对应于serviceToRespondToRecord的响应消息。

— **关闭活动窗口:**
建议根据参数eventWindowTime关闭服务器的事件窗口。 执行此操作后，服务器必须停止发送事件驱动的诊断响应消息。 通过发送包含参数stopResponseOnEvent的ResponseOnEvent (ROE_) 请求消息或通过关闭电源也可以达到相同的效果。

9.10.5.2 示例#1 - ResponseOnEvent (有限事件窗口)

表109定义了ResponseOnEvent请求消息流程示例#1的设置。

表109 - ResponseOnEvent请求消息流程示例#1的设置

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ResponseOnEvent请求SID	0x86	鱼子
#2	eventTypeRecord [eventType] = onDTCStatusChange, storageState = doNotStoreEvent suppressPosRspMsgIndicationBit = FALSE	0x01	ET_ODTCSC
#3	eventWindowTime = 80秒	0x08	EWT
#4	eventTypeRecord [eventTypeParameter] = testFailed状态	0x01	ETP1
#5		0x19	RDTCI
#6		0x01	RNDTC
#7		0x01	DTCSM

表110定义了ResponseOnEvent初始肯定响应消息流程示例#1。

表110 - ResponseOnEvent初始肯定响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ResponseOnEvent响应SID	0xC6	ROEPR
#2	eventType = onDTCStatusChange	0x01	ET_ODTCSC
#3	numberOfIdentifiedEvents = 0	0x00	NOIE
#4	eventWindowTime = 80秒	0x08	EWT
#5	eventTypeRecord [eventTypeParameter] = testFailed状态	0x01	ETP1
#6		0x19	RDTCI
#7		0x01	RNDTC
#8		0x01	DTCSM

事件逻辑设置：现在它必须被激活。

表111定义了ResponseOnEvent请求消息流程示例#1的开始。

表111 - ResponseOnEvent请求消息流程示例#1的开始

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent请求SID	0x86	鱼子
#2	eventTypeRecord [eventType] = startResponseOnEvent, storageState = doNotStoreEvent, suppressPosRspMsgIndicationBit = FALSE	0x05	ET_STRTROE
#3	eventWindowTime（不会被评估）	0x08	EWT

表112定义了ResponseOnEvent肯定响应消息流程示例#1。

表112-ResponseOnEvent肯定响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent响应SID	0xC6	ROEPR
#2	eventType = onDTCStatusChange	0x01	ET_ODTCSC
#3	numberOfIdentifiedEvents = 0	0x00	NOIE
#4	eventWindowTime	0x08	EWT

如果发生指定的事件，则服务器根据指定的serviceToRespondToRecord发送响应消息。

表113定义了ReadDTCInformation肯定响应消息流程示例#1。

表113-ReadDTCInformation肯定响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCI
#2	DTCStatusAvailibilityMask	为0xFF	DTCSAM
#3	DTCCount [DTCCountHighByte] = 0	0x00	DTCCNT_HB
#4	DTCCount [DTCCountLowByte] = 4	0x04	DTCCNT_LB

客户端在活动事件窗口期间请求报告服务器中当前活动事件的消息流程如下所示。

表114定义了活动事件消息流程示例#1的ResponseOnEvent请求数目。

表114 - ResponseOnEvent请求活动事件消息流示例#1的数量

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent请求SID	0x86	鱼子
#2	eventTypeRecord [eventType] = reportActivatedEvents, storageState = doNotStoreEvent, suppressPosRspMsgIndicationBit = FALSE	0x04	ET_RAE
#3	eventWindowTime (不会被评估)	0x08	EWT

表115定义了ResponseOnEvent reportActivatedEvents肯定响应消息流程示例#1。

表115 - ResponseOnEvent reportActivatedEvents正面响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent响应SID	0xC6	ROEPR
#2	eventType = reportActivatedEvents	0x04	ET_RAE
#3	numberOfActivatedEvents = 1	0x01	NOAE
#4	eventTypeOfActiveEvent = onDTCStatusChange	0x01	ET_ODTCSC
#5	eventWindowTime = 80秒	0x08	EWT
#6	eventTypeRecord [eventTypeParameter] = testFailed状态	0x01	ETP1
#7		0x19	RDTCI
#8		0x01	RNDTC
#9		0x01	DTCSM

如果指定的事件窗口时间已过，服务器应发送最终肯定响应。 表116定义ResponseOnEvent最终肯定响应消息流程示例#1。

表116 – ResponseOnEvent最终肯定响应消息流程示例#1

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent响应SID	0xC6	ROEPR
#2	eventType = onDTCStatusChange	0x01	ET_ODTCSC
#3	numberOfIdentifiedEvents = 1	0x01	NOIE
#4	eventWindowTime = 80秒	0x08	EWT
#5	eventTypeRecord [eventTypeParameter] = testFailed状态	0x01	ETP1
#6		0x19	RDTCI
#7		0x01	RNDTC
#8		0x01	DTCSM

9.10.5.2.1　示例#1 – 流程图

以下流程图显示了两种不同类型的服务器行为：

在有限事件窗口内不会发生事件。 在这种情况下，服务器必须在事件窗口结束时发送ResponseOnEvent的响应。

有限事件窗口内的多个事件（#1至#n）。 serviceToRespondTo的每个肯定响应与标识的事件（#1 ..#n）相关，并且应具有相同的服务标识符（SId），但可能有不同的内容。 在event_Window结束时，服务器应发送responseOnEvent服务的肯定响应消息，该消息指示numberOfIdentifiedEvents。

图11描绘了有限事件窗口 – 在活动事件窗口期间没有事件。

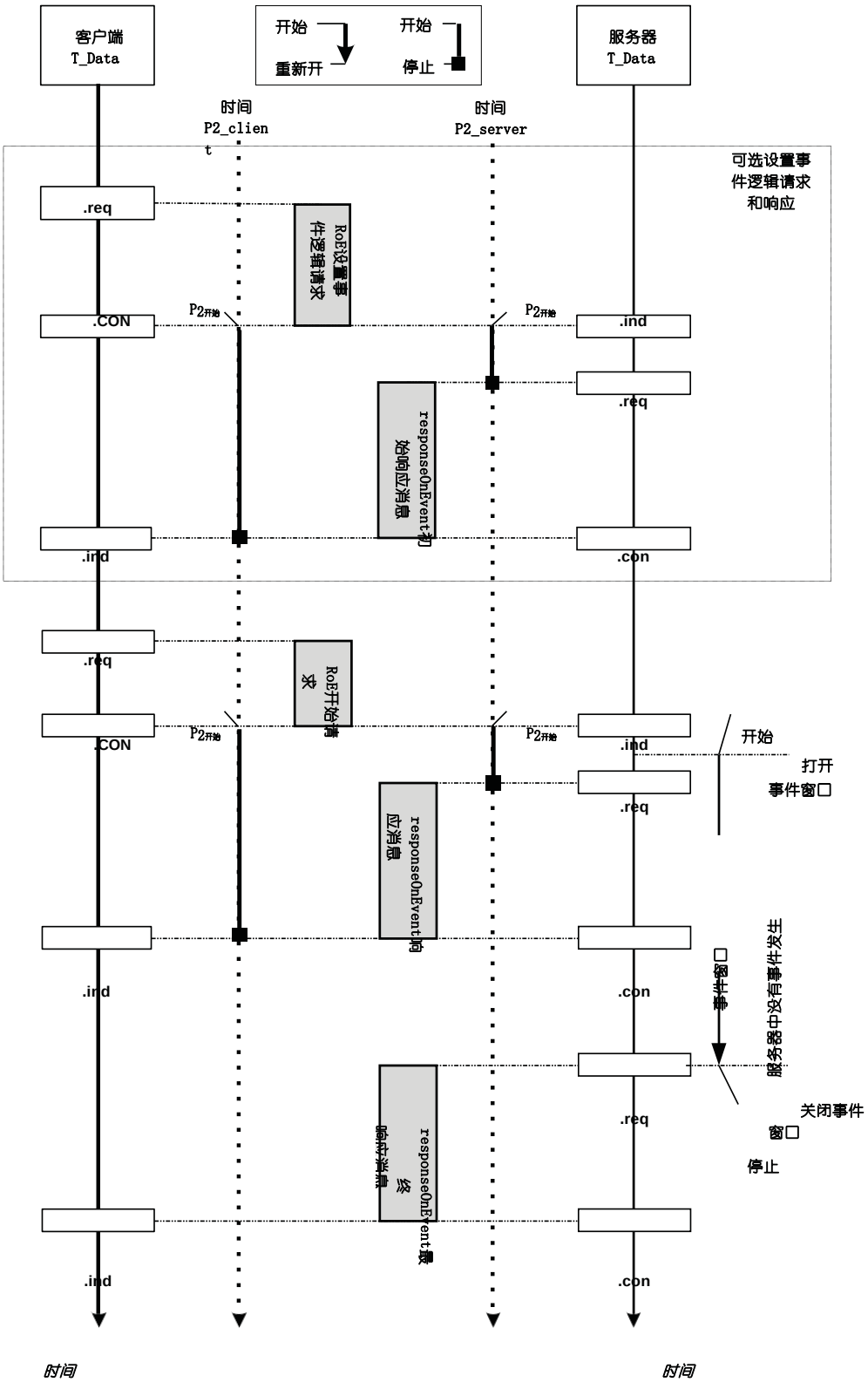


图11 – 有限事件窗口 – 活动事件窗口期间无事件

图12描绘了有限事件窗口 – 活动事件窗口期间的多个事件。

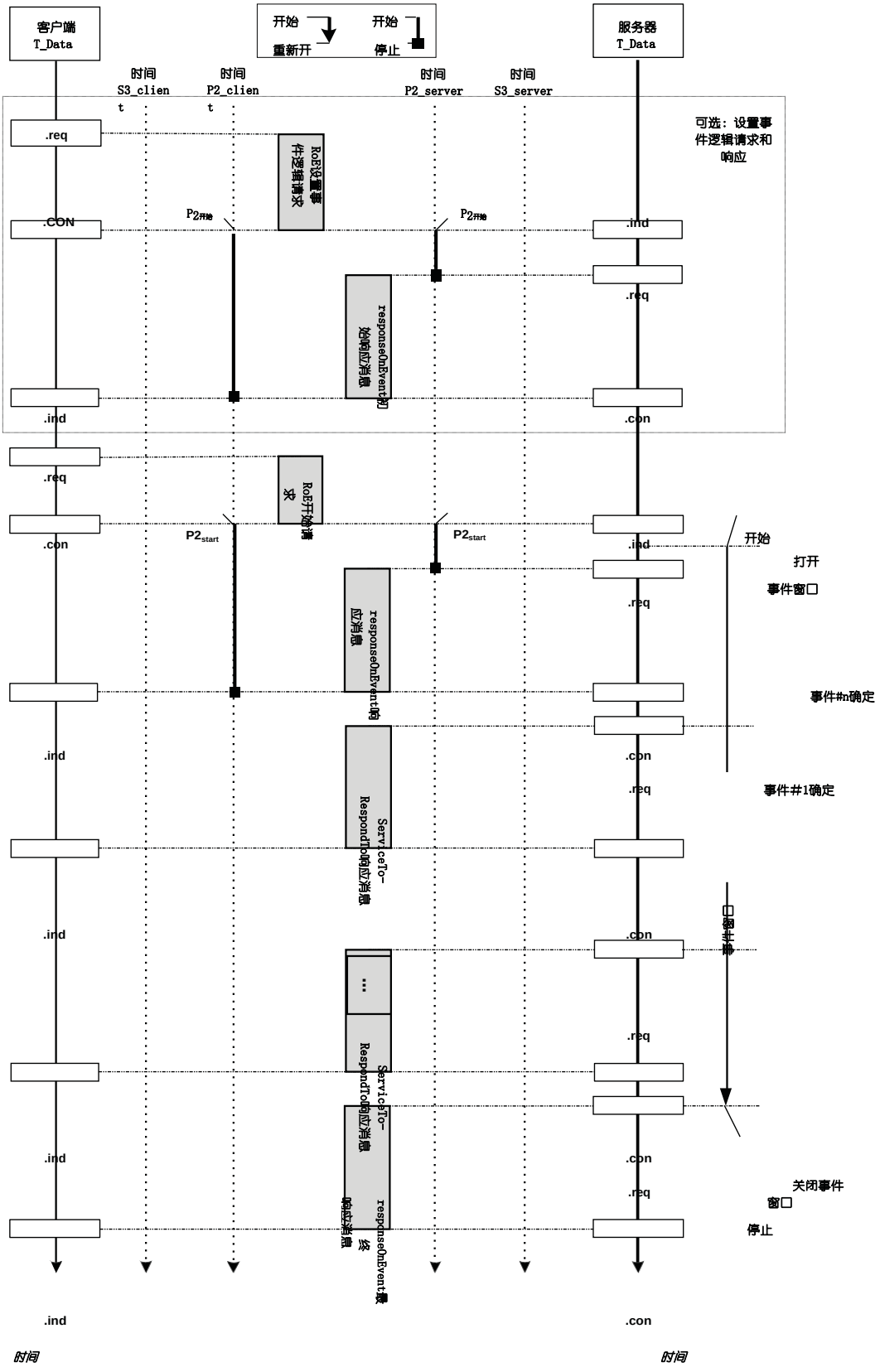


图12 – 有限事件窗口 – 活动事件窗口期间的多个事件

9.10.5.3 示例#2 - ResponseOnEvent（无限事件窗口）

表117定义了ResponseOnEvent请求消息流程示例#2。

表117 - ResponseOnEvent请求消息流程示例#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent请求SID	0x86	鱼子
#2	eventTypeRecord [eventType] = onDTCStatusChange, storageState = doNotStoreEvent, suppressPosRspMsgIndicationBit = FALSE	0x01	ET_ODTCSC
#3	eventWindowTime =无限	0x02	EWT
#4	eventTypeRecord [eventTypeParameter] = testFailed状态	0x01	ETP1
#5		0x19	RDTCI
#6		0x01	RNDTC
#7		0x01	DTCSM

表118定义了ResponseOnEvent初始肯定响应消息流程示例#2。

表118 - ResponseOnEvent初始肯定响应消息流程示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent响应SID	0xC6	ROEPR
#2	eventType = onDTCStatusChange	0x01	ET_ODTCSC
#3	numberOfIdentifiedEvents = 0	0x00	NOIE
#4	eventWindowTime =无限	0x02	EWT
#5	eventTypeRecord [eventTypeParameter] = testFailed状态	0x01	ETP1
#6		0x19	RDTCI
#7		0x01	RNDTC
#8		0x01	DTCSM

事件逻辑设置；现在它必须被激活。

表119定义了ResponseOnEvent请求消息流程示例#2的开始。

表119 - ResponseOnEvent请求消息流程示例#2的开始

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent请求SID	0x86	鱼子
#2	eventTypeRecord [eventType] = startResponseOnEvent, storageState = doNotStoreEvent, suppressPosRspMsgIndicationBit = FALSE	0x05	ET_STRTROE
#3	eventWindowTime (不会被评估)	0x02	EWT

表120定义了ResponseOnEvent肯定响应消息流程示例#2。

表120 - ResponseOnEvent肯定响应消息流程示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent响应SID	0xC6	ROEPR
#2	eventType =onDTCStatusChange	0x05	ET_ODTCSC
#3	numberOfIdentifiedEvents = 0	0x00	NOIE
#4	eventWindowTime	0x02	EWT

如果发生指定的事件，则服务器根据指定的serviceToRespondToRecord发送响应消息。

表121定义了ReadDTCInformation肯定响应消息流程示例#2。

表121 - ReadDTCInformation肯定响应消息流程示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCI
#2	DTCStatusAvailibilityMask	0xXX	DTCSAM
#3	DTCCount [DTCCountHighByte] = 0	0x00	DTCCNT_HB
#4	DTCCount [DTCCountLowByte] = 4	0x04	DTCCNT_LB

9.10.5.3.1 示例#2 - 流程图

以下流程图显示了两种不同类型的服务器行为：

无限事件窗口内没有事件发生。

无限事件窗口中的多个事件（#1至#n）。 serviceToRespondTo的每个肯定响应与标识的事件（#1 ..#n）相关，并且应该具有相同的服务标识符（SI），但可能具有不同的内容。

图13描述了无限事件窗口 – 活动事件窗口期间没有事件。

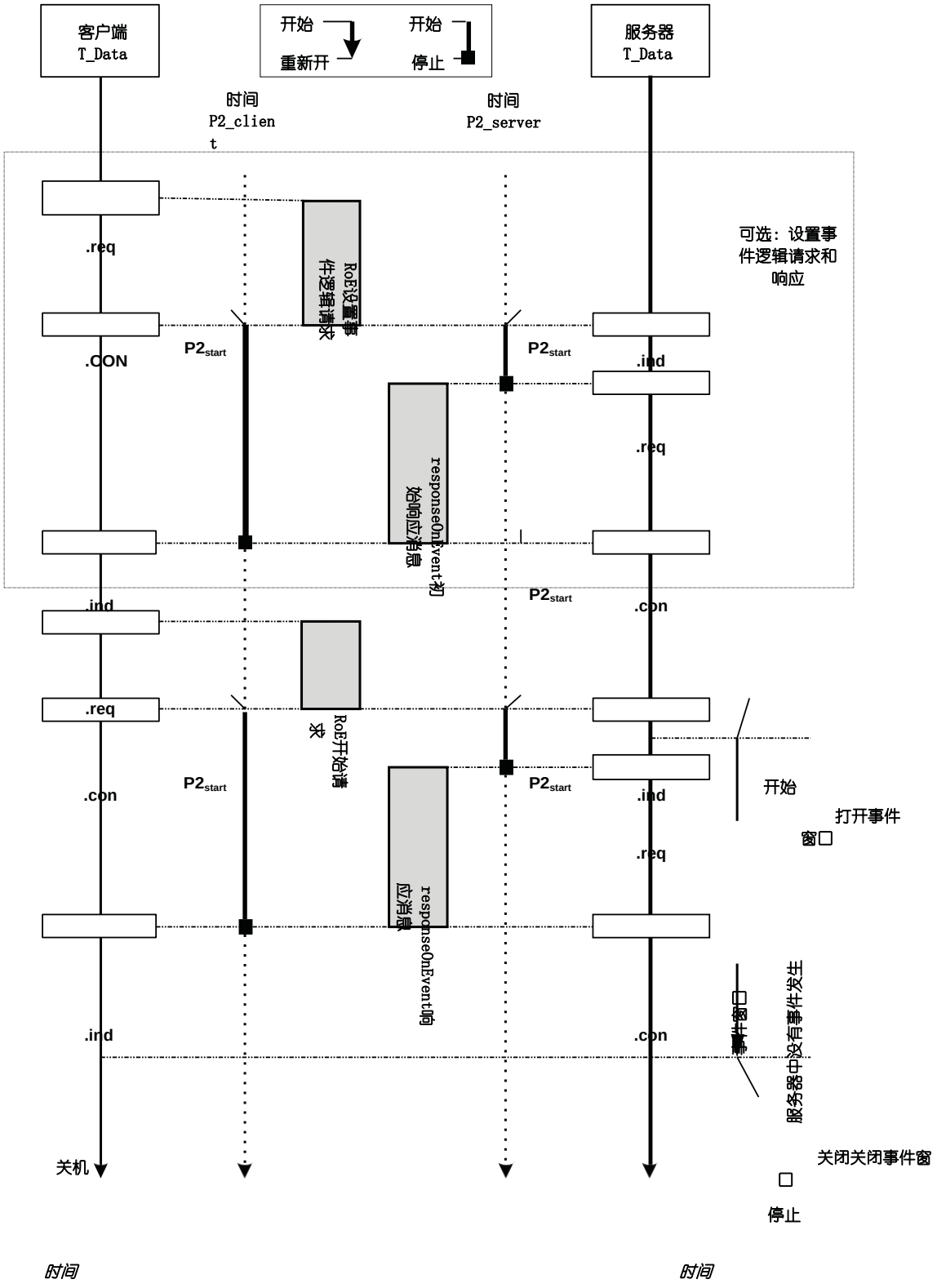


图13 – 无限事件窗口 – 活动事件窗口期间无事件

图14描绘了无限事件窗口 – 活动事件窗口期间的多个事件。

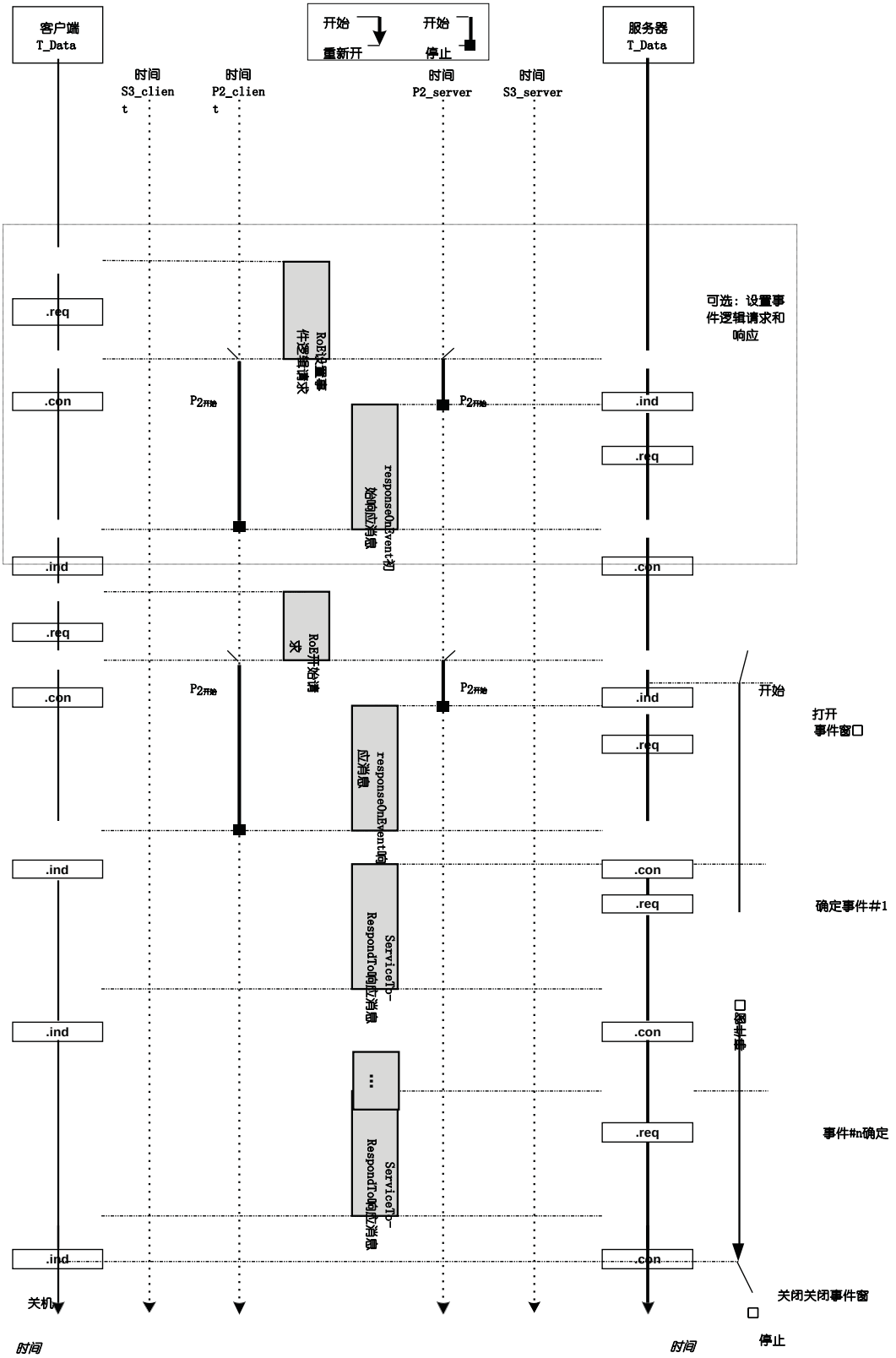


图14 – 无限事件窗口 – 活动事件窗口期间的多个事件

9.10.5.4 示例#3 - ResponseOnEvent（无限事件窗口） - 子函数参数 “onComparisonOfValues”

这个例子只说明了假设在例子＃1和例子＃2中描述的ROE服务的通信行为没有改变的情况下子功能参数“onComparisonOfValues”的使用。 因此这个例子没有描述完整的消息流。 相反，只会显示并解释事件窗口设置请求消息和发生事件的肯定响应消息。 在上面的例子中已经描述了开始和停止请求消息以及不同的响应消息。以下条件适用：

— 服务0x22 - 选择ReadDataByIdentifier作为serviceToRespondTo，

— dataIdentifier 0x0104包含要在数据字节＃11和11处进行比较的测量值＃12（该测量值也可以通过使用服务0x22读取），

— 如果测量值（MV）高于所谓的比较参数（CP），则会发生事件，因此操作员值（请参见下面的说明）选择为0x01 - “MV> CP”，

— 当滞后值选择为0x0A - 10％时，

— 作为事件窗口时间的值0x02 - “无限”被选中，

— 作为storageState（eventType子函数的第6位），值1的二进制 - “storeEvent”被选中，

— 在任何情况下都会要求回复。 示例

的定义：

— 字节＃4和5： dataIdentifier 0x0104

— 字节＃6和7： 阅读和定义阅读类型的本地化。

例1 如果读数在数据记录的第11个字节中，则适用以下内容：

— 11x8 = 88 dec = 000101 1000b位＃10 - 位＃14：以位为单位的长度 - 1。

— 用5位，最大尺寸为32位=“长”。

例2 对于“字”，长度因此为15 dec = 0 1111b位＃15：符号条目：1 =有符号的，0 =无符号示例3

总分配将是：

— 1011 1100 0101 1000b = 0xBC58因此字节＃6包含0xBC，字节＃7包含0x58

— 字节＃8：比较操作（操作员）例4

运算符MV> CP = 0x01

— 字节＃9-12： 由于4字节长度的比较参数，可以传输从’位’到’长’类型的所有数据格式。

例5 如果比较值是5 242 dec = 0x0000 147A，

— 字节＃9 = 0x00，字节＃10 = 0x00，字节＃11 = 0x14，字节＃12 = 0x7A

— 字节＃13：滞后值（指定为比较参数的百分比）。 该值是直接指定的。 它仅适用于运算符 “<”和 “>”。在比较值为零的情况下，滞后值应被定义为绝对值。

例6 滞后值10％= 0x0A

表122定义了ResponseOnEvent请求消息示例#3。

表122 - ResponseOnEvent请求消息示例#3

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ResponseOnEvent请求SID	0x86	鱼子
#2	eventTypeRecord [eventType] = onComparisonOfValues, storageState = storeEvent suppressPosRspMsgIndicationBit = FALSE	0x47	ET_OCOV
#3	eventWindowTime =无限	0x02	EWT
#4	eventTypeRecord [eventTypeParameter#1] = recordDataIdentifier (高字节)	0x01	ETR_ETP1
#5	eventTypeRecord [eventTypeParameter#2] = recordDataIdentifier (Low Byte)	0x04	ETR_ETP2
#6	eventTypeRecord [eventTypeParameter#3] = Valueinfo#1	0xBC	ETR_ETP3
#7	eventTypeRecord [eventTypeParameter#4] = Valueinfo#2	0x58	ETR_ETP4
#8	eventTypeRecord [eventTypeParameter#5] =运算符	0x01	ETR_ETP5
#9	eventTypeRecord [eventTypeParameter#6] =比较参数 (字节#4)	0x00	ETR_ETP6
#10	eventTypeRecord [eventTypeParameter#7] =比较参数 (字节#3)	0x00	ETR_ETP7
#11	eventTypeRecord [eventTypeParameter#8] =比较参数 (字节#2)	0x14	ETR_ETP8
#12	eventTypeRecord [eventTypeParameter#9] =比较参数 (字节#1)	0x7A	ETR_ETP9
#13	eventTypeRecord [eventTypeParameter#10] =滞后[%]	0x0A	ETR_ETP10
#14	serviceToRespondToRecord [serviceID] = ReadDataByIdentifier	0x22	RDBI
#15	serviceToRespondToRecord [serviceParameter#1] = dataIdentifier (MSB)	0x01	DID_B1
#16	serviceToRespondToRecord [serviceParameter#2] = dataIdentifier (LSB)	0x04	DID_B2

注意 没有显示响应消息和随后的初始化序列。

如果在成功设置和激活ROE机制的事件窗口后，测量值高于5 242_d，服务器会作出反应。 指定的事件发生并且服务器发送以下消息。

表123定义ReadDataByIdentifier肯定响应消息示例#3。

表123 - ReadDataByIdentifier积极响应消息示例#3

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier响应SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	0x01	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x04	DID_B2
#4	dataRecord [数据#1]	0xXX	DREC_DATA1
#5	dataRecord [data#2]	0xXX	DREC_DATA2
#6	dataRecord [data#3]	0xXX	DREC_DATA3
#7	dataRecord [data#4]	0xXX	DREC_DATA4
#8	dataRecord [数据#5]	0xXX	DREC_DATA5
#9	dataRecord [数据#6]	0xXX	DREC_DATA6
#10	dataRecord [数据#7]	0xXX	DREC_DATA7
#11	dataRecord [数据#8]	0xXX	DREC_DATA8
#12	dataRecord [数据#9]	0xXX	DREC_DATA9
#13	dataRecord [数据#10]	0xXX	DREC_DATA10
#14	dataRecord [数据#11]字节#11的数据内容: 0x14	0x14	DREC_DATA11
#15	dataRecord [data#12]字节#12的数据内容: 0x7B	0x7B	DREC_DATA12
:	:	:	:

在测量值至少低于比较参数值的90%之前发生另一事件。 这种行为由滞后值指定。 如果满足该条件并且测量值再次高于比较值，则会发生新事件，并由服务器发送新的ReadDataByIdentifier响应消息。

9.11 LinkControl (0x87)服务

9.11.1 服务说明

LinkControl服务用于控制客户端和服务器之间的通信，以获取用于诊断目的（例如编程）的总线带宽。 此服务可选择应用于那些数据链路层，这样可以在非默认诊断会话期间重新配置其通信参数（例如，更改CAN上的波特率或重新配置FlexRay周期设计）。

注: 有关在某个数据链路层上应用和使用此服务的更多详细信息，请参阅单个数据链路层特定诊断服务实现UDSonXYZ'数据链路'规范。

该服务用于将数据链路层转换为特定状态，从而允许利用更可能用于编程目的的更高诊断带宽。 为了克服功能上的通信限制（例如，波特率必须同时在多个服务器中转换），转换本身分为两个步骤：

步骤1：客户端验证是否可以执行转换并通知服务器有关要使用的模式转换机制。 在客户端执行第2步之前，每个服务器都必须积极响应（suppressPosRspMsgIndicationBit = FALSE）。 这一步实际上并不执行模式转换。

步骤#2：客户端实际上请求模式转换（例如，较高的波特率）。 如果步骤#1已成功执行，则只能请求该步骤。 在功能通信的情况下，建议在执行模式转换时（suppressPosRspMsgIndicationBit = TRUE）服务器不应有任何响应，因为一个服务器可能已经转换到新模式，而其他服务器仍在进行中。

请求消息中的linkControlType参数与条件linkControlModeIdentifier / linkRecord参数一起提供了一种机制，可以使用预定义的模式转换参数或特定的模式转换参数进行转换。

注意此服务绑定到非defaultSession。 会话层计时器超时将会将服务器转换回其（正常）操作模式。 在执行ECUReset服务（0x11）的情况下也是如此。 一旦发生数据链接模式转换，任何其他非defaultSession请求都不应导致重新转换到默认操作模式（例如，在编程会话期间）。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

9.11.2 请求消息

9.11.2.1 请求消息定义

表124定义了请求消息（linkControlType = verifyModeTransitionWithFixedParameter）。

表124 – 请求消息定义（linkControlType = verifyModeTransitionWithFixedParameter）

A_Data字节	参数名称	Cvt	字节值	助记符
#1	LinkControl请求SID	M	0x87	LC
#2	子函数=［linkControlType］	M	0x01	LEV_LCTP_
#3	linkControlModeIdentifier	M	0x00 – 0xFF	LCMI_

表125定义了请求消息（linkControlType = verifyModeTransitionWithSpecificParameter）。

表125 – 请求消息定义（linkControlType = verifyModeTransitionWithSpecificParameter）

A_Data字节	参数名称	Cvt	字节值	助记符
#1	LinkControl请求SID	M	0x87	LC
#2	子函数=［linkControlType］	M	0x02	LEV_LCTP_
	linkRecord[] = [			LBR_
#3	modeParameterHighByte	M	0x00 – 0xFF	MPHB
#4	modeParameterMiddleByte	M	0x00 – 0xFF	MPMB
#5	modeParameterLowByte]	M	0x00 – 0xFF	MPLB

表126定义了请求消息（linkControlType = transitionMode）。

表126 – 请求消息定义（linkControlType = transitionMode）

A_Data字节	参数名称	Cvt	字节值	助记符
#1	LinkControl请求SID	M	0x87	LC
#2	子函数=[linkControlType]	M	0x03	LEV_LCTP_

9.11.2.2 请求消息子函数参数$ Level (LEV_) 定义

LinkControl请求消息使用子函数参数linkControlType来描述要在服务器中执行的操作（suppressPosRspMsgIndicationBit (bit 7)，未在下表中显示）。

表127定义了请求消息子功能参数。

表127 – 请求消息子功能参数定义

位6 – 0	描述	Cvt	助记符
0x00	**ISOSAEReserved** 该值由本文档保留。	M	ISOSAERESRVD
0x01	**verifyModeTransitionWithFixedParameter** 该参数用于验证是否可以执行具有由linkControlModeIdentifier数据参数指定的预定义参数的转换。	U	VMTWFP
0x02	**verifyModeTransitionWithSpecificParameter** 该参数用于验证是否可以执行由linkRecord数据参数指定的特定定义参数（例如特定波特率）的转换。	U	VMTWSP
0x03	**transitionMode** 该子功能参数请求服务器将数据链路转换为在前面的验证消息中请求的模式。	U	TM值
0x04 – 0x3F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD
0x40 – 0x5F	**vehicleManufacturerSpecific** 这个值的范围保留给车辆制造商特定用途。	U	VMS
0x60 – 0x7E	**systemSupplierSpecific** 这个值的范围保留给系统供应商特定用途。	U	SSS
0x7F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

9.11.2.3 请求消息数据参数定义

表128定义了请求消息的数据参数。

表128 – 请求消息数据参数定义

定义
linkControlModeIdentifier 这个条件参数引用一个固定的定义模式参数来转换到（见B.3）。
linkRecord 在子功能参数指示使用特定参数的情况下，该条件参数记录包含特定的模式参数。 linkRecord的格式在单个数据链接特定的诊断规范（UDSonXYZ）中指定。

9.11.3 积极的回应消息

9.11.3.1 积极响应消息的定义

表129定义了肯定响应消息。

表129 – 肯定回应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	LinkControl响应SID	M	0xC7	LCPR
#2	linkControlType	M	00-7F	LCTP

9.11.3.2 肯定回复消息数据参数定义

表130定义了肯定响应消息的数据参数。

表130 – 响应消息数据参数定义

定义
linkControlType 该参数是来自请求消息的linkControlType子函数参数的位6 – 0的回显。

9.11.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表131中。如果错误情况适用于服务器，则应使用列出的否定响应。

表131 – 支持的否定响应代码

NRC	描述	助记符
0x12	子functionNotSupported 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果所请求的LinkControl的标准不符合，则应该返回该NRC。	CNC
0x24	**requestSequenceError** 如果客户请求转换操作模式而没有先前的验证步骤（该步骤指定要转换到的模式），则应返回此NRC。	RSE
0x31	**requestOutOfRange** 这个NRC将被退回 请求的linkControlModeIdentifier无效；具体的 modeParameter（linkRecord）无效；	ROOR

9.11.5 消息流示例（s）LinkControl

9.11.5.1 示例#1 – 将波特率转换为固定波特率（PC波特率115200 kBit / s）

9.11.5.1.1 步骤1：验证波特率开关是否满足所有条件

表132定义了链路控制请求消息流程示例#1-步骤#1。

表132 – 链路控制请求消息流程示例#1 – 步骤#1

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	LinkControl请求SID	0x87	LC
#2	linkControlType = verifyModeTransitionWithFixedParameter, suppressPosRspMsgIndicationBit = FALSE	0x01	VMTWFP
#3	linkControlModeIdentifier = PC115200Baud	0x05	BI_PC115200

表133定义LinkControl肯定响应消息流程示例#1-步骤#1。

表133 – LinkControl肯定响应消息流程示例#1 – 步骤#1

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	LinkControl响应SID	0xC7	LCPR
#2	linkControlType = verifyModeTransitionWithFixedParameter	0x01	VMTWFP

9.11.5.1.2 步骤#2：转换波特率

表134定义了链路控制请求消息流程示例#1-步骤#2。

表134-链路控制请求消息流程示例#1-步骤#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	LinkControl请求SID	0x87	LC
#2	linkControlType = transitionMode, suppressPosRspMsgIndicationBit = TRUE	0x83	TM值

没有来自服务器的响应。 客户端和服务器必须转换其通信链路的波特率。

9.11.5.2 示例#2 - 将波特率转换为特定波特率（150kBit / s）

9.11.5.2.1 步骤1：验证波特率开关是否满足所有条件

表135定义了LinkControl请求消息流程示例#2 - 步骤#1。

表135 - LinkControl请求消息流程示例#2 - 步骤#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	LinkControl请求SID	0x87	LC
#2	linkControlType = verifyModeTransitionWithSpecificParameter	0x02	VMTWSP
#3	linkRecord [modeParameterHighByte] (150kBit / s)	0x02	MPHB
#4	linkRecord [modeParameterMiddleByte]	0x49	MPMB
#5	linkRecord [modeParameterLowByte]	0xF0	MPLB

表136定义LinkControl肯定响应消息流程示例#2 - 步骤#1。

表136 - LinkControl肯定响应消息流程示例#2 - 步骤#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	LinkControl响应SID	0xC7	LCPR
#2	linkControlType = verifyModeTransitionWithSpecificParameter	0x02	VMTWSP

9.11.5.2.2 步骤#2：转换波特率

表137定义了LinkControl请求消息流程示例#2 - 步骤#2。

表137 – LinkControl请求消息流程示例#2 – 步骤#2

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	LinkControl请求SID	0x87	LC
#2	linkControlType = transitionMode, suppressPosRspMsgIndicationBit = TRUE	0x83	TM值

没有来自服务器的响应。 客户端和服务器必须转换其通信链路的波特率。

9.11.5.3　示例#3 – 将FlexRay循环设计转换为“编程”

以下示例反映了FlexRay网络周期设计转换为优化的“编程”模式（例如，利用增强型动态段进行编程）的场景。

9.11.5.3.1　步骤1：验证是否满足调度程序切换的所有条件

表138定义了LinkControl请求消息流程示例#3 – 步骤#1。

表138 – LinkControl请求消息流程示例#3 – 步骤#1

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	LinkControl请求SID	0x87	LC
#2	linkControlType = verifyModeTransitionWithFixedParameter, suppressPosRspMsgIndicationBit = FALSE	0x01	VMTWFP
#3	linkControlModeIdentifier = ProgrammingSetup	0x20	PROGSU

表139定义了LinkControl肯定响应消息流程示例#3 – 步骤#1。

表139 – LinkControl肯定响应消息流程示例#3 – 步骤#1

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	LinkControl响应SID	0xC7	LCPR
#2	linkControlType = verifyModeTransitionWithFixedParameter	0x01	VMTWFP

9.11.5.3.2　步骤2：转换到编程调度器

表140定义了LinkControl请求消息流程示例#3-步骤#2。

表140-链路控制请求消息流程示例#3 - 步骤#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	LinkControl请求SID	0x87	LC
#2	linkControlType = transitionMode, suppressPosRspMsgIndicationBit = TRUE	0x83	TM值

没有来自服务器的响应。 客户端和服务器必须转换FlexRay通信链路的周期设计。

10 数据传输功能单元

10.1 概观

表141定义了数据传输功能单元。

表141 - 数据传输功能单元

服务	描述
ReadDataByIdentifier	客户端请求读取由提供的dataIdentifier标识的记录的当前值。
ReadMemoryByAddress	客户端请求读取提供的内存范围的当前值。
ReadScalingDataByIdentifier	客户端请求读取由提供的dataIdentifier标识的记录的缩放信息。
ReadDataByPeriodicIdentifier	客户端请求在服务器中调度数据以进行定期传输。
DynamicallyDefineDataIdentifier	客户端请求动态定义可能随后由readDataByIdentifier服务读取的数据标识符。
WriteDataByIdentifier	客户端请求写入由提供的dataIdentifier指定的记录。
WriteMemoryByAddress	客户端请求覆盖提供的内存范围。

10.2 ReadDataByIdentifier (0x22) 服务

10.2.1 服务说明

ReadDataByIdentifier服务允许客户端从一个或多个dataIdentifiers标识的服务器请求数据记录值。

客户端请求消息包含一个或多个两字节的dataIdentifier值，用于标识服务器维护的数据记录（请参阅C.1了解允许的dataIdentifier值）。 dataRecord的格式和定义应该是车辆制造商或系统供应商特定的，并且如果服务器支持，可以包括模拟输入和输出信号，数字输入和输出信号，内部数据和系统状态信息。

服务器可以限制车辆制造商和系统供应商同意的可同时请求的数据标识符的数量。

一旦接收到ReadDataByIdentifier请求，服务器将访问由dataIdentifier参数指定的记录的数据元素，并在包含关联的dataRecord参数的单个ReadDataByIdentifier肯定响应中传送它们的值。 请求消息可能包含

相同的dataIdentifier多次。 服务器应将每个dataIdentifier作为一个单独的参数对待，并按照请求的频率对每个dataIdentifier的数据进行响应。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

10.2.2 请求消息

10.2.2.1 请求消息定义

表142定义了请求消息。

表142 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDataByIdentifier请求SID	M	0x22	RDBI
#2 #3	dataIdentifier []#1 = [字节#1 (MSB) 字节 #2]	M M	0x00 – 0xFF 0x00 – 0xFF	DID_ HB LB
:	:	:	:	:
#n-1 #n	dataIdentifier []#m = [字节#1 (MSB) 字节 #2]	U U	0x00 – 0xFF 0x00 – 0xFF	DID_ HB LB

10.2.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

10.2.2.3 请求消息数据参数定义

表143定义了请求消息的数据参数。

表143 – 请求消息数据参数定义

定义
dataIdentifier (#1至#m) 此参数标识客户端请求的服务器数据记录 (有关详细参数定义，请参阅C.1) 。

10.2.3 积极的回应消息

10.2.3.1 积极响应消息的定义

表144定义了肯定响应消息。

表144 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDataByIdentifier响应SID	M	0x62	RDBIPR
#2 #3	dataIdentifier []#1 = [字节#1 (MSB) 字 节#2]	M M	0x00 – 0xFF 0x00 – 0xFF	DID_ HB LB
#4 : #(k-1)+4	dataRecord []#1 = [数据#1 : 数据#k]	M : U	0x00 – 0xFF : 0x00 – 0xFF	DREC_ DATA_1 : DATA_m
:	:	:	:	:
#N- (0-1) -2 #N- (0-1) -1	dataIdentifier []#m = [字节#1 (MSB) 字节 #2]	U U	0x00 – 0xFF 0x00 – 0xFF	DID_ HB LB
#N- (0-1) : #n	dataRecord []#m = [数据#1 : 数据#o]	U : U	0x00 – 0xFF : 0x00 – 0xFF	DREC_ DATA_1 : DATA_k

10.2.3.2 肯定回复消息数据参数定义

表145定义了肯定响应消息的数据参数。

表145 – 响应消息数据参数定义

定义
dataIdentifier (#1至#m) 该参数是来自请求消息的数据参数dataIdentifier的回显。
dataRecord (#1到#k / o) ReadDataByIdentifier肯定响应消息使用此参数向客户端提供请求的数据记录值。 dataRecord的内容未在本文档中定义，并且是车辆制造商特定的。

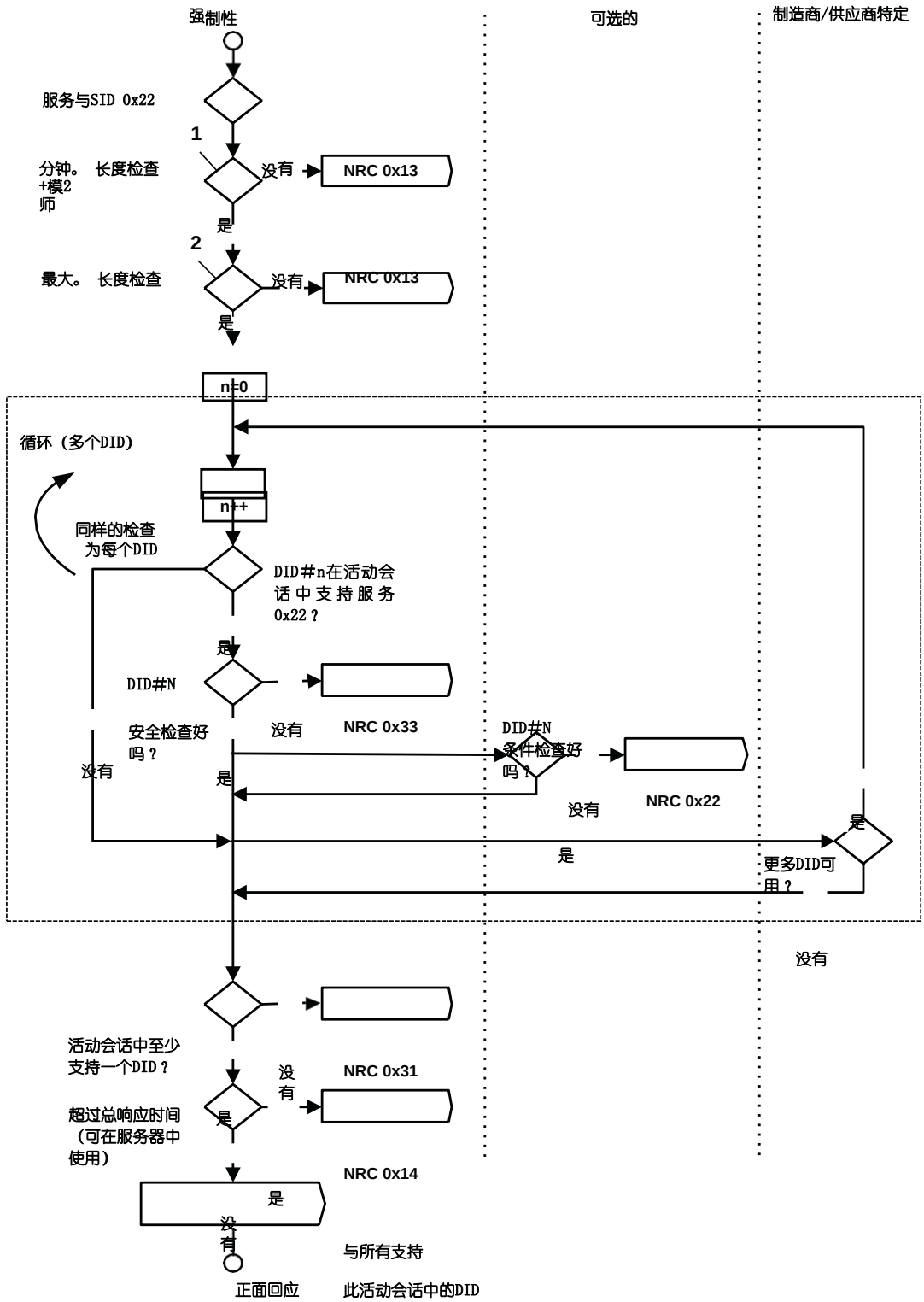
10.2.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表146中。如果错误情况适用于服务器，则应使用列出的否定响应。

表146 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果请求消息的长度无效或客户端一次超出允许请求的最大数据标识符数量，则应发送此NRC。	IMLOIF
0x14	**responseTooLong** 如果响应消息的总长度超过了底层传输协议的限制（例如，在单个请求中请求多个DID），则应发送此NRC。	RTL
0x22	**conditionsNotCorrect** 如果服务器的运行条件不符合要求，则应发送NRC。	CNC
0x31	**requestOutOfRange** 这个NRC将被发送，如果 设备不支持所请求的dataIdentifier值；当前会话中不支持所请求的数据标识符； 请求的dynamicDefinedDataIdentifier尚未分配；	ROOR
0x33	**securityAccessDenied** 如果至少有一个dataIdentifiers受到保护且服务器未处于解锁状态，则应发送此NRC。	伤心

评估顺序记录在图15中。



键

1 最小长度为3个字节（SI + DID）

2 最大长度为1个字节（SI）+ 2 * n个字节（DID（s））

图15 - ReadDataByIdentifier服务的NRC处理

10.2.5 消息流示例ReadDataByIdentifier

10.2.5.1 假设

本子条款指定了示例执行ReadDataByIdentifier服务所要满足的条件。 客户端可以在任何时候请求dataIdentifier数据，而与服务器的状态无关。

下面的dataIdentifier示例是动力总成设备（例如，发动机控制模块）特有的。 有关排放相关系统的公认术语/定义/首字母缩略词的更多详细信息，请参阅ISO 15031-2 [6]。

第一个示例读取包含单个信息（其中dataIdentifier 0xF190包含VIN号码）的单个2字节dataIdentifier。

第二个例子展示了用单个请求（其中dataIdentifier 0x010A包含发动机冷却剂温度，节气门位置，发动机转速，歧管绝对压力，质量空气流量，车辆速度传感器，大气压力，计算的负载值，怠速空气控制等等）请求多个dataIdentifiers。和油门踏板位置，dataIdentifier 0x0110包含电池正极电压）。

10.2.5.2 示例#1：读取单个数据标识符0xF190（VIN号）

表147定义了ReadDataByIdentifier请求消息流程示例#1。

表147 - ReadDataByIdentifier请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier请求SID	0x22	RDBI
#2	dataIdentifier [byte#1] (MSB)	0xF1	DID_B1
#3	dataIdentifier [byte#2]	0x90	DID_B2

表148定义了ReadDataByIdentifier肯定响应消息流程示例#1。

表148 - ReadDataByIdentifier肯定响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier响应SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	0xF1	DID_B1
#3	dataIdentifier [byte#2]	0x90	DID_B2
#4	dataRecord [数据#1] = VIN数字1 = “W”	0x57	DREC_DATA1
#5	dataRecord [data#2] = VIN数字2 = “0”	0x30	DREC_DATA2
#6	dataRecord [data#3] = VIN数字3 = “L”	0x4C	DREC_DATA3
#7	dataRecord [data#4] = VIN数字4 = “0”	0x30	DREC_DATA4
#8	dataRecord [数据#5] = VIN数字5 = “0”	0x30	DREC_DATA5
#9	dataRecord [数据#6] = VIN数字6 = “0”	0x30	DREC_DATA6
#10	dataRecord [数据#7] = VIN数字7 = “0”	0x30	DREC_DATA7

表148 - （续）

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#11	dataRecord［数据#8］ = VIN数字8 = “4”	0x34	DREC_DATA8
#12	dataRecord［数据#9］ = VIN数字9 = “3”	0x33	DREC_DATA9
#13	dataRecord［数据#10］= VIN数字10 = “M”	0x4D	DREC_DATA10
#14	dataRecord［数据#11］= VIN数字11 = “B”	0x42	DREC_DATA11
#15	dataRecord［数据#12］= VIN数字12 = “5”	0x35	DREC_DATA12
#16	dataRecord［数据#13］= VIN数字13 = “4”	0x34	DREC_DATA13
#17	dataRecord［数据#14］= VIN数字14 = “1”	0x31	DREC_DATA14
#18	dataRecord［数据#15］= VIN数字15 = “3”	0x33	DREC_DATA15
#19	dataRecord［数据#16］= VIN数字16 = “2”	0x32	DREC_DATA16
#20	dataRecord［数据#17］= VIN数字17 = “6”	0x36	DREC_DATA17

10.2.5.3　示例#2：读取多个dataIdentifiers 0x010A和0x0110

表149定义了ReadDataByIdentifier请求消息流程示例#2。

表149 - ReadDataByIdentifier请求消息流程示例#2

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier请求SID	0x22	RDBI
#2	dataIdentifier#1［byte#1］(MSB)	0x01	DID_B1
#3	dataIdentifier#1［byte#2］	0x0A	DID_B2
#4	dataIdentifier#2［byte#1］(MSB)	0x01	DID_B1
#5	dataIdentifier#2［byte#2］	0x10	DID_B2

表150定义了ReadDataByIdentifier肯定响应消息流程示例#2。

表150 - ReadDataByIdentifier肯定响应消息流程示例#2

消息方向	服务器 客户		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier响应SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	0x01	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x0A	DID_B2
#4	dataRecord [数据# 1] = ECT	0xA6	DREC_DATA1
#5	dataRecord [data# 2] = TP	0x66	DREC_DATA2
#6	dataRecord [data# 3] = RPM	0x07	DREC_DATA3
#7	dataRecord [data# 4] = RPM	0x50	DREC_DATA4
#8	dataRecord [数据# 5] = MAP	0x20	DREC_DATA5
#9	dataRecord [数据# 6] = MAF	0x1A	DREC_DATA6
#10	dataRecord [数据# 7] = VSS	0x00	DREC_DATA7
#11	dataRecord [数据# 8] = BARO	0x63	DREC_DATA8
#12	dataRecord [数据# 9] = LOAD	0x4A	DREC_DATA9
#13	dataRecord [数据# 10] = IAC	0x82	DREC_DATA10
#14	dataRecord [数据# 11] = APP	0x7E	DREC_DATA11
#15	dataIdentifier [byte#1] (MSB)	0x01	DID_B1
#16	dataIdentifier [byte#2] (LSB)	0x10	DID_B2
#17	dataRecord [data#1] = B +	0x8C	DREC_DATA1

10.3 ReadMemoryByAddress (0x23) 服务

10.3.1 服务说明

ReadMemoryByAddress服务允许客户端通过提供的起始地址和要读取的内存大小向服务器请求内存数据。

ReadMemoryByAddress请求消息用于请求由参数memoryAddress和memorySize标识的服务器的内存数据。 用于memoryAddress和memorySize参数的字节数由addressAndLengthFormatIdentifier (低位和高位半字节) 定义。

也可以使用固定的addressAndLengthFormatIdentifier，并且memoryAddress或memorySize参数内的未使用字节在较高范围地址位置填充值为0x00。

在存储区重叠的情况下，可以使用额外的存储器地址字节作为存储器标识符（例如，使用内部和外部闪存）。

服务器通过ReadMemoryByAddress肯定响应消息发送数据记录值。 dataRecord参数的格式和定义应该是车辆制造商特定的。 如果服务器支持，dataRecord参数可以包括模拟输入和输出信号，数字输入和输出信号，内部数据和系统状态信息。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

10.3.2 请求消息

10.3.2.1 请求消息定义

表151定义了请求消息。

表151 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadMemoryByAddress请求SID	M	0x23	RMBA
#2	addressAndLengthFormatIdentifier	M	0x00 – 0xFF	ALFID
#3 ： #(m-1)+3		M ： C1	0x00 – 0xFF ： 0x00 – 0xFF	MA_ B1 ： Bm
#N-（K-1） ： #n		M ： C2	0x00 – 0xFF ： 0x00 – 0xFF	MS_ B1 ： Bk
C1：存在的这个参数依靠上地址长度信息参数的该地址和长度格式标识符 C2：存在的这个参数依靠上该记忆尺寸长度信息的该地址和长度格式标识符。				

10.3.2.2 请求消息子函数参数$ Level（LEV_）定义

此服务不使用子功能参数。

10.3.2.3 请求消息数据参数定义

表152定义了请求消息的数据参数。

表152 – 请求消息数据参数定义

定义
addressAndLengthFormatIdentifier 该参数是一个单字节值，每个半字节分别进行编码（参见H.1的示例值）：位7 – 4：memorySize参数的长度（字节数） 位3 – 0：memoryAddress参数的长度（字节数）
memoryAddress 参数memoryAddress是要从中检索数据的服务器内存的起始地址。 用于该地址的字节数由addressAndLengthFormatIdentifier的低半字节（bit 3-0）定义。 memoryAddress参数中的字节#m始终是服务器中引用地址的最低有效字节。 地址的最高有效字节可用作存储器标识符。 使用内存标识符的一个例子是具有16位寻址和内存地址重叠的双处理器服务器（当给定地址对任一处理器有效但是产生不同的物理内存设备或使用内部和外部闪存时）。 在这种情况下，可以将memoryAddress参数中另外未使用的字节指定为用于选择所需存储器设备的存储器标识符。 该功能的使用应该由车辆制造商/系统供应商定义。
memorySize ReadMemoryByAddress请求消息中的参数memorySize指定要从服务器内存中由memoryAddress指定的地址开始读取的字节数。 用于此大小的字节数由addressAndLengthFormatIdentifier的高半字节（第7 – 4位）定义。

10.3.3 积极的回应消息

10.3.3.1 积极响应消息的定义

表153定义了肯定响应消息。

表153 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadMemoryByAddress响应SID	M	0x63	RMBAPR
#2 : #n	dataRecord[] = [数据#1 : 数据#m]	M : U	0x00 – 0xFF : 0x00 – 0xFF	DREC_ DATA_1 : DATA_m

10.3.3.2 肯定回复消息数据参数定义

表154定义了肯定响应消息的数据参数。

表154 – 响应消息数据参数定义

定义
dataRecord ReadMemoryByAddress肯定响应消息使用此参数向客户端提供请求的数据记录值。 dataRecord的内容未在本文档中定义，并应反映所请求的存储器内容。 数据格式应该由车辆制造商/系统供应商定义。

10.3.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表155中。如果错误情况适用于服务器，则应使用列出的否定响应。

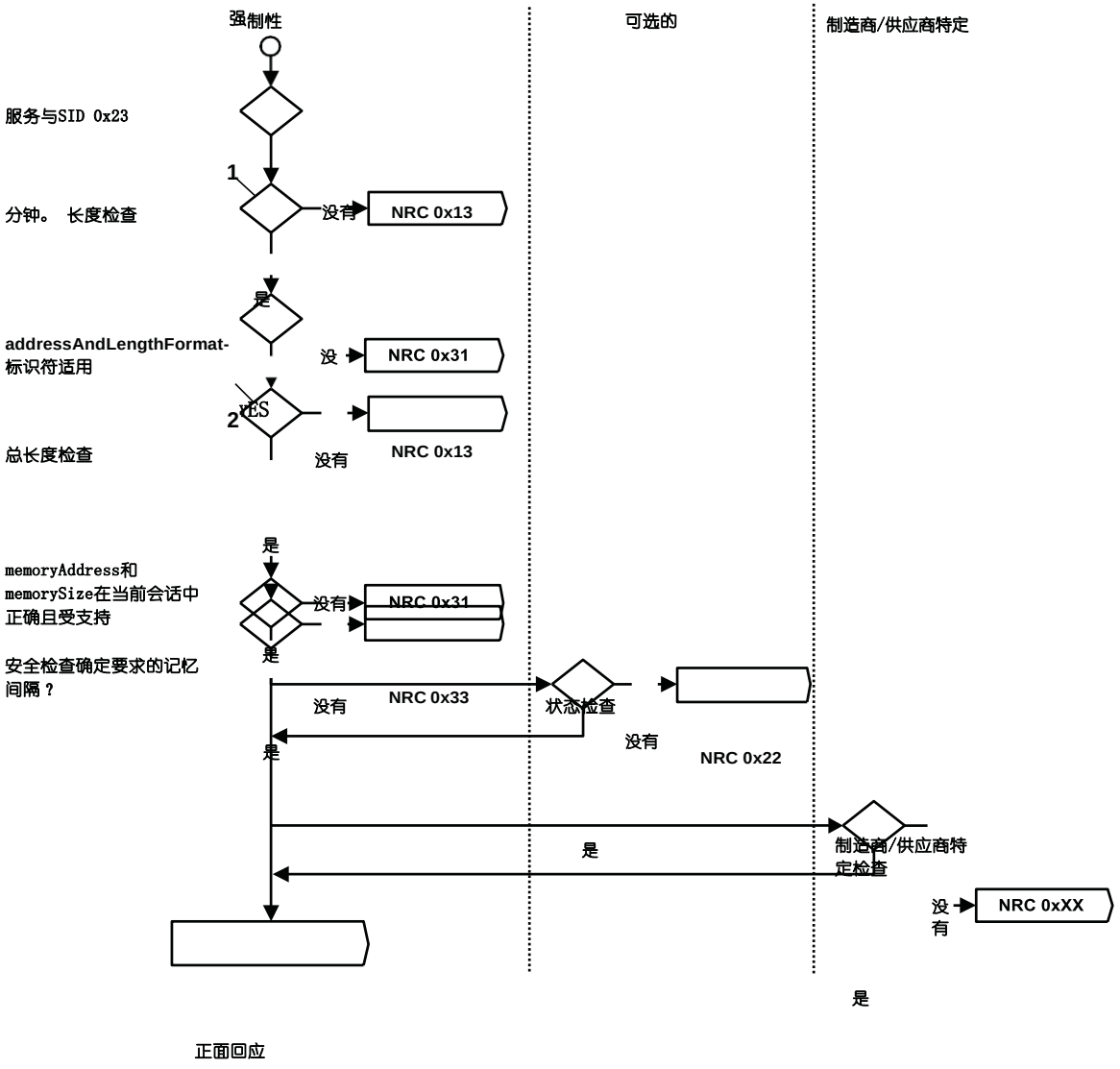
表155 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果服务器的运行条件不符合要求，则应发送NRC。	CNC
0x31	**requestOutOfRange** 如果发生以下情况，应发送NRC: — 区间[0xMA, (0xMA + 0xMS -0x1)]内的任何内存地址都是无效的; — 区间[0xMA, (0xMA + 0xMS -0x1)]内的任何内存地址都受到限制; — 请求消息中的memorySize参数值不受服务器支持; — 指定的addressAndLengthFormatIdentifier无效; — 请求消息中的memorySize参数值为零;	ROOR

表155 - （续）

0x33	**SecurityAccessDenied** 如果间隔[0xMA，（0xMA + 0xMS - 0x1）]内的任何内存地址是安全的并且服务器被锁定，则应发送此NRC。	伤心

评估顺序记录在图16中。



键

1 至少4（SI +地址和长度格式标识符+最小内存地址+最小内存大小）

2 在1字节SI + 1字节addressAndLengthFormatIdentifier + n字节memoryAddress参数长度+ n字节memorySize参数长度）

图16 - ReadMemoryByAddress服务的NRC处理

10.3.5 消息流示例ReadMemoryByAddress

10.3.5.1 假设

本小节指定了示例执行ReadMemoryByAddress服务所要满足的条件。 本例中的服务不受服务器限制。

10.3.5.2 示例#1: ReadMemoryByAddress - 4字节（32位）寻址

客户端从内存地址0x2048 1392开始从服务器的内存中读取259个数据字节。表156定义了ReadMemoryByAddress请求消息流程示例#1。

表156 - ReadMemoryByAddress请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadMemoryByAddress请求SID	0x23	RMBA
#2	addressAndLengthFormatIdentifier	0x24	ALFID
#3	memoryAddress [byte#1] (MSB)	0x20	MA_B1
#4	memoryAddress [byte#2]	0x48	MA_B2
#5	memoryAddress [byte#3]	0x13	MA_B3
#6	memoryAddress [byte#4]	0x92	MA_B4
#7	memorySize [byte#1] (MSB)	0x01	MS_B1
#8	memorySize [byte#2]	0x03	MS_B2

表157定义了ReadMemoryByAddress正响应消息流程示例#1。

表157 - ReadMemoryByAddress肯定响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadMemoryByAddress响应SID	0x63	RMBAPR
#2	dataRecord [数据#1] (存储单元#1)	0x00	DREC_DATA_1
:	:	:	: DREC_DATA_259
#259+1	dataRecord [data#259] (存储单元#259)	0x8C	

10.3.5.3 示例2：ReadMemoryByAddress - 2字节（16位）寻址。

客户端从内存地址0x4813开始读取服务器内存中的五个数据字节。 表158定义了ReadMemoryByAddress请求消息流程示例#2。

表158 - ReadMemoryByAddress请求消息流程示例#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadMemoryByAddress请求SID	0x23	RMBA
#2	addressAndLengthFormatIdentifier	0x12	ALFID
#3	memoryAddress [byte#1 (MSB)]	0x48	MA_B1
#4	memoryAddress [byte#2 (LSB)]	0x13	MA_B2
#5	memorySize [byte#1]	0x05	MS_B1

表159定义了ReadMemoryByAddress正响应消息流程示例#2。

表159 - ReadMemoryByAddress积极响应消息流程示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadMemoryByAddress响应SID	0x63	RMBAPR
#2	dataRecord [数据#1] (存储单元#1)	0x43	DREC_DATA_1
#3	dataRecord [data#2] (存储单元#2)	0x2A	DREC_DATA_2
#4	dataRecord [数据#3] (存储单元#3)	0x07	DREC_DATA_3
#5	dataRecord [数据#4] (存储单元#4)	0x2A	DREC_DATA_4
#6	dataRecord [data#5] (存储单元#5)	0x55	DREC_DATA_5

10.3.5.4 示例#3：ReadMemoryByAddress，3字节（24位）寻址

客户端从内存地址0x204813开始从服务器的外部RAM单元中读取三个数据字节。 表160定义了ReadMemoryByAddress请求消息流程示例#3。

表160 – ReadMemoryByAddress请求消息流程示例#3

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadMemoryByAddress请求SID	0x23	RMBA
#2	addressAndLengthFormatIdentifier	0x23	ALFID
#3	memoryAddress [byte#1 (MSB)]	0x20	MA_B1
#4	memoryAddress [byte#2]	0x48	MA_B2
#5	memoryAddress [byte#3 (LSB)]	0x13	MA_B3
#6	memorySize [byte#1 (MSB)]	0x00	MS_B1
#7	memorySize [byte#2 (LSB)]	0x03	MS_B2

表161定义了ReadMemoryByAddress第一个肯定响应消息，例#3。

表161 – ReadMemoryByAddress第一个肯定响应消息，示例#3

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadMemoryByAddress响应SID	0x63	RMBAPR
#2	dataRecord [数据#1] (存储单元#1)	0x00	DREC_DATA_1
#3	dataRecord [数据#2] (存储单元#2)	0x01	DREC_DATA_2
#4	dataRecord [数据#3] (存储单元#3)	0x8C	DREC_DATA_3

10.4 ReadScalingDataByIdentifier (0x24) 服务

10.4.1 服务说明

ReadScalingDataByIdentifier服务允许客户端从dataIdentifier标识的服务器请求缩放数据记录信息。

客户端请求消息包含一个dataIdentifier值，用于标识由服务器维护的数据记录（请参阅C.1了解允许的dataIdentifier值）。 dataRecord的格式和定义应该是车辆制造商特定的，并且如果服务器支持，可以包括模拟输入和输出信号，数字输入和输出信号，内部数据和系统状态信息。

一旦接收到ReadScalingDataByIdentifier请求，服务器将访问与指定的dataIdentifier参数关联的缩放信息，并在一个ReadScalingDataByIdentifier正响应中传输缩放信息值。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

10.4.2 请求消息

10.4.2.1 请求消息定义

表162定义了请求消息。

表162 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadScalingDataByIdentifier请求SID	M	0x24	RSDBI
#2 #3		M M	0x00 – 0xFF 0x00 – 0xFF	DID_ HB 磅

10.4.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

10.4.2.3 请求消息数据参数定义

表163定义了请求消息的数据参数。

表163 – 请求消息数据参数定义

定义
DataIdentifier 此参数标识客户端请求的服务器数据记录（有关详细参数定义，请参阅C.1）。

10.4.3 积极的回应消息

10.4.3.1 积极响应消息的定义

表164定义了肯定响应消息。

表164 – 肯定回应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadScalingDataByIdentifier响应SID	M	0x64	RSDBIPR
#2 #3	dataIdentifier[] = [字节#1 (MSB) 字节 #2 (LSB)]	M M	0x00 – 0xFF 0x00 – 0xFF	DID_ HB LB
#4	scalingByte#1	M	0x00 – 0xFF	SB_1
#5 : #(p-1)+5	scalingByteExtension []#1 = [scalingByteExtensionParameter#1 : scalingByteExtensionParameter#p]	C1 : C1	0x00 – 0xFF : 0x00 – 0xFF	SBE_ PAR1 : PARp
:	:	:	:	:
#nr	scalingByte#k中	C2	0x00 – 0xFF	SB_k
#N- (R-1) : #n	scalingByteExtension []#k = [scalingByteExtensionParameter#1 : scalingByteExtensionParameter#r]	C1 : C1	0x00 – 0xFF : 0x00 – 0xFF	SBE_ PAR1 : PARr
C1：此参数的存在取决于scalingByte高半字节。 如果scalingByte高半字节被编码为公式，单位/格式或bitMappedReportedWithOutMask，则必须存在。 C2：该参数的存在取决于缩放信息的编码是否需要多于一个字节。				

10.4.3.2 肯定回复消息数据参数定义

表165定义了肯定响应消息的数据参数。

表165 – 响应消息数据参数定义

定义
dataIdentifier 该参数是来自请求消息的数据参数dataIdentifier的回显。
scalingByte (#1到#k) ReadScalingDataByIdentifier肯定响应消息使用此参数向客户端提供所请求的缩放数据记录值（有关详细参数定义，请参阅C.2）。
scalingByteExtension (#1到#p /#1到#r) 此参数用于提供scalingBytes的附加信息，其高位半字节编码为公式，单位/格式或bitmappedReportedWithOutMask（详细参数定义请参见C.3）。

10.4.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表166中。如果错误情况适用于服务器，则应使用列出的否定响应。

表166 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果请求消息的长度无效，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果服务器的运行条件不符合要求，则应发送NRC。	CNC
0x31	**requestOutOfRange** 如果出现下列情况，将返回NRC: 设备不支持请求的dataIdentifier值， 所请求的dataIdentifier值由设备支持，但没有缩放信息可用于指定的dataIdentifier。	ROOR
0x33	**securityAccessDenied** 如果dataIdentifier受到保护且服务器未处于解锁状态，则应发送此NRC。	伤心

评估顺序记录在图17中。

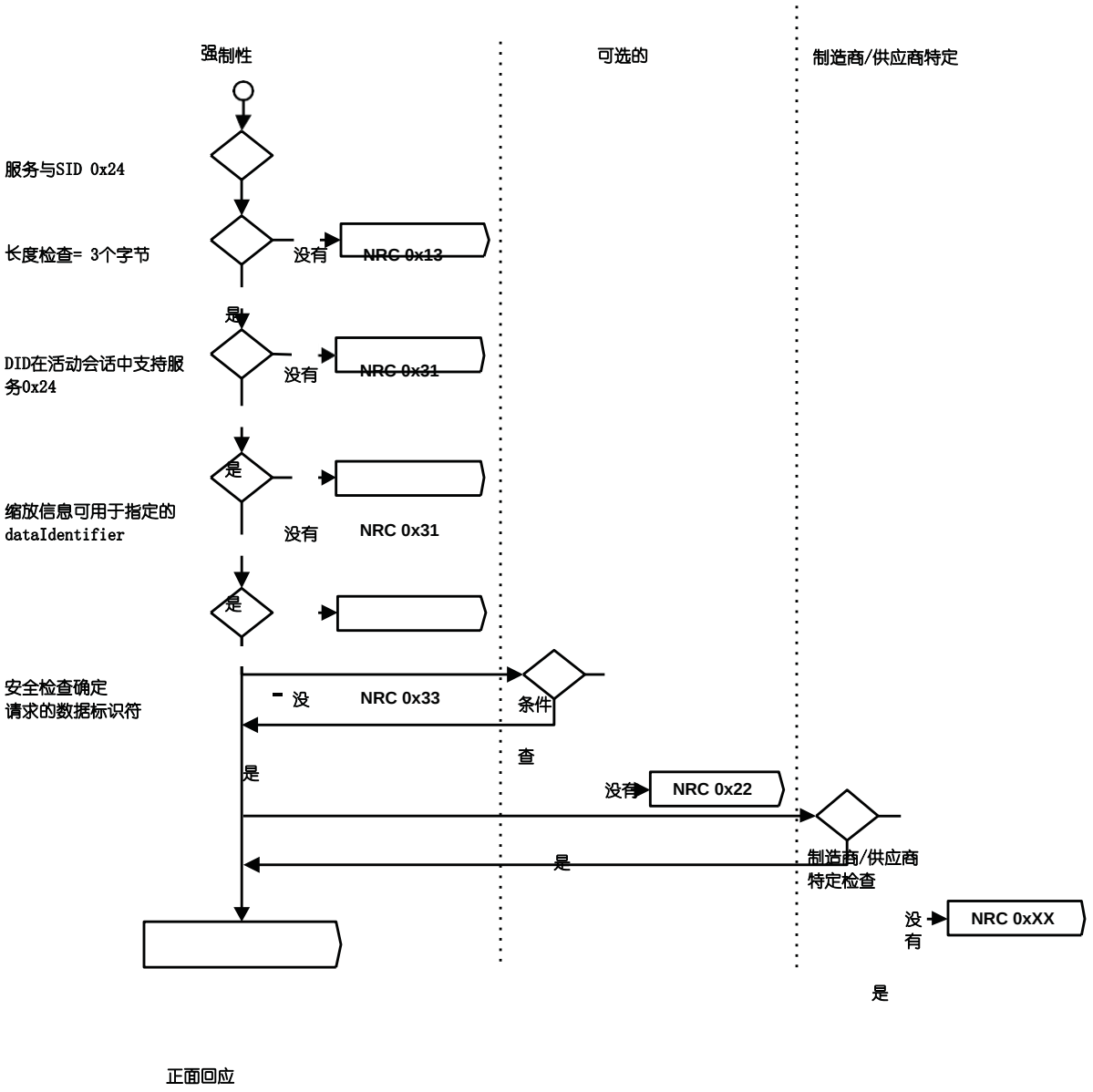


图17 - ReadScalingDataByIdentifier服务的NRC处理

10.4.5 消息流示例ReadScalingDataByIdentifier

10.4.5.1 假设

本小节指定了示例执行ReadScalingDataByIdentifier服务所要满足的条件。 客户端可以随时请求dataIdentifier缩放数据，而与服务器的状态无关。

第一个示例读取与包含单个信息（17个字符的VIN号码）的两个字节dataIdentifier 0xF190关联的缩放信息。

第二个例子演示了使用公式和单位标识符来指定服务器中的数据变量。

第三个例子说明如何使用readScalingDataByIdentifier返回位图映射的dataIdentifier的支持位（有效性掩码），该位标识符通过使用readDataByIdenditfier报告没有掩码。

10.4.5.2 示例#1: readScalingDataByIdentifier wth dataIdentifier 0xF190 (VIN号码)

表167定义了ReadScalingDataByIdentifier请求消息流程示例#1。

表167 - ReadScalingDataByIdentifier请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadScalingDataByIdentifier请求SID	0x24	RSDBI
#2	dataIdentifier [byte#1] (MSB)	0xF1	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x90	DID_B2

表168定义了ReadScalingDataByIdentifier肯定响应消息流程示例#1。

表168 - ReadScalingDataByIdentifier正响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadScalingDataByIdentifier响应SID	0x64	RSDBIPR
#2	dataIdentifier [byte#1] (MSB)	0xF1	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x90	DID_B2
#4	scalingByte#1 {ASCII, 15个数据字节}	0x6F	SB_1
#5	scalingByte#2 {ASCII, 2个数据字节}	0x62	SB_2

10.4.5.3 Example#2: readScalingDataByIdentifier wth dataIdentifier 0x0105 (Vehicle Speed)

表169定义了ReadScalingDataByIdentifier请求消息流程示例#2。

表169 - ReadScalingDataByIdentifier请求消息流程示例#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadScalingDataByIdentifier请求SID	0x24	RSDBI
#2	dataIdentifier [byte#1] (MSB)	0x01	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x05	DID_B2

表170定义了ReadScalingDataByIdentifier肯定响应消息流程示例#2。

表170 - ReadScalingDataByIdentifier肯定响应消息流程示例#2

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadScalingDataByIdentifier响应SID	0x64	RSDBIPR
#2	dataIdentifier [byte#1] (MSB)	0x01	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x05	DID_B2
#4	scalingByte#1 {无符号数字，1个数据字节}	0x01	SBYT_1
#5	scalingByte#2 {公式，5个数据字节}	0x95	SB_2
#6	scalingByteExtension#2 [byte#1] {formulaIdentifier = C0 * x + C1}	0x00	SBE_21
#7	scalingByteExtension#2 [byte#2] {C0高字节}	0xE0	SBE_22
#8	scalingByteExtension#2 [byte#3] {C0低字节} [C0 = 75 * 10P-2P]	0x4B	SBE_23
#9	scalingByteExtension#2 [byte#4] {C1高字节}	0x00	SBE_24
#10	scalingByteExtension#2 [byte#5] {C1 low byte} [C1 = 30 * 10P0P]	0x1E	SBE_25
#11	scalingByte#3 {单位/格式，1个数据字节}	0xA1	SB_3
#12	scalingByteExtension#3 [byte#1] {unit ID, km / h}	0x30	SBE_31

使用C.2中包含的信息对scalingBytes，常量（C0，C1）和单位进行解码，车辆速度的数据变量使用以下公式计算：

车速= (0.75 * x + 30) km / h

其中“x”是存储在服务器中的实际数据，由dataIdentifier 0x0105标识。

10.4.5.4　Example#3: readScalingDataByIdentifier wth dataIdentifier 0x0967

此示例显示客户端如何确定服务器中dataIdentifier支持哪些位，该服务器被格式化为报告没有有效性掩码的位映射记录。

示例dataIdentifier（0x0967）在表171中定义。

表171 – 示例数据定义

数据字节	位 (S)	描述
	7-4	Unusued
	3	中速风扇正在启动
	2	检测到中速风扇输出故障
	1	清除监视器保持时间状态标志
	0	清除显示器闲置测试由于加油事件而被阻止
#2	7	检查燃油箱盖灯是否已启动
	6	检查检测到燃油箱盖光输出故障
	5	风扇控制检测到输出故障
	4	检测到风扇控制B输出故障
	3	检测到高速风扇输出故障
	2	指示高速风扇输出
	1	清除显示器闲置测试（小泄漏）准备运行
	0	吹扫监视器小泄漏已被监视

表172定义了ReadScalingDataByIdentifier请求消息流程示例#3。

表172 – ReadScalingDataByIdentifier请求消息流程示例#3

消息方向		客户端　服务器	
消息类型		请求	
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadScalingDataByIdentifier请求SID	0x24	RSDBI
#2	dataIdentifier [byte#1] (MSB)	0x09	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x67	DID_B2

表173定义了ReadScalingDataByIdentifier肯定响应消息流程示例#3。

表173 – ReadScalingDataByIdentifier正响应消息流程示例#3

消息方向		服务器　客户端	
消息类型		响应	
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadScalingDataByIdentifier响应SID	0x64	RSDBIPR
#2	dataIdentifier [byte#1] (MSB)	0x09	DID_HB
#3	dataIdentifier [byte#2] (LSB)	0x67	DID_LB
#4	scalingByte#1 {bitMappedReportedWithOutMask, 2个数据字节}	0x22	SBYT_1
#5	scalingByteExtension#1 [byte#1] {dataRecord#1有效性掩码}	0x03	SBYE_11
#6	scalingByteExtension#1 [byte#2] {dataRecord#2有效性掩码}	0x43	SBYE_12

上面的例子假定服务器中唯一支持这个数据标识符的位（即包含信息）是字节#1，位1和0，字节#2，位6,1和0。

10.5 ReadDataByPeriodicIdentifier（0x2A）服务

10.5.1 服务说明

ReadDataByPeriodicIdentifier服务允许客户端请求从一个或多个periodicDataIdentifiers标识的服务器定期传输数据记录值。

客户端请求消息包含一个或多个1字节periodicDataIdentifier值，用于标识服务器维护的数据记录。periodicDataIdentifier代表为此服务保留的dataIdentifier范围中的dataIdentifier的低位字节（0xF2XX，请参阅C.1中的允许的periodicDataIdentifier值），
例如此服务中使用的periodicDataIdentifier 0xE3是dataIdentifier 0xF2E3。

dataRecord的格式和定义应该是车辆制造商特定的，并且如果服务器支持，可以包括模拟输入和输出信号，数字输入和输出信号，内部数据和系统状态信息。

在接收到除停止之外的ReadDataByPeriodicIdentifier请求时，发送服务器应检查条件是否正确以执行服务。

periodicDataIdentifier只能在给定时间支持单个传输模式。 在接收到一个请求消息时，应该改变periodicDataIdentifier的调度，其中transmissionMode参数设置为同一个periodicDataIdentifier的新调度。根据车辆制造商的要求，不同的periodicDataIdentifiers的多个时间表应得到支持。

重要 - 如果条件正确，则服务器应发送肯定响应消息，仅包括服务标识符。 一旦接受到初始请求消息，服务器就不会通过积极响应来传输否定响应消息。

在初始肯定响应消息之后，服务器应该访问由periodicDataIdentifier参数指定的记录的数据元素，并且在包含关联的dataRecord参数的每个periodicDataIdentifier的单独的周期性数据响应消息中传送它们的值。

定义为在初始肯定响应消息之后向客户端传输periodicDataIdentifier数据的单独的周期性数据响应消息应包括periodicDataIdentifier和periodicDataIdentifier的数据，但不包括肯定响应服务标识符。 定期响应消息到特定数据链路层的映射在ISO 14229的适当实现规范中描述。

当仅调度一个periodicDataIdentifier时，特定传输模式的记录的周期率被定义为具有相同periodicDataIdentifier的任何两个连续响应消息之间的时间。 如果同时调度多个periodicDataIdentifiers，则相同periodicDataIdentifier之间的有效时间将根据以下设计参数而变化：

— 周期性调度器的呼叫率，

— 每个调度器调用分配的可用协议特定周期性数据响应消息地址信息ID的数量（例如，CAN上的CAN标识符）

— 可以并行定义的periodicDataIdentifiers的数量可以同时传输。

如果同时传输多个periodicDataIdentifer，这些参数值将影响相同periodicDataIdentifier之间的有效期限将增加多少。 因此，所有前面提到的设计参数应由车辆制造商指定。 每次调用周期性调度程序时，它应确定是否有任何periodicDataIdentifiers准备好传输。

注意注意，周期率是周期性调度程序调用率的整数倍。

例如，两种不同的ECU实现都可以支持具有10ms的周期速率的快速传输模式和单个唯一的周期性数据响应消息地址信息ID。 如果第一个实现每10ms调用周期性调度器，则当调度两个periodicDataIdentifiers时，相同periodicDataIdentifier之间的时间将增加到20 ms，并且在调度四个periodicDataIdentifiers时将增加到40 ms。 如果第二个实现每5毫秒调用一次周期性调度器，那么当调度两个periodicDataIdentifiers时，相同periodicDataIdentifier之间的时间将保持在10 ms，并且在调度四个periodicDataIdentifiers时将增加至20 ms。 请参阅10.6.5中的更多示例。

在接收到包含传输模式停止的ReadDataByPeriodicIdentifier请求时，如果请求消息中没有指定特定的数据，发送服务器应停止请求消息中包含的periodicDataIdentifier的周期性传输，或停止传输所有periodicDataIdentifier。 该传输模式的响应消息仅包含服务标识符。

服务器可能会限制车辆制造商和系统供应商同意支持的periodicDataIdentifiers的数量。 超过可同时支持的periodicDataIdentifier的最大数量应导致一个否定的响应，并且该请求中的任何periodicDataIdentifiers都不应该被调度。 在单个请求消息中重复使用相同的periodicDataIdentifier是不允许的，并且如果客户端违反了这个规则，服务器将忽略除了一个periodicDataIdentifer以外的所有其他消息。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

10.5.2 请求消息

10.5.2.1 请求消息定义

表174定义了请求消息。

表174 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDataByPeriodicIdentifier请求SID	M	0x2A	RDBPI
#2	transmissionMode	M	0x00 – 0xFF	TM值
#3	periodicDataIdentifier []#1	C	0x00 – 0xFF	PDID1
:	:	:	:	:
#m+2	periodicDataIdentifier []#m的	U	0x00 – 0xFF	PDIDm
C: 如果传输模式等于sendAtSlowRate，sendAtMediumRate或sendAtFastRate，则第一个periodicDataIdentifier必须存在于请求消息中。 在传输模式等于stopSending的情况下，可以不存在periodicDataIdentifier以便停止所有调度的periodicDataIdentifier，或者客户端可以显式指定一个或多个periodicDataIdentifier (s) 被停止。				

10.5.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

10.5.2.3 请求消息数据参数定义

表175定义了请求消息的数据参数。

表175 – 请求消息数据参数定义

定义
transmissionMode 此参数标识服务器使用的请求的periodicDataIdentifiers的传输速率（请参阅C.4）。
periodicDataIdentifier (#1至#m) 此参数标识客户端请求的服务器数据记录（有关详细参数定义，请参阅上面的C.1和服务描述）。 应该可以通过一个请求请求多个periodicDataIdentifiers。

10.5.3 积极的回应消息

10.5.3.1 积极响应消息的定义

它必须区分初始肯定响应消息，该消息指示服务器接受服务以及随后的周期性数据响应消息，其中包括periodicDataIdentifier数据。

表176定义了当服务器接受请求时要发送的初始肯定响应消息。

表176 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDataByPeriodicIdentifier响应SID	M	0x6A	RDBPIPR

周期性数据标识符的数据以由请求的传输模式参数确定的速率周期性发送（具有更新的数据）。

在最初的肯定响应之后，对于请求中的每个支持的periodicDataIdentifier，服务器将开始发送如下定义的单个周期性数据响应消息。

表177定义了周期性数据响应消息数据定义。

表177 – 定期数据响应消息数据定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	periodicDataIdentifier	M	0x00 – 0xFF	PDID
#2 : #k+2	dataRecord[] = [数据#1 : 数据#k]	M : U	0x00 – 0xFF : 0x00 – 0xFF	DREC_ DATA_1 : DATA_k

10.5.3.2 肯定回复消息数据参数定义

此服务不支持肯定响应消息中的响应消息数据参数。 表178定义了定义的周期性数据响应消息的周期性消息数据参数。

表178 – 定期消息数据参数定义

定义
periodicDataIdentifier 该参数引用请求消息中的periodicDataIdentifier。
dataRecord ReadDataByPeriodicIdentifier肯定响应消息使用此参数向客户端提供请求的数据记录值。 dataRecord的内容未在本文档中定义，并且是车辆制造商特定的。

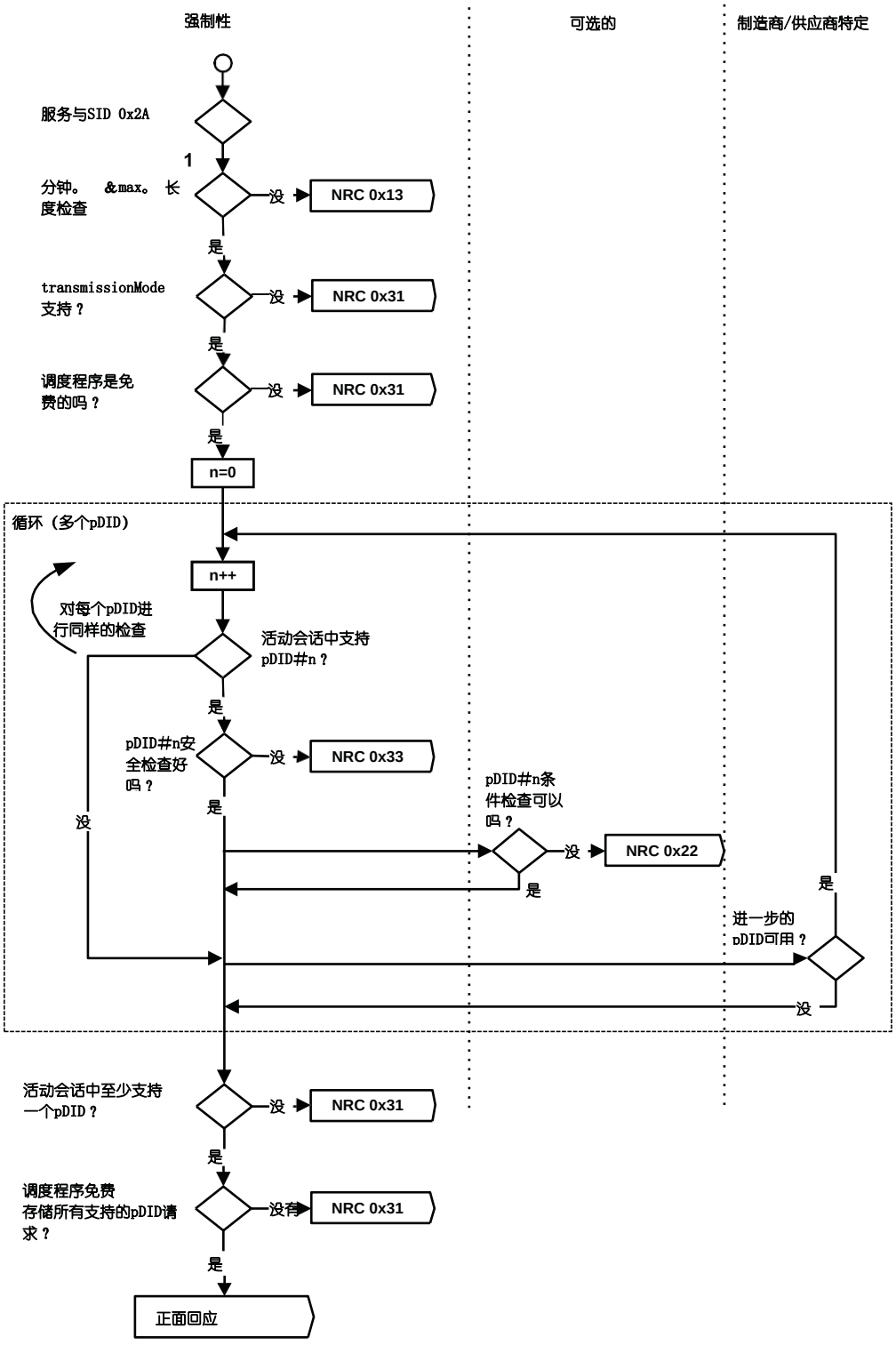
10.5.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 表179中记录了每个响应代码发生的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表179 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果请求消息的长度无效或客户端一次超过允许请求的最大数量的periodicDataIdentifiers，则应发送此NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果服务器的运行条件不符合要求，则应发送NRC。 例如，如果客户端请求具有不同传输模式的periodicDataIdentifiers并且服务器不同时支持多个传输模式，则可能发生这种情况。	CNC
0x31	**requestOutOfRange** 这个NRC将被发送，如果 设备不支持所请求的periodicDataIdentifier值；当前会话中不支持请求的 periodicDataIdentifiers；设备不支持指定的传输模式； 请求的dynamicDefinedDataIdentifier尚未分配； 客户端超出了允许同时调度的periodicDataIdentifiers的最大数量	ROOR
0x33	**securityAccessDenied** 如果至少有一个periodicDataIdentifier受到保护且服务器未处于解锁状态，则应发送此NRC。	伤心

评估顺序记录在图18中。



键

1 如果TM <>停止发送，则最小长度为3字节（SI + TM + pDID），如果TM =停止发送（SI + TM），则最小长度为2字节（最大长度为1字节（SI）+ 1字节（TM）+ n字节PDID（S））

图18 - ReadDataByPeriodicIdentifier服务的NRC处理

10.5.5 消息流示例ReadDataByPeriodicIdentifier

10.5.5.1 假设

以下示例显示了ReadDataByPeriodicIdentifer的行为。 客户端可以随时请求一个periodicDataIdentifier数据，与服务器的状态无关。

下面的periodicDataIdentifier示例是动力总成设备（例如，发动机控制模块）特有的。 有关排放相关系统的公认术语/定义/首字母缩略词的更多详细信息，请参阅ISO 15031-2 [6]。

10.5.5.2 示例#1 - 以中速读取多个periodicDataIdentifiers 0xE3和0x24

10.5.5.2.1 假设

该示例演示了使用单个请求（其中periodicDataIdentifier 0xE3 (= dataIdentifier 0xF2E3) 包含发动机冷却液温度，节气门位置，发动机转速，车辆速度传感器和periodicDataIdentifier 0x24 (= dataIdentifier 0xF224)）包含电池正极电压，歧管绝对值压力，质量空气流量，车辆气压和计算的负载值）。

客户端以中等速率请求传输，并且在检索周期性数据一段时间之后，客户端仅停止传输periodicDataIdentifier 0xE3。

10.5.5.2.2 步骤#1：请求定期传输periodicDataIdentifiers

表180定义了ReadDataByPeriodicIdentifier请求消息流程示例 - 步骤#1。

表180 - ReadDataByPeriodicIdentifier请求消息流程示例 - 步骤#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByPeriodicIdentifier请求SID	0x2A	RDBPI
#2	transmissionMode = sendAtMediumRate	0x02	TM_SAMR
#3	periodicDataIdentifier#1	0xE3	PDID1
#4	periodicDataIdentifier#2	0x24	PDID2

表181定义了ReadDataByPeriodicIdentifier初始肯定响应消息流程示例 - 步骤#1。

表181 - ReadDataByPeriodicIdentifier初始肯定响应消息流程示例 - 步骤#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByPeriodicIdentifier响应SID	0x6A	RDBPIPR

表182定义了ReadDataByPeriodicIdentifier子顺序肯定响应消息#1的流程 - 步骤#1。

表182 - ReadDataByPeriodicIdentifier次序肯定响应消息#1流程 - 步骤#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	periodicDataIdentifier#1	0xE3	PDID1
#2	dataRecord [data#1] = ECT	0xA6	DREC_DATA_1
#3	dataRecord [data#2] = TP	0x66	DREC_DATA_2
#4	dataRecord [data#3] = RPM	0x07	DREC_DATA_3
#5	dataRecord [data#4] = RPM	0x50	DREC_DATA_4
#6	dataRecord [data#5] = VSS	0x00	DREC_DATA_5

表183定义了ReadDataByPeriodicIdentifier子顺序肯定响应消息#2流程 - 步骤#1。

表183 - ReadDataByPeriodicIdentifier次后肯定响应消息#2流程 - 步骤#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	periodicDataIdentifier#2	0x24	PDID2
#2	dataRecord [data#1] = B +	0x8C	DREC_DATA_1
#3	dataRecord [data#2] = MAP	0x20	DREC_DATA_2
#4	dataRecord [data#3] = MAF	0x1A	DREC_DATA_3
#5	dataRecord [data#4] = BARO	0x63	DREC_DATA_4
#6	dataRecord [data#5] = LOAD	0x4A	DREC_DATA_5

服务器以适用于服务器的中等速率发送上述所示的后续响应消息。

10.5.5.2.3 步骤#2：停止periodicDataIdentifiers的传输

表184定义了ReadDataByIdentifier请求消息流程示例 - 步骤#2。

表184 - ReadDataByIdentifier请求消息流程示例 - 步骤#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByPeriodicIdentifier请求SID	0x2A	RDBPI
#2	transmissionMode = stopSending	0x04	TM_SS
#3	periodicDataIdentifier#1	0xE3	PDID

表185定义了ReadDataByIdentifier肯定响应消息流程示例 - 步骤#2。

表185 - ReadDataByIdentifier积极响应消息流程示例 - 步骤#2

消息方向		服务器 客户端	
消息类型		响应	
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByPeriodicIdentifier响应SID	0x6A	RDBPIPR

服务器仅停止periodicDataIdentifier 0xE3的传输。 periodicDataIdentifier 0x24仍以服务器中等速率传输。

10.5.5.3 示例#2 - ReadDataByPeriodicIdentifier服务定期计划费率的图形和表格示例

10.5.5.3.1 ReadDataByPeriodicIdentifier示例概述

本小节包含一个计划周期性数据的例子，其中包含ReadDataByPeriodicIdentifier (0x2A) 服务的图形和表格示例。

该示例包含图形描述了客户端和服务器应用程序之间传输什么消息（请求/响应），后面跟着一张表，其中显示了服务器周期性调度程序的可能实现，其变量以及它们在每次后台功能时如何改变检查周期性调度程序是否被执行。

在下面的例子中，定义了以下实现:

— 快速周期率为25毫秒，中等周期率为300毫秒。

— 周期性调度程序每12.5 ms检查一次，这意味着周期性调度程序后台函数在此期间被调用（轮询）。 每次调用后台周期性调度程序时，它都会遍历调度程序条目，直到发送一个周期性标识符，或者直到调度程序中的所有标识符都被检查并且没有任何准备好传输。 在示例实现中，表中的“周期性调度器传输索引”变量是在遍历调度器以查看标识符是否准备好传输时检查的第一个索引。

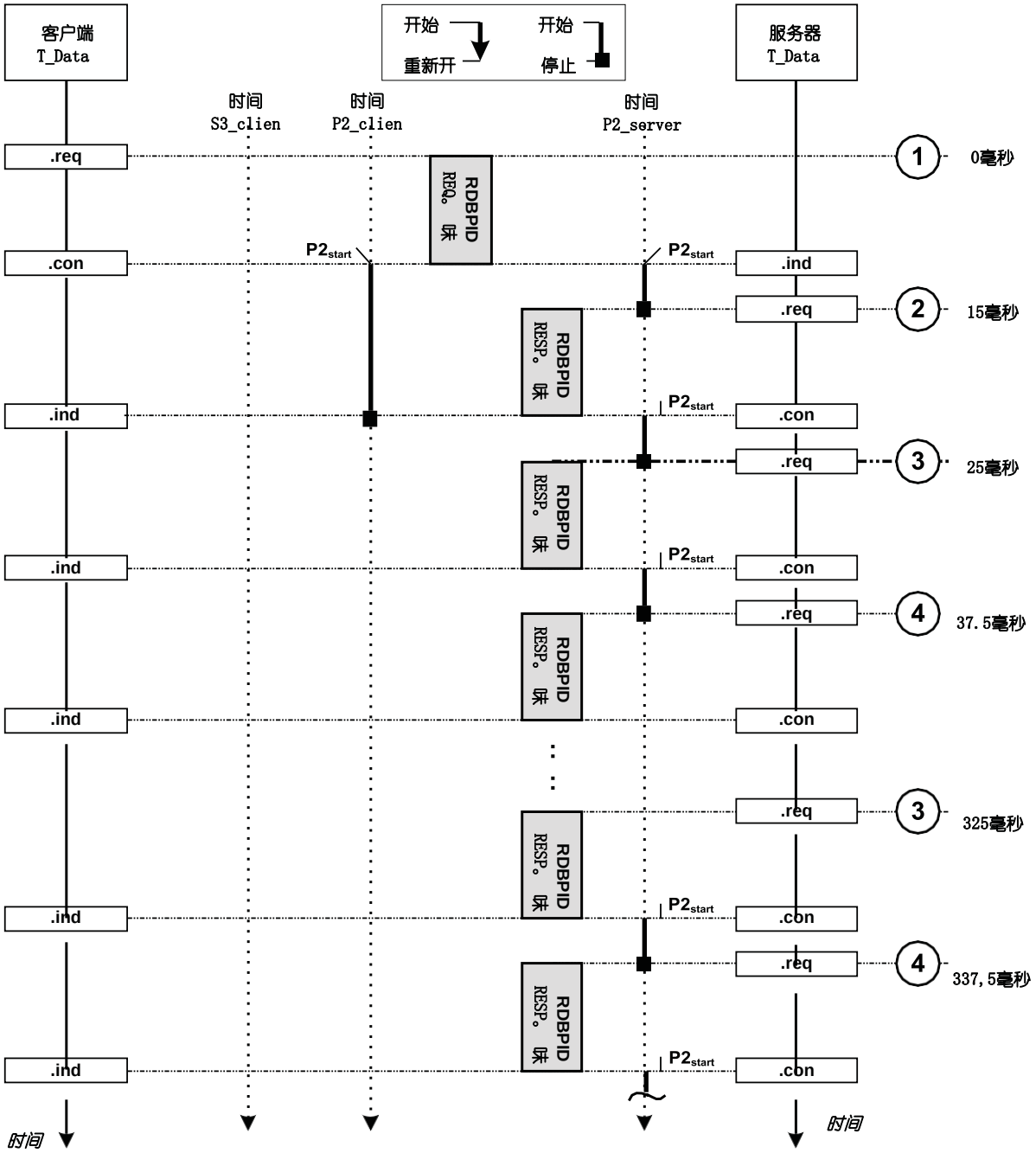
— 可同时调度的periodicDataIdentifiers的最大数量是4。

— 分配一个唯一的周期性数据响应消息地址信息ID。

由于周期性调度器轮询速率为12.5 ms，因此快速速率循环计数器将设置为2（该值基于调度速率 (25 ms) 除以周期性调度程序轮询速率 (12.5 ms) 或25 / 12,5），每次发送一个快速率periodicDataIdentifier并且每次中等速率periodicDataIdentifier时，中等速率循环计数器将被重置为24（预定速率除以周期性调度器轮询速率或300 / 12,5）已发送。

10.5.5.3.2 示例#2 - 以中等速率读取多个periodicDataIdentifiers 0xE3和0x24

在t = 0毫秒时，客户端开始发送请求，以中等速率（300毫秒）调度2个periodicDataIdentifiers（0xF2E3和0xF224）。 为了本示例的目的，服务器在第一次t = 25.0 ms时接收请求并执行周期性调度程序后台功能。



键

1 ReadDataByPeriodicIdentifier (0x2A, 0x02, 0xF2E3, 0xF224) 请求消息 (sendAtMediumRate)
2 ReadDataByPeriodicIdentifier肯定响应消息 (0x6A, 不包含数据)
3 ReadDataByPeriodicIdentifier周期性数据响应消息 (0xE3, 0xXX, ..., 0xXX)
4 ReadDataByPeriodicIdentifier周期性数据响应消息 (0x24, 0xXX, ..., 0xXX)

图19 - 示例#2 - 以中等速率调度的periodicDataIdentifiers

表186显示了服务器中的周期性调度程序的可能实现。 该表包含周期性调度程序变量以及每次执行检查周期性调度程序的后台函数时它们如何更改。

表186 – 示例2：定期调度程序表

时间 t 女士	周期性调度程序传输索引	定期发送标识符	定期调度程序循环#	调度器[0]传输计数	调度器[1]传输计数
25,0	0	0xE3	1	0->24	0
37,5	1	0x24	2	23	0->24
50,0	0	没有	3	22	23
62,5	0	没有	4	21	22
75,0	0	没有	5	20	21
87,5	0	没有	6	19	20
100,0	0	没有	7	18	19
112,5	0	没有	8	17	18
125,0	0	没有	9	16	17
137,5	0	没有	10	15	16
150,0	0	没有	11	14	15
162,5	0	没有	12	13	14
175,0	0	没有	13	12	13
187,5	0	没有	14	11	12
200,0	0	没有	15	10	11
212,5	0	没有	16	9	10
225,0	0	没有	17	8	9
237,5	0	没有	18	7	8
250,0	0	没有	19	6	7
262,5	0	没有	20	5	6
275,0	0	没有	21	4	5
287,5	0	没有	22	3	4
300,0	0	没有	23	2	3
312,5	0	没有	24	1	2
325,0	0	0xE3	25	0->24	1
337,5	1	0x24	26	23	0->24
350,0	0	没有	27	22	23
362,5	0	没有	28	21	22

10.5.5.4 示例#3 - ReadDataByPeriodicIdentifier服务定期计划费率的图形和表格示例

10.5.5.4.1 ReadDataByPeriodicIdentifier示例概述

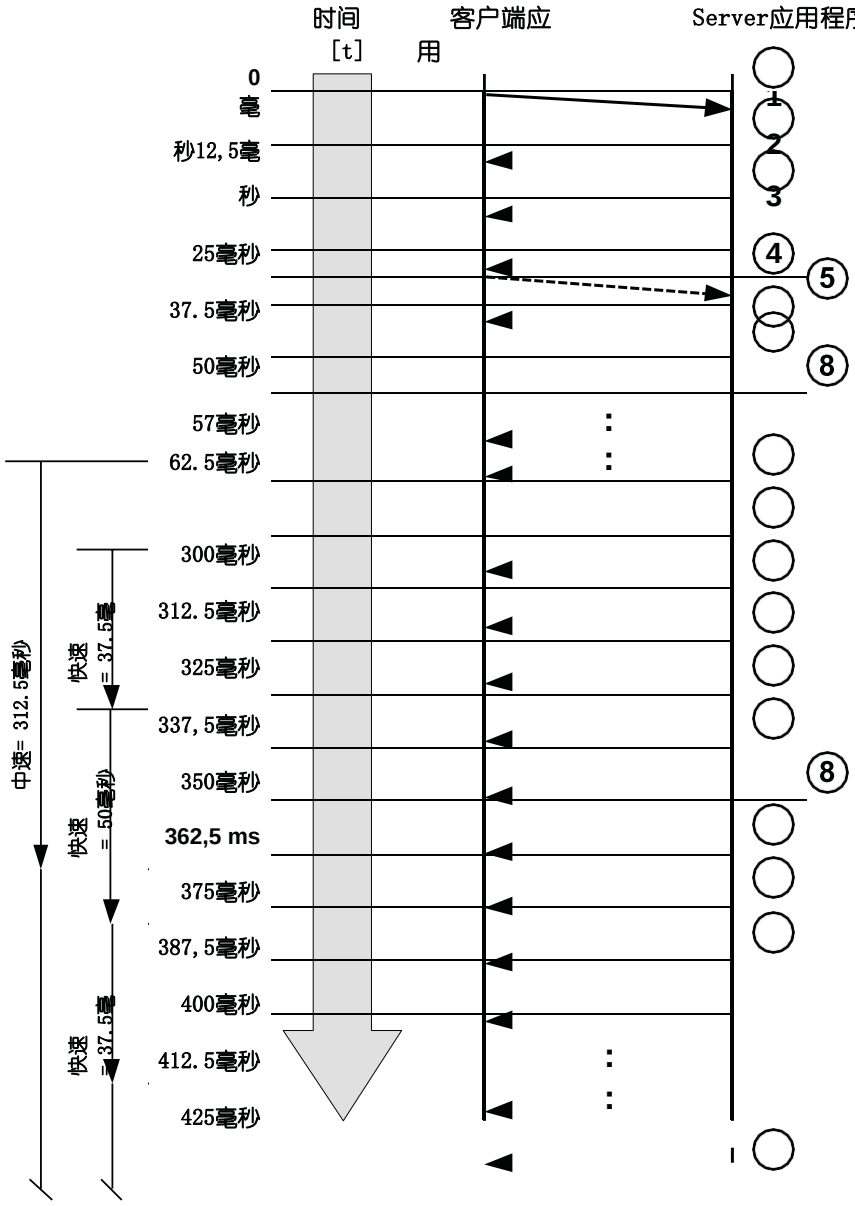
本小节包含一个计划周期性数据示例，其中包含ReadDataByPeriodicIdentifier (0x2A) 服务的图形和表格示例。

该示例基于10.5.5.3中给出的示例。 该示例包含图形描述了客户端和服务器应用程序之间传输什么消息（请求/响应），后面跟着一张表，其中显示了服务器周期性调度程序的可能实现，其变量以及它们在每次后台功能时如何改变检查周期性调度程序是否被执行。

10.5.5.4.2 以不同的周期性读取多个periodicDataIdentifiers

在此示例中，以快速周期速率（25 ms）调度三个periodicDataIdentifiers（为简单起见0x01,0x02,0x03），然后发送另一个请求发送单个periodicDataIdentifier（0x04）以中等周期速率（300 ms ）。 为了这个例子的目的，服务器接收到第一个ReadDataByPeriodicIdentifier请求（1），发送一个没有任何周期性数据的肯定响应（2），并且第一次执行周期性调度器后台函数t = 25,0ms（3）。 当收到第二个ReadDataByPeriodicIdentifier请求（5）时，服务器发送一个没有任何周期性数据的肯定响应（7），并开始以t = 62.5ms（8）以300ms的中间速率执行周期性调度器后台功能。

图20描述了以快速和中等速率调度的示例#3 - periodicDataIdentifiers。



键

1 ReadDataByPeriodicIdentifier (0x2A, 0x03, 0xF201, 0xF202, 0xF203) 请求消息 (sendAtFastRate)
2 ReadDataByPeriodicIdentifier肯定响应消息 (0x6A, 不包含数据)
3 ReadDataByPeriodicIdentifier周期性数据响应消息 (0x01, 0xXX, ..., 0xXX)
4 ReadDataByPeriodicIdentifier周期性数据响应消息 (0x02, 0xXX, ..., 0xXX)
5 ReadDataByPeriodicIdentifier (0x2A, 0x02, 0xF204) 请求消息 (sendAtMediumRate)
6 ReadDataByPeriodicIdentifier周期性数据响应消息 (0x03, 0xXX, ..., 0xXX)
7 ReadDataByPeriodicIdentifier肯定响应消息 (0x6A, 不包含数据)
8 ReadDataByPeriodicIdentifier周期性数据响应消息 (04, 0xXX, ..., 0xXX)

图20 – 示例#3 – 以快速和中等速率调度的periodicDataIdentifiers

表187显示了服务器中的周期性调度程序的可能实现。 该表包含周期性调度程序变量以及每次执行检查周期性调度程序的后台函数时它们如何更改。

表187 – 示例#3：定期调度程序表

时间 t 女士	周期性调度器传输索引	定期发送标识符	定期调度程序循环#	调度器[0]传输计数	调度器[1]传输计数	调度器[2]传输计数	调度器[3]传输计数
25,0	0	0x01	1	0->2	0	0	N/A
37,5	1	0x02	2	1	0->2	0	N/A
50,0	2	0x03	3	0	1	0->2	0
62,5	3	0x04	4	0	0	1	0->24
75,0	0	0x01	5	0->2	0	0	23
87,5	1	0x02	6	1	0->2	0	22
100,0	2	0x03	7	0	1	0->2	21
112,5	3	0x01	8	0->2	0	1	20
125,0	1	0x02	9	1	0->2	0	19
137,5	2	0x03	10	0	1	0->2	18
150,0	3	0x01	11	0->2	0	1	17
162,5	1	0x02	12	1	0->2	0	16
175,0	2	0x03	13	0	1	0->2	15
187,5	3	0x01	14	0->2	0	1	14
200,0	1	0x02	15	1	0->2	0	13
212,5	2	0x03	16	0	1	0->2	12
225,0	3	0x01	17	0->2	0	1	11
237,5	1	0x02	18	1	0->2	0	10
250,0	2	0x03	19	0	1	0->2	9
262,5	3	0x01	20	0->2	0	1	8
275,0	1	0x02	21	1	0->2	0	7
287,5	2	0x03	22	0	1	0->2	6
300,0	3	0x01	23	0->2	0	1	5
312,5	1	0x02	24	1	0->2	0	4
325,0	2	0x03	25	0	1	0->2	3
337,5	3	0x01	26	0->2	0	1	2
350,0	1	0x02	27	1	0->2	0	1
362,5	2	0x03	28	0	1	0->2	0
375,0	3	0x04	29	0	0	1	0->24
387,5	0	0x01	30	0->2	0	0	23

10.5.5.5 示例#4 – ReadDataByPeriodicIdentifier服务定期计划比率的表格示例

10.5.5.5.1 ReadDataByPeriodicIdentifier示例概述

本小节包含一个带有ReadDataByPeriodicIdentifier (0x2A) 服务表格例子的预定周期数据的例子。 该示例包含一个显示可能的表格

服务器周期性调度程序的实现，其变量以及它们在每次执行检查周期性调度程序的后台功能时如何改变。

在下面的例子中，定义了以下信息：

— 快速周期率是10毫秒。

— 周期性调度程序每10毫秒检查一次，这意味着周期性调度程序后台函数在此期间被调用（轮询）。

— 可同时调度的periodicDataIdentifiers的最大数目是16。

— 分配两个唯一的周期性数据响应消息地址信息ID。

由于周期性调度器轮询速率为10毫秒，所以快速速率循环计数器将设置为1（该值基于调度速率（10毫秒）除以周期性调度器轮询速率（10毫秒）），每次快速率periodicDataIdentifier被发送。

在t = 0毫秒时，客户端开始发送请求，以快速周期速率（10毫秒）调度2 periodicDataIdentifier（简单地说是0x01,0x02）。 为了本示例的目的，服务器在第一次t = 10 ms时收到请求并执行周期性调度程序后台功能。

表188 – 示例#4：定期调度程序表

时间t ms	响应消息ID#	定期标识符已发送	定期调度程序循环#
10	1	0x01	1
10	2	0x02	1
20	1	0x01	2
20	2	0x02	2
30	1	0x01	3
30	2	0x02	3
40	1	0x01	4
40	2	0x02	4
50	1	0x01	5
50	2	0x02	5
60	1	0x01	6
60	2	0x02	6
70	1	0x01	7
70	2	0x02	7
80	1	0x01	8
80	2	0x02	8
90	1	0x01	9
90	2	0x02	9
100	1	0x01	10
100	2	0x02	10

10.5.5.6 示例#5 - ReadDataByPeriodicIdentifier服务定期计划比率的表格示例

10.5.5.6.1 ReadDataByPeriodicIdentifier示例概述

本小节使用与10.5.5.5.1相同的假设。 在这个例子中，请求比响应消息集合中的唯一周期性数据响应消息地址信息ID更多的周期性数据标识符。

在t = 0毫秒时，客户端开始发送请求，以快速周期速率（10毫秒）调度3 periodicDataIdentifier（简单的0x01,0x02,0x03）。 为了本示例的目的，服务器在第一次t = 10 ms时收到请求并执行周期性调度程序后台功能。

表189 - 示例#5：周期性调度程序表

时间t ms	响应消息ID#	定期标识符已发送	定期调度程序循环#
10	1	0x01	1
10	2	0x02	1
20	1	0x03	2
20	2	0x01	2
30	1	0x02	3
30	2	0x03	3
40	1	0x01	4
40	2	0x02	4
50	1	0x03	5
50	2	0x01	5
60	1	0x02	6
60	2	0x03	6
70	1	0x01	7
70	2	0x02	7
80	1	0x03	8
80	2	0x01	8
90	1	0x02	9
90	2	0x03	9
100	1	0x01	10
100	2	0x02	10

10.6 DynamicallyDefineDataIdentifier (0x2C) 服务

10.6.1 服务说明

DynamicallyDefineDataIdentifier服务允许客户端动态地在服务器中定义一个数据标识符，以后可以通过ReadDataByIdentifier服务读取数据标识符。

此服务的目的是为客户提供将一个或多个数据元素分组到数据超集中的能力，该数据超集可通过ReadDataByIdentifier或ReadDataByPeriodicIdentifier服务集体请求。 要组合在一起的数据元素可以通过以下方式引用:

— 源数据标识符，位置和大小或

— 一个内存地址和一个内存长度，或者

— 上述两种方法的组合使用多个请求来定义单个数据元素。 动态定义的dataIdentifier将包含数据参数定义的串联。

此服务在处理超出可通过静态定义的数据标识符读取的信息的诊断应用程序的特定数据需求方面提供了更大的灵活性，还可用于通过避免与频繁的请求/响应事务相关的开销损失来减少带宽利用率。

动态定义的数据标识符的定义可以通过单个请求消息或通过多个请求消息来完成。 这允许定义一个数据元素引用源标识符和内存地址。 服务器必须连接单个数据元素的定义。 通过清除当前定义并重新定义新定义，可以重新定义动态定义的数据标识符。 当使用多个DynamicallyDefineDataIdentifier请求消息来配置单个DataIdentifier，并且服务器检测到对该DataIdentifier的后续请求（例如periodicDataIdentifier的定义）期间溢出的最大字节数时，服务器应将DataIdentifier的定义保留为它在请求之前会导致超限。

尽管此服务不禁止此类功能，但建议客户不要引用另一个动态定义的数据记录，因为删除引用的记录可能会在引用记录中创建数据一致性问题。

该服务还提供了清除现有动态定义的数据记录的功能。 如果指定的数据记录标识符在服务器支持的有效动态数据标识符的范围内（请参阅C.1了解更多详细信息），则应清除数据记录的请求应予以积极响应。

服务器应维护动态定义的数据记录，直到其被清除或由车辆制造商指定（例如，在会话转换或服务器掉电时删除动态定义的数据记录）。

服务器可以通过两种不同的方式实现数据记录:

— 包含多个未单独引用的基本数据记录的复合数据记录。

— 服务器内支持的单个数据记录（示例基本数据记录或DID是发动机转速或进气温度）的唯一2字节标识“标签”或数据标识符（DID）值。 数据记录的这种实现是复合数据记录实现的子集，因为它仅引用单个元素数据记录，而不是包含多个元素数据记录的数据记录。

DynamicallyDefineDataIdentifier服务支持这两种类型的实现数据记录以定义动态数据标识符。

— 组合数据块：位置参数必须引用组合数据块中的起点，并且大小参数必须反映要放置在动态定义的数据标识符中的数据长度。 测试人员有责任不在动态数据记录中仅包含数据复合数据块的一部分基本数据记录。

— 2字节DID：位置参数必须设置为1，尺寸参数必须反映DID的长度（元素数据记录的长度）。 测试人员有责任在动态数据记录中不包含2字节DID值的一部分。

动态定义数据记录中的数据排序应与客户端请求消息中指定的顺序相同。 此外，客户请求中指定的数据的第一个位置应按照前面提到的排序要求进行定位，使其最接近动态数据记录的开始位置。

除了通过逻辑参考（记录数据标识符）定义动态数据标识符之外，该服务还提供了通过绝对存储器地址和存储器长度信息来定义动态定义的数据标识符的能力。 这种定义动态数据标识符的机制建议仅在服务器的开发阶段使用。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

10.6.2 请求消息

10.6.2.1 请求消息定义

表190定义了请求消息 - 子函数= defineByIdentifier。

表190 - 请求消息定义 - 子功能= defineByIdentifier

A_Data字节	参数名称	Cvt	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	M	0x2C	DDDI
#2	子功能= [definitionType = defineByIdentifier]	M	0x01	LEV_ DBID
#3 #4	dynamicDefinedDataIdentifier [] = [字节#1 (MSB) 字节 #2 (LSB)]	M M	0xF2 / 0xF3 0x00 - 0xFF	DDDDI_ HB 磅
#5 #6	sourceDataIdentifier []#1 = [字节#1 (MSB) 字节 #2 (LSB)]	M M	0x00 - 0xFF 0x00 - 0xFF	SDI_ HB LB
#7	positionInSourceDataRecord#1	M	0x01 - 0xFF	PISDR#1
#8	memorySize#1	M	0x00 - 0xFF	MS#1
:	:	:	:	:
#n-3 #n-2	sourceDataIdentifier []#m = [字节#1 (MSB) 字节 #2 (LSB)]	U U	0x00 - 0xFF 0x00 - 0xFF	SDI_ HB LB
#n-1	positionInSourceDataRecord#m的	U	0x01 - 0xFF	PISDR#m的
#n	memorySize#m的	U	0x00 - 0xFF	MS#m的

表191定义了请求消息 - 子函数= defineByMemoryAddress。

表191 - 请求消息定义 - 子功能= defineByMemoryAddress

A_Data字节	参数名称	Cvt	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	M	0x2C	DDDI
#2	子功能= [definitionType = defineByMemoryAddress]	M	0x02	LEV_ DBMA
#3 #4	dynamicDefinedDataIdentifier [] = [字节#1 (MSB) 字节#2 (LSB)]	M M	0xF2 / 0xF3 0x00 - 0xFF	DDDDI_ HB 磅
#5	addressAndLengthFormatIdentifier	M1	0x00 - 0xFF	ALFID
#6 : #(m-1)+6	memoryAddress[] = [字节#1 (MSB) : 字节#m]	M : C1	0x00 - 0xFF : 0x00 - 0xFF	MA_ B1 : Bm
#m+6 : #M + 6 + (K-1)	memorySize[] = [字节#1 (MSB) : 字节#k]	M : C2	0x00 - 0xFF : 0x00 - 0xFF	MS_ B1 : Bk
:	:	:	:	:
#NK- (M-1) : #nk	memoryAddress[] = [字节#1 (MSB) : 字节#m]	U : U / C1	0x00 - 0xFF : 0x00 - 0xFF	MA_ B1 : Bm
#N- (K-1) : #n	memorySize[] = [字节#1 (MSB) : 字节#k]	U : U / C2	0x00 - 0xFF : 0x00 - 0xFF	MS_ B1 : Bk

M1: addressAndLengthFormatIdentifier参数仅在请求消息的最开始处出现一次，并在整个请求消息中为每个存储器位置引用定义地址和长度信息的长度。

C1: 该参数的存在取决于addressAndLengthFormatIdentifier的地址长度信息参数。

C2: 此参数的存在取决于addressAndLengthFormatIdentifier的内存大小长度信息。

表192定义了请求消息 - 子函数= clearDynamicallyDefinedDataIdentifier。

表192 - 请求消息定义 - 子功能= clearDynamicallyDefinedDataIdentifier

A_Data字节	参数名称	Cvt	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	M	0x2C	DDDI
#2	子功能= [definitionType = clearDynamicallyDefinedDataIdentifier]	M	0x03	LEV_ CDDDID
#3 #4	dynamicDefinedDataIdentifier [] = [字节#1 (MSB) 字节#2 (LSB)]	C C	0xF2 / 0xF3 0x00 - 0xFF	DDDDI_ HB LB

C: 该参数的存在要求服务器清除包含在字节#1和字节#2中的dynamicDefinedDataIdentifier。 如果该参数不存在，则应清除服务器中的所有dynamicDefinedDataIdentifier。

10.6.2.2 请求消息子函数参数$ Level (LEV_) 定义

定义为对该服务的请求消息有效的子参数在表193中指出 (suppressPosRspMsgIndicationBit (bit 7) 未示出)。

表193 - 请求消息子功能参数定义

位6 - 0	描述	Cvt	助记符
00	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
01	**defineByIdentifier** 该值应用于向服务器指定动态数据标识符的定义应通过数据标识符引用进行。	U	DBID
02	**defineByMemoryAddress** 该值应用于向服务器指定动态数据标识符的定义应通过地址引用进行。	U	DBMA
03	**clearDynamicallyDefinedDataIdentifier** 该值应用于清除指定的动态数据标识符。 请注意，即使请求时指定的动态数据标识符不存在，服务器也应对来自客户端的明确请求作出肯定响应。 但是，指定的动态数据标识符必须在有效范围内 (关于允许范围，请参阅C.1)。 如果在请求时定期报告指定的动态数据标识符，则应首先停止并清除动态标识符。	U	CDDDI
04-7F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD

10.6.2.3 请求消息数据参数定义

表194定义了请求消息的数据参数。

表194 - 请求消息数据参数定义

定义
dynamicallyDefinedDataIdentifier 此参数指定如何在将来调用服务ReadDataByIdentifier或ReadDataByPeriodicDataIdentifier时引用由客户机定义的动态数据记录。 dynamicDefinedDataIdentifier应该作为ReadDataByIdentifier服务中的dataIdentifier来处理 (更多细节见C.1)。 它应作为ReadDataByPeriodicDataIdentifier服务中的periodicRecordIdentifier进行处理 (请参阅ReadDataByPeriodicDataIdentifier服务以获取有关此参数值的要求，以便能够定期请求动态定义的数据标识符)。
sourceDataIdentifier 该参数仅存在于子函数= defineByIdentifier中。 此参数逻辑上指定要包含在动态数据记录中的信息源。 例如，这可以是用于参考发动机转速的2字节DID，也可以是用于引用包含发动机转速，车辆速度，进气温度等的复合信息块的2字节DID (参见C.1更多细节)。
positionInSourceDataRecord 该参数仅存在于子函数= defineByIdentifier中。 该1字节参数用于指定要包含在动态数据记录中的源数据记录摘录的起始字节位置。 一个位置应引用由sourceDataIdentifier引用的数据记录的第一个字节。

表194 - （续）

定义
addressAndLengthFormatIdentifier 该参数是一个单字节值，每个半字节分别进行编码（参见H.1的示例值）：bit 7 - 4: memorySize参数的长度（字节数）； 位3 - 0: memoryAddress参数的长度（字节数）；
memoryAddress 该参数仅对子功能= defineByMemoryAddress存在。 此参数指定要包含到动态数据记录中的信息的内存源地址。 用于该地址的字节数由addressAndLengthFormatIdentifier的低半字节（bit 3-0）定义。 memoryAddress参数中的字节#m始终是服务器中引用地址的最低有效字节。 地址的最高有效字节可用作存储器标识符。
memorySize 此参数用于指定要包含在动态数据记录中的源数据记录/内存地址的总字节数。 在子函数= defineByIdentifier的情况下，另外使用positionInSourceDataRecord参数以指定memorySize应用于的源数据标识符中的起始位置。 用于此大小的字节数是一个字节。 在子函数= defineByMemoryAddress的情况下，那么这个参数反映了从指定的memoryAddress开始在动态定义的数据标识符中包含的字节数。 用于此大小的字节数由addressAndLengthFormatIdentifier的高半字节（第7 - 4位）定义。

10.6.3 积极的回应消息

10.6.3.1 积极响应消息的定义

表195定义了肯定响应消息。

表195 - 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	M	0x6C	DDDIPR
#2	子函数= [definitionType]	M	0x00 – 0x7F	DM
#3 #4	dynamicDefinedDataIdentifier [] = [字节#1（MSB）字节#2（LSB）]	C C	0xF2 / 0xF3 0x00 - 0xFF	DDDDI_HB 磅
C: 如果请求消息中存在dynamicDefinedDataIdentifier参数，则此参数的存在是必需的，否则参数不应包括在内。				

10.6.3.2 肯定回复消息数据参数定义

表196定义了肯定响应消息的数据参数。

表196 - 响应消息数据参数定义

定义
definitionType 该参数是来自请求消息的子功能参数的比特6-0的回声。
dynamicallyDefinedDataIdentifier 此参数是来自请求消息的数据参数dynamicDefinedDataIdentifier的回显。

10.6.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 表197中记录了每个响应代码发生的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表197 – 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果服务器的运行条件不符合要求，则应发送NRC。	CNC
0x31	**requestOutOfRange** 如果发生以下情况，应发送NRC： — 请求消息中的任何数据标识符（动态定义的数据标识符或任何sourceDataIdentifier）不受支持/无效； — positionInSourceDataRecord不正确（小于1，或大于服务器允许的最大值）； — 服务器不支持请求消息中的任何内存地址。 — 指定的memorySize无效； — 要打包到动态数据标识符中的数据量超过服务器允许的最大数量； — 指定的addressAndLengthFormatIdentifier无效； — 动态定义的periodicDataIdentifier的总长度超过了适合用于传输周期性响应消息的数据链路的单个帧的最大长度；	ROOR
0x33	**securityAccessDenied** 如果发生以下情况，应发送NRC： — 请求消息中的任何数据标识符（动态定义的数据标识符或任何sourceDataIdentifier）都是安全的，并且服务器未处于解锁状态； — 请求消息中的任何内存地址都是安全的，并且服务器未处于解锁状态；	伤心

10.6.5 消息流示例DynamicallyDefineDataIdentifier

10.6.5.1 假设

这个 节 指定 该 条件 至 是 完成 对于 该 例 至 演出 一个DynamicallyDefineDataIdentifier服务。

本例中的服务不受服务器限制。

在第一个示例中，服务器支持引用单个数据信息的2字节标识符（DID）。 该示例使用defineByIdentifier方法构建动态数据标识符，然后发送ReadDataByIdentifier请求以读取刚配置的动态数据标识符。

在第二个例子中，服务器支持数据标识符，数据标识符引用包含多个数据信息的数据的组合数据块。 该示例还使用defineByIdentifier方法构建动态标识符，并发送ReadDataByIdentifier请求以读取刚定义的数据标识符。

第三个示例使用defineByMemoryAddress方法构建动态数据标识符，并发送ReadDataByIdentifier请求以读取刚刚定义的数据标识符。

在第四个例子中，服务器支持数据标识符，数据标识符引用包含多个数据信息的数据的组合块。 该示例使用defineByIdentifier方法构建动态数据标识符，然后使用ReadDataByPeriodicIdentifier服务请求动态定义的数据标识符由服务器定期发送。

第五个例子演示删除动态定义的数据标识符。

下面的例子将使用表198。 报道的数值可能会随着时间的推移而在实际车辆上发生变化，但为了清晰起见，它们显示为常数。

有关排放相关系统的公认术语/定义/首字母缩略词的更多详细信息，请参阅ISO 15031-2 [6]。

对于所有示例，客户端通过将suppressPosRspMsgIndicationBit（子函数参数的第7位）设置为“FALSE”（“0”）来请求获得响应消息。

表198 – 复合数据块 – 数据标识符定义

数据 标识符（块）	数据字节	数据记录内容	字节值
	#1	dataRecord [数据#1] = B+	0x8C
	#2	dataRecord [data#2] = ECT	0xA6
	#3	dataRecord [data#3] = TP	0x66
	#4	dataRecord [data#4] = RPM	0x07
	#5	dataRecord [数据#5] = RPM	0x50
	#6	dataRecord [数据#6] = MAP	0x20
	#7	dataRecord [数据#7] = MAF	0x1A
	#8	dataRecord [数据#8] = VSS	0x00
	#9	dataRecord [数据#9] = BARO	0x63
	#10	dataRecord [data#10] = LOAD	0x4A
	#11	dataRecord [数据#11] = IAC	0x82
	#12	dataRecord [data#12] = APP	0x7E
	#1	dataRecord [数据#1] = SPARKADV	0x00
	#2	dataRecord [data#2] = KS	0x91

表199定义了元素数据记录 – DID定义。

表199 – 元素数据记录 – DID定义

数据 标识符 (DID)	数据字节	数据记录内容	字节值
	#1	EOT (MSB)	0x4C
	#2	EOT (LSB)	0x36
0x5678	#1	AAT	0x4D
	#1	EOL (MSB)	0x49
	#2	EOL	0x21
	#3	EOL	0x00
	#4	EOL (LSB)	0x17

表200定义了存储器数据记录 – 存储器地址定义。

表200 – 存储器数据记录 – 存储器地址定义

内存地址	数据字节	数据记录内容	字节值
	#1	dataRecord [数据#1] = B+	0x8C
	#2	dataRecord [data#2] = ECT	0xA6
	#3	dataRecord [data#3] = TP	0x66
	#4	dataRecord [data#4] = RPM	0x07
	#5	dataRecord [数据#5] = RPM	0x50
	#6	dataRecord [数据#6] = MAP	0x20
	#7	dataRecord [数据#7] = MAF	0x1A
	#8	dataRecord [数据#8] = VSS	0x00
	#9	dataRecord [数据#9] = BARO	0x63
	#10	dataRecord [data#10] = LOAD	0x4A
	#11	dataRecord [数据#11] = IAC	0x82
	#12	dataRecord [data#12] = APP	0x7E
	#1	dataRecord [数据#1] = SPARKADV	0x00
	#2	dataRecord [data#2] = KS	0x91

10.6.5.2 示例#1: DynamicallyDefineDataIdentifier, 子函数= defineByIdentifier

表201中的示例将使用2字节的DID作为所需数据的参考，生成包含发动机机油温度，环境空气温度和发动机机油油位的动态数据标识符（DDDI 0xF301）。

表201 - DynamicallyDefineDataIdentifier请求DDDDI 0xF301消息流示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	0x2C	DDDI
#2	子函数= defineByIdentifier, suppressPosRspMsgIndicationBit = FALSE	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2
#5	sourceDataIdentifier#1 [byte#1] (MSB) - 发动机油温	0x12	SDI_B1
#6	sourceDataIdentifier#1 [byte#2]	0x34	SDI_B2
#7	positionInSourceDataRecord#1	0x01	PISDR#1
#8	memorySize#1	0x02	MS#1
#9	sourceDataIdentifier#2 [byte#1] (MSB) - 环境空气温度	0x56	SDI_B1
#10	sourceDataIdentifier#2 [byte#2]	0x78	SDI_B2
#11	positionInSourceDataRecord#2	0x01	PISDR#2
#12	memorySize#2	0x01	MS#2
#13	sourceDataIdentifier#3 [byte#1] (MSB) - 发动机油位	0x9A	SDI_B1
#14	sourceDataIdentifier#3 [byte#2]	0xBC	SDI_B2
#15	positionInSourceDataRecord#3	0x01	PISDR#3
#16	memorySize#3	0x04	MS#3

表202定义了DynamicallyDefineDataIdentifier肯定响应DDDDI 0xF301消息流程示例#1。

表202 - DynamicallyDefineDataIdentifier肯定响应DDDDI 0xF301消息流示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	0x6C	DDDIPR
#2	definitionMode = defineByIdentifier	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2

表203定义了ReadDataByIdentifier请求DDDDI 0xF301消息流程示例#1。

表203 - ReadDataByIdentifier请求DDDDI 0xF301消息流示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier请求SID	0x22	RDBI
#2	dataIdentifier [byte#1] (MSB)	0xF3	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x01	DID_B2

表204定义ReadDataByIdentifier肯定响应DDDDI 0xF301消息流程示例#1。

表204 - ReadDataByIdentifier肯定响应DDDDI 0xF301消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier响应SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	0xF3	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x01	DID_B2
#4	dataRecord [数据#1] = EOT	0x4C	DREC_DATA_1
#5	dataRecord [data#2] = EOT	0x36	DREC_DATA_2
#6	dataRecord [data#3] = AAT	0x4D	DREC_DATA_3
#7	dataRecord [data#4] = EOL	0x49	DREC_DATA_4
#8	dataRecord [data#5] = EOL	0x21	DREC_DATA_5
#9	dataRecord [data#6] = EOL	0x00	DREC_DATA_6
#10	dataRecord [data#7] = EOL	0x17	DREC_DATA_7

10.6.5.3 示例#2: DynamicallyDefineDataIdentifier, 子函数= defineByIdentifier

表205中的示例将生成一个动态数据标识符（DDDDI 0xF302），其中包含发动机冷却液温度（来自数据记录0x010A），发动机转速（来自数据记录0x010A），IAC Pintle位置（来自数据记录0x010A）和爆震传感器数据记录0x050B）。

表205 - DynamicallyDefineDataIdentifier请求DDDDI 0xF302消息流示例#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	0x2C	DDDI
#2	子函数= defineByIdentifier, suppressPosRspMsgIndicationBit = FALSE	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x02	DDDDI_B2
#5	sourceDataIdentifier#1 [byte#1] (MSB)	0x01	SDI_B1
#6	sourceDataIdentifier#1 [byte#2] (LSB)	0x0A	SDI_B2
#7	positionInSourceDataRecord#1 - 发动机冷却液温度	0x02	PISDR#1
#8	memorySize#1	0x01	MS#1
#9	sourceDataIdentifier#2 [byte#1] (MSB)	0x01	SDI_B1
#10	sourceDataIdentifier#2 [byte#2] (LSB)	0x0A	SDI_B2
#11	positionInSourceDataRecord#2 - 发动机转速	0x04	PISDR#2
#12	memorySize#2	0x02	MS#2
#13	sourceDataIdentifier#3 [byte#1] (MSB)	0x01	SDI_B1
#14	sourceDataIdentifier#3 [byte#2] (LSB)	0x0A	SDI_B2
#15	positionInSourceDataRecord#3 - 怠速空气控制	0x0B	PISDR#3
#16	memorySize#3	0x01	MS#3
#17	sourceDataIdentifier#4 [byte#1] (MSB)	0x05	SDI_B1
#18	sourceDataIdentifier#4 [byte#2] (LSB)	0x0B	SDI_B2
#19	positionInSourceDataRecord#4 - 爆震传感器	0x02	PISDR#4
#20	memorySize#4	0x01	MS#4

表206定义了DynamicallyDefineDataIdentifier肯定响应DDDDI 0xF302消息流程示例#2。

表206 - DynamicallyDefineDataIdentifier肯定响应DDDDI 0xF302消息流示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	0x6C	DDDIPR
#2	definitionMode = defineByIdentifier	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x02	DDDDI_B2

表207定义了ReadDataByIdentifier请求DDDDI 0xF302消息流示例#2。

表207 - ReadDataByIdentifier请求DDDDI 0xF302消息流示例#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier请求SID	0x22	RDBI
#2	dataIdentifier [byte#1] (MSB)	0xF3	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x02	DID_B2

表208定义ReadDataByIdentifier肯定响应DDDDI 0xF302消息流程示例#2。

表208 - ReadDataByIdentifier正响应DDDDI 0xF302消息流示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier响应SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	0xF3	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x02	DID_B2
#4	dataRecord [data#1] = ECT	0xA6	DREC_DATA_1
#5	dataRecord [data#2] = RPM	0x07	DREC_DATA_2
#6	dataRecord [data#3] = RPM	0x50	DREC_DATA_3
#7	dataRecord [data#4] = IAC	0x82	DREC_DATA_4
#8	dataRecord [data#5] = KS	0x91	DREC_DATA_5

10.6.5.4 示例#3: DynamicallyDefineDataIdentifier, 子函数= defineByMemoryAddress

表209中的示例将建立一个动态数据标识符（DDDDI 0xF302），其中包含发动机冷却液温度（从存储器地址0x21091969开始的存储器模块），发动机转速（从存储器地址0x2109196B开始的存储器模块）和爆震传感器（来自存储器块从内存地址0x13101995开始）。

表209 - DynamicallyDefineDataIdentifier请求DDDDI 0xF302消息流示例#3

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	0x2C	DDDI
#2	sub-function = defineByMemoryAddress, suppressPosRspMsgIndicationBit = FALSE	0x02	DBMA
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x02	DDDDI_B2
#5	addressAndLengthFormatIdentifier	0x14	ALFID
#6	memoryAddress#1 [byte#1] (MSB) - 发动机冷却液温度	0x21	MA_1_B1
#7	memoryAddress#1 [byte#2]	0x09	MA_1_B2

表209 - （续）

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#8	memoryAddress#1 [byte#3]	0x19	MA_1_B3
#9	memoryAddress#1 [byte#4]	0x69	MA_1_B4
#10	memorySize#1	0x01	MS#1
#11	memoryAddress#2 [byte#1] (MSB) - 发动机转速	0x21	MA_2_B1
#12	memoryAddress#2 [byte#2]	0x09	MA_2_B2
#13	memoryAddress#2 [byte#3]	0x19	MA_2_B3
#14	memoryAddress#2 [byte#4]	0x6B	MA_2_B4
#15	memorySize#2	0x02	MS#2
#16	memoryAddress#3 [byte#1] (MSB) - 爆震传感器	0x13	MA_3_B1
#17	memoryAddress#3 [byte#2]	0x10	MA_3_B2
#18	memoryAddress#3 [byte#3]	0x19	MA_3_B3
#19	memoryAddress#3 [byte#4]	0x95	MA_3_B4
#20	memorySize#3	0x01	MS#3

表210定义了DynamicallyDefineDataIdentifier肯定响应DDDDI 0xF302消息流程示例#3。

表210 - DynamicallyDefineDataIdentifier肯定响应DDDDI 0xF302消息流示例#3

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	0x6C	DDDIPR
#2	definitionMode = defineByMemoryAddress	0x02	DBMA
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x02	DDDDI_B2

表211定义了ReadDataByIdentifier请求DDDDI 0xF302消息流程示例#3。

表211 - ReadDataByIdentifier请求DDDDI 0xF302消息流示例#3

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier请求SID	0x22	RDBI
#2	dataIdentifier [byte#1] (MSB)	0xF3	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x02	DID_B2

表212定义了ReadDataByIdentifier肯定响应DDDDI 0xF302消息流程示例#3。

表212 - ReadDataByIdentifier肯定响应DDDDI 0xF302消息流示例#3

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier响应SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	0xF3	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x02	DID_B2
#4	dataRecord [data#1] = ECT	0xA6	DREC_DATA_1
#5	dataRecord [data#2] = RPM	0x07	DREC_DATA_2
#6	dataRecord [data#3] = RPM	0x50	DREC_DATA_3
#7	dataRecord [data#4] = KS	0x91	DREC_DATA_4

10.6.5.5 示例#4: DynamicallyDefineDataIdentifier, 子函数= defineByIdentifier

表213中的示例将生成包含发动机冷却液温度（来自数据记录0x010A），发动机转速（来自数据记录0x010A）和爆震传感器（来自数据记录0x050B）的动态数据标识符（DDDDI 0xF2E7）。

动态数据标识符的值在可用于周期性请求数据的范围之外选择。 在定义动态数据标识符之后，客户端请求定期发送数据标识符（快速）。

表213 - DynamicallyDefineDataIdentifier请求DDDDI 0xF2E7消息流示例#4

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	0x2C	DDDI
#2	子函数= defineByIdentifier, suppressPosRspMsgIndicationBit = FALSE	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF2	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0xE7	DDDDI_B2
#5	sourceDataIdentifier#1 [byte#1] (MSB)	0x01	SDI_B1
#6	sourceDataIdentifier#1 [byte#2] (LSB)	0x0A	SDI_B2
#7	positionInSourceDataRecord#1 - 发动机冷却液温度	0x02	PISDR
#8	memorySize#1	0x01	女士
#9	sourceDataIdentifier#2 [byte#1] (MSB)	0x01	SDI_B1
#10	sourceDataIdentifier#2 [byte#2] (LSB)	0x0A	SDI_B2
#11	positionInSourceDataRecord#2 - 发动机转速	0x04	PISDR
#12	memorySize#2	0x02	女士
#13	sourceDataIdentifier#3 [byte#1] (MSB)	0x05	SDI_B1
#14	sourceDataIdentifier#3 [byte#2] (LSB)	0x0B	SDI_B2
#15	positionInSourceDataRecord#3 - 爆震传感器	0x02	PISDR
#16	memorySize#3	0x01	女士

表214定义了DynamicallyDefineDataIdentifier正响应DDDDI 0xF2E7消息流示例#4。

表214 - DynamicallyDefineDataIdentifier肯定响应DDDDI 0xF2E7消息流示例#4

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	0x6C	DDDIPR
#2	definitionMode = defineByIdentifier	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF2	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0xE7	DDDDI_B2

表215定义了ReadDataByPeriodicIdentifier请求DDDDI 0xF2E7消息流示例#4。

表215 - ReadDataByPeriodicIdentifier请求DDDDI 0xF2E7消息流示例#4

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByPeriodicIdentifier请求SID	0x2A	RDBPI
#2	transmissionMode = sendAtFastRate	0x04	TM值
#3	PeriodicDataIdentifier	0xE7	PDID

表216定义了ReadDataByPeriodicIdentifier初始肯定消息流程示例#4。

表216 - ReadDataByPeriodicIdentifier初始肯定消息流示例#4

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByPeriodicIdentifier响应SID	0x6A	RDBPIPR

表217和表218定义ReadDataByPeriodicIdentifier周期性数据响应#1 DDDDI 0xF2E7消息流示例#4。

表217 - ReadDataByPeriodicIdentifier周期性数据响应#1 DDDDI 0xF2E7消息流示例#4

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	PeriodicDataIdentifier	0xE7	PDID
#2	dataRecord [data#1] = ECT	0xA6	DREC_DATA_1
#3	dataRecord [data#2] = RPM	0x07	DREC_DATA_2
#4	dataRecord [data#3] = RPM	0x50	DREC_DATA_3
#5	dataRecord [data#4] = KS	0x91	DREC_DATA_4

注意 本例中没有显示具有不同字节值的多个响应消息。

表218 - ReadDataByPeriodicIdentifier周期性数据响应#n DDDDI 0xF2E7消息流示例#4

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	periodicDataIdentifier	0xE7	PDID
#2	dataRecord [data#1] = ECT	0xA6	DREC_DATA_1
#3	dataRecord [data#2] = RPM	0x07	DREC_DATA_2
#4	dataRecord [data#3] = RPM	0x55	DREC_DATA_3
#5	dataRecord [data#4] = KS	0x98	DREC_DATA_4

10.6.5.6 示例#5: DynamicallyDefineDataIdentifier, 子函数= clearDynamicallyDefined-DataIdentifier

表219中的示例演示了清除dynamicDefinedDataIdentifier, 并假定请求时存在DDDDI 0xF303。

表219 - DynamicallyDefineDataIdentifier请求清除DDDDI 0xF303消息流程示例#5

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	0x2C	DDDI
#2	子函数= clearDynamicallyDefinedDataIdentifier, suppressPosRspMsgIndicationBit = FALSE	0x03	CDDDI
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x03	DDDDI_B2

表220定义了DynamicallyDefineDataIdentifier肯定响应清除DDDDI 0xF303消息流示例#5。

表220 - DynamicallyDefineDataIdentifier肯定响应清除DDDDI 0xF303消息流示例#5

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	0x6C	DDDIPR
#2	definitionMode = clearDynamicallyDefinedDataIdentifier	0x03	CDDDI
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x03	DDDDI_B2

10.6.5.7 Example#6: DynamicallyDefineDataIdentifier, 连接定义 (defineByIdentifier / defineByAddress)

这个例子将使用这两种定义类型建立一个动态数据标识符（DDDI 0xF301）。 以下列表显示了动态定义的数据标识符中的数据顺序（定义动态数据标识符的请求消息的隐式顺序）：

— 1ST部分：发动机油温和环境空气温度通过2字节DID（defineByIdentifier）进行引用，

— 2ND部分：内存地址引用的发动机冷却液温度和发动机转速，

— 3RD部分：发动机机油油位由2字节DID引用。

10.6.5.7.1 步骤#1: DynamicallyDefineDataIdentifier, 子函数= defineByIdentifier（第一部分）

表221定义了DynamicallyDefineDataIdentifier请求DDDI 0xF301消息流示例#6定义的1ST部分（defineByIdentifier）。

表221 - DynamicallyDefineDataIdentifier请求DDDI 0xF301消息流示例#6定义为1ST部分 (defineByIdentifier)

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	0x2C	DDDI
#2	子函数= defineByIdentifier, suppressPosRspMsgIndicationBit = FALSE	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2
#5	sourceDataIdentifier#1 [byte#1] (MSB) - 发动机油温	0x12	SDI_B1
#6	sourceDataIdentifier#1 [byte#2]	0x34	SDI_B2
#7	positionInSourceDataRecord#1	0x01	PISDR#1

表221 - （续）

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#8	memorySize#1	0x02	MS#1
#9	sourceDataIdentifier#2 [byte#1] (MSB) - 环境空气温度	0x56	SDI_B1
#10	sourceDataIdentifier#2 [byte#2] (LSB)	0x78	SDI_B2
#11	positionInSourceDataRecord#2	0x01	PISDR#2
#12	memorySize#2	0x01	MS#2

表222定义了第一部分（defineByIdentifier）的DynamicallyDefineDataIdentifier肯定响应DDDI 0xF301消息流示例#6定义。

表222 - DynamicallyDefineDataIdentifier正响应DDDI 0xF301消息流示例第一部分的#6定义（defineByIdentifier）

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	0x6C	DDDIPR
#2	definitionMode = defineByIdentifier	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2

10.6.5.7.2　步骤2: DynamicallyDefineDataIdentifier, 子函数= defineByMemoryAddress（第2部分）

表223定义了DynamicallyDefineDataIdentifier请求DDDDI 0xF301消息流示例#6定义了2ND部分（defineByMemoryAddress）。

表223 - DynamicallyDefineDataIdentifier请求DDDDI 0xF301消息流示例#6定义2ND部分 (defineByMemoryAddress)

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	0x2C	DDDI
#2	sub-function = defineByMemoryAddress, suppressPosRspMsgIndicationBit = FALSE	0x02	DBMA
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2
#5	addressAndLengthFormatIdentifier	0x14	ALFID
#6	memoryAddress#1 [byte#1] (MSB) - 发动机冷却液温度	0x21	MA_B1 # 1
#7	memoryAddress#1 [byte#2]	0x09	MA_B2 # 1
#8	memoryAddress#1 [byte#3]	0x19	MA_B3#1
#9	memoryAddress#1 [byte#4]	0x69	MA_B4#1
#10	memorySize#1	0x01	MS#1
#11	memoryAddress#2 [byte#1] (MSB) - 引擎速度memoryAddress#2	0x21	MA_B1 # 2
#12	[byte#2]	0x09	MA_B2 # 2
#13	memoryAddress#2 [byte#3]	0x19	MA_B3#2
#14	memoryAddress#2 [byte#4]	0x6B	MA_B4#2
#15	memorySize#2	0x02	MS#2

表222定义了DynamicallyDefineDataIdentifier肯定响应DDDI 0xF301消息流程示例#6。

表224 - DynamicallyDefineDataIdentifier正面响应DDDI 0xF301消息流示例 #6

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	0x6C	DDDIPR
#2	definitionMode = defineByMemoryAddress	0x02	DBMA
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2

10.6.5.7.3 步骤#3: DynamicallyDefineDataIdentifier, 子函数= defineByIdentifier (第3部分)

表225定义了DynamicallyDefineDataIdentifier请求DDDI 0xF301消息流示例#6定义的3RD部分 (defineByIdentifier)。

表225 - DynamicallyDefineDataIdentifier请求DDDI 0xF301消息流示例#6定义3RD部分 (defineByIdentifier)

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	0x2C	DDDI
#2	子函数= defineByIdentifier, suppressPosRspMsgIndicationBit = FALSE	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2
#5	sourceDataIdentifier#1 [byte#1] (MSB) - 发动机油位	0x9A	SDI_B1
#6	sourceDataIdentifier#1 [byte#2]	0xBC	SDI_B2
#7	positionInSourceDataRecord#1	0x01	PISDR#3
#8	memorySize#1	0x04	MS#3

表226定义了DynamicallyDefineDataIdentifier肯定响应DDDI 0xF301消息流程示例#6。

表226 - DynamicallyDefineDataIdentifier肯定响应DDDI 0xF301消息流示例 #6

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	0x6C	DDDIPR
#2	definitionMode = defineByIdentifier	0x01	DBID
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2

10.6.5.7.4 步骤#4: ReadDataByIdentifier - dataIdentifier = DDDDI 0xF301

表227定义了ReadDataByIdentifier请求DDDDI 0xF301消息流示例#6。

表227 - ReadDataByIdentifier请求DDDDI 0xF301消息流程示例#6

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDataByIdentifier请求SID	0x22	RDBI
#2	dataIdentifier [byte#1] (MSB)	0xF3	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x01	DID_B2

表228定义ReadDataByIdentifier肯定响应DDDDI 0xF301消息流程示例#6。

表228 - ReadDataByIdentifier正响应DDDDI 0xF301消息流程示例#6

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier响应SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] (MSB)	0xF3	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x01	DID_B2
#4	dataRecord [data#1] = EOT (MSB)	0x4C	DREC_DATA_1
#5	dataRecord [data#2] = EOT	0x36	DREC_DATA_2
#6	dataRecord [data#3] = AAT	0x4D	DREC_DATA_3
#7	dataRecord [data#4] = ECT	0xA6	DREC_DATA_4
#8	dataRecord [data#5] = RPM	0x07	DREC_DATA_5
#9	dataRecord [data#6] = RPM	0x50	DREC_DATA_6
#10	dataRecord [data#7] = EOL (MSB)	0x49	DREC_DATA_7
#11	dataRecord [data#8] = EOL	0x21	DREC_DATA_8
#12	dataRecord [数据#9] = EOL	0x00	DREC_DATA_9
#13	dataRecord [data#10] = EOL	0x17	DREC_DATA_10

10.6.5.7.5 第5步：DynamicallyDefineDataIdentifier - 清晰定义DDDDI 0xF301

表229定义了DynamicallyDefineDataIdentifier请求清除DDDDI 0xF301消息流示例#6.

表229 - DynamicallyDefineDataIdentifier请求清除DDDDI 0xF301消息流示例#6

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier请求SID	0x2C	DDDI
#2	子函数= clearDynamicallyDefinedDataIdentifier, suppressPosRspMsgIndicationBit = FALSE	0x03	CDDDI
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2

表230定义DynamicallyDefineDataIdentifier肯定响应清除DDDDI 0xF301消息流示例#6。

表230 - DynamicallyDefineDataIdentifier肯定响应清除DDDDI 0xF301消息流示例#6

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	DynamicallyDefineDataIdentifier响应SID	0x6C	DDDIPR
#2	definitionMode = clearDynamicallyDefinedDataIdentifier	0x03	CDDDI
#3	dynamicDefinedDataIdentifier [byte#1] (MSB)	0xF3	DDDDI_B1
#4	dynamicDefinedDataIdentifier [byte#2] (LSB)	0x01	DDDDI_B2

10.7 WriteDataByIdentifier (0x2E) 服务

10.7.1 服务说明

WriteDataByIdentifier服务允许客户端将信息写入由提供的数据标识符指定的内部位置的服务器中。

WriteDataByIdentifier服务由客户端用来将dataRecord写入服务器。 数据由dataIdentifier标识，并且可能会或可能不会被保护。

动态定义的dataIdentifer (s) 不得用于此服务。 车辆制造商有责任在执行此服务时满足服务器条件。 此服务的可能用途是:

— 将配置信息编入服务器 (例如，VIN号码)，

— 清除非易失性存储器，

— 重置学习值，

— 设置选项内容。

注意 服务器可以限制或禁止对某些dataIdentifier值 (由系统供应商/车辆制造商为只读标识符等所定义) 进行写访问。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

10.7.2 请求消息

10.7.2.1 请求消息定义

表231定义了请求消息。

表231 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	WriteDataByIdentifier请求SID	M	0x2E	WDBI
#2 #3	dataIdentifier[] = [字节#1 (MSB) 字节#2]	M M	0x00 – 0xFF 0x00 – 0xFF	DID_ HB LB
#4 : #m+3	dataRecord[] = [数据#1 : 数据#m]	M : U	0x00 – 0xFF : 0x00 – 0xFF	DREC_ DATA_1 : DATA_m

10.7.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

10.7.2.3 请求消息数据参数定义

表232定义了请求消息的数据参数。

表232 – 请求消息数据参数定义

定义
dataIdentifier 此参数标识客户端请求写入的服务器数据记录（有关详细参数定义，请参阅C.1）。
dataRecord 此参数提供与客户机请求写入的dataIdentifier关联的数据记录。

10.7.3 积极的回应消息

10.7.3.1 积极响应消息的定义

表233定义了肯定响应消息。

表233 – 肯定回应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	WriteDataByIdentifier响应SID	M	0x6E	WDBIPR
#2 #3	dataIdentifier[] = [字节#1 (MSB) 字节#2]	M M	0x00 – 0xFF 0x00 – 0xFF	DID_ HB LB

10.7.3.2 肯定回复消息数据参数定义

表234定义了肯定响应消息的数据参数。

表234 – 响应消息数据参数定义

定义
dataIdentifier 该参数是来自请求消息的数据参数dataIdentifier的回显。

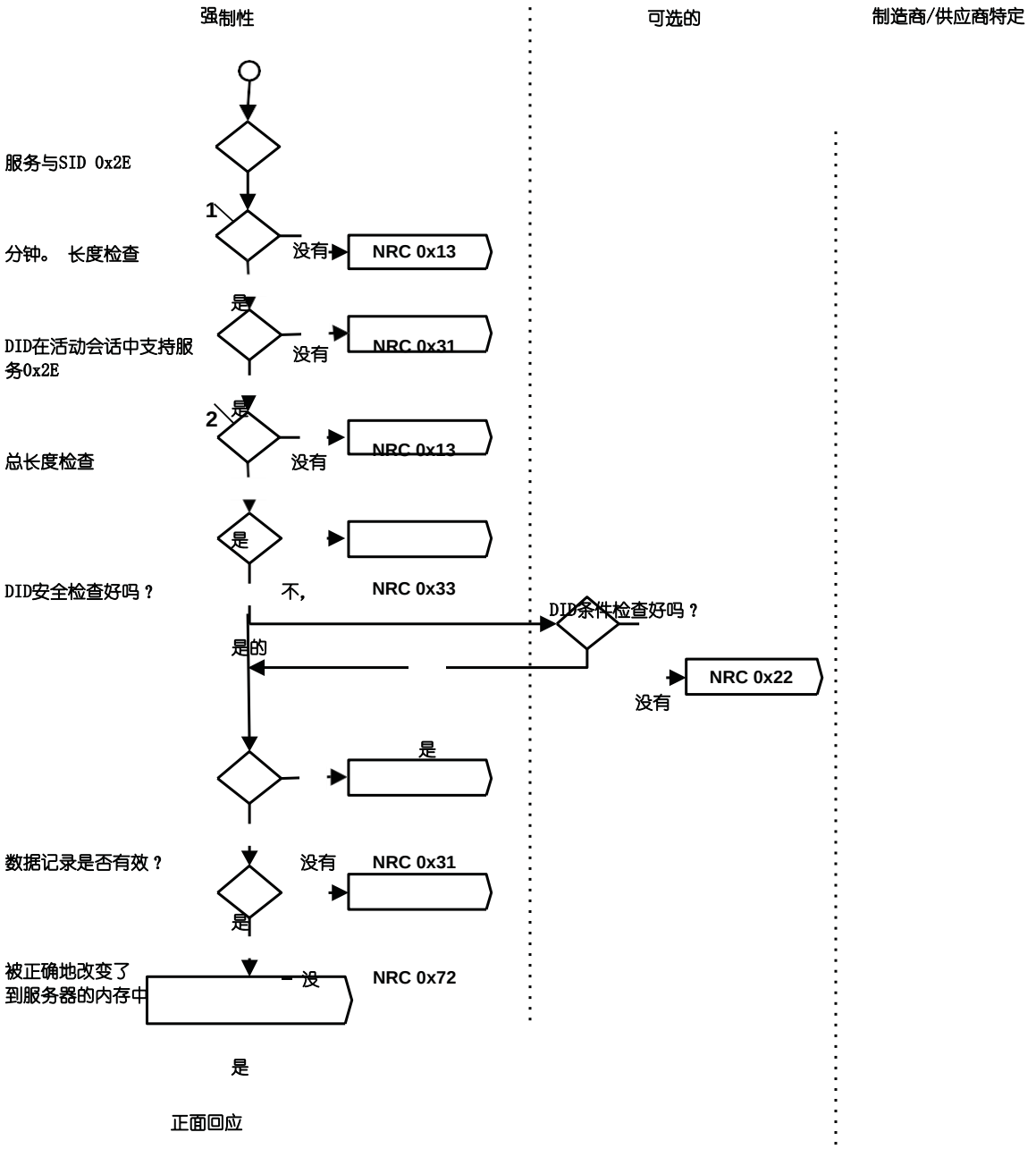
10.7.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表235中。如果错误情况适用于服务器，则应使用列出的否定响应。

表235 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果服务器的运行条件不符合要求，则应发送NRC。	CNC
0x31	**requestOutOfRange** 如果发生以下情况，应发送NRC： — 请求消息中的dataIdentifier在服务器中不受支持，或dataIdentifier仅用于只读目的（通过ReadDataByIdentifier服务）； — 在dataIdentifier无效后请求消息中传输的任何数据（如果适用于该节点）；	ROOR
0x33	**securityAccessDenied** 如果引用特定地址的dataIdentifier受到保护且服务器未处于解锁状态，则应发送此NRC。	伤心
0x72	**generalProgrammingFailure** 如果服务器在写入内存位置时检测到错误，则应返回此NRC。	GPF

评估顺序记录在图21中。



键

1 最小长度为4字节 (SI + DID + DREC)

2 总长度为1个字节 (SI + 2字节DID +第n个字节DREC)

图21 - 用于WriteDataByIdentifier服务的NRC处理

10.7.5 消息流示例WriteDataByIdentifier

10.7.5.1 假设

本条款规定了示例执行WriteDataByIdentifier服务所要满足的条件。

本例中的服务不受服务器限制。 这个例子通过一个两字节的dataIdentifier 0xF190来演示VIN编程。

10.7.5.2 示例#1: 写入dataIdentifier 0xF190 (VIN)

表236定义WriteDataByIdentifier请求消息流程示例#1。

表236 - WriteDataByIdentifier请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	WriteDataByIdentifier请求SID	0x2E	WDBI
#2	dataIdentifier [byte#1] (MSB)	0xF1	DID_B1
#3	dataIdentifier [byte#2]	0x90	DID_B2
#4	dataRecord [data#1] = VIN数字1 =“W”	0x57	DREC_DATA1
#5	dataRecord [data#2] = VIN Digit 2 =“0”	0x30	DREC_DATA2
#6	dataRecord [数据#3] = VIN数字3 =“L”	0x4C	DREC_DATA3
#7	dataRecord [data#4] = VIN Digit 4 =“0”	0x30	DREC_DATA4
#8	dataRecord [data#5] = VIN数字5 =“0”	0x30	DREC_DATA5
#9	dataRecord [数据#6] = VIN数字6 =“0”	0x30	DREC_DATA6
#10	dataRecord [data#7] = VIN Digit 7 =“0”	0x30	DREC_DATA7
#11	dataRecord [data#8] = VIN Digit 8 =“4”	0x34	DREC_DATA8
#12	dataRecord [数据#9] = VIN数字9 =“3”	0x33	DREC_DATA9
#13	dataRecord [数据#10] = VIN数字10 =“M”	0x4D	DREC_DATA10
#14	dataRecord [数据#11] = VIN数字11 =“B”	0x42	DREC_DATA11
#15	dataRecord [数据#12] = VIN数字12 =“5”	0x35	DREC_DATA12
#16	dataRecord [数据#13] = VIN数字13 =“4”	0x34	DREC_DATA13
#17	dataRecord [数据#14] = VIN数字14 =“1”	0x31	DREC_DATA14
#18	dataRecord [数据#15] = VIN数字15 =“3”	0x33	DREC_DATA15
#19	dataRecord [数据#16] = VIN数字16 =“2”	0x32	DREC_DATA16
#20	dataRecord [数据#17] = VIN数字17 =“6”	0x36	DREC_DATA17

表237定义了WriteDataByIdentifier肯定响应消息流程示例#1。

表237 - WriteDataByIdentifier肯定响应消息流程示例#1

消息方向		服务器 客户端	
消息类型		响应	
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	WriteDataByIdentifier响应SID	0x6E	WDBIPR
#2	dataIdentifier [byte#1] (MSB)	0xF1	DID_B1
#3	dataIdentifier [byte#2] (LSB)	0x90	DID_B2

10.8 WriteMemoryByAddress (0x3D) 服务

10.8.1 服务说明

WriteMemoryByAddress服务允许客户端在一个或多个连续的内存位置将信息写入服务器。

WriteMemoryByAddress请求消息将由参数dataRecord []指定的信息写入由参数memoryAddress和memorySize指定的存储器位置的服务器中。 用于memoryAddress和memorySize参数的字节数由addressAndLengthFormatIdentifier（低位和高位半字节）定义。 也可以使用固定的addressAndLengthFormatIdentifier，并且memoryAddress或memorySize参数内的未使用字节在较高范围地址位置填充值为0x00。

dataRecord的格式和定义应该是车辆制造商特定的，并且可能也可能不安全。 车辆制造商有责任确保在执行此服务时满足服务器条件。 此服务的可能用途是：

— 清除非易失性存储器；

— 更改校准值；

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

10.8.2 请求消息

10.8.2.1 请求消息定义

表238定义了请求消息定义。

表238 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	WriteMemoryByAddress请求SID	M	0x3D	WMBA
#2	addressAndLengthFormatIdentifier	M	0x00 – 0xFF	ALFID
#3 : #m+2		M : C1	0x00 – 0xFF : 0x00 – 0xFF	MA_ B1 : Bm
#NR -2- (K-1) : #NR-2		M : C2	0x00 – 0xFF : 0x00 – 0xFF	MS_ B1 : Bk
#N- (R-1) : #n	dataRecord[] = [数据#1 : 数据#r]	M : U	0x00 – 0xFF : 0x00 – 0xFF	DREC_ DATA_1 : DATA_r
C1: 存在的这个参数依靠上地址长度信息参数的该地址和长度格式标识符				
C2: 存在的这个参数依靠上该记忆尺寸长度信息的该地址和长度格式标识符。				

10.8.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

10.8.2.3 请求消息数据参数定义

表239定义了请求消息的数据参数。

表239 – 请求消息数据参数定义

定义
addressAndLengthFormatIdentifier 该参数是一个字节值，每个半字节分别进行编码（请参见H.1或示例值）：位7 – 4：memorySize参数的长度（字节数） 位3 – 0：memoryAddress参数的长度（字节数）
memoryAddress 参数memoryAddress是要写入数据的服务器内存的起始地址。 用于该地址的字节数由addressAndLengthFormatIdentifier的低半字节（bit 3-0）定义。 memoryAddress参数中的字节#m始终是服务器中引用地址的最低有效字节。 地址的最高有效字节可用作存储器标识符。 使用内存标识符的一个例子是具有16位寻址和内存地址重叠的双处理器服务器（当给定地址对任一处理器有效但是产生不同的物理内存设备或使用内部和外部闪存时）。 在这种情况下，可以将memoryAddress参数中另外未使用的字节指定为用于选择所需存储器设备的存储器标识符。 该功能的使用应该由车辆制造商/系统供应商定义。
memorySize WriteMemoryByAddress请求消息中的参数memorySize指定了从memoryAddress指定的地址开始写入的服务器内存中的字节数。 用于此大小的字节数由addressAndLengthFormatIdentifier的高半字节（第7 – 4位）定义。

表239 – （续）

dataRecord
该参数提供了客户端实际尝试在间隔{0xMA, (0xMA + 0xMS – 0x01)}内写入服务器内存地址的数据。

10.8.3 积极的回应消息

10.8.3.1 积极响应消息的定义

表240定义了肯定响应消息。

表240 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	WriteMemoryByAddress响应SID	M	0x7D	WMBAPR
#2	addressAndLengthFormatIdentifier	M	0x00 – 0xFF	ALFID
#3 : #(m-1)+3		M : C1	0x00 – 0xFF : 0x00 – 0xFF	MA_ B1 : Bm
#N- (K-1) : #n		M : C2	0x00 – 0xFF : 0x00 – 0xFF	MS_ B1 : Bk
C1: 存在的这个参数依靠上地址长度信息参数的该地址和长度格式标识符 C2: 存在的这个参数依靠上该记忆尺寸长度信息的该地址和长度格式标识符。				

10.8.3.2 肯定回复消息数据参数定义

表241定义了肯定响应消息的数据参数。

表241 – 响应消息数据参数定义

定义
addressAndLengthFormatIdentifier 该参数是来自请求消息的addressAndLengthFormatIdentifier的回显。
memoryAddress 该参数是来自请求消息的memoryAddress的回显。
memorySize 该参数是来自请求消息的memorySize的回声。

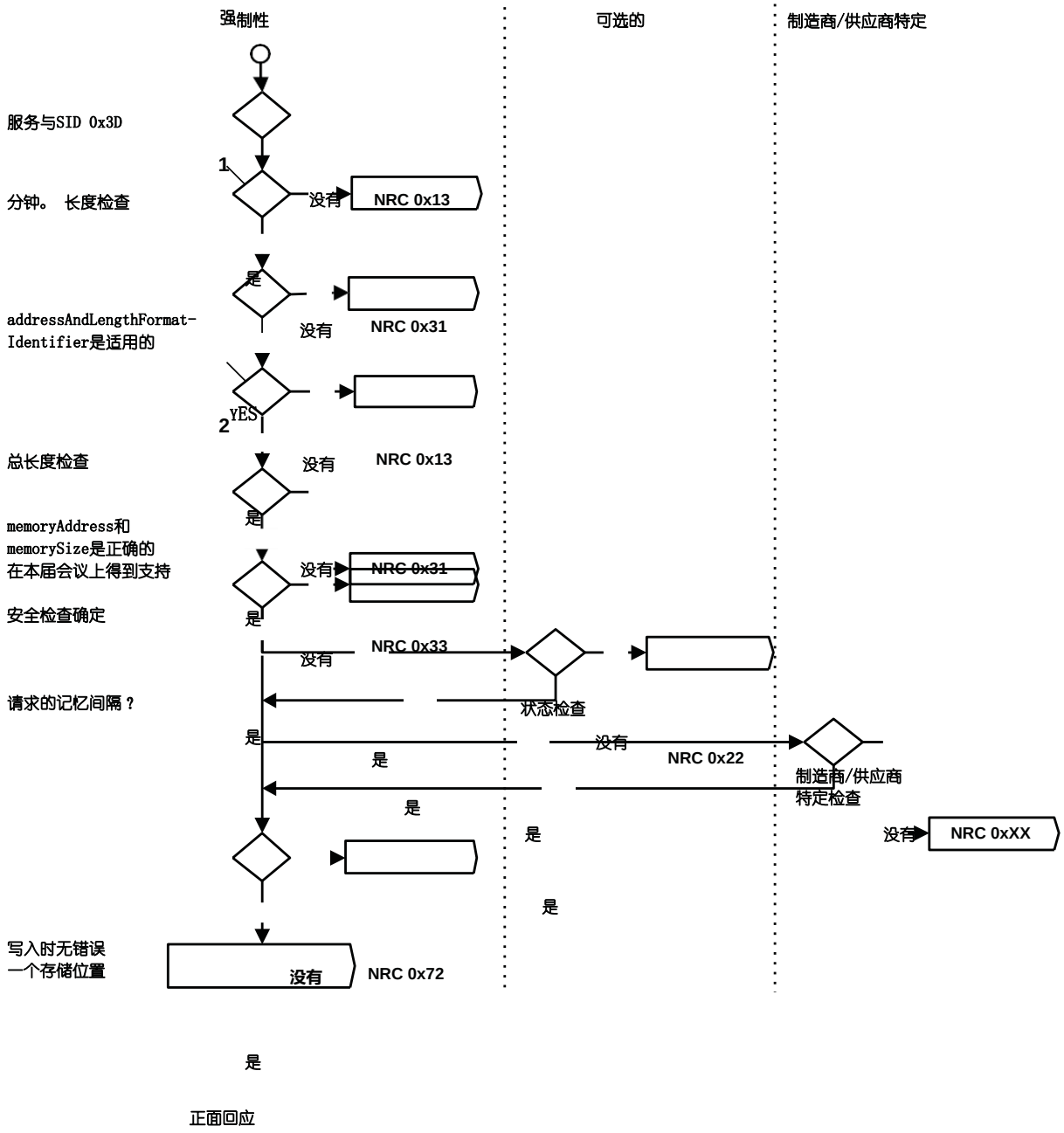
10.8.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 在表242中记录了每个响应代码发生的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表242 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果服务器的运行条件不符合要求，则应发送NRC。	CNC
0x31	**requestOutOfRange** 如果发生以下情况，应发送NRC： — 区间[0xMA，（0xMA + 0xMS -0x1）]内的任何内存地址都是无效的； — 区间[0xMA，（0xMA + 0xMS -0x1）]内的任何内存地址都受到限制； — 请求消息中的memorySize参数值不受服务器支持； — 指定的addressAndLengthFormatIdentifier无效； — 请求消息中的memorySize参数值为零；	ROOR
0x33	**securityAccessDenied** 如果间隔[0xMA，（0xMA + 0xMS-0x1）]内的任何内存地址是安全的并且服务器被锁定，则应发送此NRC。	伤心
0x72	**generalProgrammingFailure** 如果服务器在写入内存位置时检测到错误，则应返回此NRC。	GPF

评估顺序记录在图22中。



键

1 至少5（SI +地址和长度格式标识符+最小内存地址+最小内存大小+最小数据记录）

2 1字节SI + 1字节addressAndLengthFormatIdentifier + n字节memoryAddress参数长度+ n字节memorySize参数长度 + n字节dataRecord长度

图22 - 用于WriteMemoryByAddress服务的NRC处理

10.8.5 消息流示例WriteMemoryByAddress

10.8.5.1 假设

本小节指定了执行WriteMemoryByAddress服务的例子要满足的条件。 本例中的服务不受服务器限制。

以下示例分别演示了如何将数据字节写入服务器内存，以获取2字节，3字节和4字节的寻址格式。

10.8.5.2 示例#1: WriteMemoryByAddress，2字节（16位）寻址

表243定义了WriteMemoryByAddress请求消息流程示例#1。

表243 - WriteMemoryByAddress请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	WriteMemoryByAddress请求SID	0x3D	WMBA
#2	addressAndLengthFormatIdentifier	0x12	ALFID
#3	memoryAddress [byte#1] (MSB)	0x20	MA_B1
#4	memoryAddress [byte#2] (LSB)	0x48	MA_B2
#5	memorySize [byte#1]	0x02	MS_B1
#6	dataRecord [数据#1]	0x00	DREC_DATA_1
#7	dataRecord [data#2]	0x8C	DREC_DATA_2

表244定义了WriteMemoryByAddress肯定响应消息流程示例#1。

表244 - WriteMemoryByAddress正响应消息流程示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	WriteMemoryByAddress响应SID	0x7D	WMBAPR
#2	addressAndLengthFormatIdentifier	0x12	ALFID
#3	memoryAddress [byte#1] (MSB)	0x20	MA_B1
#4	memoryAddress [byte#2] (LSB)	0x48	MA_B2
#5	memorySize [byte#1]	0x02	MS_B1

10.8.5.3 例#2: WriteMemoryByAddress，3字节（24位）寻址

表245定义了WriteMemoryByAddress请求消息流程示例#2。

表245 - WriteMemoryByAddress请求消息流程示例#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	WriteMemoryByAddress请求SID	0x3D	WMBA
#2	addressAndLengthFormatIdentifier	0x13	ALFID
#3	memoryAddress [byte#1]	0x20	MA_B1
#4	memoryAddress [byte#2]	0x48	MA_B2
#5	memoryAddress [byte#3]	0x13	MA_B3
#6	memorySize [byte#1]	0x03	MS_B1
#7	dataRecord [数据#1]	0x00	DREC_DATA_1
#8	dataRecord [data#2]	0x01	DREC_DATA_2
#9	dataRecord [data#3]	0x8C	DREC_DATA_3

表246定义了WriteMemoryByAddress肯定响应消息流程示例#2。

表246 - WriteMemoryByAddress正响应消息流程示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	WriteMemoryByAddress响应SID	0x7D	WMBAPR
#2	addressAndLengthFormatIdentifier	0x13	ALFID
#3	memoryAddress [byte#1]	0x20	MA_B1
#4	memoryAddress [byte#2]	0x48	MA_B2
#5	memoryAddress [byte#3]	0x13	MA_B3
#6	memorySize [byte#1]	0x03	MS_B1

10.8.5.4 例#3: WriteMemoryByAddress, 4字节（32位）寻址

表247定义了WriteMemoryByAddress请求消息流程示例#3。

表247 - WriteMemoryByAddress请求消息流程示例#3

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	WriteMemoryByAddress请求SID	0x3D	WMBA
#2	addressAndLengthFormatIdentifier	0x14	ALFID
#3	memoryAddress [byte#1] (MSB)	0x20	MA_B1
#4	memoryAddress [byte#2]	0x48	MA_B2
#5	memoryAddress [byte#3]	0x13	MA_B3
#6	memoryAddress [byte#4] (LSB)	0x09	MA_B4
#7	memorySize [byte#1]	0x05	MS_B1
#8	dataRecord [数据#1]	0x00	DREC_DATA_1
#9	dataRecord [data#2]	0x01	DREC_DATA_2
#10	dataRecord [data#3]	0x8C	DREC_DATA_3
#11	dataRecord [data#4]	0x09	DREC_DATA_4
#12	dataRecord [数据#5]	0xAF执行	DREC_DATA_5

表248定义了WriteMemoryByAddress正响应消息流程示例#3。

表248 - WriteMemoryByAddress正响应消息流程示例#3

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	WriteMemoryByAddress响应SID	0x7D	WMBAPR
#2	addressAndLengthFormatIdentifier	0x14	ALFID
#3	memoryAddress [byte#1] (MSB)	0x20	MA_B1
#4	memoryAddress [byte#2]	0x48	MA_B2
#5	memoryAddress [byte#3]	0x13	MA_B3
#6	memoryAddress [byte#4] (LSB)	0x09	MA_B4
#7	memorySize [byte#1]	0x05	MS_B1

11 存储数据传输功能单元

11.1 概观

表249 - 存储数据传输功能单元

服务	描述
ClearDiagnosticInformation	允许客户端清除服务器的诊断信息（包括DTC，捕获的数据等）
ReadDTCInformation	允许客户端从服务器请求诊断信息（包括DTC，捕获的数据等）

11.2 ClearDiagnosticInformation (0x14) 服务

11.2.1 服务说明

客户端使用ClearDiagnosticInformation服务清除一个或多个服务器内存中的诊断信息。

当ClearDiagnosticInformation服务完全处理时，服务器应发送肯定响应。 即使没有存储DTC，服务器也应发送肯定的响应。 如果服务器在内存中支持DTC状态信息的多个副本（例如，一个RAM副本和一个EEPROM副本），则服务器应清除ReadDTCInformation状态报告服务使用的副本。 其他副本（例如长期内存中的备份副本）根据适当的备份策略进行更新（例如，在电源锁定阶段）。

注意 在功率锁存阶段受到干扰的情况下（例如，在功率锁定阶段期间电池断开），这可能导致数据不一致。

各个DTC状态位的行为应根据D.2，图D.1 – 图D.8中的定义执行。

客户端的请求消息包含一个参数。 参数groupOfDTC允许客户端清除一组DTC（例如动力总成，车身，底盘等）或特定的DTC。 进一步的细节请参考D.1。 除非另有说明，否则服务器应为所请求的组清除内存中与排放有关的和与排放无关的DTC信息。

通过此服务重置/清除DTC信息包括但不限于以下内容:

— DTC状态字节（见11.3中的ReadDTCInformation服务），

— 捕获的DTC快照数据（DTCSnapshotData，请参阅11.3中的ReadDTCInformation服务），

— 捕获的DTC扩展数据（DTCExtendedData，请参阅11.3中的ReadDTCInformation服务），

— 其他DTC相关数据，例如DTC专用的第一/最近DTC，标志，计数器，定时器等，

存储在服务器的可选DTC镜像存储器中的任何DTC信息都不受此服务的影响（请参阅11.3中的ReadDTCInformation (0x19) 服务，以用于DTC镜像存储器定义）。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

11.2.2 请求消息

11.2.2.1 请求消息定义

表250定义了请求消息。

表250 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ClearDiagnosticInformation请求SID	M	0x14	CDTCI
#2		M	0x00 – 0xFF	GODTC_HB
#3		M	0x00 – 0xFF	MB
#4		M	0x00 – 0xFF	磅

11.2.2.2 请求消息子函数参数$ Level（LEV_）定义

这个服务没有使用子功能参数。

11.2.2.3 请求消息数据参数定义

表251定义了请求消息的数据参数。

表251 – 请求消息数据参数定义

定义
groupOfDTC 该参数包含一个3字节的值，指示DTC组（例如，动力总成，车身，底盘）或要清除的特定DTC。 D.1中包含了每个值/值范围的值的定义。

11.2.3 积极的回应消息

11.2.3.1 积极响应消息的定义

表252定义了肯定响应消息。

表252 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ClearDiagnosticInformation正面响应SID	M	0x54	CDTCIPR

11.2.3.2 肯定回复消息数据参数定义

在肯定响应消息中没有该服务使用的数据参数。

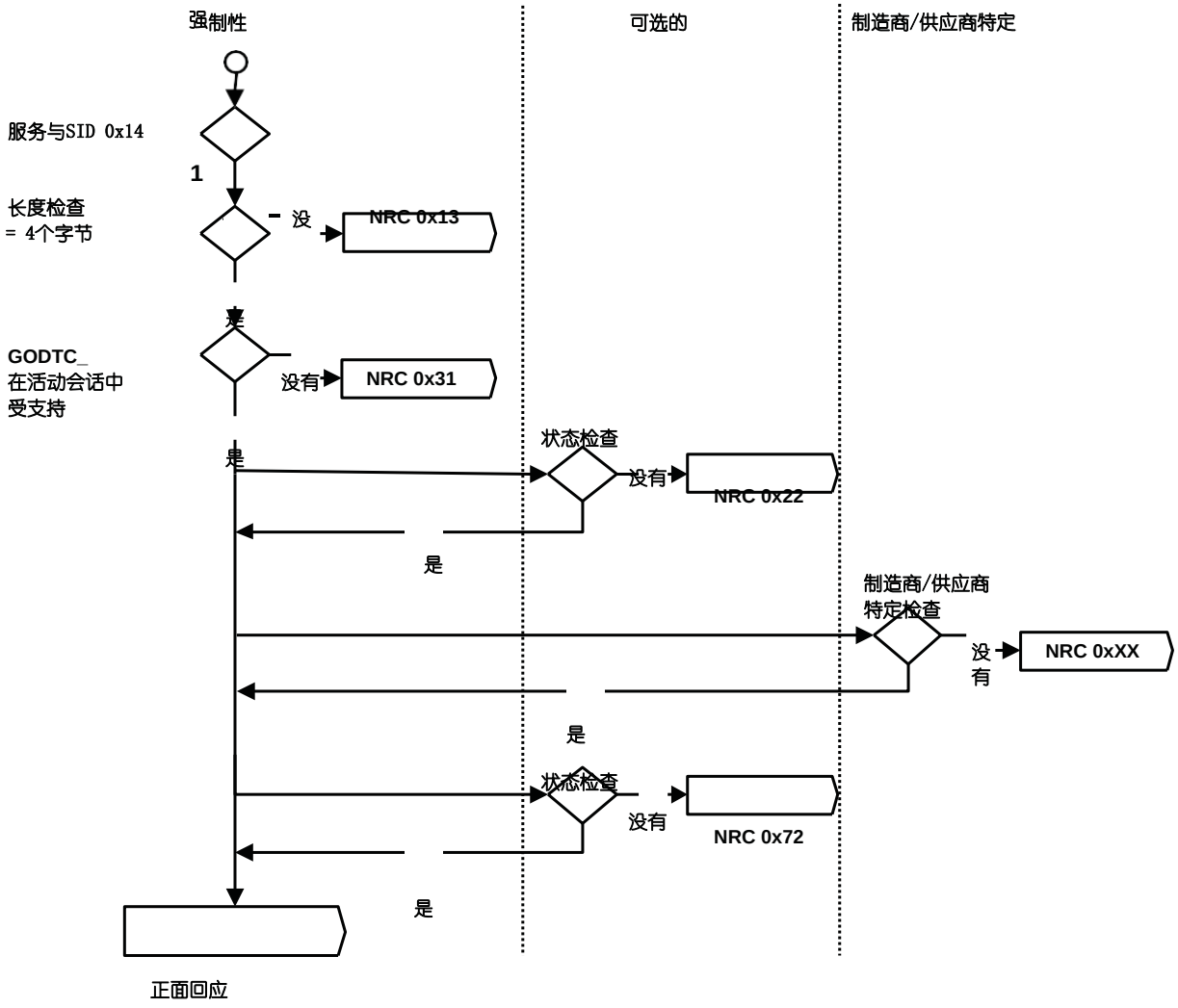
11.2.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表253中。如果错误情况适用于服务器，则应使用列出的否定响应。

表253 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果服务器内部条件阻止清除存储在服务器中的DTC相关信息，则应使用此NRC。	CNC
0x31	**requestOutOfRange** 如果不支持指定的groupOfDTC参数，则应返回该NRC。	ROOR
0x72	**generalProgrammingFailure** 如果服务器在写入内存位置时检测到错误，则应返回此NRC。	GPF

评估顺序记录在图23中。



键
1 CDTCI + GODTC_

图23 – 用于ClearDiagnosticInformation服务的NRC处理

11.2.5 消息流示例ClearDiagnosticInformation

客户端将ClearDiagnosticInformation请求消息发送到单个服务器。 表254定义了ClearDiagnosticInformation请求消息流程示例#1。 客户端将ClearDiagnosticInformation请求消息发送到单个服务器。

表254 – ClearDiagnosticInformation请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ClearDiagnosticInformation请求SID	0x14	CDTCI
#2	groupOfDTC [DTCHighByte]（“排放相关系统”）	为0xFF	DTCHB
#3	groupOfDTC [DTCMiddleByte]	为0xFF	DTCMB
#4	groupOfDTC [DTCLowByte]	0x33	DTCLB

表255定义了ClearDiagnosticInformation肯定响应消息流程示例#1。

表255 - ClearDiagnosticInformation肯定响应消息流程示例#1

消息方向		服务器 客户端	
消息类型		响应	
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ClearDiagnosticInformation响应SID	0x54	CDTCIPR

11.3 ReadDTCInformation (0x19) 服务

11.3.1 服务说明

11.3.1.1 一般描述

该服务允许客户从任何服务器或车辆内的一组服务器读取服务器驻留的诊断故障代码（DTC）信息的状态。 除非特定子功能另有要求，否则服务器应返回所有DTC信息（例如与排放有关的和与排放无关的信息）。 该服务允许客户执行以下操作:

— 检索与客户定义的DTC状态掩码相匹配的DTC数量，

— 检索与客户定义的DTC状态掩码相匹配的所有DTC的列表，

— 检索匹配客户定义的DTC状态掩码的特定功能组中的DTC列表，

— 检索所有具有“永久性DTC”状态的DTC。

— 检索与客户端定义的DTC关联的DTCSnapshot数据（有时称为冻结帧）: DTC快照是存储在服务器内存中的与DTC关联的特定数据记录。 DTC快照的典型用法是在检测到系统故障时存储数据。 DTC快照将作为系统故障发生时刻的数据值的快照。 存储在DTC快照中的数据参数应与DTC相关联。 DTC特定的数据参数旨在减轻技术人员的故障隔离过程。

— 从DTC存储器或DTC镜像存储器中检索与客户机定义的DTC和状态掩码组合关联的DTCExtendedData。DTCExtendedData由与DTC相关的扩展状态信息组成。 DTCExtendedData包含请求时已识别的DTC参数值。DTCExtendedData的典型用途是存储与DTC相关的动态数据，例如

  — DTC B1故障指示器计数器，用于显示故障激活期间OBD系统运行的时间量（发动机运行小时数）

  — DTC发生计数器，计数已报告“testFailed”的驾驶循环的次数，

  — DTC老化计数器，计算自故障最近以来的驾驶循环次数，排除测试未报告“testPassed”或“testFailed”的驾驶循环，

  — 用于OBD的特定计数器（例如，如果驾驶循环可以在无故障模式下执行，则直到“检查引擎”灯关闭为止的剩余驾驶循环次数）。

  — 最后一次发生的时间（等）。

  — 测试失败的计数器，计数报告的“testFailed”的数量和可能的其他计数器，如果验证是在几个步骤中执行的，

— 未完成的测试计数器，计数自测试最近完成（即自测试报告“testPassed”或“testFailed”以来）的驾驶周期数，

— 检索与客户机定义的严重性掩码匹配的DTC数量，

— 检索与客户机定义的严重性掩码记录匹配的DTC列表，

— 检索客户端定义的DTC的严重性信息，

— 检索服务器支持的所有DTC的状态，

— 检索服务器失败的第一个DTC，

— 检索服务器内最近失败的DTC，

— 检索由服务器确认的第一个DTC，

— 检索服务器内最近确认的DTC，

— 从DTC镜像内存中检索与客户机定义的DTC状态掩码匹配的DTC列表，

— 从DTC镜像存储器中检索客户机定义的DTC掩码和客户机定义的DTCExtendedData记录编号的镜像内存DTCExtendedData记录数据，

— 从DTC镜像存储器中检索与客户机定义的DTC状态掩码相匹配的DTC数量，

— 检索与客户定义的DTC状态掩码相匹配的“唯一”与排放相关的OBD DTC的数量。 与排放相关的OBD DTC会导致故障指示灯在检测到DTC时打开/显示信息，

— 检索与客户定义的DTC状态掩码相匹配的“唯一”与排放相关的OBD DTC的状态。 与排放相关的OBD DTC会导致故障指示灯在检测到DTC时打开/显示信息，

— 检索所有目前已经或尚未被检测为“待定”或“已确认”的“预先故障”DTC，

— 从DTC内存中检索与客户端定义的DTCExtendedData记录状态关联的DTCExtendedData。

— 从用户定义的DTC存储器中检索与客户机定义的DTC状态掩码相匹配的DTC列表，

— 从用户定义的DTC镜像内存中检索用户定义的DTC内存DTCExtendedData记录数据，用于客户机定义的DTC掩码掩码和客户机定义的DTCExtendedData记录编号，

— 从用户定义的DTC内存中检索用户定义的DTC内存DTCSnapshotRecord数据，用于客户机定义的DTC掩码，

此服务使用子功能来确定客户端请求哪种类型的诊断信息。 在下面的小节中提供了关于每个子功能参数的更多细节。

此服务使用以下条款：

— **启用标准：服务器/车辆制造商/系统供应商用于控制服务器何时实际执行特定内部诊断的特定标准。**

— 测试通过标准：服务器/车辆制造商/系统供应商特定的条件，定义正在被诊断的系统是否在正常的可接受的工作范围内正常运行（例如，没有故障存在并且诊断系统被分类为“OK”）。

— 测试失败标准：服务器/车辆制造商/系统供应商特定故障条件，定义正在诊断的系统是否未通过测试。

— 确认的故障标准：服务器/车辆制造商/系统供应商特定的故障条件，用于定义被诊断的系统是否确定存在问题（确认），保证将DTC记录存储在长期存储器中。

— 发生次数：由某些服务器维护的计数器，用于记录给定DTC测试报告发生测试失败的唯一事件的实例数量。

— 老化：某些服务器评估每个内部诊断结果的过程，以确定是否可以从长期记忆中清除已确认的DTC，例如在校准的无故障循环次数的情况下。

除读取DTCSnapshotRecords之外，给定的DTC值（例如，0x080511）在读取DTCInformation的肯定响应中不会多次报告，其中响应可能包含针对相同DTC的多个DTCSnapshotRecords。

当使用分页缓冲区处理来读取DTC时（特别是对于子功能= reportDTCByStatusMask），创建响应时可能会减少DTC的数量。 在这种情况下，响应应填入DTC 0x000000和DTC状态0x00。 客户端应将这些DTC视为不存在于响应消息中。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

11.3.1.2 检索与客户端定义的状态掩码相匹配的DTC数（子函数= 0x01 reportNumberOfDTCByStatusMask）

客户端可以通过发送针对此服务的请求并将子功能设置为reportNumberOfDTCByStatusMask来检索匹配客户端定义的状态掩码的DTC数量。 对此请求的响应包含DTCStatusAvailabilityMask，它提供服务器为掩蔽目的支持的DTC状态位指示。 在DTCStatusAvailabilityMask之后，响应包含报告关于DTC格式化和编码的信息的DTCFormatIdentifier。 DTCFormatIdentifier之后是DTCCount参数，该参数是一个2字节的无符号数字数字，包含基于客户端提供的状态掩码的服务器内存中可用的DTC数量。

子函数reportNumberOfMirrorMemoryDTCByStatusMask具有与子函数reportNumberOfDTCByStatusMask相同的功能，区别在于它从DTC镜像内存中返回DTC的数量。

11.3.1.3 检索与客户端定义的状态掩码相匹配的DTC列表（子功能= 0x02 reportDTCByStatusMask）

客户端可以通过发送具有设置为reportDTCByStatusMask的子函数字节的请求来检索满足客户端定义的状态掩码的DTC列表。 该子功能允许客户端请求服务器报告“testFailed”或“确认”或“等”的所有DTC。

评估应按如下方式进行：服务器应在客户端请求中指定的掩码与服务器支持的每个DTC关联的实际状态之间执行按位逻辑AND操作。 除DTCStatusAvailabilityMask外，服务器应返回AND操作结果非零的所有DTC（即（statusOfDTC&DTCStatusMask）！= 0）。 如果客户端指定包含服务器不支持的位的状态掩码，则服务器应仅使用它支持的位处理DTC信息。 如果服务器内没有DTC符合掩蔽标准

在客户请求中指定，在肯定响应消息中的DTCStatusAvailabilityMask字节之后不应提供DTC或状态信息。

DTC状态信息应当在来自客户端的成功的ClearDiagnosticInformation请求时被清除（关于在服务器中接收到ClearDiagnostic信息服务请求的情况下的DTC状态位处理的进一步描述，请参见D.2中的DTC状态位定义）。

11.3.1.4 检索DTCSnapshot记录标识（子功能= 0x03 reportDTCSnapshotIdentification）

客户端可以通过发送对具有设置为reportDTCSnapshotIdentification的子功能的此服务的请求来检索所有捕获的DTCSnapshot记录的DTCSnapshot记录标识信息。 服务器应返回所有存储的DTCSnapshot记录的DTCSnapshot记录标识信息列表。 服务器放置在单个DTCSnapshot记录的响应消息中的每个项目应包含一个DTCRecord（包含DTC编号（高，中，低字节））和DTCSnapshot记录号。 如果为单个DTC存储多个DTCSnapshot记录，则服务器应在每次出现的响应中放置一个项目，每次使用不同的DTCSnapshot记录号（用于稍后检索记录数据）。

注意服务器可以支持为单个DTC存储多个DTCSnapshot记录，以跟踪每次DTC出现的情况。 支持此功能，“发生”标准的定义以及要支持的DTCSnapshot记录的数量应由系统供应商/车辆制造商定义。

DTCSnapshot记录标识信息应当在客户端成功执行ClearDiagnosticInformation请求时清除。 车辆制造商有责任在发生内存溢出（存储的DTC和DTCSnapshot数据在服务器中完全占用存储空间）时指定删除存储的DTC和DTCSnapshot数据的规则。

11.3.1.5 检索客户端定义的DTC掩码的DTCSnapshot记录数据（子函数= 0x04 reportDTCSnapshotRecordByDTCNumber）

客户端可以通过发送对具有设置为reportDTCSnapshotRecordByDTCNumber的子功能的此服务的请求，检索客户端定义的DTCMaskRecord的捕获的DTCSnapshot记录数据以及DTCSnapshot记录号。 服务器应搜索其支持的DTC与客户指定的DTCMaskRecord（包含DTC编号（高，中，低字节））完全匹配。 客户请求中提供的DTCSnapshotRecordNumber参数应指定一个特定的DTCSnapshot记录数据被请求的DTC的出现。

注1: DTCSnapshotRecordNumber不共享与DTCStoredDataRecordNumber相同的地址空间。

如果服务器支持为单个DTC存储多个DTCSnapshot记录（支持此功能由系统供应商/车辆制造商定义），则建议服务器还实施reportDTCSnapshotIdentification子功能参数。 建议客户在通过reportDTCSnapshotRecordByDTCNumber请求请求特定的DTCSnapshotRecordNumber之前，首先请求使用子函数参数reportDTCSnapshotIdentification存储的DTCSnapshot记录的标识。

还建议支持子函数参数reportDTCSnapshotRecordIdentification，以便客户机会直接识别存储的DTCSnapshot记录，而不是通过解析服务器的所有存储的DTC来确定是否存储DTCSnapshot记录。

系统供应商/车辆制造商有责任确定在此类服务器内捕获的DTCSnapshot记录是否将与故障发生信息相关的数据存储为ECU文档的一部分。

如果客户端定义的DTCMaskRecord和DTCSnapshotRecordNumber参数（DTCSnapshotRecordNumber不等于0xFF）已被识别出故障，则服务器将返回单个预定义的DTCSnapshotRecord以响应客户端的请求，以及DTC号和statusOfDTC。

注2：准确的故障标准应由系统供应商/车辆制造商定义。

DTCSnapshot记录可能包含多个数据参数，可用于在发生故障时重建车辆状况（例如B +，RPM，时间戳）。

车辆制造商应定义DTCSnapshotRecord的格式和内容。 DTCSnapshotRecord中报告的数据首先包含一个dataIdentifier以标识随后的数据。 该数据标识符/数据组合可以在DTCSnapshotRecord内重复使用.DTCSnapshotRecord中的一个或多个dataIdentifiers的使用允许针对单个DTC存储不同类型的DTCSnapshotRecords以用于不同发生的故障。 指示每个DTCSnapshotRecord中包含的记录DataIdentifiers的数量的参数应与每个DTCSnapshotRecord一起提供以协助数据检索。

除了客户端已将DTCSnapshotRecordNumber设置为0xFF之外，服务器应在单个响应消息中报告一个DTCSnapshot记录，因为这将导致服务器响应于在单个响应消息中为客户端定义的DTCMaskRecord存储的所有DTCSnapshot记录。 DTCAndStatusRecord只在响应消息中包含一次。 如果客户端在其请求中为参数DTCSnapshotRecordNumber使用0xFF，则服务器应以数字升序报告为特定DTC捕获的所有DTCSnapshot记录。

如果客户端指定的DTCMaskRecord或DTCSnapshotRecordNumber参数无效或服务器不支持，服务器将作出负面响应。 这与客户端指定的DTCMaskRecord和/或DTCSnapshotRecordNumber参数确实有效且受服务器支持但不具有与其关联的DTCSnapshot数据的情况不同（例如，因为指定的故障事件从未发生过DTC或记录号码）。 服务器应发送仅包含DTCAndStatusRecord（请求的DTC编号（高，中，低字节）的回声加上statusOfDTC的肯定响应）。

DTCSnapshot信息应当在客户端成功的ClearDiagnosticInformation请求时被清除。 车辆制造商有责任在发生内存溢出（用于存储的DTC和DTC快照数据在服务器中完全占用的存储空间）时指定删除存储的DTC和DTCSnapshot数据的规则。

11.3.1.6 为客户机定义的记录号检索DTCStoredData记录数据（子函数= 0x05 reportDTCStoredDataByRecordNumber）

客户端可以通过发送对具有设置为reportDTCStoredDataByRecordNumber的子功能的该服务的请求来检索DTCStoredDataRecordNumber的捕获的DTCStoredData记录数据。 服务器应搜索其存储的DTCStoredData记录以查找客户提供的记录号的匹配。

DTCStoredDataRecordNumber不共享与DTCSnapshotRecordNumber相同的地址空间。

系统供应商/车辆制造商有责任定义在这些服务器中捕获的DTCStoredData记录是否存储与第一次或最近发生的故障相关的数据。

注：确切的故障标准应由系统供应商/车辆制造商定义。

DTCStoredData记录可能包含多个数据参数，这些参数可用于在故障发生时重建车辆状况（例如B +，RPM，时间戳）。

车辆制造商应定义DTCStoredDataRecordNumber的格式和内容。 DTCStoredDataRecord中报告的数据首先包含一个dataIdentifier以标识随后的数据。 该数据标识符/数据组合可以在DTCStoredDataRecord内重复。 在DTCStoredDataRecord中使用一个或多个dataIdentifiers可以为单个DTC存储不同类型的DTCStoredDataRecords，用于发生不同的故障。 指示每个DTCStoredDataRecord中包含的记录DataIdentifiers的数量的参数应与每个DTCStoredDataRecord一起提供以协助数据检索。

除了客户端已将DTCStoredDataRecordNumber设置为0xFF之外，服务器应在单个响应消息中报告一个DTCStoredDataRecord，因为这将导致服务器响应存储在单个响应消息中的所有DTCStoredDataRecords。

如果客户端请求通过记录号报告所有DTCStoredDataRecords，则必须在每个存储的DTCStoredDataRecord的响应消息中重复DTCAndStatusRecord。

如果客户端指定的DTCStoredDataRecordNumber参数无效或服务器不支持，服务器将作出否定响应。 这与客户端指定的DTCStoredDataRecordNumber参数确实有效并受服务器支持，但没有与其关联的DTCStoredDataRecord数据（例如，因为指定记录号从未发生过失败事件）不同的情况不同。 服务器应发送只包含DTCStoredDataRecordNumber的正响应（请求记录号的回显）。

DTCStoredDataRecord信息应在客户端成功执行ClearDiagnosticInformation请求时清除。 车辆制造商有责任在发生内存溢出时（存储的DTC和DTCStoredDataRecord数据在服务器中完全占用存储空间），指定删除存储的DTC和DTCStoredDataRecord数据的规则。

11.3.1.7 检索客户端定义的DTC掩码和客户端定义的DTCExtendedData记录号的DTCExtendedData记录数据（子函数= 0x06 reportDTCExtDataRecordByDTCNumber）

客户端可以通过发送针对此服务的请求并将子功能集设置为reportDTCExtDataRecordByDTCNumber来检索客户端定义的DTCMaskRecord的DTCExtendedData以及DTCExtendedData记录号。 服务器应搜索其支持的DTC与客户指定的DTCMaskRecord（包含DTC编号（高，中，低字节））完全匹配。 在这种情况下，客户端请求中提供的DTCExtDataRecordNumber参数应指定DTCExtendedData所请求的指定DTC的特定DTCExtendedData记录。

与DTC编号和statusOfDTC一起，服务器将响应客户端请求（DTCExtDataRecordNumber不等于0xFE或0xFF）返回单个预定义的DTCExtendedData记录。

车辆制造商应定义DTCExtDataRecord的格式和内容。 DTCExtDataRecord中报告的数据结构由DTCExtDataRecordNumber定义，与记录DataIdentifier中的数据定义类似。 响应中可能包含多个DTCExtDataRecordNumbers和相关的DTCExtDataRecords。 使用一个或多个DTCExtDataRecordNumbers允许为单个DTC存储不同类型的DTCExtDataRecords。

除了客户端已将DTCExtDataRecordNumber设置为0xFE或0xFF之外，服务器应在单个响应消息中报告一个DTCExtendedData记录，因为这将导致服务器在单个响应消息中响应为客户端定义的DTCMaskRecord存储的所有DTCExtendedData记录。

如果客户端指定的DTCMaskRecord或DTCExtDataRecordNumber参数无效或服务器不支持，服务器应负面响应。这包括客户端发送0xFE的DTCExtDataRecordNumber但服务器不支持OBD扩展数据记录（0x90 – 0xEF）的情况。这与客户端指定的DTCMaskRecord和/或DTCExtDataRecordNumber参数确实有效且受服务器支持但不具有与其关联的DTC扩展数据的情况（例如，由于扩展数据的内存溢出）。 在reportDTCExtDataRecordByDTCNumber的情况下，服务器应发送只包含DTCAndStatusRecord（请求的DTC编号的回声（高，中，低字节）加上statusOfDTC的肯定响应）。

在接收到ClearDiagnosticInformation服务时清除DTCExtendedData信息在11.2.1中指定。 车辆制造商有责任在存储器溢出的情况下（用于存储的DTC和DTC扩展数据在服务器中完全占用的存储空间）指定删除存储的DTC和DTC扩展数据的规则。

11.3.1.8 检索与客户机定义的严重性掩码记录相匹配的DTC数量 (subfunction = 0x07 reportNumberOfDTCBySeverityMaskRecord)

客户端可以通过发送对具有设置为reportNumberOfDTCBySeverityMaskRecord的子功能的此服务的请求来检索匹配客户端定义的严重性状态掩码记录的DTC的数目的计数。 服务器应扫描所有支持的DTC，在客户指定的掩模记录与每个存储的DTC的实际信息之间执行按位逻辑AND操作。

(((statusOfDTC&DTCStatusMask) ! = 0) && ((severity&DTCSeverityMask) ! = 0)) == TRUE

对于产生TRUE结果的每个AND操作，服务器应将计数器加1。如果客户端在掩码记录中指定了一个包含服务器不支持的位的状态掩码，则服务器应使用DTC信息处理只有它支持的位。 一旦所有支持的DTC被检查过一次，服务器应该返回DTCStatusAvailabilityMask和得到的2字节计数给客户端。

说明如果服务器内没有DTC与客户端请求中指定的掩码准则匹配，则服务器返回给客户端的计数应为0.报告的符合DTC状态掩码的DTC数量对于请求时间点有效被制造了。 报告的DTC数量与通过子函数reportDTCByStatusMas读取的实际DTC列表之间没有关系，因为读取DTC的请求是在不同的时间点完成的。

11.3.1.9 检索与客户机定义的严重性掩码记录匹配的严重性和功能单元信息 (子函数= 0x08 reportDTCBySeverityMaskRecord)

客户端可以通过发送具有设置为reportDTCBySeverityMaskRecord的子函数字节的请求来检索DTC严重性和功能单元信息的列表，其满足客户机定义的严重性掩码记录。 该子功能允许客户端请求服务器报告具有“testFailed”或“确认”或“等”的特定严重性和状态的所有DTC。 评估工作如下:

服务器应在客户请求中指定的DTCSeverityMask和DTCStatusMask与服务器支持的每个DTC关联的实际DTCSeverity和statusOfDTC之间执行按位逻辑AND操作。

除了DTCStatusAvailabilityMask之外，服务器还应返回AND操作结果为TRUE的所有DTC，

(((statusOfDTC&DTCStatusMask) ! = 0) && ((severity&DTCSeverityMask) ! = 0)) == TRUE

如果客户端在掩码记录内指定了一个包含服务器不支持的位的状态掩码，那么服务器将仅使用它支持的位来处理DTC信息。 如果服务器内没有DTC与客户请求中指定的屏蔽标准相匹配，则不应在正响应消息中的DTCStatusAvailabilityMask字节之后提供DTC或状态信息。

11.3.1.10 检索客户定义的DTC的严重程度和功能单位信息 (子功能= 0x09 reportSeverityInformationOfDTC)

客户端可以通过发送针对此服务的请求并将子功能集设置为reportSeverityInformationOfDTC来检索客户端定义的DTCMaskRecord的严重性和功能单元信息。 服务器应搜索其支持的DTC与客户指定的DTCMaskRecord (包含DTC编号 (高，中，低字节)) 完全匹配。

11.3.1.11 检索服务器支持的所有DTC的状态（子功能= 0x0A reportSupportedDTC）

客户端可以通过发送对该服务的请求并将子功能集设置为reportSupportedDTC来检索服务器支持的所有DTC的状态。 对此请求的响应包含DTCStatusAvailabilityMask，它提供服务器为掩蔽目的支持的DTC状态位指示。 在DTCStatusAvailabilityMask之后，响应还包含listOfDTCAndStatusRecord，其中包含服务器支持的每个诊断故障代码的DTC编号和相关状态。

11.3.1.12 检索第一个/最近的故障DTC（子功能= 0x0B reportFirstTestFailedDTC，子功能= 0x0D reportMostRecentTestFailedDTC）

客户端可以通过发送分别设置为“reportFirstTestFailedDTC”或“reportMostRecentTestFailedDTC”的子功能字节的请求来从服务器检索第一个/最近的失败DTC。 与DTCStatusAvailabilityMask一起，服务器应将第一个或最近一次失败的DTC编号和相关状态返回给客户端。

如果自上次客户端请求服务器清除诊断信息以来没有记录失败的DTC，则在肯定响应消息中的DTCStatusAvailabilityMask字节之后不应提供DTC /状态信息。 此外，如果自上次客户端请求服务器清除诊断信息以来只有一个DTC失败，则一个失败的DTC应返回给来自客户端的reportFirstTestFailedDTC和reportMostRecentTestFailedDTC请求。

第一个/最近的故障DTC记录应独立于已确认DTC的老化过程。

如上所述，当客户端的ClearDiagnosticInformation请求成功时，第一个/最近的故障DTC信息将被清除（请参阅D.2中的DTC状态位定义，以了解在ClearDiagnosticInformation服务请求接收时DTC状态位处理的进一步描述服务器）。

11.3.1.13 检索第一个/最近检测到的确认DTC（子功能= 0x0C reportFirstConfirmedDTC，subfunction = 0x0E reportMostRecentConfirmedDTC）

客户端可以通过发送分别设置为“reportFirstConfirmedDTC”或“reportMostRecentConfirmedDTC”的子功能字节的请求来从服务器检索第一个/最近确认的DTC。 与DTCStatusAvailabilityMask一起，服务器应将第一个或最近一次确认的DTC编号和相关状态返回给客户端。

如果自上次客户端请求服务器清除诊断信息后没有记录确认的DTC，则在肯定响应消息中的DTCStatusAvailabilityMask字节之后不应提供DTC /状态信息。 另外，如果自上次客户端请求服务器清除诊断信息以来只有1个DTC被确认，则一个确认的DTC应该返回给来自客户端的reportFirstConfirmedDTC和reportMostRecentConfirmedDTC请求。

如果DTC在过去的某个时间点失败，但是在客户的请求时间之前满足了老化标准（不管其后的任何其他DTC是否被确认），则首先确认的DTC的记录应保存下来。上述DTC被证实）。 同样，如果DTC在过去的某个时间点得到确认，但在客户的请求时间之前满足老化标准（假设没有其他DTC被确认后，则最新确认的DTC的记录应保存下来前面提到的DTC失败）。

如上所述，在客户端成功获得ClearDiagnosticInformation请求后，首先/最近确认的DTC信息应被清除。

11.3.1.14 从服务器DTC镜像内存中检索与客户机定义的状态掩码相匹配的DTC列表（子函数= 0x0F reportMirrorMemoryDTCByStatusMask）

子函数reportMirrorMemoryDTCByStatusMask的处理与为reportDTCByStatusMask定义的处理相同，除了所有状态掩码检查都使用存储在服务器的DTC镜像存储器中的DTC执行。 DTC镜像内存是服务器中的另一个可选错误内存，无法通过ClearDiagnosticInformation (0x14) 服务进行擦除。 DTC镜像存储器镜像正常的DTC存储器，例如，如果正常的错误存储器被擦除，则可以使用该存储器。

11.3.1.15 检索镜像内存DCE镜像内存中的客户机定义的DTC掩码和客户机定义的DTCExtendedData记录编号的DTCExtendedData记录数据（subfunction = 0x10 reportMirrorMemoryDTCExtDataRecordByDTCNumber）

子函数reportMirrorMemoryDTCExtDataRecordByDTCNumber的处理与为reportDTCExtDataRecordByDTCNumber定义的处理相同，只是数据从DTC镜像内存中检索出来。 DTC镜像内存是服务器中的另一个可选错误内存，无法通过ClearDiagnosticInformation (0x14) 服务进行擦除。 DTC镜像存储器镜像正常的DTC存储器，例如，如果正常的错误存储器被擦除，则可以使用该存储器。

11.3.1.16 检索与客户端定义的状态掩码相匹配的镜像内存DTC的数量（subfunction = 0x11 reportNumberOfMirrorMemoryDTCByStatusMask）

客户端可以通过发送对具有设置为reportNumberOfMirrorMemoryDTCByStatusMask的子功能的此服务的请求来检索与客户端定义的状态掩码相匹配的镜像内存DTC的数量。 对此请求的响应包含DTCStatusAvailabilityMask，它提供服务器为掩蔽目的支持的DTC状态位指示。 在DTCStatusAvailabilityMask之后，响应包含报告关于DTC格式化和编码的信息的DTCFormatIdentifier。 DTCFormatIdentifier后跟DTCCount参数，该参数是一个2字节的无符号数字数字，包含基于客户端提供的状态掩码的服务器内存中可用的DTC数量。

11.3.1.17 检索与客户端定义的状态掩码相匹配的“唯一与排放相关的OBD”DTC的数量（子函数= 0x12 reportNumberOfEmissionsOBDDTCByStatusMask）

客户端可以通过发送针对此服务的请求并将子功能集设置为reportNumberOfEmissionsOBDDTCByStatusMask来检索匹配客户端定义的状态掩码的“唯一与排放相关的OBD”DTC的数量。 对此请求的响应包含DTCStatusAvailabilityMask，它提供服务器为掩蔽目的支持的DTC状态位指示。 在DTCStatusAvailabilityMask之后，响应包含报告关于DTC格式化和编码的信息的DTCFormatIdentifier。 DTCFormatIdentifier后跟DTCCount参数，该参数是一个2字节的无符号数字数字，包含基于客户端提供的状态掩码的服务器内存中可用的“唯一与排放相关的OBD”DTC的数量。

11.3.1.18 检索与客户端定义的状态掩码相匹配的“唯一与排放相关的OBD”DTC列表（子功能= 0x13 reportEmissionsOBDDTCByStatusMask）

客户端可以通过发送请求并将子功能字节设置为reportEmissionsOBDDTCByStatusMask来检索“仅与排放相关的OBD”DTC列表，该列表满足客户端定义的状态掩码。 该子功能允许客户请求服务器报告“testFailed”或“确认”或“等”的所有“与排放相关的OBD”DTC。 评估应按如下方式进行：服务器应在客户端请求中指定的掩码与服务器支持的每个“与排放相关的OBD”DTC相关的实际状态之间执行按位逻辑AND操作。 除了DTCStatusAvailabilityMask之外，服务器还应返回AND操作结果为非零（即（statusOfDTC & DTCStatusMask）！= 0）的所有“与排放相关的OBD”DTC。 如果客户端指定包含服务器不支持的位的状态掩码，则服务器应仅使用它支持的位处理DTC信息。 如果服务器内没有“与排放相关的OBD”DTC与客户请求中指定的屏蔽标准相匹配，则不应在正响应消息中的DTCStatusAvailabilityMask字节后面提供DTC或状态信息。

“与排放相关的OBD”DTC状态信息应当在来自客户端的ClearDiagnosticInformation请求成功时被清除（请参阅D.2中的DTC状态位定义，以了解在服务器中接收到ClearDiagnosticInformation服务请求时DTC状态位处理的进一步描述）。

11.3.1.19 检索“预先故障的”DTC状态列表（子功能= 0x14 reportDTCFaultDetection-Counter）

客户可以检索在客户请求时已经或尚未检测到“当前”或“已确认”的所有当前“预先故障”DTC的列表。DTCFaultDetectionCounter的意图是一种简单的方法，用于识别特定DTC的statusOfDTC字节无法识别/读取的增长或间歇性问题。 DTCFaultDetectionCounter的内部实现应该是车辆制造商特定的。 “预先故障”DTC的使用案例是为了在制造工厂中对DTC进行测试时加快故障检测速度，这些DTC需要熟化时间，这对于制造测试来说是不可接受的。 维修或安装新组件后，服务具有类似的用例。

11.3.1.20 检索具有“永久性DTC”状态的DTC列表（子功能= 0x15 reportDTCWithPermanentStatus）

如3.1所述，客户可以检索具有“永久性DTC”状态的DTC列表。

11.3.1.21 为客户机定义的DTCExtendedData记录号检索DTCExtendedData记录数据（子函数= 0x16 reportDTCExtDataRecordByRecordNumber）

客户端可以通过发送针对此服务的请求并将子功能集设置为reportDTCExtDataRecordByRecordNumber来为客户端定义的DTCExtendedData记录号检索DTCExtendedData。 服务器应搜索所有支持的DTC，以与客户端指定的DTCExtDataRecordNumber完全匹配。 在这种情况下，客户端请求中提供的DTCExtDataRecordNumber参数应为请求DTCExtendedData的所有支持的DTC指定特定的DTCExtendedData记录。

服务器应为包含所请求的DTCExtDataRecordNumber的数据的每个支持的DTC返回一个DTCExtendedData记录以及DTC编号和statusOfDTC。

车辆制造商应定义DTCExtDataRecord的格式和内容。 DTCExtDataRecord中报告的数据结构由DTCExtDataRecordNumber定义，与记录DataIdentifier中的数据定义类似。

如果客户端指定的DTCExtDataRecordNumber参数无效或服务器不支持，服务器应负面响应。

在接收到ClearDiagnosticInformation服务时清除DTCExtendedData信息在11.2.1中指定。 车辆制造商有责任在存储器溢出的情况下（用于存储的DTC和DTC扩展数据在服务器中完全占用的存储空间）指定删除存储的DTC和DTC扩展数据的规则。

11.3.1.22 从匹配客户端定义的状态掩码的功能组中检索WWH-OBD DTC列表（子功能= 0x42 reportWWHOBDDTCByMaskRecord）

DTCSeverityMask（严重程度和级别）的实施和使用在ISO 27145-3 [17]中定义。

11.3.1.23 检索具有“永久性DTC”状态的WWH-OBD DTC列表（子功能= 0x55 reportWWHOBDDTCWithPermanentStatus）

如3.1所述，客户可以检索具有“永久性DTC”状态的WWH-OBD DTC列表。

11.3.1.24 从服务器的用户定义的DTC存储器中检索与客户机定义的DTC状态掩码相匹配的DTC列表（子函数= 0x17 reportUserDefMemoryDTCByStatusMask）

客户端可以从用户定义的存储器中检索DTC列表，该列表通过发送具有设置为reportUserDefMemoryDTCByStatusMask的子功能字节的请求来满足客户端定义的状态掩码。 该子功能允许客户请求服务器报告用户定义的存储器中所有“testFailed”或“确认”或“等”的DTC。

评估应按如下方式进行：服务器应在客户请求中指定的掩码与服务器在该用户定义的存储器中支持的每个DTC相关的实际状态之间执行按位逻辑AND操作。 除了DTCStatusAvailabilityMask之外，服务器应该返回AND操作结果非零的所有DTC（即，（statusOfDTC&DTCStatusMask）！= 0）在该特定存储器中。 如果客户端指定包含服务器不支持的位的状态掩码，则服务器应仅使用它支持的位处理DTC信息。 如果服务器内没有DTC与特定存储器中客户请求中指定的屏蔽标准匹配，则不应在正响应消息中的DTCStatusAvailabilityMask字节后面提供DTC或状态信息。

DTC状态信息不应在清除客户端的ClearDiagnosticInformation请求时清除，而应由制造商特定的例程控制清除。

11.3.1.25 从DTC用户定义的存储器中检索用户定义的存储器的DTCSnapshot记录数据（subfunction = 0x18 reportUserDefMemoryDTCSnapshotRecordByDTCNumber）用于客户机定义的DTC掩码和客户机定义的DTCSnapshotNumber

客户端可以通过发送对具有设置为reportUserDefMemoryDTCSnapshotRecordByDTCNumber的子功能的此服务的请求，检索客户端定义的DTCMaskRecord的捕获的DTCSnapshot记录数据以及DTCSnapshot记录号和用户定义的内存标识符。 服务器应搜索其支持的DTC与客户指定的DTCMaskRecord（包含DTC编号（高，中，低字节））完全匹配。 客户请求中提供的DTCSnapshotRecordNumber参数应指定特定事件的指定DTC和为其请求DTCSnapshot记录数据的已定义内存。

注1：DTCSnapshotRecordNumber不共享与DTCStoredDataRecordNumber相同的地址空间。

系统供应商/车辆制造商有责任定义在这些服务器内捕获的DTCSnapshot记录是否存储与第一次或最近发生的故障相关的数据。

如果客户端定义的DTCMaskRecord和DTCSnapshotRecordNumber参数（DTCSnapshotRecordNumber不等于0xFF）和客户端定义的DTCSnapshotRecordNumber失败（DTCSnapshotRecordNumber不等于0xFF），则服务器将响应客户端的请求返回单个预定义的DTCSnapshotRecord那个具体的记忆。

注2：准确的故障标准应由系统供应商/车辆制造商定义。

DTCSnapshot记录可能包含多个数据参数，可用于在发生故障时重建车辆状况（例如B +，RPM，时间戳）。

车辆制造商应在用户定义的存储器中定义DTCSnapshotRecord的格式和内容（即DTCSnapshotRecords的内容可以在不同存储器之间有所不同）记录。 DTCSnapshotRecord中报告的数据首先包含一个dataIdentifier以标识随后的数据。 该数据标识符/数据组合可以在DTCSnapshotRecord内重复使用。在用户定义的存储器中的DTCSnapshotRecord中使用一个或多个dataIdentifiers允许针对单个DTC存储不同类型的DTCSnapshotRecords以用于不同发生的故障。 指示每个DTCSnapshotRecord中包含的记录DataIdentifiers的数量的参数应与每个DTCSnapshotRecord一起提供以协助数据检索。

除了客户端已将DTCSnapshotRecordNumber设置为0xFF之外，服务器应在单个响应消息中报告一个DTCSnapshot记录，因为这将导致服务器响应于在单个响应中为客户端定义的DTCMaskRecord和用户定义的存储器存储的所有DTCSnapshot记录信息。 DTCAndStatusRecord只在响应消息中包含一次。

如果客户端指定的DTCMaskRecord，DTCSnapshotRecordNumber，UserDefMemory参数无效或服务器不支持，服务器将作出负面响应。 这与客户端指定的DTCMaskRecord和/或DTCSnapshotRecordNumber参数确实有效并由服务器支持该特定内存但不具有与其关联的DTCSnapshot数据的情况不同（例如，因为从未出现故障事件发生在指定的DTC或记录号上）。 服务器应发送仅包含DTCAndStatusRecord（请求的DTC编号（高，中，低字节）的回声加上statusOfDTC的肯定响应）。

DTCSnapshot信息应根据客户的制造商特定条件（如日常控制）要求清除。 车辆制造商有责任在发生内存溢出（用于存储DTC和DTC快照数据的存储空间在服务器中完全占用该特定内存的存储空间）时指定删除存储的DTC和DTCSnapshot数据的规则。

11.3.1.26 检索用户定义的内存DTCExtendedData记录数据，用于客户机定义的DTC掩码和客户机定义的DCE内存中的DTCExtendedData记录编号（subfunction = 0x19 reportUserDefMemoryDTCExtDataRecordByDTCNumber）

客户端可以通过发送针对该服务的请求并将子功能集设置为reportUserDefMemoryDTCExtDataRecordByDTCNumber来检索客户端定义的DTCMaskRecord的DTCExtendedData以及DTCExtendedData记录号和UserDefMemoryIdenitfier。 服务器应搜索其支持的DTC与客户端指定的DTCMaskRecord（包含DTC编号（高，中，低字节））和UserDefMemoryIdentifier的完全匹配。 在这种情况下，客户端请求中提供的DTCExtDataRecordNumber参数应指定DTCExtendedData所请求的指定DTC的特定DTCExtendedData记录。

与DTC编号和statusOfDTC一起，服务器将响应客户端请求（DTCExtDataRecordNumber不等于0xFE或0xFF）返回单个预定义的DTCExtendedData记录。

车辆制造商应定义UserDefDTCExtDataRecord的格式和内容。 DTCExtDataRecord中报告的数据结构由DTCExtDataRecordNumber定义，该DTCExtDataRecordNumber用于与记录DataIdentifier中的数据定义类似的特定用户定义内存。 响应中可能包含多个DTCExtDataRecordNumbers和相关的DTCExtDataRecords。 使用一个或多个DTCExtDataRecordNumbers允许为单个DTC存储不同类型的DTCExtDataRecords。

除了客户端已将DTCExtDataRecordNumber设置为0xFE或0xFF之外，服务器应在单个响应消息中报告一个DTCExtendedData记录，因为这将导致服务器响应存储在客户端定义的DTCMaskRecord中的所有DTCExtendedData记录，一个响应消息。

如果客户端指定的DTCMaskRecord或DTCExtDataRecordNumber参数无效或服务器不支持，或者不在该特定内存中，服务器应负面响应。 这与客户端指定的DTCMaskRecord和/或DTCExtDataRecordNumber参数确实有效且受服务器支持但不具有与其关联的DTC扩展数据的情况（例如，由于扩展数据的内存溢出）。 在reportDTCExtDataRecordByDTCNumber的情况下，服务器应发送只包含DTCAndStatusRecord（请求的DTC编号的回声（高，中，低字节）加上statusOfDTC的肯定响应）。

车辆制造商有责任指定在用户定义的存储器中删除存储的DTC和DTC扩展数据的规则。

11.3.2 请求消息

11.3.2.1 请求消息定义

表256基于所使用的子功能参数定义了ReadDTCInformation请求消息的结构。

表256 - 请求消息定义 - 子功能= reportNumberOfDTCByStatusMask, reportDTCByStatusMask, reportMirrorMemoryDTCByStatusMask, reportNumberOfMirrorMemoryDTCByStatusMask, reportNumberOfEmissionsOBDDTCByStatusMask, reportEmissionsOBDDTCByStatusMask

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportNumberOfDTCByStatusMask reportDTCByStatusMask reportMirrorMemoryDTCByStatusMask reportNumberOfMirrorMemoryDTCByStatusMask reportNumberOfEmissionsOBDDTCByStatusMask reportEmissionsOBDDTCByStatusMask]	M	 0x01 0x02 0x0F 0x11 0x12 0x13	LEV_ RNODTCBSM RDTCBSM RMMDTCBSM RNOMMDTCBSM RNOOEBDDTCBSM ROBDDTCBSM
#3	DTCStatusMask	M	0x00 - 0xFF	DTCSM

表257基于所使用的子功能参数来定义ReadDTCInformation请求消息的结构。

表257 - 请求消息定义 - 子功能= reportDTCSnapshotIdentification, reportDTCSnapshotRecordByDTCNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportDTCSnapshotIdentification reportDTCSnapshotRecordByDTCNumber]	M	 0x03 0x04	LEV_ RDTCSSI RDTCSSBDTC
	DTCMaskRecord[] = [			DTCMREC_
#3	DTCHighByte	C	0x00 - 0xFF	DTCHB
#4	DTCMiddleByte	C	0x00 - 0xFF	DTCMB
#5	DTCLowByte]	C	0x00 - 0xFF	DTCLB
#6	DTCSnapshotRecordNumber	C	0x00 - 0xFF	DTCSSRN
C: DTCMaskRecord记录和DTCSnapshotRecordNumber参数仅在subfunction参数等于reportDTCSnapshotRecordByDTCNumber的情况下存在。				

表258基于所使用的子功能参数定义了ReadDTCInformation请求消息的结构。

表258 - 请求消息定义 - 子功能= reportDTCStoredDataByRecordNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportDTCStoredDataByRecordNumber]	M	 0x05	LEV_ RDTCSDBRN
#3	DTCStoredDataRecordNumber	M	0x00 - 0xFF	DTCSDRN

表259基于所使用的子功能参数来定义ReadDTCInformation请求消息的结构。

表259 - 请求消息定义 - 子功能= reportDTCExtDataRecordByDTCNumber, reportMirrorMemoryDTCExtDataRecordByDTCNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportDTCExtDataRecordByDTCNumber reportMirrorMemoryDTCExtDataRecordByDTCNumber]	M	0x06 0x10	LEV_ RDTCEDRBDN RMDEDRBDN
	DTCMaskRecord[] = [			DTCMREC_
#3	DTCHighByte	M	0x00 - 0xFF	DTCHB
#4	DTCMiddleByte	M	0x00 - 0xFF	DTCMB
#5	DTCLowByte]	M	0x00 - 0xFF	DTCLB
#6	DTCExtDataRecordNumber	M	0x00 - 0xFF	DTCEDRN

表260基于所使用的子功能参数定义了ReadDTCInformation请求消息的结构。

表260 - 请求消息定义 - 子功能= reportNumberOfDTCBySeverityMaskRecord, reportDTCSeverityInformation

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportNumberOfDTCBySeverityMaskRecord reportDTCBySeverityMaskRecord]	M	0x07 0x08	LEV_ RNODTCBSMR RDTCBSMR
	DTCSeverityMaskRecord[] = [			DTCSVMREC_
#3	DTCSeverityMask	M	0x00 - 0xFF	DTCSVM
#4	DTCStatusMask]	M	0x00 - 0xFF	DTCSM

表261基于所使用的子功能参数定义了ReadDTCInformation请求消息的结构。

表261 - 请求消息定义 - 子功能= reportSeverityInformationOfDTC

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportSeverityInformationOfDTC]	M	0x09	LEV_ RSIODTC
	DTCMaskRecord[] = [			DTCMREC_
#3	DTCHighByte	M	0x00 - 0xFF	DTCHB
#4	DTCMiddleByte	M	0x00 - 0xFF	DTCMB
#5	DTCLowByte]	M	0x00 - 0xFF	DTCLB

表262基于所使用的子功能参数来定义ReadDTCInformation请求消息的结构。

表262 - 请求消息定义 - 子功能= reportSupportedDTC, reportFirstTestFailedDTC, reportFirstConfirmedDTC, reportMostRecentTestFailedDTC, reportMostRecentConfirmedDTC, reportDTCFaultDetectionCounter, reportDTCWithPermanentStatus

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportSupportedDTC reportFirstTestFailedDTC reportFirstConfirmedDTC reportMostRecentTestFailedDTC reportMostRecentConfirmedDTC reportDTCFaultDetectionCounter reportDTCWithPermanentStatus]	M	 0x0A 0x0B 0x0C 0x0D 0x0E 0x14 0x15	LEV_ RSUPDTC RFTFDTC RFCDTC RMRTFDTC RMRCDTC RDTCFDC RDTCWPS

表格263基于所使用的子功能参数来定义ReadDTCInformation请求消息的结构。

表263 - 请求消息定义 - 子功能= reportDTCExtDataRecordByRecordNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportDTCExtDataRecordByRecordNumber]	M	0x16	LEV_ RDTCEDRBR
#3	DTCExtDataRecordNumber	M	0x00 - 0xEF	DTCEDRN

表264基于所使用的子功能参数定义了ReadDTCInformation请求消息的结构。

表264 - 请求消息定义 - 子功能= reportUserDefMemoryDTCByStatusMask

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportUserDefMemoryDTCByStatusMask	M	0x17	LEV_ RUDMDTCBSM
#3	DTCStatusMask	M	0x00 -0xFF	DTCSM
#4	MemorySelection	M	0x00 - 0xFF	MEMYS

表265基于所使用的子功能参数来定义ReadDTCInformation请求消息的结构。

表265 – 请求消息定义 – 子功能= reportUserDefMemoryDTCSnapshotRecordByDTCNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportUserDefMemoryDTCSnapshotRecordByDTCNumber]	M	0x18	LEV_ RUDMDTCSSBDTC
	DTCMaskRecord[] = [			DTCMREC_
#3	DTCHighByte	M	0x00 – 0xFF	DTCHB
#4	DTCMiddleByte	M	0x00 – 0xFF	DTCMB
#5	DTCLowByte]	M	0x00 – 0xFF	DTCLB
#6	DTCSnapshotRecordNumber	M	0x00 – 0xFF	DTCSSRN
#7	MemorySelection	M	0x00 – 0xFF	MEMYS

表266基于所使用的子功能参数来定义ReadDTCInformation请求消息的结构。

表266 – 请求消息定义 – 子功能= reportUserDefMemoryDTCExtDataRecordByDTCNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportUserDefMemoryDTCExtDataRecordByDTCNumber]	M	0x19	LEV_ RUDMDTCEDRBDN
	DTCMaskRecord[] = [			DTCMREC_
#3	DTCHighByte	M	0x00 – 0xFF	DTCHB
#4	DTCMiddleByte	M	0x00 – 0xFF	DTCMB
#5	DTCLowByte]	M	0x00 – 0xFF	DTCLB
#6	DTCExtDataRecordNumber	M	0x00 – 0xFF	DTCEDRN
#7	MemorySelection	M	0x00 – 0xFF	MEMYS

表267基于所使用的子功能参数来定义ReadDTCInformation请求消息的结构。

表267 – 请求消息定义 – 子功能= reportWWHOBDDTCByMaskRecord

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportWWHOBDDTCByMaskRecord]	M	0x42	LEV_ ROBDDTCBMR
#3	FunctionalGroupIdentifier	M	0x00 – 0xFF	FGID
	DTCSeverityMaskRecord[] = [			DTCSVMREC_
#4	DTCStatusMask	M	0x00 – 0xFF	DTCSM
#5	DTCSeverityMask]	M	0x00 – 0xFF	DTCSVM

表268基于所使用的子功能参数来定义ReadDTCInformation请求消息的结构。

表268 – 请求消息定义 – 子功能= reportWWHOBDDTCWithPermanentStatus

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation请求SID	M	0x19	RDTCI
#2	子函数= [reportType = reportWWHOBDDTCWithPermanentStatus]	M	0x55	LEV_ RWWHOBDDTCWPS
#3	FunctionalGroupIdentifier	M	0x00 – 0xFF	FGID

11.3.2.2 请求消息子函数参数$ Level (LEV_) 定义

该服务使用子功能参数来选择表269中指定的DTC报告类型之一。下面详细说明可能级别的说明和用法（suppressPosRspMsgIndicationBit (bit 7) 未显示）。

表269 – 请求消息子功能定义

位6 – 0	描述	Cvt	助记符
0x00	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
0x01	**reportNumberOfDTCByStatusMask** 此参数指定服务器应将与客户机定义的状态掩码相匹配的DTC数传送给客户机。	U	RNODTCBSM
0x02	**reportDTCByStatusMask** 该参数指定服务器应向客户端发送与客户端定义的状态掩码相匹配的DTC列表和相应的状态。	U	RDTCBSM
0x03	**reportDTCSnapshotIdentification** 该参数指定服务器应将所有DTCSnapshot数据记录标识（DTC编号和DTCSnapshot记录编号）传送给客户端。	U	RDTCSSI
0x04	**reportDTCSnapshotRecordByDTCNumber** 此参数指定服务器应将与客户机定义的DTC编号和DTCSnapshot记录编号（所有记录为0xFF）关联的DTCSnapshot记录传输到客户机。	U	RDTCSSBDTC
0x05	**reportDTCStoredDataByRecordNumber** 此参数指定服务器应向客户端传输与客户端定义的DTCStoredData记录号（所有记录为0xFF）相关联的DTCStoredDatarecord。	U	RDTCSDBRN
0x06	**reportDTCExtDataRecordByDTCNumber** 此参数指定服务器应向客户端传输与客户端定义的DTC号和DTCExtendedData记录号（所有记录为0xFF，所有OBD记录为0xFE）相关联的DTCExtendedData记录。	U	RDTCEDRBDN
0x07	**reportNumberOfDTCBySeverityMaskRecord** 此参数指定服务器应向客户端传输与客户机定义的严重性掩码记录相匹配的DTC数量。	U	RNODTCBSMR

表269 – （续）

位6 – 0	描述	Cvt	助记符
0x08	**reportDTCBySeverityMaskRecord** 此参数指定服务器应向客户端传输与客户机定义的严重性掩码记录匹配的DTC列表和相应的状态。	U	RDTCBSMR
0x09	**reportSeverityInformationOfDTC** 此参数指定服务器应向客户端传输客户端请求消息中指定的特定DTC的严重性信息。	U	RSIODTC
0x0A	**reportSupportedDTC** 此参数指定服务器应向客户端发送服务器内支持的所有DTC和相应状态的列表。	U	RSUPDTC
0x0B	**reportFirstTestFailedDTC** 此参数指定服务器应向客户端发送自上次清除诊断信息后服务器检测到的第一个故障DTC。 请注意，通过此子功能参数报告的信息应独立于DTC是否已确认或老化。	U	RFTFDTC
0x0C	**reportFirstConfirmedDTC** 此参数指定服务器应向客户端发送自上次清除诊断信息后服务器检测到的第一个确认的DTC。 通过此子功能参数报告的信息应独立于已确认的DTC的老化过程（例如，如果DTC老化以使其状态被允许重置，则首先确认的DTC记录应由服务器继续保存，无论的其他任何DTC会在以后得到确认）。	U	RFCDTC
0x0D	**reportMostRecentTestFailedDTC** 此参数指定服务器应向客户端发送自上次清除诊断信息后服务器检测到的最新失败DTC。 请注意，通过此子功能参数报告的信息应独立于DTC是否已确认或老化。	U	RMRTFDTC
0x0E	**reportMostRecentConfirmedDTC** 该参数指定服务器应向客户端发送自上次清除诊断信息后服务器将检测到的最近一次确认的DTC。 请注意，通过此子功能参数报告的信息应与已确认的DTC的老化过程无关（例如，如果DTC老化以使其状态可以重置，则首先确认的DTC记录应由服务器继续保存假设之后没有其他DTC被证实）。	U	RMRCDTC
0x0F	**reportMirrorMemoryDTCByStatusMask** 此参数指定服务器应将客户端DTC镜像内存列表中的DTC列表以及与客户端定义的状态掩码匹配的相应状态发送给客户端。	U	RMMDTCBSM

表269 – （续）

位6 – 0	描述	Cvt	助记符
0x10	**reportMirrorMemoryDTCExtDataRecordByDTCNumber** 此参数指定服务器应向客户端传输与客户端定义的DTC编号和DTCExtendedData记录编号（所有记录为0xFF，所有OBD记录均为0xFE）DTC关联的DTCExtendedData记录（在DTC镜像存储器外）。	U	RMMDEDRBDN
0x11	**reportNumberOfMirrorMemoryDTCByStatusMask** 此参数指定服务器应将与匹配客户端定义的状态掩码的镜像内存中的DTC数量传送给客户端。	U	RNOMMDTCBSM
0x12	**reportNumberOfEmissionsOBDDTCByStatusMask** 此参数指定服务器应向客户端传输与客户端定义的状态掩码相匹配的与排放相关的OBD DTC的数量。 报告的OBD DTC数量只能是与排放相关的法律要求相一致的OBD DTC数量。	U	RNOOEOBDDTCBSM
0x13	**reportEmissionsOBDDTCByStatusMask** 此参数指定服务器应将与排放相关的OBD DTC列表以及与客户端定义的状态掩码相匹配的相应状态发送给客户端。 所报告的OBD DTC列表只能是与排放相关的法律要求相一致的列表。	U	ROBDDTCBSM
0x14	**reportDTCFaultDetectionCounter** 此参数指定服务器应将已经或尚未检测到“当前”或“已确认”的当前“预先故障”DTC列表发送给客户端。 DTCFaultDetectionCounter的意图是一种简单的方法，用于识别特定DTC的statusOfDTC字节无法识别/读取的增长或间歇性问题。DTCFaultDetectionCounter的内部实现应该是车辆制造商特定的（例如，字节数，带符号与无符号等），但报告的值应为缩放的1字节有符号值，以便+127（0x7F）表示“失败“并且任何其他非零正值表示”预失败“的测试结果。 但DTCFaultDetectionCounter的值为+127的DTC不应按照以下规定报告。 每次测试逻辑运行时，DTCFaultDetectionCounter应按车辆制造商特定数量增加，并指示该测试运行失败。 报告的DTCFaultDetectionCounter值大于零且小于+127（即0x01 – 0x7E）表示已满足DTC启用条件，并且未完成的测试结果至少在一个条件或阈值内预先失效。 必须报告具有非零正值小于+127（0x7F）的DTCFaultDetectionCounters的DTC。 每次测试逻辑运行时，DTCFaultDetectionCounter应按车辆制造商的具体数量递减，并指示该测试运行的合格。 如果DTCFaultDetectionCounter递减到零或低于DTC，则不再在肯定响应消息中报告。DTCFaultDetectionCounter的值不得在操作周期之间保持不变。 如果收到ClearDiagnosticInformation服务请求，则所有DTC的DTCFaultDetectionCounter值都应重置为零。 额外的复位条件应由车辆制造商定义。 有关示例实现细节，请参阅D.5。	U	RDTCFDC

表269 – （续）

位6 – 0	描述	Cvt	助记符
0x15	**reportDTCWithPermanentStatus** 此参数指定服务器应将3.1中描述的具有“永久性DTC”状态的DTC列表传送给客户端。	U	RDTCWPS
0x16	**reportDTCExtDataRecordByRecordNumber** 此参数指定服务器应向客户端传输与客户端定义的DTCExtendedData记录号相关的DTCExtendedData记录，该记录号小于0xF0。	U	RDTCEDBR
0x17	**reportUserDefMemoryDTCByStatusMask** 此参数指定服务器应向客户端发送用户定义的DTC内存中的DTC列表以及与客户端定义的状态掩码相匹配的相应状态。	U	RUDMDTCBSM
0x18	**reportUserDefMemoryDTCSnapshotRecordByDTCNumber** 此参数指定服务器应将客户定义的DTC编号和DTCSnapshot记录编号（所有记录的0xFF）关联的DTCSnapshot记录传输到客户端 – 在用户定义的DTC内存之外。	U	RUDMDTCSSBDTC
0x19	**reportUserDefMemoryDTCExtDataRecordByDTCNumber** 此参数指定服务器应向客户端传输与用户定义的DTC编号和DTCExtendedData记录编号（所有记录的0xFF）关联的DTCExtendedData记录 – 在用户定义的DTC存储器之外。	U	RUDMDTCEDRBDN
0x1A – 0x41	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
0x42	**reportWWHOBDDTCByMaskRecord** 此参数指定服务器应将客户端定义的状态掩码和严重性掩码记录匹配的WWHOBD DTC列表以及相应的状态和严重级别信息发送给客户端。	U	RWWHOBDDTCBMR
0x43 – 0x54	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
0x55	**reportWWHOBDDTCWithPermanentStatus** 该参数指定服务器应向客户端发送3.1中描述的具有“永久性DTC”状态的WWHOBD DTC列表。	U	RWWHOBDDTCWPS
0x56 – 0x7F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

11.3.2.3 请求消息数据参数定义

表270指定了请求消息的数据参数。

表270 – 请求数据参数定义

定义
DTCStatusMask DTCStatusMask包含八（8）个DTC状态位。 每个八位的定义可以在D.2中找到。 该字节用于请求消息中，以允许客户端请求状态与DTCStatusMask相匹配的DTC的DTC信息。 如果任何一个DTC实际状态位被设置为'1'并且DTCStatusMask中的相应状态位也被设置为'1'（即，如果DTCStatusMask按位进行逻辑“与”运算，则DTC状态与DTCStatusMask匹配DTC的实际状态和结果非零，则发生匹配）。 如果客户端指定包含服务器不支持的位的状态掩码，则服务器应仅使用它支持的位处理DTC信息。
DTCMaskRecord [DTCHighByte, DTCMiddleByte, DTCLowByte] DTCMaskRecord是一个3字节的值，包含DTCHighByte，DTCMiddleByte和DTCLowByte，它们共同表示服务器支持的特定诊断故障代码的唯一标识号。 3字节DTC编号的定义允许多种编码DTC信息的方法。 它可以做到 — 通过使用根据ISO 15031-6 [12]规范的DTCHighByte，DTCMiddleByte和DTCLowByte的解码。 这种格式由DTCFormatIdentifier = SAE_J2012-DA_DTCFormat_00标识，或 — 通过使用根据ISO 14229的该部分的DTCHighByte，DTCMiddleByte和DTCLowByte的解码，该部分没有指定任何解码方法，因此允许车辆制造商定义的解码方法。 这种格式由DTCFormatIdentifier = ISO_14229-1_DTCFormat或者 — 通过使用根据SAE J1939-73 [19]规范的DTCHighByte，DTCMiddleByte和DTCLowByte的解码。 这种格式由DTCFormatIdentifier = SAE_J1939-73_DTCFormat标识，或 — 根据ISO 11992-4 [5]规范使用DTCHighByte，DTCMiddleByte和DTCLowByte的解码。 这种格式由DTCFormatIdentifier = ISO_11992-4_DTCFormat标识。 — 通过使用根据ISO 27145-2 [16]规范的DTCHighByte，DTCMiddleByte和DTCLowByte的解码。 这种格式由DTCFormatIdentifier = SAE_J2012-DA_WWH-OBD_DTCFormat标识。
DTCSnapshotRecordNumber DTCSnapshotRecordNumber是一个1字节的值，指示通过reportDTCSnapshotByDTCNumber子函数为客户机定义的DTCMaskRecord请求的特定DTCSnapshot数据记录的编号。 DTCSnapshot数据记录号0x00应保留用于立法目的（例如，WWH-OBD）。 在0x01到0xFE范围内的DTCSnapshot记录应可用于车辆制造商的特定用途。 值0xFF请求服务器立即报告所有存储的DTCSnapshot数据记录。
DTCStoredDataRecordNumber DTCStoredDataRecordNumber是一个1字节的值，指示通过reportDTCStoredDataByRecordNumber子函数请求的特定DTCStoredDataRecord的编号。 DTCStoredDataRecordNumber 0x00应保留用于立法目的。 在0x01到0xFE范围内的DTCStoredData记录应可用于车辆制造商的特定用途。 值0xFF请求服务器立即报告所有存储的DTCStoredData数据记录。

表270 – （续）

定义
DTCExtDataRecordNumber DTCExtDataRecordNumber 是一个1字节值，表示通过reportDTCExtDataRecordByDTCNumber和reportDTCExtDataRecordByRecordNumber子函数为客户机定义的DTCMaskRecord请求的特定DTCExtendedData记录的编号。对于与排放相关的服务器（符合OBD标准的ECU），DTCExtDataRecordNumber 0x00应保留给将来的OBD使用。 以下DTCExtDataRecordNumber范围是保留的: — ISO / SAE保留0x00的值。 — 值0x01 – 0x8F请求服务器报告车辆制造商特定存储的DTCExtendedData记录。 — 值0x90 – 0xEF请求服务器报告立法的OBD存储的DTCExtendedData记录。 — ISO / SAE保留0xF0 – 0xFD的值，以便将来在单个响应消息中报告组。 — 0xFE的值请求服务器在单个响应消息中报告所有立法的OBD存储的DTCExtendedData记录。 — 值0xFF请求服务器在单个响应消息中报告所有存储的DTCExtendedData记录。
DTCSeverityMaskRecord [DTCSeverityMask, DTCStatusMask] DTCSeverityMaskRecord是一个2字节的值，包含DTCSeverityMask和DTCStatusMask（见D.3和D.2）。
DTCSeverityMask DTCSeverityMask包含三个DTC严重性位。 三位中每一位的定义可以在D.3中找到。 该字节用于请求消息中，以允许客户端请求严重性定义与DTCSeverityMask匹配的DTC的DTC信息。 如果DTC实际严重性位中的任何一个被设置为'1'并且DTCSeverityMask中的对应严重性位也被设置为'1'（即，如果DTCSeverityMask按位进行逻辑“与”运算，则DTC严重性定义匹配DTCSeverityMask DTC的实际严重性和结果非零，则发生匹配）。
FunctionalGroupIdentifier FunctionalGroupIdentifier已被引入，以区分由多种不同的ECU组成的电气架构内的不同功能系统组之间由测试设备发送的命令。 如果ECU执行了排放系统的软件以及在I / M测试期间可能检查的其他系统，重要的是仅报告所请求的功能系统组的DTC信息。 I / M测试不应该失败，因为另一个功能系统组存储了DTC信息。 FunctionalGroupIdentifiers在D.5中指定。
MemorySelection 检索DTC时，应使用此参数来寻址各自的用户定义的DTC存储器。

11.3.3 积极的回应消息

11.3.3.1 积极响应消息的定义

服务ReadDTCInformation请求的正面响应取决于服务请求中的子功能。

表271定义了子函数参数的肯定响应消息格式。

表271 - 响应消息定义 - 子函数= reportNumberOfDTCByStatusMask, reportNumberOfDTCBySeverityMaskRecord, reportNumberOfMirrorMemoryDTCByStatusMask, reportNumberOfEmissionsOBDDTCByStatusMask

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportNumberOfDTCByStatusMask reportNumberOfDTCBySeverityMaskRecord reportNumberOfMirrorMemoryDTCByStatusMask reportNumberOfEmissionsOBDDTCByStatusMask]	M	 0x01 0x07 0x11 0x12	LEV_ RNODTCBSM RNODTCBSMR RNOMMDTCBSM RNOOEOBDDTCBSM
#3	DTCStatusAvailabilityMask	M	0x00 - 0xFF	DTCSAM
#4	DTCFormatIdentifier = [SAE_J2012-DA_DTCFormat_00 ISO_14229-1_DTCFormat SAE_J1939-73_DTCFormat ISO_11992-4_DTCFormat SAE_J2012-DA_DTCFormat_04]	M	 0x00 0x01 0x02 0x03 0x04	DTCFID_ J2012-DADTCF00 14229-1DTCF J1939-73DTCF 11992-4DTCF J2012-DADTCF04
 #5 #6	DTCCount[] = [DTCCountHighByte DTCCountLowByte]	 M M	 0x00 - 0xFF 0x00 - 0xFF	DTCC_ DTCCHB DTCCLB

表272定义了子函数参数的肯定响应消息格式。

表272 - 响应消息定义 - 子功能= reportDTCByStatusMask, reportSupportedDTCs, reportFirstTestFailedDTC, reportFirstConfirmedDTC, reportMostRecentTestFailedDTC, reportMostRecentConfirmedDTC, reportMirrorMemoryDTCByStatusMask, reportEmissionsOBDDTCByStatusMask, reportDTCWithPermanentStatus

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2		M	 0x02 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x13 0x15	LEV_ RDTCBSM RSUPDTC RFTFDTC RFCDTC RMRTFDTC RMRCDTC RMMDTCBSM ROBDDTCBSM RDTCWPS
#3	DTCStatusAvailabilityMask	M	0x00 - 0xFF	DTCSAM
 #4 #5 #6 #7 #8 #9 #10 #11 : #n-3 #n-2 #n-1 #n		 C_1 C_1 C_1 C_1 C_2 C_2 C_2 C_2 : C_2 C_2 C_2 C_2	 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF : 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF	DTCASR_ DTCHB DTCMB DTCLB SODTC DTCHB DTCMB DTCLB SODTC : DTCHB DTCMB DTCLB SODTC

表272 – （续）

A_Data字节	参数名称	Cvt	字节值	助记符
C_1: 该参数仅在可以报告DTC信息时才存在。 C_2: 仅当reportType = reportSupportedDTCs, reportDTCByStatusMask, reportMirrorMemoryDTCByStatusMask, reportEmissionsOBDDTCByStatusMask, reportDTCWithPermanentStatus和多个DTC信息可用于报告时，才会显示此参数。				

表273定义了子函数参数的肯定响应消息格式。

表273 – 响应消息定义 – 子功能= reportSnapshotIdentification

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportDTCSnapshotIdentification]	M	0x03	LEV_ RDTCSSI
#3 #4 #5	DTCRecord []#1 = [DTCHighByte#1 DTCMiddleByte #1 DTCLowByte#1]	C_1 C_1 C_1	0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF	DTCASR_ DTCHB DTCMB DTCLB
#6	DTCSnapshotRecordNumber#1	C_1	0x00 – 0xFF	DTCSSRN
:	:	:	:	:
#n-3 #n-2 #n-1	DTCRecord []#m = [DTCHighByte#m DTCMiddleByte #m DTCLowByte#m]	C_2 C_2 C_2	0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF	DTCASR_ DTCHB DTCMB DTCLB
#n	DTCSnapshotRecordNumber#m的	C_2	0x00 – 0xFF	DTCSSRN
C_1: 如果至少可以报告一个DTCSnapshot记录，则仅存在DTCRecord和DTCSnapshotRecordNumber参数。 C_2: 如果可以报告多个DTCSnapshot记录，则只有DTCRecord和DTCSnapshotRecordNumber参数存在。				

表274定义了子函数参数的肯定响应消息格式。

表274 – 响应消息定义 – 子功能= reportDTCSnapshotRecordByDTCNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportDTCSnapshotRecordByDTCNumber]	M	0x04	LEV_ RDTCSSBDTC
#3 #4 #5 #6	DTCAndStatusRecord[] = [DTCHighByte DTCMiddleByte DTCLowByte statusOfDTC]	M M M M	0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF	DTCASR_ DTCHB DTCMB DTCLB SODTC
#7	DTCSnapshotRecordNumber#1	C_1	0x00 – 0xFF	DTCSSRN
#8	DTCSnapshotRecordNumberOfIdentifiers#1	C_1	0x00 – 0xFF	DTCSSRNI

表274 - （续）

A_Data字节	参数名称	Cvt	字节值	助记符
#9		C_1	0x00 - 0xFF	DTCSSR_DIDB11
#10		C_1	0x00 - 0xFF	DIDB12
#11		C_1	0x00 - 0xFF	SSD11
:		C_1	:	:
#11+(p-1)		C_1	0x00 - 0xFF	SSD1p
:		:	:	:
#R-(M-1)-2		C_2	0x00 - 0xFF	DIDB21
#R-(M-1)-1		C_2	0x00 - 0xFF	DIDB22
#R-(M-1)		C_2	0x00 - 0xFF	SSD21
:		C_2	:	:
#r		C_2	0x00 - 0xFF	SSD2m
:	:	:	:	:
#t	DTCSnapshotRecordNumber#X	C_3	0x00 - 0xFF	DTCSSRN
#t+1	DTCSnapshotRecordNumberOfIdentifiers#X	C_3	0x00 - 0xFF	DTCSSRNI
#t+2		C_3	0x00 - 0xFF	DTCSSR_DIDB11
#t+3		C_3	0x00 - 0xFF	DIDB12
#t+5		C_3	0x00 - 0xFF	SSD11
:		C_3	:	:
#T+5+(P-1)		C_3	0x00 - 0xFF	SSD1p
:		:	:	:
#N-(U-1)-2		C_4	0x00 - 0xFF	DIDB21
#N-(U-1)-1		C_4	0x00 - 0xFF	DIDB22
#N-(U-1)		C_4	0x00 - 0xFF	SSD21
:		C_4	:	:
#n		C_4	0x00 - 0xFF	SSD2u

C_1: 只有至少有一个DTCSnapshot记录可用于报告时，DTCSnapshotRecord参数中的DTCSnapshotRecordNumber和第一个dataIdentifier / snapshotData组合才存在。

C_2/ C_4允许在单个DTCSnapshotRecord中存在多个dataIdentifier / snapshotData组合。 例如，对于单个数据标识符仅引用数据的整数部分的情况，情况可能如此。 当dataIdentifier引用一个数据块时，可以使用单个dataIdentifier / snapshotData组合。

C_3: DTCSnapshotRecord参数中的DTCSnapshotRecordNumber和第一个dataIdentifier / snapshotData组合仅在请求报告所有记录（请求中将DTCSnapshotRecordNumber设置为0xFF）并且可以报告多条记录时才存在。

表275定义了子函数参数的肯定响应消息格式。

表275 - 响应消息定义 - 子功能= reportDTCStoredDataByRecordNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportDTCStoredDataByRecordNumber]	M	0x05	LEV_RDTCSDBRN
#3	DTCStoredDataRecordNumber#1	M	0x00 - 0xFF	DTCSDRN
	DTCAndStatusRecord []#1 = [			DTCASR_
#4	DTCHighByte	C_1	0x00 - 0xFF	DTCHB
#5	DTCMiddleByte	C_1	0x00 - 0xFF	DTCMB
#6	DTCLowByte	C_1	0x00 - 0xFF	DTCLB
#7	statusOfDTC]			SODTC

		C_1	0x00 – 0xFF	

表275 – （续）

A_Data字节	参数名称	Cvt	字节值	助记符
#8	DTCStoredDataRecordNumberOfIdentifiers#1	C_1	0x00 – 0xFF	DTCSDRNI
#9		C_1	0x00 – 0xFF	DTCSDR_DIDB11
#10		C_1	0x00 – 0xFF	DIDB12
#11		C_1	0x00 – 0xFF	DSD11
:		:	:	:
#11+(p-1)		C_1	0x00 – 0xFF	DSD1p
:		:	:	:
#R- (M-1)-2		C_2	0x00 – 0xFF	DIDB21
#R- (M-1)-1		C_2	0x00 – 0xFF	DIDB22
#R- (M-1)		C_2	0x00 – 0xFF	DSD21
:		:	:	:
#r		C_2	0x00 – 0xFF	DSD2m
:	:	:	:	:
#t	DTCStoredDataRecordNumber#X	C_3	0x00 – 0xFF	DTCSDRN
#t+1		C_3	0x00 – 0xFF	DTCASR_DTCHB
#t+2		C_3	0x00 – 0xFF	DTCMB
#t+3		C_3	0x00 – 0xFF	DTCLB
#t+4		C_3	0x00 – 0xFF	SODTC
#t+5	DTCStoredDataRecordNumberOfIdentifiers#X	C_3	0x00 – 0xFF	DTCSDRNI
#t+6		C_3	0x00 – 0xFF	DTCSDR_DIDB11
#t+7		C_3	0x00 – 0xFF	DIDB12
#t+8		C_3	0x00 – 0xFF	DSD11
:		:	:	:
#吨+ 8 + (P-1)		C_3	0x00 – 0xFF	DSD1p
:		:	:	:
#N- (U-1)-2		C_4	0x00 – 0xFF	DIDB21
#N- (U-1)-1		C_4	0x00 – 0xFF	DIDB22
#N- (U-1)		C_4	0x00 – 0xFF	DSD21
:		:	:	:
#n		C_4	0x00 – 0xFF	DSD2u

C_1: 如果至少有一个DTCStoredData记录可用于报告，则仅在DTCStoredDataRecord参数中存在DTCAndStatusRecord和第一个dataIdentifier / DTCStoredData组合。

C_2/ C_4允许在单个DTCStoredDataRecord中存在多个dataIdentifier / DTCStoredData组合。 例如，对于单个数据标识符仅引用数据的整数部分的情况，情况可能如此。 当dataIdentifier引用一个数据块时，可以使用单个dataIdentifier / DTCStoredData组合。

C_3: DTCStoredDataRecord参数中的DTCStoredDataRecordNumber，DTCAndStatusRecord和第一个dataIdentifier / DTCStoredData组合仅在请求报告所有记录（请求中DTCStoredDataRecordNumber设置为0xFF）并且有多个记录可用时才存在报道。

表276定义了子函数参数的肯定响应消息格式。

表276 - 响应消息定义 - 子函数= reportDTCExtDataRecordByDTCNumber和 reportMirrorMemoryDTCExtDataRecordByDTCNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportDTCExtDataRecordByDTCNumber reportMirrorMemoryDTCExtDataRecordByDTCNumber]	M	 0x06 0x10	LEV_ RDTCEDRBD RMDEDRBDN
 #3 #4 #5 #6	DTCAndStatusRecord[] = [DTCHighByte DTCMiddleByte DTCLowByte statusOfDTC]	 M M M M	 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF	DTCASR_ DTCHB DTCMB DTCLB SODTC
#7	DTCExtDataRecordNumber#1	C_1	0x00-0xFD	DTCEDRN
 #8 : #8+(p-1)	DTCExtDataRecord []#1 = [extendedData#1字节#1 : 扩展数据#1字节#p]	 C_1 C_1 C_1	 0x00 - 0xFF : 0x00 - 0xFF	DTCSSR_ EDD11 : EDD1p
:	:	:	:	:
#t	DTCExtDataRecordNumber#X	C_2	0x00 - 0xFD	DTCEDRN
 #t+1 : #T + 1 + (Q-1)	DTCExtDataRecord []#x = [extendedData#x字节#1 : extendedData#x byte#q]	 C_2 C_2 C_2	 0x00 - 0xFF : 0x00 - 0xFF	DTCSSR_ EDDx1 : EDDxq

C_1: DTCExtDataRecord参数中的DTCExtDataRecordNumber和extendedData仅在至少有一个DTCExtDataRecord可用于报告时才存在。

C_2: DTCExtDataRecordNumber 和 DTCExtDataRecord 参数中的 extendedData 仅在请求报告所有记录（请求中DTCExtDataRecordNumber设置为0xFE或0xFF）并且可以报告多条记录时才存在。

表277定义了子函数参数的肯定响应消息格式。

表277 - 响应消息定义 - 子功能= reportDTCBySeverityMaskRecord, reportSeverityInformationOfDTC

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportDTCBySeverityMaskRecord reportSeverityInformationOfDTC]	M	 0x08 0x09	LEV_ RDTCBSMR RSIODTC
#3	DTCStatusAvailabilityMask	M	0x00 - 0xFF	DTCSAM

表277 – （续）

A_Data字节	参数名称	Cvt	字节值	助记符
#4		C_1	0x00 – 0xFF	DTCASR_DTCS
#5		C_1	0x00 – 0xFF	DTCFU
#6		C_1	0x00 – 0xFF	DTCHB
#7		C_1	0x00 – 0xFF	DTCMB
#8		C_1	0x00 – 0xFF	DTCLB
#9		C_1	0x00 – 0xFF	SODTC
:		:	:	:
#n-5		C_2	0x00 – 0xFF	DTCS
#n-4		C_2	0x00 – 0xFF	DTCFU
#n-3		C_2	0x00 – 0xFF	DTCHB
#n-2		C_2	0x00 – 0xFF	DTCMB
#n-1		C_2	0x00 – 0xFF	DTCLB
#n		C_2	0x00 – 0xFF	SODTC

C_1: 在报告DTCBySeverityMaskRecord的情况下，如果至少有一个DTC与客户机定义的DTC严重性掩码相匹配，则必须存在此参数。 在reportSeverityInformationOfDTC的情况下，如果服务器支持请求消息中指定的DTC，则该参数必须存在。

C_2: 此参数记录仅在reportType = reportDTCBySeverityMaskRecord时出现。 如果多个DTC与客户机定义的DTC严重性掩码相匹配，它必须存在。

表278定义了子函数参数的肯定响应消息格式。

表278 – 响应消息定义 – 子功能= reportDTCFaultDetectionCounter

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportDTCFaultDetectionCounter]	M	0x14	LEV_RDTCFDC
#3		C_1	0x00 – 0xFF	DTCFDCR_DTCHB
#4		C_1	0x00 – 0xFF	DTCMB
#5		C_1	0x00 – 0xFF	DTCLB
#6		C_1	0x01 – 0xFF	DTCFDC
#7		C_2	0x00 – 0xFF	DTCHB
#8		C_2	0x00 – 0xFF	DTCLB
#9		C_2	0x00 – 0xFF	DTCFT
#10		C_2	0x01 – 0xFF	DTCFDC
#n-3		:	:	:
#n-2		C_2	0x00 – 0xFF	DTCHB
#n-1		C_2	0x00 – 0xFF	DTCMB
#n		C_2	0x00 – 0xFF	DTCLB
		C_2	0x01 – 0xFF	DTCFDC

C_1: 仅当至少有一个DTC的DTCFaultDetectionCounter的正值小于0x7F时才存在此参数。

C_2: 只有当一个以上DTC具有正值小于0x7F的DTCFaultDetectionCounter时，此参数记录才存在。

表279定义了子函数参数的肯定响应消息格式。

表279 – 响应消息定义 – 子功能= reportDTCExtDataRecordByRecordNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportDTCExtDataRecordByRecordNumber]	M	0x16	LEV_ RDTCEDRBR
#3	DTCExtDataRecordNumber	M	0x00 – 0xEF	DTCEDRN
#4 #5 #6 #7		C_1 C_1 C_1 C_1	0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF	DTCASR_ DTCHB DTCMB DTCLB SODTC
#8 : #8+(p-1)		C_1 : C_1	0x00 – 0xFF : 0x00 – 0xFF	DTCEDR_ EDD11 : EDD1p
:	:	:	:	:
#t #t+1 #t+2 #t+3		C_2 C_2 C_2 C_2	0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF	DTCSSR
#t+4 : #吨+ 4 + (P-1)		C_2 : C_2	0x00 – 0xFF : 0x00 – 0xFF	DTCEDR_ EDDx1 : EDDxp

C_1：只有至少有一个DTCExtDataRecord可用于报告时，才会显示DTCAndStatusRecord和DTCExtDataRecord参数。

C_2：如果可以报告多个DTCExtDataRecord，则只有DTCAndStatusRecord和DTCExtDataRecord参数存在。

注意　　实施者需要指定一个响应不会超过所使用的诊断通信所能达到的时间长度。

表280定义了子功能参数的肯定响应消息格式。

表280 – 响应消息定义 – 子功能= reportUserDefMemoryDTCByStatusMask

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	59	RDTCIPR
#2	reportType = [reportUserDefMemoryDTCByStatusMask]	M	17	LEV_ RUDMDTCBSM
#3	MemorySelection	M	00-FF	MEMYS
#4	DTCStatusAvailabilityMask	M	00-FF	DTCSAM

表280 – （续）

A_Data字节	参数名称	Cvt	字节值	助记符
#5		C_1	00-FF	DTCASR_ DTCHB
#6		C_1	00-FF	DTCMB
#7		C_1	00-FF	DTCLB
#8		C_1	00-FF	SODTC
#9		C_2	00-FF	DTCHB
#10		C_2	00-FF	DTCMB
#11		C_2	00-FF	DTCLB
#12		C_2	00-FF	SODTC
:		:	:	:
#n-3		C_2	00-FF	DTCHB
#n-2		C_2	00-FF	DTCMB
#n-1		C_2	00-FF	DTCLB
#n		C_2	00-FF	SODTC
C_1: 该参数仅在可报告DTC信息时才存在。 C_2: 该参数仅在可以报告多个DTC信息时才存在。				

表281定义了子函数参数的肯定响应消息格式。

表281 – 响应消息定义 – 子功能= reportUserDefMemoryDTCSnapshotRecordByDTCNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportUserDefMemoryDTCSnapshotRecordByDTCNumber]	M	0x18	LEV_ RUDMDTCSSBDTC
#3	MemorySelection	M	0x00-0xFF	MEMYS
#4		M	0x00 – 0xFF	DTCASR_ DTCHB
#5		M	0x00 – 0xFF	DTCMB
#6		M	0x00 – 0xFF	DTCLB
#7		M	0x00 – 0xFF	SODTC
#8	DTCSnapshotRecordNumber#1	C_1	0x00 – 0xFF	DTCSSRN
#9	DTCSnapshotRecordNumberOfIdentifiers#1	C_1	0x00 – 0xFF	DTCSSRNI
#10		C_1	0x00 – 0xFF	DTCSSR_ DIDB11
#11		C_1	0x00 – 0xFF	DIDB12
#12		C_1	0x00 – 0xFF	SSD11
:		C_1	:	:
# 12+(p-1)		C_1	0x00 – 0xFF	SSD1p
:		:	:	:
#R- (M-1) -2		C_2	0x00 – 0xFF	DIDB21
#R- (M-1) -1		C_2	0x00 – 0xFF	DIDB22
#R- (M-1)		C_2	0x00 – 0xFF	SSD21
:		C_2	:	:
#r		C_2	0x00 – 0xFF	SSD2m
:	:	:	:	:

表281 - （续）

A_Data字节	参数名称	Cvt	字节值	助记符
A_Data字节	参数名称	Cvt	字节值	助记符
#t	DTCSnapshotRecordNumber#X	C_3	0x00 - 0xFF	DTCSSRN
#t+1	DTCSnapshotRecordNumberOfIdentifiers#X	C_3	0x00 - 0xFF	DTCSSRNI
#t+2 #t+3 #t+5 : #T + 5 + (P-1) : #N- (U-1) -2 #N- (U-1) -1 #N- (U-1) : #n		C_3 C_3 C_3 C_3 C_3 : C_4 C_4 C_4 C_4 C_4	0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF : 0x00 - 0xFF : 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF : 0x00 - 0xFF	DTCSSR_ DIDB11 DIDB12 SSD11 : SSD1p : DIDB21 DIDB22 SSD21 : SSD2u

C_1: 只有至少有一个DTCSnapshot记录可用于报告时，DTCSnapshotRecord参数中的DTCSnapshotRecordNumber和第一个dataIdentifier / snapshotData组合才存在。

C_2/ C_4允许在单个DTCSnapshotRecord中存在多个dataIdentifier / snapshotData组合。 例如，对于单个数据标识符仅引用数据的整数部分的情况，情况可能如此。 当dataIdentifier引用一个数据块时，可以使用单个dataIdentifier / snapshotData组合。

C_3: DTCSnapshotRecordNumber和DTCSnapshotRecord参数中的第一个dataIdentifier / snapshotData组合仅在请求报告所有记录（请求中DTCSnapshotRecordNumber设置为0xFF）并且可以报告多条记录时才存在。

表282定义了子函数参数的肯定响应消息格式。

表282 - 响应消息定义 - 子功能= reportUserDefMemoryDTCExtDataRecordByDTCNumber

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportUserDefMemoryDTCExtDataRecordByDTCNumber]	M	0x19	LEV_ RUDMDTCEDRBDN
#3	MemorySelection	M	0x00-0xFF	MEMYS
#4 #5 #6 #7	DTCAndStatusRecord[] = [DTCHighByte DTCMiddleByte DTCLowByte statusOfDTC]	M M M M	0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF	DTCASR_ DTCHB DTCMB DTCLB SODTC
#8	DTCExtDataRecordNumber#1	C_1	0x00-0xFE	DTCEDRN
#9 : #9+(p-1)	DTCExtDataRecord []#1 = [extendedData#1字节#1 : 扩展数据#1字节#p]	C_1 C_1 C_1	0x00 - 0xFF : 0x00 - 0xFF	DTCSSR_ EDD11 : EDD1p
:	:	:	:	:

表282 – （续）

A_Data字节	参数名称	Cvt	字节值	助记符
#t+1 : #T + 1 + (Q-1)	DTCExtDataRecord []#x = [extendedData#x字节#1 : extendedData#x byte#q]	 C_2 C_2 C_2	 0x00 – 0xFF : 0x00 – 0xFF	DTCSSR_ EDDx1 : EDDxq
C_1: DTCExtDataRecord参数中的DTCExtDataRecordNumber和extendedData仅在至少可以报告一个DTCExtDataRecord时才存在。 C_2: DTCExtDataRecordNumber 和 DTCExtDataRecord 参数中的 extendedData 仅在请求报告所有记录（请求中的DTCExtDataRecordNumber设置为0xFE或0xFF）并且可以报告多条记录时才存在。				

表283定义了子函数参数的肯定响应消息格式。

表283 – 响应消息定义 – 子功能= reportWWHOBDDTCByMaskRecord

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportWWHOBDDTCByMaskRecord]	M	 0x42	LEV_ RWWHOBDDTCBSMR
#3	FunctionalGroupIdentifier	M	0x00 – 0xFF	FGID
#4	DTCStatusAvailabilityMask	M	0x00 – 0xFF	DTCSAM
#5	DTCSeverityAvailabilityMask	M	0x00 – 0xFF	DTCSVAM
#6	DTCFormatIdentifier = [SAE_J2012-DA_DTCFormat_04 SAE_J1939-73_DTCFormat]	M	 0x04 0x02	DTCFID_ J2012-DADTCF04 J1939-73DTCF
#7 #8 #9 #10 #11 : #n-4 #n-3 #n-2 #n-1 #n		 C_1 C_1 C_1 C_1 C_1 : C_2 C_2 C_2 C_2 C_2	 0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF : 0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF 0x00 – 0xFF	DTCASR_ DTCS DTCHB DTCMB DTCLB SODTC : DTCS DTCHB DTCMB DTCLB SODTC
C_1: 该参数仅在可报告DTC信息时才存在。 C_2: 该参数仅在可以报告多个DTC信息时才存在。				

表284定义了子函数参数的肯定响应消息格式。

表284 - 响应消息定义 - 子函数= reportWWHOBDDTCWithPermanentStatus

A_Data字节	参数名称	Cvt	字节值	助记符
#1	ReadDTCInformation响应SID	M	0x59	RDTCIPR
#2	reportType = [reportWWHOBDDTCWithPermanentStatus]	M	0x55	LEV_ RWWHOBDDTCWPS
#3	FunctionalGroupIdentifier	M	0x00 - 0xFF	FGID
#4	DTCStatusAvailabilityMask	M	0x00 - 0xFF	DTCSAM
#5	DTCFormatIdentifier = [SAE_J2012-DA_DTCFormat_04 SAE_J1939-73_DTCFormat]	M	0x04 0x02	DTCFID_ J2012-DADTCF04 J1939-73DTCF
#6 #7 #8 #9 : #n-3 #n-2 #n-1 #n		C_1 C_1 C_1 C_1 : C_2 C_2 C_2 C_2	0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF : 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF 0x00 - 0xFF	DTCASR_ DTCHB DTCMB DTCLB SODTC : DTCHB DTCMB DTCLB SODTC

C_1: 该参数仅在可报告DTC信息时才存在。

C_2: 仅当可以报告多个DTC信息时才会显示此参数。

11.3.3.2 肯定回复消息数据参数定义

表285指定了肯定响应消息的数据参数。

表285 - 响应数据参数定义

定义
报告类型 该参数是来自客户端的请求消息中提供的子函数参数的位6 - 0的回显。
DTCAndSeverityRecord 如果SAE_J2012-DA_DTCFormat_00, ISO_14229-1_DTCFormat, SAE_J1939-73_DTCFormat (详见下面的详细说明), ISO11992-4DTCFormat或SAE_J2012-DA_DTCFormat_04, 此参数记录包含一个或多个DTCSeverity, DTCFunctionalUnit, DTCHighByte, DTCMiddleByte, DTCLowByte和statusOfDTC的分组。 DTCSeverity识别车辆操作和/或系统功能故障的重要性, 并允许向驾驶员显示建议的操作。 DTCSeverity的定义可以在D.3中找到。 DTCFuncitonalUnit是一个1字节的值, 用于标识报告DTC的相应基本车辆/系统功能。 DTCFunctionalUnit的定义是具体实现的, 应在相应的执行标准中规定。 DTCHighByte, DTCMiddleByte和DTCLowByte一起表示服务器支持的特定诊断故障代码的唯一标识号。 DTCHighByte和DTCMiddleByte表示正在诊断的电路或系统。 DTCLowByte表示电路或系统中的故障类型 (例如传感器开路, 传感器对地短路, 基于算法的故障等)。 该定义可以在ISO 15031-6 [12]规范中找到。 如果SAE_J1939-73_DTCFormat包含DTCSeverity, DTCFunctionalUnit, SPN (可疑参数号), FMI (失败模式标识符) 和OC (发生计数器) 的一个或多个分组, 则该参数记录包含一个或多个分组。 SPN, FMI和OC在SAE J1939 [18]中定义。

表285 – （续）

定义
DTCAndStatusRecord 如果是ISO_14229-1_DTC格式，SAE_J2012-DA_DTCFormat_00，SAE_J1939-73_DTC格式，SAE_J2012-DA_DTCFormat_04或ISO_11992-4_DTC格式，则此参数记录包含DTCHighByte，DTCMiddleByte，DTCLowByte和statusOfDTC的一个或多个分组。 SAE_J1939-73_DTCFormat支持SPN（可疑参数号），FMI（失败模式标识符）和OC（发生计数器）参数。 SPN，FMI和OC在SAE J1939中定义。 DTCHighByte，DTCMiddleByte和DTCLowByte一起表示服务器支持的特定诊断故障代码的唯一标识号。 可以完成3字节DTC编号的编码： — 通过使用根据ISO 15031-6 [12]规范的DTCHighByte，DTCMiddleByte和DTCLowByte的解码。 这种格式由DTCFormatIdentifier = SAE_J2012-DA_DTCFormat_00标识，或 — 通过使用根据ISO 14229-1规范的DTCHighByte，DTCMiddleByte和DTCLowByte的解码，其不规定任何解码方法并因此允许车辆制造商定义的解码方法。 这种格式由DTCFormatIdentifier = ISO_14229-1_DTCFormat或者 — 通过使用根据SAE J1939-73 [19]规范的DTCHighByte，DTCMiddleByte和DTCLowByte的解码。 这种格式由DTCFormatIdentifier = SAE_J1939-73_DTCFormat标识，或 — 根据ISO 11992-4 [5]规范使用DTCHighByte，DTCMiddleByte和DTCLowByte的解码。 这种格式由DTCFormatIdentifier = ISO_11992-4_DTCFormat标识。 — 通过使用根据ISO 27145-2 [16]规范的DTCHighByte，DTCMiddleByte和DTCLowByte的解码。 这种格式由DTCFormatIdentifier = SAE_2012-DA_WWHOBD_DTCFormat标识。
DTCRecord 此参数记录包含一个或多个DTCHighByte，DTCMiddleByte和DTCLowByte的分组。 DTCRecord的解释取决于此表中定义的DTCFormatIdentifier参数中包含的值。
StatusOfDTC 特定DTC的状态（例如，测试失败了此操作周期等）。 包含在statusOfDTC字节中的位的定义可以在本规范的D.2中找到。服务器不支持的位应报告为'0'。
DTCStatusAvailabilityMask 一个字节，其位的定义与statusOfDTC相同，代表服务器支持的状态位。 服务器不支持的位必须设置为'0'。 每个支持的位（由'1'表示）应该为服务器支持的每个DTC执行。
DTCFormatIdentifier 这个1字节的参数值定义了服务器报告的DTC格式。 — SAE_J2012-DA_DTCFormat_00：该参数值标识由ISO 15031-6 [12]规范定义的服务器报告的DTC格式。 — ISO_14229-1_DTCFormat：此参数值通过参数DTCAndStatusRecord标识服务器报告的DTC格式，如本表中所定义。 — SAE_J1939-73_DTCFormat：该参数值标识SAE J1939-73 [19]中定义的服务器报告的DTC格式。 — ISO_11992-4_DTCFormat：该参数值标识ISO 11992-4 [5]规范中定义的服务器报告的DTC格式。 — SAE_J2012-DA_DTCFormat_04：该参数值标识ISO 27145-2 [16]规范中定义的服务器报告的DTC格式。 包含在DTCFormatIdentifier字节中的字节值的定义可以在本规范的D.4中找到。 给定的服务器只能支持一个DTCFormatIdentifier。

表285 – （续）

定义
DTCCount 此2字节参数统称为响应reportNumberOfDTCByStatusMask或reportNumberOfMirrorMemoryDTC请求而发送的DTCCountHighByte和DTCCountLowByte参数。 DTCCount提供与客户请求中定义的DTCStatusMask相匹配的DTC数量。
DTCSnapshotRecordNumber 客户端在reportDTCSnapshotRecordByDTCNumber请求中指定的DTCSnapshotRecordNumber参数的回显，或存储的DTCSnapshot记录的实际DTCSnapshotRecordNumber。
DTCSnapshotRecordNumberOfIdentifiers 此单字节参数显示紧随其后的DTCSnapshotRecord中的dataIdentifiers的数量。 应使用值0x00来指示相应的DTCSnapshotRecord中包含未定义数量的dataIdentifiers（例如，主要用例是当DTCSnapshotRecord包含多于255个dataIdentifiers时）。
DTCSnapshotRecord DTCSnapshotRecord包含系统故障发生时刻的数据值快照。
DTCStoredDataRecord DTCStoredDataRecord包含从系统故障发生时起的数据值的冻结帧。
DTCStoredDataRecordNumber 客户端在reportDTCStoredDataByRecordNumber请求中指定的DTCStoredDataRecordNumber参数的回显或存储的DTCStoredDataRecord的实际DTCStoredDataRecordNumber。
DTCStoredDataRecordNumberOfIdentifiers 此单字节参数显示紧随其后的DTCStoredDataRecord中的dataIdentifiers的数量。
DTCExtDataRecordNumber 客户端在报告DTCExtDataRecordByDTCNumber或reportDTCExtDataRecordByRecordNumber请求中指定的DTCExtDataRecordNumber参数的回显，或存储的DTCExtendedData记录的实际DTCExtDataRecordNumber。
DTCExtDataRecord DTCExtDataRecord是一个服务器特定的信息块，可以包含与DTC相关的扩展状态信息。 DTCExtendedData包含请求时已识别的DTC参数值。
DTCFaultDetectionCounterRecord DTCFaultDetectionCounterRecord是包含一个或多个DTC编号和DTC特定DTCFaultDetectionCounter参数值的记录。
DTCFaultDetectionCounter DTCFaultDetectionCounter报告DTC故障检测计数的数量。
FunctionalGroupIdentifier 一个字节标识符，其中包含与DTC功能系统组相关的例如刹车，排放，乘员限制，轮胎充气，前进/外部照明等。这些值在D.5中定义。
DTCSeverityAvailabilityMask 一个字节，其位的定义与DTCSeverity相同，代表服务器支持的严重性位。 服务器不支持的位必须设置为'0'。
MemorySelection 该参数是来自客户端的请求消息中提供的MemorySelection参数的回显。

11.3.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表286中。如果错误情况适用于服务器，则应使用列出的否定响应。

表286 – 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果不支持子功能参数，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x31	**requestOutOfRange** 如果发生以下情况，应发送NRC: — 客户端指定了服务器无法识别的DTCMaskRecord; — 客户端指定了无效的DTCSnapshotRecordNumber / DTCExtDataRecordNumber。 请注意，这与服务器支持DTCSnapshotRecordNumber和DTCMaskRecord组合或DTCExtDataRecordNumber和DTCMaskRecord组合的情况不同，但当前没有数据与其关联（即没有数据需要肯定的响应）。 — 客户端指定了服务器无法识别的FunctionalGroupIdentifier; — MemorySelection标识符未被服务器识别。	ROOR

11.3.5 消息流示例 – ReadDTCInformation

11.3.5.1 一般假设

对于所有例子，该客户要求至有a响应信息通过设置suppressPosRspMsgIndicationBit（子函数参数的第7位）为“FALSE”（“0”）。

11.3.5.2 示例#1 – ReadDTCInformation，子函数= reportNumberOfDTCByStatusMask

11.3.5.2.1 示例#1概述

此示例演示了已确认的DTC（DTC状态掩码0x08）的reportNumberOfDTCByStatusMask子函数参数的用法以及各种屏蔽原理。 该服务器的DTCStatusAvailabilityMask = 0x2F。

11.3.5.2.2 示例#1的假设

该服务器总共支持三个DTC（为了简单起见！），它们在客户端请求时具有以下状态:

以下假设适用于DTC P0805-11离合器位置传感器 – 对地短路（0x080511），statusOfDTC 0x24（0010 0100_b）短路。

表287定义了DTC P0805-11的statusOfDTC = 0x24。

表287 - DTC P0805-11的statusOfDTC = 0x24

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不再失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC P0A9B-17混合电池温度传感器 - 电路电压高于阈值（0x0A9B17），状态0xDTC为0x26（0010 0110_b）。

表288定义了DTC P0A9B-17的statusOfDTC = 0x26。

表288 -DTC P0A9B-17的statusOfDTC = 0x26

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不再失败
testFailedThisOperationCycle	1	1	DTC在当前操作循环中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC P2522-1F A / C请求“B”- 间歇性电路（0x25221F），0x2F（0010 1111_b）的statusOfDTC。

表289定义了DTC P2522-1F的statusOfDTC = 0x2F。

表289 - DTC P2522-1F的statusOfDTC = 0x2F

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	1	DTC在请求时失败
testFailedThisOperationCycle	1	1	DTC在当前操作循环中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	1	DTC在请求时被确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

11.3.5.2.3 示例#1消息流

在以下示例中，计数为1会返回给客户端，因为只有DTC P2522-1F A / C请求“B”
- 电路间歇性（0x25221F），0x2F（0010 1111$_b$）的statusOfDTC与客户机定义的0x08（0000 1000$_b$）状态掩码匹配。

表290定义了ReadDTCInformation，子函数= reportNumberOfDTCByStatusMask，请求消息流程示例#1。

表290 - ReadDTCInformation，子功能= reportNumberOfDTCByStatusMask，请求消息流程示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportNumberOfDTCByStatusMask, suppressPosRspMsgIndicationBit = FALSE	0x01	RNODTCBSM
#3	DTCStatusMask	0x08	DTCSM

表291定义了ReadDTCInformation，子函数= reportNumberOfDTCByStatusMask，肯定响应，示例#1。

表291 - ReadDTCInformation，子函数= reportNumberOfDTCByStatusMask，肯定响应，示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportNumberOfDTCByStatusMask	0x01	RNODTCBSM
#3	DTCStatusAvailabilityMask	0x2F	DTCSAM
#4	DTCFormatIdentifier = ISO_14229-1_DTCFormat	0x01	14229-1DTCF
#5	DTCCount [DTCCountHighByte]	0x00	DTCCHB
#6	DTCCount [DTCCountLowByte]	0x01	DTCCLB

11.3.5.3 示例#2 - ReadDTCInformation，子函数= reportDTCByStatusMask，匹配的DTC返回

11.3.5.3.1 示例#2概述

此示例演示reportDTCByStatusMask子函数参数的用法，以及各种掩蔽原理以及不支持的掩码位。 此示例也适用于子函数参数reportMirrorMemoryDTCByStatusMask和子函数参数reportUserDefMemoryDTCByStatusMask，不同之处在于状态掩码检查是使用存储在DTC镜像存储器或用户定义的存储器中的DTC执行的。

11.3.5.3.2 示例#2的假设

除了第7位“warningIndicatorRequested”之外，服务器支持所有状态位用于掩码目的。

该服务器总共支持三个DTC（为了简单起见！），它们在客户端请求时具有以下状态：

以下假设适用于DTC P0A9B-17混合电池温度传感器 – 电路电压高于阈值（0x0A9B17），状态0xDTC 0x24（0010 0100$_b$）。

表292定义了DTC P0A9B-17的statusOfDTC = 0x24。

表292 – DTC P0A9B-17的statusOfDTC = 0x24

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不再失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC P2522-1F A / C请求“B”– 断路（0x25221F），状态0xDTC为0x00（0000 0000$_b$）。

表293定义了DTC P2522-1F的statusOfDTC = 0x00。

表293 –DTC P2522-1F的statusOfDTC = 0x00

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	0	DTC在当前或之前的操作周期中没有失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	0	自上次代码清除以来，DTC测试从未失败
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC P0805-11离合器位置传感器 – 对地短路（0x080511），0x02F（0010 1111$_b$）状态输出。

表294定义了DTC P0805-11的statusOfDTC = 0x2F。

表294 - DTC P0805-11的statusOfDTC = 0x2F

statusOfDTC: 位字段名称	位#	位状态	描述
testFailed	0	1	DTC在请求时失败
testFailedThisOperationCycle	1	1	DTC在当前操作循环中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	1	DTC在请求时被确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

11.3.5.3.3 示例#2消息流

在以下示例中，将DTC P0A9B-17（0x0A9B17）和P0805-11（0x080511）返回给客户机的请求。 DTC P2522-1F（0x25221F）不会返回，因为其状态0x00与0x84的DTCStatusMask不匹配（如以下示例中客户端请求消息中所指定的那样）。 服务器应绕过它不支持的那些状态位的掩码。

表295定义了ReadDTCInformation，子函数= reportDTCByStatusMask，请求消息流程示例#2。

表295 - ReadDTCInformation，子函数= reportDTCByStatusMask，请求消息流程示例#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportDTCByStatusMask, suppressPosRspMsgIndicationBit = FALSE	0x02	RDTCBSM
#3	DTCStatusMask	0x84	DTCSM

表296定义了ReadDTCInformation，子函数= reportDTCByStatusMask，正响应，示例#2。

表296 - ReadDTCInformation，子函数= reportDTCByStatusMask，肯定响应，示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportDTCByStatusMask	0x02	RDTCBSM
#3	DTCStatusAvailabilityMask	0x7F	DTCSAM
#4	DTCAndStatusRecord#1 [DTCHighByte]	0x0A	DTCHB
#5	DTCAndStatusRecord#1 [DTCMiddleByte]	0x9B	DTCMB
#6	DTCAndStatusRecord#1 [DTCLowByte]	0x17	DTCLB
#7	DTCAndStatusRecord#1 [statusOfDTC]	0x24	SODTC
#8	DTCAndStatusRecord#2 [DTCHighByte]	0x08	DTCHB
#9	DTCAndStatusRecord#2 [DTCMiddleByte]	0x05	DTCMB
#10	DTCAndStatusRecord#2 [DTCLowByte]	0x11	DTCLB
#11	DTCAndStatusRecord#2 [statusOfDTC]	0x2F	SODTC

11.3.5.4 示例#3 - ReadDTCInformation，子函数= reportDTCByStatusMask，没有匹配的DTC返回

11.3.5.4.1 示例#3概述

此示例演示如果DTC与客户机定义的DTCStatusMask不匹配，则使用reportDTCByStatusMask子函数参数。

11.3.5.4.2 示例#3的假设

除了第7位“warningIndicatorRequested”之外，服务器支持所有状态位用于掩码目的。 服务器总共支持两个DTC（为了简单起见！），它们在客户端请求时具有以下状态：

以下假设适用于DTC P2522-1F A / C请求“B”- 断路（0x25221F），状态0xDTC 0x24（0010 0100$_b$）。

表297定义了DTC P2522-1F的statusOfDTC = 0x24。

表297 - DTC P2522-1F的statusOfDTC = 0x24

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不再失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC P0A9B-17混合电池温度传感器 - 电路电压高于阈值 (0x0A9B17)，状态0xDTC为0x00 (0000 0000_b)。

表298定义了DTC P0A9B-17的statusOfDTC = 0x00。

表298 - DTC P0A9B-17的statusOfDTC = 0x00

statusOfDTC: 位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	0	DTC在当前或之前的操作周期中没有失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	0	自上次代码清除以来，DTC测试从未失败
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

客户端请求服务器reportByStatusMask位0 (TestFailed) 设置为逻辑'1'的所有DTC。

11.3.5.4.3 示例#3消息流

在以下示例中，上述DTC都不会返回到客户端的请求，因为在请求发生时没有任何DTC通过测试。

表299定义了ReadDTCInformation，子函数= reportDTCByStatusMask，请求消息流程示例#3。

表299 - ReadDTCInformation，子功能= reportDTCByStatusMask，请求消息流程示例#3

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportDTCByStatusMask, suppressPosRspMsgIndicationBit = FALSE	0x02	RDTCBSM
#3	DTCStatusMask	0x01	DTCSM

表300定义了ReadDTCInformation，子函数= reportDTCByStatusMask，肯定响应，示例#3。

表300 - ReadDTCInformation, 子函数= reportDTCByStatusMask, 肯定响应, 示例#3

消息方向		服务器 客户端	
消息类型		响应	
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportDTCByStatusMask	0x02	RDTCBSM
#3	DTCStatusAvailabilityMask	0x7F	DTCSAM

11.3.5.5 示例#4 - ReadDTCInformation, 子函数= reportDTCSnapshotIdentification

11.3.5.5.1 示例#4概述

此示例演示reportDTCSnapshotIdentification子函数参数的用法。

11.3.5.5.2 示例#4的假设

以下假设适用:

— 服务器支持为给定的DTC存储两个DTCSnapshot记录。

— 服务器应指示当前为DTC编号0x123456存储两个DTCSnapshot记录。 就本例而言, 假设此DTC发生了三次(因为服务器内缺少存储空间, 所以只存储第一个和最近的DTCSnapshot记录)。

— 服务器应指示当前存储一个DTCSnapshot记录的DTC编号为0x789ABC。

— 所有DTCSnapshot记录按升序存储。

11.3.5.5.3 示例#4消息流

在以下示例中, 将三个DTCSnapshot记录返回给客户端的请求。

表301定义了ReadDTCInformation, 子功能= reportDTCSnapshotIdentification, 请求消息流程示例#4。

表301 - ReadDTCInformation, 子功能= reportDTCSnapshotIdentification, 请求消息流程示例#4

消息方向		客户端 服务器	
消息类型		请求	
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportDTCSnapshotIdentification, suppressPosRspMsgIndicationBit = FALSE	0x03	RDTCSSI

表302定义了ReadDTCInformation，子函数= reportDTCSnapshotIdentification，肯定响应，示例#4。

表302-ReadDTCInformation，子功能= reportDTCSnapshotIdentification，肯定响应，示例#4

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportDTCSnapshotIdentification	0x03	RDTCSSI
#3	DTCAndStatusRecord#1 [DTCHighByte]	0x12	DTCHB
#4	DTCAndStatusRecord#1 [DTCMiddleByte]	0x34	DTCMB
#5	DTCAndStatusRecord#1 [DTCLowByte]	0x56	DTCLB
#6	DTCSnapshotRecordNumber#1	0x01	DTCEDRC
#7	DTCAndStatusRecord#2 [DTCHighByte]	0x12	DTCHB
#8	DTCAndStatusRecord#2 [DTCMiddleByte]	0x34	DTCMB
#9	DTCAndStatusRecord#2 [DTCLowByte]	0x56	DTCLB
#10	DTCSnapshotRecordNumber#2	0x02	DTCEDRC
#11	DTCAndStatusRecord#3 [DTCHighByte]	0x78	DTCHB
#12	DTCAndStatusRecord#3 [DTCMiddleByte]	0x9A	DTCMB
#13	DTCAndStatusRecord#3 [DTCLowByte]	0xBC	DTCLB
#14	DTCSnapshotRecordNumber#3	0x01	DTCEDRC

11.3.5.6 示例#5 - ReadDTCInformation，子函数= reportDTCSnapshotRecord - ByDTCNumber

11.3.5.6.1 示例#5概述

此示例演示reportDTCSnapshotRecordByDTCNumber子函数参数的用法。 此示例也适用于子函数参数reportUserDefMemory-DTCSnapshotRecordByDTCNumber，不同之处在于检查是使用存储在用户定义的存储器中的DTC执行的。

11.3.5.6.2 示例#5的假设

以下假设适用：

— 服务器支持为给定的DTC存储两个DTCSnapshot记录。

— 这个例子假定前面例子的延续。

— 假设服务器请求服务器存储的两个DTCSnapshot记录中的第二个DTC编号为0x123456（请参阅前面的示例，其中DTCSnapshotRecordCount的0x02返回给客户端）。

— 假设DTC 0x123456的statusOfDTC为0x24，并且每次发生DTC时都会捕获以下环境数据。

— DTCSnapshot记录数据通过dataIdentifier 0x4711引用。

表303定义了DTCSnapshot记录内容。

表303 - DTCSnapshot记录内容

数据字节	DTCSnapshot记录内容	字节值
#1	DTCSnapshotRecord [数据#1] = ECT (发动机冷却液温度)	0xA6
#2	DTCSnapshotRecord [数据#2] = TP (节气门位置)	0x66
#3	DTCSnapshotRecord [data#3] = RPM (发动机转速)	0x07
#4	DTCSnapshotRecord [data#4] = RPM (发动机转速)	0x50
#5	DTCSnapshotRecord [data#5] = MAP (歧管绝对压力)	0x20

11.3.5.6.3 示例#5消息流

在以下示例中，根据客户端的reportDTCSnapshotRecordByDTCNumber请求返回一个DTCSnapshot记录。

表304定义了ReadDTCInformation，子功能= reportDTCSnapshotRecordByDTCNumber，请求消息流程示例#5。

表304 - ReadDTCInformation，子功能= reportDTCSnapshotRecordByDTCNumber，请求消息流程示例#5

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	sub-function = reportDTCSnapshotRecordByDTCNumber, suppressPosRspMsgIndicationBit = FALSE	0x04	RDTCSSRBDN
#3	DTCMaskRecord [DTCHighByte]	0x12	DTCHB
#4	DTCMaskRecord [DTCMiddleByte]	0x34	DTCMB
#5	DTCMaskRecord [DTCLowByte]	0x56	DTCLB
#6	DTCSnapshotRecordNumber	0x02	DTCSSRN

表305定义了ReadDTCInformation，子函数= reportDTCSnapshotRecordByDTCNumber，肯定响应，示例#5。

表305-ReadDTCInformation，子功能= reportDTCSnapshotRecordByDTCNumber，肯定响应，示例#5

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportDTCSnapshotRecordByDTCNumber	0x04	RDTCSSRBDN
#3	DTCAndStatusRecord [DTCHighByte]	0x12	DTCHB
#4	DTCAndStatusRecord [DTCMiddleByte]	0x34	DTCMB
#5	DTCAndStatusRecord [DTCLowByte]	0x56	DTCLB
#6	DTCAndStatusRecord [statusOfDTC]	0x24	SODTC
#7	DTCSnapshotRecordNumber	0x02	DTCEDRN
#8	DTCSnapshotRecordNumberOfIdentifiers	0x01	DTCSSRNI
#9	dataIdentifier [byte#1] (MSB)	0x47	DIDB1
#10	dataIdentifier [byte#2] (LSB)	0x11	DIDB2
#11	DTCSnapshotRecord [data#1] = ECT DTCSnapshotRecord	0xA6	ED_1
#12	[data#2] = TP DTCSnapshotRecord [data#3] = RPM	0x66	ED_2
#13	DTCSnapshotRecord [data#4] = RPM	0x07	ED_3
#14	DTCSnapshotRecord [data#5] = MAP	0x50	ED_4
#15		0x20	ED_5

11.3.5.7 示例#6 - ReadDTCInformation，子函数= reportDTCStoredDataByRecordNumber

11.3.5.7.1 示例#6概述

此示例演示reportDTCStoredDataByRecordNumber子函数参数的用法。

11.3.5.7.2 示例#6的假设

以下假设适用：

— 服务器支持为给定的DTC存储两个DTCStoredDataRecords。

— 这个例子假定前面例子的延续。

— 假定服务器请求服务器存储的两个DTCStoredDataRecords中的第二个DTC编号为0x123456（请参阅前面的示例，其中DTCStoredDataRecordCount为2的DTCStoredDataRecordCount返回给客户端）。

— 假设DTC 0x123456的statusOfDTC为0x24，并且每次发生DTC时都会捕获以下环境数据。

— DTCStoredData记录数据通过dataIdentifier 0x4711引用。

表306定义了DTCStoredData记录内容。

表306-DTCStoredData记录内容

数据字节	DTCSnapshot记录内容	字节值
#1	DTCStoredDatatRecord [data#1] = ECT (发动机冷却液温度)	0xA6
#2	DTCStoredDataRecord [数据#2] = TP (节气门位置)	0x66
#3	DTCStoredDataRecord [数据#3] = RPM (发动机转速)	0x07
#4	DTCStoredDataRecord [数据#4] = RPM (发动机转速)	0x50
#5	DTCStoredDataRecord [data#5] = MAP (歧管绝对压力)	0x20

11.3.5.7.3 示例#6消息流

在以下示例中，请求DTCStoredData记录号为2，并且服务器返回DTC和DTCStoredData记录内容。

表307定义了ReadDTCInformation，子功能= reportDTCStoredDataByRecordNumber，请求消息流程示例#6。

表307 - ReadDTCInformation，子功能= reportDTCStoredDataByRecordNumber，请求消息流程示例#6

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	sub-function = reportDTCStoredDataByRecordNumber, suppressPosRspMsgIndicationBit = FALSE	0x05	RDTCSDRBRN
#3	DTCStoredDataRecordNumber	0x02	DTCSDRN

表308定义了ReadDTCInformation，子函数= reportDTCStoredDataByRecordNumber，肯定响应，示例#6。

表308-ReadDTCInformation，子功能= reportDTCStoredDataByRecordNumber，肯定响应，示例#6

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportDTCStoredDataByRecordNumber	0x05	RDTCSDRBRN
#3	DTCStoredDataRecordNumber	0x02	DTCSDRN
#4	DTCAndStatusRecord [DTCHighByte]	0x12	DTCHB
#5	DTCAndStatusRecord [DTCMiddleByte]	0x34	DTCMB
#6	DTCAndStatusRecord [DTCLowByte]	0x56	DTCLB
#7	DTCAndStatusRecord [statusOfDTC]	0x24	SODTC

表308 – （续）

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#8	DTCStoredDataRecordNumberOfIdentifiers	0x01	DTCSDRNI
#9	dataIdentifier [byte#1 (MSB)]	0x47	DIDB1
#10	dataIdentifier [byte#2] (LSB)	0x11	DIDB2
#11	DTCStoredDataRecord [数据#1] = ECT	0xA6	ED_1
#12	DTCStoredDataRecord [数据#2] = TP	0x66	ED_2
#13	DTCStoredDataRecord [data#3] = RPM	0x07	ED_3
#14	DTCStoredDataRecord [data#4] = RPM	0x50	ED_4
#15	DTCStoredDataRecord [data#5] = MAP	0x20	ED_5

11.3.5.8 示例#7 – ReadDTCInformation，子函数= reportDTCExtDataRecordByDTCNumber

11.3.5.8.1 示例#7概述

此示例演示reportDTCExtDataRecordByDTCNumber子函数参数的用法。 此示例也适用于子函数参数reportUserDefMemory-DTCExtDataRecordByDTCNumber，但检查是使用存储在用户定义的存储器中的DTC执行的。

11.3.5.8.2 示例#7的假设

以下假设适用：

— 服务器支持为给定的DTC存储两个DTCExtendedData记录的功能。

— 假设服务器请求服务器为DTC号码0x123456存储的所有可用的DTCExtendedData记录。

— 假设DTC 0x123456的statusOfDTC为0x24，并且下列扩展数据可用于DTC。

— DTCExtendedData通过DTCExtDataRecordNumbers 0x05和0x10来引用

表309-DTCExtDataRecordNumber 0x05内容

数据字节	DTCExtDataRecordNumber的DTCExtDataRecord内容0x05	字节值
#1	预热循环计数器 – 自DTC命令MIL关闭以来的预热循环次数	0x17

表310-DTCExtDataRecordNumber 0x10内容

数据字节	DTCExtDataRecordNumber 0x10的DTCExtDataRecord内容	字节值
#1	DTC故障检测计数器 – 每次DTC测试检测到故障时递增，每次测试报告无故障时递减。	0x79

11.3.5.8.3 示例#7消息流

在以下示例中，客户端请求包含DTC编号和值为0xFF（报告所有DTCExtDataRecords）的DTCExtDataRecordNumber的DTCMaskRecord。 服务器返回两个DTCExtDataRecords，这些DTCExtDataRecords已经记录了客户端提交的DTC编号。

表311定义了ReadDTCInformation，子函数= reportDTCExtDataRecordByDTCNumber，请求消息流程示例#7。

表311-ReadDTCInformation，子功能= reportDTCExtDataRecordByDTCNumber，请求消息流程示例#7

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	sub-function = reportDTCExtDataRecordByDTCNumber, suppressPosRspMsgIndicationBit = FALSE	0x06	RDTCEDRBDN
#3	DTCMaskRecord [DTCHighByte]	0x12	DTCHB
#4	DTCMaskRecord [DTCMiddleByte]	0x34	DTCMB
#5	DTCMaskRecord [DTCLowByte]	0x56	DTCLB
#6	DTCExtDataRecordNumber	为0xFF	DTCEDRN

表312定义了ReadDTCInformation，子函数= reportDTCExtDataRecordByDTCNumber，肯定响应，示例#7。

表312-ReadDTCInformation，子功能= reportDTCExtDataRecordByDTCNumber，肯定响应，示例#7

消息直接重刑	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportDTCExtDataRecordByDTCNumber	0x06	RDTCEDRBDN
#3	DTCAndStatusRecord [DTCHighByte]	0x12	DTCHB
#4	DTCAndStatusRecord [DTCMiddleByte]	0x34	DTCMB
#5	DTCAndStatusRecord [DTCLowByte]	0x56	DTCLB
#6	DTCAndStatusRecord [statusOfDTC]	0x24	SODTC
#7	DTCExtDataRecordNumber	0x05	DTCEDRN
#8	DTCExtDataRecord [byte#1]	0x17	ED_1
#9	DTCExtDataRecordNumber	0x10	DTCEDRN
#10	DTCExtDataRecord [byte#1]	0x79	ED_1

11.3.5.9 示例#8 - ReadDTCInformation, 子函数= reportNumberOfDTC-BySeverityMaskRecord

11.3.5.9.1 示例#8概述

此示例演示reportNumberOfDTCBySeverityMaskRecord子函数参数的用法。

11.3.5.9.2 示例#8的假设

服务器总共支持三个DTC，这些DTC在客户端请求时具有以下状态：

以下假设适用于DTC P0A9B-17混合电池温度传感器 - 电路电压高于阈值 (0x0A9B17)，statusOfDTC 0x24 (0010 0100b)，DTCFunctionalUnit = 0x10，DTCSeverity = 0x20：

注意 严重性字节的第7位至第5位是有效的。

表313定义了DTC P0A9B-17的statusOfDTC = 0x24。

表313 - DTC P0A9B-17的statusOfDTC = 0x24

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不再失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC P2522-1F A / C请求“B”- 断路 (0x25221F)，状态0xDTC为0x00 (0000 0000二进制)，DTCFunctionalUnit = 0x10，DTCSeverity = 0x20：

注意 严重性字节的第7位至第5位是有效的。

表314定义了DTC P2522-1F的statusOfDTC = 0x00。

表314 -DTC P2522-1F的statusOfDTC = 0x00

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	0	DTC在当前或之前的操作周期中没有失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	0	自上次代码清除以来，DTC测试从未失败
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC P0805-11离合器位置传感器 - 电路对地短路 (0x080511)，statusOfDTC为0x2F (0010 1111$_b$)，DTCFunctionalUnit = 0x10，DTCSeverity = 0x40：

说明只有严重性字节的第7位至第5位有效。

表315定义了DTC P0805-11的statusOfDTC = 0x2F。

表315 - DTC P0805-11的statusOfDTC = 0x2F

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	1	DTC在请求时失败
testFailedThisOperationCycle	1	1	DTC在当前操作循环中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	1	DTC在请求时被确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

服务器支持用于掩蔽目的的testFailed和confirmedDTC状态位。

11.3.5.9.3 示例#8消息流

在以下示例中，由于DTC P0805-11 (0x080511) 与0xC001 (DTCSeverityMask = 110x xxxx$_b$= 0xC0，DTCStatusMask = 0000 0001$_b$) 的客户机定义严重性掩码记录相匹配，所以计数为1返回给客户机。

表316定义了ReadDTCInformation，子功能= reportNumberOfDTCBySeverityMaskRecord，请求消息流程示例#8。

表316-ReadDTCInformation，子功能= reportNumberOfDTCBySeverityMaskRecord，请求消息流程示例#8

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportNumberOfDTCBySeverityMaskRecord, suppressPosRspMsgIndicationBit = FALSE	0x07	RNODTCBSMR
#3	DTCSeverityMaskRecord (DTCSeverityMask)	0xC0	DTCSVM
#4	DTCSeverityMaskRecord (DTCStatusMask)	0x01	DTCSM

表317定义了ReadDTCInformation，子函数= reportNumberOfDTCBySeverityMaskRecord，正响应，正响应，示例#8。

表317 – ReadDTCInformation, 子函数= reportNumberOfDTCBySeverityMaskRecord, 肯定响应, 示例#8

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportNumberOfDTCBySeverityMaskRecord	0x07	RNODTCBSMR
#3	DTCStatusAvailabilityMask	0x09	DTCSAM
#4	DTCFormatIdentifier = ISO_14229-1_DTCFormat	0x01	14229-1DTCF
#5	DTCCount [DTCCountHighByte]	0x00	DTCCHB
#6	DTCCount [DTCCountLowByte]	0x01	DTCCLB

11.3.5.10 示例#9 – ReadDTCInformation, 子函数= reportDTCBySeverityMaskRecord 11.3.5.10.1

示例#9概述

此示例演示reportDTCBySeverityMaskRecord子函数参数的用法。

11.3.5.10.2 示例#9假设

11.3.5.9.2中定义的假设和本子条款中定义的假设适用。

在以下示例中, DTC P0805-11 (0x080511) 与0xC001 (DTCSeverityMask = 0xC0 = 110x XXXX$_b$, DTCStatusMask = 0x01, 0000 0001$_b$) 客户机定义的严重性掩码记录相匹配, 并将其报告给客户机。 DTC P0805-11 (0x080511) 的严重性为0x40 (010x XXXX$_b$) 。 除了第7位“warningIndicatorRequested”之外, 服务器支持所有状态位用于掩码目的。

注意 只有严重性掩码字节的第7位到第5位才有效。

11.3.5.10.3 示例#9消息流

在以下示例中, 将一个DTCSeverityRecord返回给客户端的请求。

表318定义了ReadDTCInformation, 子功能= reportDTCBySeverityMaskRecord, 请求消息流程示例#9。

表318 – ReadDTCInformation, 子功能= reportDTCBySeverityMaskRecord, 请求消息流程示例#9

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportDTCBySeverityMaskRecord, suppressPosRspMsgIndicationBit = FALSE	0x08	RDTCBSMR
#3	DTCSeverityMaskRecord (DTCSeverityMask)	0xC0	DTCSVM
#4	DTCSeverityMaskRecord (DTCStatusMask)	0x01	DTCSM

表319-ReadDTCInformation，子功能= reportDTCBySeverityMaskRecord，肯定响应，示例#9

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportDTCBySeverityMaskRecord	0x08	RDTCBSMR
#3	DTCStatusAvailabilityMask	0x7F	DTCSAM
#4	DTCSeverityRecord#1 [DTCSeverity]	0x40	DTCS
#5	DTCSeverityRecord#1 [DTCFunctionalUnit]	0x10	DTCFU
#6	DTCSeverityRecord#1 [DTCHighByte]	0x08	DTCHB
#7	DTCSeverityRecord#1 [DTCMiddleByte]	0x05	DTCMB
#8	DTCSeverityRecord#1 [DTCLowByte]	0x11	DTCLB
#9	DTCSeverityRecord#1 [statusOfDTC]	0x2F	SODTC

11.3.5.11　示例#10 - ReadDTCInformation，子函数= reportSeverityInformationOfDTC

11.3.5.11.1示例#10概述

此示例演示reportSeverityInformationOfDTC子函数参数的用法。

11.3.5.11.2　示例#10假设

在11.3.5.10.2中定义的假设适用。

11.3.5.11.3　示例#10消息流

在以下示例中，将与客户机定义的DTC掩码记录相匹配的DTC P0805-11（0x080511）报告给客户机。

表320-ReadDTCInformation，子功能= reportSeverityInformationOfDTC，请求消息流程
示例#10

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportSeverityInformationOfDTC, suppressPosRspMsgIndicationBit = FALSE	0x09	RSIODTC
#3	DTCMaskRecord [DTCHighByte]	0x08	DTCHB
#4	DTCMaskRecord [DTCMiddleByte]	0x05	DTCMB
#5	DTCMaskRecord [DTCLowByte]	0x11	DTCLB

表321 - ReadDTCInformation，子功能= reportSeverityInformationOfDTC，肯定响应，示例#10

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportDTCBySeverityMaskRecord	0x09	RSIODTC
#3	DTCStatusAvailabilityMask	0x7F	DTCSAM
#4	DTCSeverityRecord [DTCSeverity]	0x40	DTCS
#5	DTCSeverityRecord [DTCFunctionalUnit]	0x10	DTCFU
#6	DTCSeverityRecord [DTCHighByte]	0x08	DTCHB
#7	DTCSeverityRecord [DTCMiddleByte]	0x05	DTCMB
#8	DTCSeverityRecord [DTCLowByte]	0x11	DTCLB
#9	DTCSeverityRecord [statusOfDTC]	0x2F	SODTC

11.3.5.12 示例#11 - ReadDTCInformation - 子函数= reportSupportedDTCs

11.3.5.12.1示例#11概述

此示例演示reportSupportedDTCs子函数参数的用法。

11.3.5.12.2 示例#11假设

在11.3.5.10.2中定义的假设适用。 除了以下假设适用：

— 服务器总共支持三个DTC（为了简单起见！），它们在客户端请求时具有以下状态。

— 以下假设适用于DTC 0x123456，statusOfDTC 0x24 (0010 0100_b)

表322 - statusOfDTC = 0x24

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	0	DTC从未被证实
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除后，DTC至少失败一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC 0x234505，statusOfDTC 0x00 (0000 0000_b)

表323 – statusOfDTC = 0x00

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	0	DTC在当前或之前的操作周期中没有失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	0	自上次代码清除以来，DTC测试从未失败
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC 0xABCD01，0x2F（0010 1111$_b$）的statusOfDTC

表324 – statusOfDTC = 0x2F

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	1	DTC在请求时失败
testFailedThisOperationCycle	1	1	DTC在当前操作循环中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	1	DTC在请求时被确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

11.3.5.12.3 示例#11消息流

在以下示例中，以上所有三种DTC均返回到客户端的请求，因为全部都受支持。

表325 – ReadDTCInformation，子功能= reportSupportedDTC，请求消息流程示例#11

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子功能= reportSupportedDTCs， suppressPosRspMsgIndicationBit = FALSE	0x0A	RSUPDTC

表326-ReadDTCInformation，子功能= readSupportedDTCs，肯定响应，示例#11

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = readSupportedDTCs	0x0A	RSUPDTC
#3	DTCStatusAvailabilityMask	0x7F	DTCSAM
#4	DTCAndStatusRecord#1 [DTCHighByte]	0x12	DTCHB
#5	DTCAndStatusRecord#1 [DTCMiddleByte]	0x34	DTCMB
#6	DTCAndStatusRecord#1 [DTCLowByte]	0x56	DTCLB
#7	DTCAndStatusRecord#1 [statusOfDTC]	0x24	SODTC
#8	DTCAndStatusRecord#2 [DTCHighByte]	0x23	DTCHB
#9	DTCAndStatusRecord#2 [DTCMiddleByte]	0x45	DTCMB
#10	DTCAndStatusRecord#2 [DTCLowByte]	0x05	DTCLB
#11	DTCAndStatusRecord#2 [statusOfDTC]	0x00	SODTC
#12	DTCAndStatusRecord#3 [DTCHighByte]	是0xAB	DTCHB
#13	DTCAndStatusRecord#3 [DTCMiddleByte]	0xCD	DTCMB
#14	DTCAndStatusRecord#3 [DTCLowByte]	0x01	DTCLB
#15	DTCAndStatusRecord#3 [statusOfDTC]	0x2F	SODTC

11.3.5.13 示例#12 - ReadDTCInformation，子函数= reportFirstTestFailedDTC，可用信息

11.3.5.13.1 示例#12概述

此示例演示reportFirstTestFailedDTC子函数参数的用法，其中假定自从服务器发出最后一次ClearDiagnosticInformation请求以来发生了至少一次失败的DTC。

如果自从服务器发出最后一次ClearDiagnosticInformation请求后服务器中只有一个DTC发生故障，则服务器将返回相同的信息以响应来自客户端的reportMostRecentTestFailedDTC请求。

在此示例中，响应reportFirstTestFailedDTC返回的DTC的状态在请求时不再是当前状态（请求服务器报告最近失败/已确认的DTC时可能出现同样的现象）。

以下示例中的请求/响应消息的一般格式也适用于子功能参数reportFirstConfirmedDTC，reportMostRecentTestFailedDTC和reportMostRecent-ConfirmedDTC（适用于DTC状态并处于类似假设下）。

11.3.5.13.2 示例#12的假设

以下假设适用：

— 自从服务器发出最后一次ClearDiagnosticInformation请求后，至少有一个DTC失败。

— 服务器支持所有状态位用于掩蔽目的。

— DTC编号0x123456 =自上次代码清除后首次检测到故障DTC。

— 以下假设适用于DTC 0x123456，statusOfDTC 0x26 (0010 0110$_b$)：

表327 - statusOfDTC = 0x26

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不失败
testFailedThisOperationCycle	1	1	DTC从未在当前操作周期中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	0	DTC从未被证实
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除后，DTC至少失败一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

11.3.5.13.3 示例#12消息流

在以下示例中，DTC 0x123456返回到客户端的请求。

表328-ReadDTCInformation，子功能= reportFirstTestFailedDTC，请求消息流程
示例#12

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportFirstTestFailedDTC， suppressPosRspMsgIndicationBit = FALSE	0x0B	RFCDTC

表329 - ReadDTCInformation，子函数= reportFirstTestFailedDTC，肯定响应，
示例#12

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportFirstTestFailedDTC	0x0B	RFCDTC
#3	DTCStatusAvailabilityMask	为0xFF	DTCSAM
#4	DTCAndStatusRecord [DTCHighByte]	0x12	DTCHB
#5	DTCAndStatusRecord [DTCMiddleByte]	0x34	DTCMB
#6	DTCAndStatusRecord [DTCLowByte]	0x56	DTCLB
#7	DTCAndStatusRecord [statusOfDTC]	0x26	SODTC

11.3.5.14 示例#13 - ReadDTCInformation，子函数= reportFirstTestFailedDTC，没有可用的信息

11.3.5.14.1 示例#13概述

此示例演示了reportFirstTestFailedDTC子函数参数的用法，其中假定自从服务器发出最后一次ClearDiagnosticInformation请求以来没有发生失败的DTC。

以下示例中的请求/响应消息的一般格式也适用于子函数参数reportFirstConfirmedDTC，reportMostRecentTestFailedDTC和reportMostRecentConfirmedDTC（适用于DTC状态并且处于类似假设下）。

11.3.5.14.2 示例#13的假设

以下假设适用：

自从服务器发出最后一次ClearDiagnosticInformation请求以来没有发生失败的DTC。 服务器支持

所有状态位用于掩蔽目的。

11.3.5.14.3 示例#13消息流

在以下示例中，没有DTC返回到客户端的请求。

表330-ReadDTCInformation，子功能= reportFirstTestFailedDTC，请求消息流程示例#13

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportFirstTestFailedDTC, suppressPosRspMsgIndicationBit = FALSE	0x0B	RFCDTC

表331 - ReadDTCInformation，子函数= reportFirstTestFailedDTC，肯定响应，示例#13

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportFirstTestFailedDTC	0x0B	RFCDTC
#3	DTCStatusAvailabilityMask	为0xFF	DTCSAM

11.3.5.15 示例#14 - ReadDTCInformation，子函数= reportNumberOfEmissionsOBD-DTCByStatusMask

11.3.5.15.1 示例#14概述

此示例演示reportNumberOfEmissionsOBDDTCByStatusMask子函数参数的用法以及各种屏蔽原则。

11.3.5.15.2 示例#14的假设

服务器支持所有状态位用于掩蔽目的。 此外，该服务器总共支持三种与排放相关的OBD DTC（为了简单起见！），在客户请求时它们具有以下状态：

以下假设适用于与排放相关的OBD DTC P0005-00燃油关闭阀“A”控制回路/打开（0x000500），statusOfDTC 0xAE（1010 1110$_b$）。

表332 - DTC P0005-00的statusOfDTC = 0xAE

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不失败
testFailedThisOperationCycle	1	1	DTC在当前操作循环中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	1	DTC在请求时被确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除后，DTC至少失败一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	1	服务器正在请求warningIndicator处于活动状态（OBD DTC）

以下假设适用于与排放相关的OBD DTC P022F-00中冷器旁路控制“B”电路高（0x022F00），状态0xDTC为0xAC（1010 1100$_b$）。

表333 - DTC P022F-00的statusOfDTC = 0xAC

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	DTC在请求时不失败
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	1	DTC在请求时被确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除后，DTC至少失败一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期，
warningIndicatorRequested	7	1	服务器正在请求warningIndicator处于活动状态（OBD DTC）

以下假设适用于与排放相关的OBD DTC P0A09-00 DC / DC转换器状态电路低输入（0x0A0900），状态0xDTC 0xAF（1010 1111$_b$）。

表334 – statusOfDTC = DTC P0A09-00的AF

statusOfDTC: 位字段名称	位#	位状态	描述
testFailed	0	1	DTC在请求时失败
testFailedThisOperationCycle	1	1	DTC在当前操作循环中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	1	DTC在请求时被确认
testNotCompletedSinceLastClear	4	0	自上次代码清除以来，DTC测试已完成
testFailedSinceLastClear	5	1	自上次代码清除以来，DTC测试至少失败了一次
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	1	服务器正在请求warningIndicator处于活动状态（OBD DTC）

11.3.5.15.3 示例#14消息流

在以下示例中，计数为3会返回给客户端，因为在假设中定义的所有DTC都与客户端定义的状态掩码0x08 – confirmedDTC（0000 1000_b）匹配。

表335 – ReadDTCInformation，子功能= reportNumberOfEmissionsOBD-DTCByStatusMask，请求消息流程示例#14

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportNumberOfEmissionsOBDDTCByStatusMask, suppressPosRspMsgIndicationBit = FALSE	0x12	RNOOEOBDDTCBSM
#3	DTCStatusMask	0x08	DTCSM

表336 – ReadDTCInformation，子功能= reportNumberOfEmissionsOBD-DTCByStatusMask，肯定响应，示例#14

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportNumberOfEmissionsOBDDTCByStatusMask	0x12	RNOOEOBDDTCBSM
#3	DTCStatusAvailabilityMask	为0xFF	DTCSAM
#4	DTCFormatIdentifier = SAE_J2012-DA_DTCFormat_00	0x00	J2012-DADTCF00
#5	DTCCount [DTCCountHighByte]	0x00	DTCCHB
#6	DTCCount [DTCCountLowByte]	0x03	DTCCLB

11.3.5.16 示例#15 - ReadDTCInformation，子功能= reportEmissionsOBDDTC-ByStatusMask，返回所有匹配的OBD DTC

11.3.5.16.1 示例#15概述

此示例演示reportEmissionsOBDDTCByStatusMask子函数参数的用法，以及各种掩蔽原理以及不支持的掩码位。

11.3.5.16.2 示例#15的假设

服务器支持所有状态位用于掩蔽目的。 服务器总共支持三个DTC（为了简单起见！），如11.3.5.15.2中定义的那样。

11.3.5.16.3 示例#15消息流

在以下示例中，与排放相关的OBD DTC P0005-AE燃油关闭阀“A”控制回路/打开（0x000500），P022F-00中冷器旁路控制“B”电路高（0x022F00）和P0A09-00 DC / DC转换器状态电路低输入（0x0A0900）返回到客户端请求，因为在假设中定义的所有DTC都与客户端定义的状态掩码0x80 - warningIndicatorRequested（1000 0000_b）匹配。

注意 服务器应绕过它不支持的那些状态位的掩码。

表337 - ReadDTCInformation，子功能= reportEmissionsOBDDTCByStatusMask，请求消息流程示例#15

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportEmissionsOBDDTCByStatusMask, suppressPosRspMsgIndicationBit = FALSE	0x13	ROBDDTCBSM
#3	DTCStatusMask	0x80	DTCSM

表338 - ReadDTCInformation，子函数= reportDTCByStatusMask，肯定响应，示例#15

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportEmissionsOBDDTCByStatusMask	0x13	ROBDDTCBSM
#3	DTCStatusAvailabilityMask	为0xFF	DTCSAM
#4	DTCAndStatusRecord#1 [DTCHighByte]	0x00	DTCHB
#5	DTCAndStatusRecord#1 [DTCMiddleByte]	0x05	DTCMB
#6	DTCAndStatusRecord#1 [DTCLowByte]	0x00	DTCLB
#7	DTCAndStatusRecord#1 [statusOfDTC]	0xAE	SODTC
#8	DTCAndStatusRecord#2 [DTCHighByte]	0x02	DTCHB
#9	DTCAndStatusRecord#2 [DTCMiddleByte]	0x2F	DTCMB
#10	DTCAndStatusRecord#2 [DTCLowByte]	0x00	DTCLB
#11	DTCAndStatusRecord#2 [statusOfDTC]	0xAC	SODTC
#12	DTCAndStatusRecord#3 [DTCHighByte]	0x0A	DTCHB
#13	DTCAndStatusRecord#3 [DTCMiddleByte]	0x09	DTCMB
#14	DTCAndStatusRecord#3 [DTCLowByte]	0x00	DTCLB
#15	DTCAndStatusRecord#3 [statusOfDTC]	0xAF执行	SODTC

11.3.5.17 示例#16 - ReadDTCInformation，子函数= reportEmissionsOBDDTC-ByStatusMask（确认的DTC和warningIndicatorRequested），匹配的DTC返回

11.3.5.17.1 示例#16概述

此示例演示reportEmissionsOBDDTCByStatusMask子函数参数的使用情况以及屏蔽原理，以请求服务器报告状态为“confirmedDTC”和“warningIndicatorRequested (MIL = ON)”且与不受支持的状态相关的与排放相关的OBD DTC屏蔽位。 此示例显示了一个典型的OBD扫描工具类型请求，用于排放相关的OBD DTC，这些DTC会导致MIL开启，因此不会通过I / M（检查和维护）测试。

11.3.5.17.2 示例#16的假设

服务器不支持掩码用途的位0（testFailed），位4（testNotCompletedSinceLastClear）和位5（testFailedSinceLastClear）。 这导致DTCStatusAvailabilityMask值为0xCE (1100 1110$_b$)。

客户端使用值为0x88（1000 1000$_b$）的DTC状态掩码，因为只有状态为“confirmedDTC = 1”和“warningIndicatorRequested = 1”的DTC应显示给技术人员。 该服务器总共支持三个DTC（为了简单起见！），它们在客户端请求时具有以下状态：

以下假设适用于DTC P010A-14质量或体积空气流量“A” - 接地短路或断路（0x010A14），状态0xDTC 0x00 (0000 0000$_b$)：

表339 – DTC P010A-14的statusOfDTC = 0x00

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	不适用
testFailedThisOperationCycle	1	0	DTC从未在当前操作周期中失败
pendingDTC	2	0	DTC在当前或之前的操作周期中没有失败
confirmedDTC	3	0	DTC在请求时未得到确认
testNotCompletedSinceLastClear	4	0	不适用
testFailedSinceLastClear	5	0	不适用
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	0	服务器不请求warningIndicator处于活动状态

以下假设适用于DTC P0180-17燃油温度传感器A-电路电压高于阈值（0x018017），statusOfDTC为0x8E（$1000\ 1110_b$）：

表340 – DTC P0180-17的statusOfDTC = 0x8E

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	不适用
testFailedThisOperationCycle	1	1	DTC在当前操作循环中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	1	DTC在请求时被确认
testNotCompletedSinceLastClear	4	0	不适用
testFailedSinceLastClear	5	0	不适用
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	1	服务器正在请求warningIndicator处于活动状态（OBD DTC）

以下假设适用于DTC P0190-1D燃油导轨压力传感器“A”– 电路电流超出范围（0x01901D），状态0xDTC为0x8E（$1000\ 1110_b$）：

表341 – DTC P0190-1D的statusOfDTC = 0x8E

statusOfDTC：位字段名称	位#	位状态	描述
testFailed	0	0	不适用
testFailedThisOperationCycle	1	1	DTC在当前操作循环中失败
pendingDTC	2	1	DTC在当前或之前的运行周期中失败
confirmedDTC	3	1	DTC在请求时被确认
testNotCompletedSinceLastClear	4	0	不适用
testFailedSinceLastClear	5	0	不适用
testNotCompletedThisOperationCycle	6	0	DTC测试完成了这个操作周期
warningIndicatorRequested	7	1	服务器正在请求warningIndicator处于活动状态（OBD DTC）

11.3.5.17.3 示例#16消息流

在以下示例中，将P0180-17（0x018017）和P0190-1D（0x01901D）返回给客户端的请求。

注意　　服务器应绕过它不支持的那些状态位的掩码。

表342 - ReadDTCInformation，子功能= reportEmissionsOBDDTCByStatusMask，请求消息流程示例#16

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportEmissionsOBDDTCByStatusMask, suppressPosRspMsgIndicationBit = FALSE	0x13	ROBDDTCBSM
#3	DTCStatusMask	0x88	DTCSM

表343 - ReadDTCInformation，子函数= reportDTCByStatusMask，肯定响应，示例#16

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportEmissionsOBDDTCByStatusMask	0x13	ROBDDTCBSM
#3	DTCStatusAvailabilityMask	0xCE	DTCSAM
#8	DTCAndStatusRecord#1 [DTCHighByte]	0x01	DTCHB
#9	DTCAndStatusRecord#1 [DTCMiddleByte]	0x80	DTCMB
#10	DTCAndStatusRecord#1 [DTCLowByte]	0x17	DTCLB
#11	DTCAndStatusRecord#1 [statusOfDTC]	0x8E	SODTC
#12	DTCAndStatusRecord#2 [DTCHighByte]	0x01	DTCHB
#13	DTCAndStatusRecord#2 [DTCMiddleByte]	0x90	DTCMB
#14	DTCAndStatusRecord#2 [DTCLowByte]	0x1D	DTCLB
#15	DTCAndStatusRecord#2 [statusOfDTC]	0x8E	SODTC

11.3.5.18 示例#17 - ReadDTCInformation，子函数= reportDTCExtDataRecordByRecordNumber

11.3.5.18.1 示例#17概述

此示例演示reportDTCExtDataRecordByRecordNumber子函数参数的用法。

11.3.5.18.2 示例#17的假设

以下假设适用：

a) 服务器支持为所有DTC存储两个DTCExtendedData记录的功能。

b) 假设服务器请求服务器存储的所有可用DTCExtendedData记录的记录号为0x05。

c) 假设DTC 0x123456的statusOfDTC为0x24，并且下列扩展数据可用于DTC。

d) DTCExtendedData通过DTCExtDataRecordNumber 0x05引用

e) 假设DTC 0x234561的状态0xDTC为0x24，并且下列扩展数据可用于DTC。

f) DTCExtendedData通过DTCExtDataRecordNumber 0x05引用。

表344 -DTC 0x123456的DTCExtDataRecordNumber 0x05内容

数据字节	DTCExtDataRecordNumber的DTCExtDataRecord内容0x05	字节值
#1	预热循环计数器 - 自DTC命令MIL关闭以来的预热循环次数	0x17

表345 - DTC 0x234561的DTCExtDataRecordNumber 0x05内容

数据字节	DTCExtDataRecordNumber的DTCExtDataRecord内容0x05	字节值
#1	预热循环计数器 - 自DTC命令MIL关闭以来的预热循环次数	0x79

11.3.5.18.3 示例#17消息流

在以下示例中，客户端请求包含DTC编号和值为0x05（报告所有DTCExtDataRecords）的DTCExtDataRecordNumber的DTCMaskRecord。 服务器返回两个记录客户端提交的DTCExtDataRecordNumber的DTC。

表346 - ReadDTCInformation，子函数= reportDTCExtDataRecordByRecordNumber，请求消息流程示例#7

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	sub-function = reportDTCExtDataRecordByRecordNumber,	0x16	RDTCEDRBDN
#3	DTCExtDataRecordNumber	0x05	DTCEDRN

表347 - ReadDTCInformation, 子函数= reportDTCExtDataRecordByRecordNumber, 肯定响应, 示例#7

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportDTCExtDataRecordByRecordNumber	0x16	RDTCEDRBDN
#3	DTCAndStatusRecord [DTCHighByte]	0x12	DTCHB
#4	DTCAndStatusRecord [DTCMiddleByte]	0x34	DTCMB
#5	DTCAndStatusRecord [DTCLowByte]	0x56	DTCLB
#6	DTCAndStatusRecord [statusOfDTC]	0x24	SODTC
#7	DTCExtDataRecordNumber	0x05	DTCEDRN
#8	DTCExtDataRecord [byte#1]	0x17	ED_1
#9	DTCAndStatusRecord [DTCHighByte]	0x23	DTCHB
#10	DTCAndStatusRecord [DTCMiddleByte]	0x45	DTCMB
#11	DTCAndStatusRecord [DTCLowByte]	0x61	DTCLB
#12	DTCAndStatusRecord [statusOfDTC]	0x24	SODTC
#13	DTCExtDataRecordNumber	0x05	DTCEDRN
#14	DTCExtDataRecord [byte#1]	0x79	ED_1

11.3.5.19 示例#18 - ReadDTCInformation, 子函数= reportWWHOBDDTCByMaskRecord 11.3.5.19.1示例#18概述

此示例演示确认的DTC (DTC状态掩码0x08) 的reportWWHOBDDTCByMaskRecord子功能参数的用法。 该车辆使用连接两个排放相关服务器的CAN总线。

客户端使用以下请求参数设置:

— FunctionalGroupIdentifier = 0x33 (排放系统组),

— DTCSeverityMaskRecord.DTCSeverityMask = 0xFF (报告具有任何严重性和类状态的DTC),

— DTCSeverityMaskRecord.DTCStatusMask = 0x08 (报告具有确认DTC状态的DTC ='1')。 这些服

务器支持以下设置:

— FunctionalGroupIdentifier = 0x33 (排放系统组),

— DTCStatusAvailabilityMask = 0xFF,

— DTCSeverityAvailabilityMask = 0xFF,

— DTCFormatIdentifier = SAE_J2012-DA_DTCFormat_04 = 0x04。

11.3.5.19.2 示例#18的假设

示例#1的所有假设都适用。

11.3.5.19.3 示例#18消息流

在以下示例中，由于0x2F (0010 1111二进制) 的statusOfDTC与0x08 (0000 1000_b) 客户端定义的状态掩码匹配，因此服务器1仅报告DTC P2522-1F A / C请求“B” - 电路间歇 (0x25221F))。 服务器#2报告DTC P0235-12涡轮增压器/增压器升压传感器“A” - 由于0x2E (0010 1110_b) 的statusOfDTC与0x08 (0000 1000_b) 的客户端定义的状态掩码相匹配，电路与电池短路。

表348 - ReadDTCInformation请求，子功能= reportNumberOfDTCByStatusMask

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDTCInformation请求SID	0x19	RDTCI
#2	子函数= reportWWHOBDDTCByMaskRecord, suppressPosRspMsgIndicationBit = FALSE	0x42	RWWHOBDDTCBMR
#3	FunctionalGroupIdentifier (FunctionalGroupIdentifier = emissions = 0x33)	0x33	FGID
#4	DTCSeverityMaskRecord[] = [DTCStatusMask]	0x08	DTCSM
#5	DTCSeverityMaskRecord[] = [DTCSeverityMask]	为0xFF	DTCSVM

表349 - ReadDTCInformation响应，子功能= reportWWHOBDDTCByStatusMask

消息方向	服务器#1 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportWWHOBDDTCByMaskRecord	0x42	RWWHOBDDTCBMR
#3	FunctionalGroupIdentifier (FunctionalGroupIdentifier = emissions = 0x33)	0x33	FGID
#4	DTCStatusAvailabilityMask	为0xFF	DTCSAM
#5	DTCSeverityAvailabilityMask	为0xFF	DTCSVAM
#6	DTCFormatIdentifier = [SAE_J2012-DA_DTCFormat_04]	0x04	J2012-DADTCF04
#7	DTCAndSeverityRecord [DTCSeverity#1]	0x20	DTCASR_DTCS
#8	DTCAndSeverityRecord [DTCHighByte#1]	0x25	DTCASR_DTCHB
#9	DTCAndSeverityRecord [DTCMiddleByte#1]	0x22	DTCASR_DTCMB
#10	DTCAndSeverityRecord [DTCLowByte#1]	0x1F	DTCASR_DTCLB
#11	DTCAndSeverityRecord [statusOfDTC#1]	0x2F	DTCASR_SODTC

表350 - ReadDTCInformation响应，子功能= reportOBDDTCByStatusMask

消息方向	服务器#2 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDTCInformation响应SID	0x59	RDTCIPR
#2	reportType = reportWWHOBDDTCByMaskRecord	0x42	RWWHOBDDTCBMR
#3	FunctionalGroupIdentifier (FunctionalGroupIdentifier = emissions = 0x33)	0x33	FGID
#4	DTCStatusAvailabilityMask	为0xFF	DTCSAM
#5	DTCSeverityAvailabilityMask	为0xFF	DTCSVAM
#6	DTCFormatIdentifier = [SAE_J2012-DA_DTCFormat_04]	0x04	J2012-DADTCF04
#7	DTCAndSeverityRecord [DTCSeverity#1]	0x20	DTCASR_DTCS
#8	DTCAndSeverityRecord [DTCHighByte#1]	0x02	DTCASR_DTCHB
#9	DTCAndSeverityRecord [DTCMiddleByte#1]	0x35	DTCASR_DTCMB
#10	DTCAndSeverityRecord [DTCLowByte#1]	0x12	DTCASR_DTCLB
#11	DTCAndSeverityRecord [statusOfDTC#1]	0x2E	DTCASR_SODTC

12 InputOutput控制功能单元

12.1 概观

表351定义了InputOutput控制功能单元。

表351 - InputOutput控制功能单元

服务	描述
InputOutputControlByIdentifier	客户端请求控制特定于服务器的输入/输出。

12.2 InputOutputControlByIdentifier (0x2F) 服务

12.2.1 服务说明

InputOutputControlByIdentifier服务被客户用来将输入信号，内部服务器功能和/或力控制的值替换为电子系统的输出（致动器）的值。 通常，此服务用于相对简单的（例如，静态的）输入替换/输出控制，而如果需要更复杂的输入替换/输出控制，则使用routineControl。

客户端请求消息包含数据标识符以引用服务器的输入信号，内部服务器功能和/或输出信号（致动器）（在设备控制访问的情况下，其可能引用一组信号） 。 controlOptionRecord参数应包括服务器输入信号，内部功能和/或输出信号所需的所有信息。 如果要被控制的数据标识符引用多于一个参数（即，数据标识符被分组化或位图化），则车辆制造商可以要求请求消息包含controlEnableMask。 如果车辆管理员选择支持EnableMask概念，则对此服务的所有类型的InputOutputControlByIdentifier请求都必须使用controlEnableMask参数。 如果在引用测量输出值或反馈值的dataIdentifier上请求inputOutputControlByIdentifier，则服务器应负责在服务器控制策略内替换正确的目标值，以便正常服务器控制策略将尝试从客户端达到所需状态请求消息。

如果请求控制成功启动或达到其所需状态，服务器应发送肯定响应消息。 即使dataIdentifier当前不在测试人员控制下，服务器也应该向具有returnControlToECU的inputOutputControlParameter的请求消息发送肯定响应消息。 另外，当接收到returnControlToECU请求时，服务器应始终向客户端提供将controlMask（如果支持）位全部设置为'1'的能力，以便将分组化或位图数据标识符的控制完全返回给ECU。 请求消息的controlOptionRecord参数内的inputOutputControlParameter后面的controlState字节的格式和长度应与请求的dataIdentifier的dataRecord的长度和格式完全匹配。 这样就可以确保通过使用具有相同DID的服务ReadDatabyIdentifier来检索实际输出或输入状态。

当使用inputOutputControlByIdentifier服务执行输入替换或输出控制时，对接受请求的ECU有两个基本要求。首先是将dataIdentifier中的参数所引用的适当数据对象与否则会更新数据对象值的所有上游控制策略断开连接。 第二种方法是将值代入用于控制策略的所有下游活动的适当数据对象中。 例如，直接强制打开前照灯的测试仪要求需要防止前照灯开关位置影响前照灯输出，并将所需的“打开”状态替换为最终决定前照灯的功能所使用的数据对象状态所需的输出。

该服务允许在单个请求消息中控制单个dataIdentifier及其相应的参数。 这样做，服务器将响应一个单一的响应消息，包括请求消息的dataIdentifier和controlStatus信息。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

12.2.2 请求消息

12.2.2.1 请求消息定义

表352 - 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	InputOutputControlByIdentifier请求SID	M	0x2F	IOCBI
#2 #3		M M	0x00 - 0xFF 0x00 - 0xFF	IOI_ B1 B2
#4 : #4+(m-1)		M_1 C_1 : C_1	0x00 - 0xFF 0x00 - 0xFF : 0x00 - 0xFF	CSR_ IOCP_ CS_ : CS_
#4+m : #4 + M + (R-1)		C_2 : C_2	0x00 - 0xFF : 0x00 - 0xFF	CEM_ 厘米_ : 厘米_

M_1InputOutputControlParameter应按照E.1中的定义来实现。

C_1：此参数的存在取决于dataIdentifier和inputOutputControlParameter（参见E.1）。

C_2：如果车辆制造商支持controlEnableMask概念，则如果dataIdentifier包含多个参数（参见controlEnableMaskRecord定义），则应包含此参数

12.2.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

12.2.2.3 请求消息数据参数定义

为此服务定义了以下数据参数:

表353 – 请求消息数据参数定义

定义
dataIdentifier 该参数标识服务器本地输入信号，内部参数和/或输出信号。 该参数的适用值范围可以在C.1中定义的dataIdentifiers表中找到。
controlOptionRecord controlOptionRecord由一个或多个字节（inputOutputControlParameter和controlState#1到controlState#m）组成。controlOptionRecord参数详细信息应按照E.1中的定义来实现。
controlEnableMaskRecord controlEnableMaskRecord包含一个或多个字节（controlMask#1到controlMask#r）。 只有当要控制的dataIdentifier由多个参数组成时（即，dataIdentifier按照定义进行位映射或分组），才能支持controlEnableMaskRecord。controlEnableMaskRecord中应该有一个位对应于dataIdentifier中定义的每个单独的参数。 当要控制的dataIdentifier只包含一个参数时，不能支持controlEnableMaskRecord。 注意 dataIdentifier中的每个参数都可以是任意数量的位。 controlEnableMaskRecord中每个位的值将决定dataIdentifier中的相应参数是否会受到请求的影响。controlEnableMaskRecord中的位值'0'表示相应的参数不受此请求影响，并且位值'1'表示相应的参数受此请求影响。ControlMask#1的最高有效位应该与ControlState中的第一个参数对应，从ControlState#1的最高有效位开始，ControlMask#1的第二个最高有效位对应于ControlState中的第二个参数，并继续以这种方式利用尽可能多的ControlMask字节来屏蔽所有参数。 例如，ControlMask#2的最低有效位将对应于controlState中的16[□]参数。 对于位图数据标识符，不支持的位还应在controlEnableMaskRecord中具有相应的位，以便controlEnableMaskRecord中每个参数的掩码位位置应与controlState中相应参数的位置精确匹配。

12.2.3 积极的回应消息

12.2.3.1 积极响应消息的定义

表354 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	InputOutputControlByIdentifier响应SID	M	0x6F	IOCBIPR
#2 #3		M M	0x00 – 0xFF 0x00 – 0xFF	IOI_ B1 B2
#4 #5 : #5+(m-1)		M C_1 : C_1	0x00 – 0xFF 0x00 – 0xFF : 0x00 – 0xFF	CSR_ IOCP_ CS_ : CS_
C_1: 此参数的存在取决于dataIdentifier和inputOutputControlParameter（参见E.1）。				

12.2.3.2 肯定回复消息数据参数定义

表355 – 响应消息数据参数定义

定义
dataIdentifier 该参数是来自请求消息的dataIdentifier（s）的回显。
controlStatusRecord controlState参数由多个字节组成（InputOutputControlParameter和controlState#1到controlState#m），其中包括例如反馈数据。 controlStatusRecord参数详细信息应按照E.1中的定义来实现

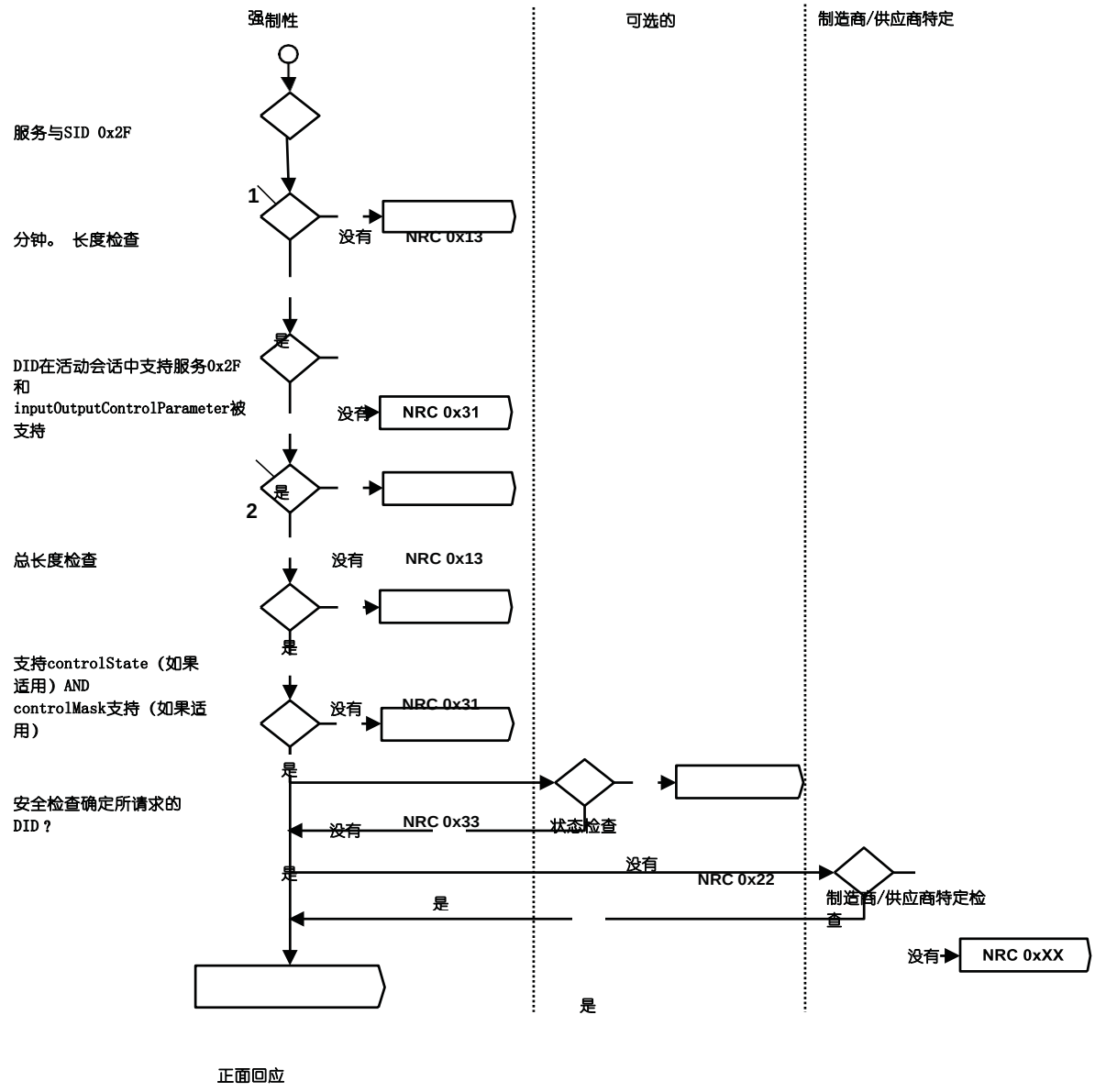
12.2.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 表356列出了每个响应代码出现的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表356 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果不满足请求InputOutputControl的条件，则应返回此NRC。	CNC
0x31	**requestOutOfRange** 如果发生以下情况，应发送NRC： — 设备不支持请求的dataIdentifier值； — inputOuptputControlParameter中包含的值无效（请参阅inputOutputControlParameter的定义）； — controlOptionRecord记录的一个或多个可应用的controlState值是无效的； — 设备不支持在ControlEnableMaskRecord中启用控制的位的组合；	ROOR
0x33	**securityAccessDenied** 如果客户端使用有效的安全数据标识符发送请求并且服务器的安全功能当前处于活动状态，则应返回此NRC。	伤心

评估顺序记录在图24中。



键

1 至少4 (SI + DID + IOCP)

2 如果IOCP = shortTermAdjustment, 1字节SI + 2字节DID + 1字节IOCP +第n字节controlState +第n字节controlMask（如果适用），
如果IOCP <> shortTermAdjustment, 1字节SI + 2字节DID + 1字节IOCP +第n个字节controlMask（如果适用）

图24 - 针对InputOutputControlByIdentifier服务的NRC处理

12.2.5 消息流示例 (s) InputOutputControlByIdentifier

12.2.5.1 假设

下面的示例显示了如何将InputOutputControlByIdentifier与HVAC控制模块一起使用，并假定使用单个服务器执行物理通信。

12.2.5.2 示例#1 - “进气门位置” shortTermAdjustment

被控制的参数是与dataIdentifier (0x9B00) 关联的 “Air Inlet Door Position”。 转换: 进气门位置[%] =十进制 (十六进制) * 1 [%]

12.2.5.2.1 步骤#1: ReadDataByIdentifier

本示例使用ReadDataByIdentifier服务读取进气门位置的当前状态。

表357 - ReadDataByIdentifier请求消息流程示例#1 - 步骤#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDataByIdentifier请求SID	0x22	RDBI
#2	dataIdentifier [byte#1] = 0x9B	0x9B	DID_B1
#3	dataIdentifier [byte#2] = 0x00 (“进气门位置”)	0x00	DID_B2

表358- ReadDataByIdentifier肯定响应消息流程示例#1-步骤#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	ReadDataByIdentifier响应SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] = 0x9B	0x9B	DID_B1
#3	dataIdentifier [byte#2] = 0x00 (“进气门位置”)	0x00	DID_B2
#4	dataRecord [数据#1] = 10%	0x0A	DREC_DATA1

12.2.5.2.2 步骤#2：短期调整

表359 – InputOutputControlByIdentifier请求消息流程示例#1 – 步骤#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier请求SID	0x2F	IOCBI
#2	dataIdentifier [byte#1] = 0x9B	0x9B	IOI_B1
#3	dataIdentifier [byte#2] = 0x00（“进气门位置”）	0x00	IOI_B2
#4	controlOptionRecord [inputOutputControlParameter] = shortTermAdjustment	0x03	IOCP_STA
#5	controlOptionRecord [controlState#1] = 60%	0x3C	CS_1

表360 – InputOutputControlByIdentifier肯定响应消息流程示例#1 – 步骤#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier响应SID	0x6F	IOCBLIPR
#2	dataIdentifier [byte#1] = 0x9B	0x9B	IOI_B1
#3	dataIdentifier [byte#2] = 0x00（“进气门位置”）	0x00	IOI_B2
#4	controlStatusRecord [inputOutputControlParameter] = shortTermAdjustment	0x03	IOCP_STA
#5	controlStatusRecord [controlState#1] = 12%	0x0C	CS_1

注意客户端发送了一个inputOutputControlByIdentifier请求消息，如上所述。 服务器发送了一个即时的肯定响应消息，其中包括值为12%的controlState参数“进气门位置”。 进气门需要一定的时间才能移动到所需的60%的值。

12.2.5.2.3 步骤#3：ReadDataByIdentifier

本示例使用readDataByIdentifier服务读取进气门位置的当前状态。

表361 – ReadDataByIdentifier请求消息流程示例#1 – 步骤#3

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier请求SID	0x22	RDBI
#2	dataIdentifier [byte#1] = 0xB9	0x9B	DID_B1
#3	dataIdentifier [byte#2] = 0x00（“进气门位置”）	0x00	DID_B2

表362- ReadDataByIdentifier肯定响应消息流程示例#1-步骤#3

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	ReadDataByIdentifier响应SID	0x62	RDBIPR
#2	dataIdentifier [byte#1] = 0x9B	0x9B	DID_B1
#3	dataIdentifier [byte#2] = 0x00（“进气门位置”）	0x00	DID_B2
#4	dataRecord [data#1] = 60%	0x3C	DREC_DATA1

注意当inputOutputControlByIdentifier处于活动状态时，客户端发送了一个如上所述的readDataByIdentifier请求消息。服务器控制策略将花费有限的时间最终达到期望值。 上面的例子反映了服务器何时最终达到了期望的目标值。

12.2.5.2.4 步骤#4：returnControlToECU

表363 - InputOutputControlByIdentifier请求消息流程示例#1 - 步骤#4

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier请求SID	0x2F	IOCBI
#2	dataIdentifier [byte#1] = 0x9B	0x9B	IOI_B1
#3	dataIdentifier [byte#2] = 0x00（“进气门位置”）	0x00	IOI_B2
#4	controlOptionRecord [inputOutputControlParameter] = returnControlToECU	0x00	RCTECU

表364- InputOutputControlByIdentifier肯定响应消息流程示例#1-步骤#4

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier响应SID	0x6F	IOCBIPR
#2	dataIdentifier [byte#1] = 0x9B	0x9B	IOI_B1
#3	dataIdentifier [byte#2] = 0x00（“进气门位置”）	0x00	IOI_B2
#4	controlStatusRecord [inputOutputControlParameter] = returnControlToECU	0x00	RCTECU
#5	controlStatusRecord [controlState#1] = 58%	0x3A	CS_1

12.2.5.2.5 步骤#5: freezeCurrentState

表365 – InputOutputControlByIdentifier请求消息流程示例#1 – 步骤#5

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier请求SID	0x2F	IOCBI
#2	dataIdentifier [byte#1] = 0x9B	0x9B	IOI_B1
#3	dataIdentifier [byte#2] = 0x00 (“进气门位置”)	0x00	IOI_B2
#4	controlOptionRecord [inputOutputControlParameter] = freezeCurrentState	0x02	IOCP_FCS

表366– InputOutputControlByIdentifier肯定响应消息流程示例#1–步骤#5

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier响应SID	0x6F	IOCBIPR
#2	dataIdentifier [byte#1] = 0x9B	0x9B	IOI_B1
#3	dataIdentifier [byte#2] = 0x00 (“进气门位置”)	0x00	IOI_B2
#4	controlStatusRecord [inputOutputControlParameter] = freezeCurrentState	0x02	IOCP_FCS
#5	controlStatusRecord [controlState#1] = 50%	0x32	CS_1

12.2.5.3 示例#2 - EGR和IAC shortTerm调整

12.2.5.3.1 假设

本示例使用打包的dataIdentifier 0x0155来演示单个请求中单个参数或多个参数的控制。

本小节规定了shortTermAdjustment函数的测试条件和示例dataIdentifier 0x0155的相关消息流。dataIdentifier支持五个单独的参数，如下面的表367所述。

表367 - 复合数据块 - 数据标识符定义 - 示例#2

DID	数据字节	参数		数据记录内容
		数	尺寸	
	#1（所有位）	#1	8位	dataRecord [data#1] = IAC Pintle位置（n =计数）
	#2 - #3（所有位）	#2	16位	dataRecord [data#2-#3] = RPM（0 = 0 U / min，65 535 = 65 535 U / min）
	#4（比特7-4）	#3	4比特	dataRecord [数据#4（位7-4）] =踏板位置A：线性缩放，0 = 0%，15 = 120%
	#4（比特3-0）	#4	4比特	dataRecord [data#4（bits 3-0）] =踏板位置B：线性缩放，0 = 0%，15 = 120%
	#5（所有位）	#5	8位	dataRecord [数据#5] = EGR占空比：线性标度，0计数= 0%，255计数= 100%

DataIdentifier 0x0155按照定义进行打包，由五个基本参数组成。 为了个人控制目的，这些元素参数中的每一个都可以通过ControlEnableMaskRecord中的单个位来选择。 如果给定的数据标识符的定义不是打包或位图映射的，则请求消息中不存在ControlEnableMaskRecord。 ControlMask#1的最高有效位始终需要对应于从ControlState#1的最高有效位开始的dataIdentifier中的第一个参数。 这在表368中得到了证明。

表368 - ControlEnableMaskRecord - 示例#2

用于dataIdenitifier 0x0155的ControlEnableMaskRecord。 总大小= 1字节（即，仅由ControlEnableMask#1组成）		
位位置		ControlEnableMask#1 - 位含义（1 =受影响，0 =不受影响）
7	（最高有效位）	确定参数#1（IAC Pintle位置）是否会受到请求的影响
6		确定参数#2（RPM）是否会受到请求的影响
5		确定参数#3（踏板位置A）是否会受到请求的影响
4		确定参数#4（踏板位置B）是否会受到请求的影响
3		确定参数#5（EGR占空比）是否会受到请求的影响
2		由于没有相应的参数而没有影响
1		由于没有相应的参数而没有影响
0	（最低有效位）	由于没有相应的参数而没有影响

12.2.5.3.2 案例#1: 仅控制IAC Pintle位置

表369定义了InputOutputControlByIdentifier请求消息流程示例#2 - 案例#1。

表369 - InputOutputControlByIdentifier请求消息流程示例#2 - 案例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	InputOutputControlByIdentifier请求SID	0x2F	IOCBI
#2	dataIdentifier [byte#1] = 0x01	0x01	IOI_B1
#3	dataIdentifier [byte#2] = 0x55 (IAC / RPM / PPA / PPB / EGR)	0x55	IOI_B2
#4	controlOptionRecord [inputOutputControlParameter] = shortTermAdjustment	0x03	IOCP_STA
#5	controlOptionRecord [controlState#1] = IAC Pintle Position (7个计数)	0x07	CS_1
#6	controlOptionRecord [controlState#2] = RPM (XX)	0xXX	CS_2
#7	controlOptionRecord [controlState#3] = RPM (XX)	0xXX	CS_3
#8	controlOptionRecord [controlState#4] =踏板位置A (Y) 和B (Z)	0xYZ	CS_4
#9	controlOptionRecord [controlState#5] = EGR占空比 (XX)	0xXX	CS_5
#10	controlEnableMask [controlMask#1] =仅控制IAC Pintle位置	80	CM_1

注意controlState#2 - #5中RPM, 踏板位置A, 踏板位置B和EGR占空比传输的值无关紧要, 因为controlMask#1参数指定dataIdentifier中只有第一个参数会受到请求的影响。

表370定义了InputOutputControlByIdentifier肯定响应消息流程示例#2-案例#1。

表370 - InputOutputControlByIdentifier肯定响应消息流程示例#2 - 案例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明 (所有值均为十六进制)	字节值	助记符
#1	InputOutputControlByIdentifier响应SID	0x6F	IOCBLIPR
#2	dataIdentifier [byte#1] = 0x01	0x01	IOI_B1
#3	dataIdentifier [byte#2] = 0x55 (IAC / RPM / PPA / PPB / EGR)	0x55	IOI_B2
#4	controlStatusRecord [inputOutputControlParameter] = shortTermAdjustment	0x03	IOCP_STA
#5	controlStatusRecord [controlState#1] = IAC Pintle Position (7个计数)	0x07	CS_1
#6	controlStatusRecord [controlState#2] = RPM (750 U / min)	0x02	CS_2
#7	controlStatusRecord [controlState#3] = RPM	0xEE	CS_3
#8	controlStatusRecord [controlState#4] =踏板位置A (8%), 踏板位置B (16%)	0x12	CS_4
#9	controlStatusRecord [controlState#5] = EGR占空比 (35%)	0x59	CS_5

注意　　为controlState#1 - controlState#5中的所有参数传输的值应反映系统的当前状态。

12.2.5.3.3　情况#2：仅控制RPM

表371定义了InputOutputControlByIdentifier请求消息流程示例#2 - 情况#2。

表371 - InputOutputControlByIdentifier请求消息流程示例#2 - 案例#2

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier请求SID	0x2F	IOCBI
#2	dataIdentifier [byte#1] = 0x01	0x01	IOI_B1
#3	dataIdentifier [byte#2] = 0x55 (IAC / RPM / EGR)	0x55	IOI_B2
#4	controlOptionRecord [inputOutputControlParameter] = shortTermAdjustment	0x03	IOCP_STA
#5	controlOptionRecord [controlState#1] = IAC Pintle Position (XX计数)	0xXX	CS_1
#6	controlOptionRecord [controlState#2] = RPM (0x03E8 = 1000U / min)	0x03	CS_2
#7	controlOptionRecord [controlState#3] = RPM	0xE8	CS_3
#8	controlOptionRecord [controlState#4] =踏板位置A (Y) 和B (Z)	0xYZ	CS_4
#9	controlOptionRecord [controlState#5] = EGR占空比 (XX)	0xXX	CS_5
#10	controlEnableMask [controlMask#1] =仅控制RPM	0x40	CM_1

注意controlState#1和controlState#4 - #5中为IAC Pintle位置，踏板位置A，踏板位置B和EGR占空比传输的值无关紧要，因为controlMask#1参数指定只有dataIdentifier中的第二个参数将受到请求的影响。

表372定义了InputOutputControlByIdentifier肯定响应消息流程示例#2 - 情况#2。

表372 - InputOutputControlByIdentifier肯定响应消息流程示例#2 - 案例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier响应SID	0x6F	IOCBLIPR
#2	dataIdentifier [byte#1] = 0x01	0x01	IOI_B1
#3	dataIdentifier [byte#2] = 0x55 (IAC / RPM / PPA / PPB / EGR)	0x55	IOI_B2
#4	controlStatusRecord [inputOutputControlParameter] = shortTermAdjustment	0x03	IOCP_STA
#5	controlStatusRecord [controlState#1] = IAC Pintle位置（9计数）	0x09	CS_1
#6	controlStatusRecord [controlState#2] = RPM (950 U / min)	0x03	CS_2
#7	controlStatusRecord [controlState#3] = RPM	0xB6	CS_3
#8	controlStatusRecord [controlState#4] =踏板位置A (8%)，踏板位置B (16%)	0x12	CS_4
#9	controlStatusRecord [controlState#5] = EGR占空比 (35%)	0x59	CS_5

注意 为controlState#1 - controlState#5中的所有参数传输的值应反映系统的当前状态。

12.2.5.3.4 案例#3：控制踏板位置A和EGR占空比

表373定义了InputOutputControlByIdentifier请求消息流程示例#2 - 情况#3。

表373 - InputOutputControlByIdentifier请求消息流程示例#2 - 案例#3

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier请求SID	0x2F	IOCBI
#2	dataIdentifier [byte#1] = 0x01	0x01	IOI_B1
#3	dataIdentifier [byte#2] = 0x55 (IAC / RPM / PPA / PPB / EGR)	0x55	IOI_B2
#4	controlOptionRecord [inputOutputControlParameter] = shortTermAdjustment	0x03	IOCP_STA
#5	controlOptionRecord [controlState#1] = IAC Pintle Position	0xXX	CS_1
#6	(XX) controlOptionRecord [controlState#2] = RPM (XX)	0xXX	CS_2
#7	controlOptionRecord [controlState#3] = RPM (XX)	0xXX	CS_3
#8	controlOptionRecord [controlState#4] =踏板位置A (0x3 = 24%)，踏板位置B (Z)	0x3Z	CS_4
#9	controlOptionRecord [controlState#5] = EGR占空比 (45%)	0x72	CS_5
#10	controlEnableMask [controlMask#1] =控制踏板位置A和EGR	28	CM_1

注意因为controlMask#1参数指定只有dataIdentifier中的第三个和第五个参数，所以在控制状态#1 - #3和控制状态#4（位3-0）中为IAC Pintle位置，RPM和踏板位置B传输的值是不相关的会受到请求的影响。

表374定义了InputOutputControlByIdentifier肯定响应消息流程示例#2 - 情况#3。

表374 - InputOutputControlByIdentifier肯定响应消息流程示例#2 - 案例#3

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier响应SID	0x6F	IOCBLIPR
#2	dataIdentifier [byte#1] = 0x01	0x01	IOI_B1
#3	dataIdentifier [byte#2] = 0x55 (IAC / RPM / PPA / PPB / EGR)	0x55	IOI_B2
#4	controlStatusRecord [inputOutputControlParameter] = shortTermAdjustment	0x03	IOCP_STA
#5	controlStatusRecord [controlState#1] = IAC Pintle Position (7个计数)	0x07	CS_1
#6	controlStatusRecord [controlState#2] = RPM (850 U / min)	0x03	CS_2
#7	controlStatusRecord [controlState#3] = RPM	0x52	CS_3
#8	controlStatusRecord [controlState#4] =踏板位置A (24%) 踏板位置B (16%)	0x32	CS_4
#9	controlStatusRecord [controlState#4] = EGR占空比 (41%)	0x69	CS_5

注意 为controlState#1 - controlState#5中的所有参数传输的值应反映系统的当前状态。

12.2.5.3.5 案例#4：将所有参数的控制返回给ECU

表375定义了InputOutputControlByIdentifier请求消息流程示例#2 - 情况#4。

表375 - InputOutputControlByIdentifier请求消息流程示例#2 - 案例#4

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier请求SID	0x2F	IOCBI
#2	dataIdentifier [byte#1] = 0x01	0x01	IOI_B1
#3	dataIdentifier [byte#2] = 0x55 (IAC / RPM / PPA / PPB / EGR)	0x55	IOI_B2
#4	controlOptionRecord [inputOutputControlParameter] = returnControlToECU	0x00	RCTECU
#5	controlEnableMask [controlMask#1] =所有元素参数	为0xFF	CM_1

表376定义了InputOutputControlByIdentifier肯定响应消息流程示例#2-案例#4。

表376 - InputOutputControlByIdentifier肯定响应消息流程示例#2 - 案例#4

消息方向		服务器 客户端	
消息类型		响应	
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	InputOutputControlByIdentifier响应SID	0x6F	IOCBLIPR
#2	dataIdentifier [byte#1] = 0x01	0x01	IOI_B1
#3	dataIdentifier [byte#2] = 0x55 (IAC / RPM / PPA / PPB / EGR)	0x55	IOI_B2
#4	controlStatusRecord [inputOutputControlParameter] = returnControlToECU	0x00	RCTECU
#5	controlStatusRecord [controlState#1] = IAC Pintle位置（9计数）	0x09	CS_1
#6	controlStatusRecord [controlState#2] = RPM (850 U / min)	0x03	CS_2
#7	controlStatusRecord [controlState#3] = RPM	0x52	CS_3
#8	controlStatusRecord [controlState#4] =踏板位置A (8%) 踏板位置B (16%)	0x12	CS_4
#9	controlStatusRecord [controlState#4] = EGR占空比 (35%)	0x59	CS_5

注意 为controlState#1 - controlState#5中的所有参数传输的值应反映系统的当前状态。

13 常规功能单元

13.1 概观

表377定义了例程功能单元。

表377 - 日常功能单元

服务	描述
RoutineControl	客户端请求启动，停止服务器中的例程或请求例程结果。

这个功能单元规定了远程激活例程的服务，因为它们将在服务器和客户端中实现。 以下子条款描述了两种不同的实现方法（方法“A”和“B”）。 可能还有其他的实现方法。 方法“A”和“B”应作为实施日常服务的指导原则。

注意每个方法都可能具有在例程停止后请求例行结果服务的功能。 方法的选择和实施是车辆制造商和系统供应商的责任。

以下是方法“A”和“B”的简要说明：

— **方法“A”：**

— 该方法基于这样的假设，即客户端在服务器内存中启动例程之后，客户端应负责停止例程。

— 服务器例程应在启动例程的RoutineControl请求消息完成和第一个响应消息完成（如果基于服务器条件的“肯定”）之间的某段时间内在服务器的内存中启动。

— 在完成StopRoutine请求消息和完成第一个响应消息（如果根据服务器的条件为“正面”）后的一段时间内，服务器例程应停止在服务器的内存中。

— 例行程序停止后，客户可能要求例行结果。

— 方法“B”：

— 该方法基于这样的假设，即客户端在服务器的内存中启动例程后，服务器应负责停止例程。

— 服务器例程应在启动例程的RoutineControl请求消息完成和第一个响应消息完成（如果基于服务器条件的“肯定”）之间的某段时间内在服务器的内存中启动。

— 服务器例程应该随时被编程或在服务器的内存中初始化。

13.2 RoutineControl (0x31) 服务

13.2.1 服务说明

客户端使用RoutineControl服务来执行已定义的步骤序列并获取任何相关结果。 这项服务具有很大的灵活性，但典型的使用可能包括擦除内存，重置或学习自适应数据，运行自检，覆盖正常服务器控制策略以及控制服务器值随时间变化等功能，包括预定义的序列（例如，关闭敞篷车顶）等等。 通常，当用于控制输出时，该服务用于更复杂的类型控制，而inputOutputControlByIdentifier用于相对简单的（例如静态）输出控制。

13.2.1.1 概观

RoutineControl服务被客户用来：

— 开始一个例程，

— 停止一个例程，然后

— 请求例行结果

一个例程由一个2字节的routineIdentifier引用。

以下子条款指定了routineIdentifier引用的启动例程，停止例程和请求例程结果。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

13.2.1.2 启动由routineIdentifier引用的例程

例程应在StartRoutine请求消息完成和第一个响应消息完成之间的一段时间内在服务器的存储器中启动，如果响应消息为正数或负数，表明请求已执行或正在执行。

例程既可以是运行测试而不是正常的操作代码，也可以是在正常运行代码运行时启用和执行的例程。 特别是在第一种情况下，可能需要使用DiagnosticSessionControl服务在特定诊断会话中切换服务器，或者在使用StartRoutine服务之前使用SecurityAccess服务解锁服务器。

13.2.1.3 停止由routineIdentifier引用的例程

在StopRoutine请求消息完成后的一段时间内，服务器例程应停止在服务器的存储器中，如果响应消息为正或负，则表示停止例程的请求已执行或在要执行的进度。

注意 服务器例程应该随时被编程或在服务器的内存中初始化。

13.2.1.4 请求由routineIdentifier引用的例程结果

客户端使用该子函数来请求由例程标识符引用并由服务器内存中执行的例程生成的结果（例如退出状态信息）。

根据stopRoutine子函数参数的肯定响应消息中可能收到的例行结果（例如正常/异常Exit With Results），应使用requestRoutineResults子函数。

例程结果的例子可以是服务器收集的数据，由于服务器性能的限制，在例行执行期间无法传输数据。

13.2.2 请求消息

13.2.2.1 请求消息定义

表378 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	RoutineControl请求SID	M	0x31	RC
#2	子功能= [routineControlType]	M	0x00 – 0xFF	LEV_ RCTP_
#3 #4	routineIdentifier [] = [字节#1 (MSB) 字 节#2 (LSB)]	M M	0x00 – 0xFF 0x00 – 0xFF	RI_ B1 B2
#5 : #n	routineControlOptionRecord[] = [routineControlOption#1 : routineControlOption#m]	C/U : C/U	0x00 – 0xFF : 0x00 – 0xFF	RCEOR_ RCO_ : RCO _
C: 该参数是用户可选的，用于子功能参数startRoutine和stopRoutine。				

13.2.2.2 请求消息子函数参数$ Level（LEV_）定义

该服务使用子功能参数来选择对例程的控制。 表379（suppressPosRspMsgIndicationBit（bit 7）未显示）详细说明和使用可能的级别。

表379 – 请求消息子功能定义

位6 – 0	描述	Cvt	助记符
0x00	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
0x01	**startRoutine** 该参数指定服务器应启动由routineIdentifier指定的例程。	M	STR
0x02	**stopRoutine** 该参数指定服务器应停止由routineIdentifier指定的例程。	U	STPR
0x03	**requestRoutineResults** 该参数指定服务器应返回由routineIdentifier指定的例程的结果值。	U	存款准备金率
0x04 – 0x7F	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

13.2.2.3 请求消息数据参数定义

表380定义了请求消息的数据参数。

表380 – 请求消息数据参数定义

定义
routineIdentifier 该参数标识服务器本地例程，并且超出了定义的dataIdentifiers的范围（参见F.1）。
routineControlOptionRecord 该参数记录包含 — 例程输入选项参数，可选地指定例程的开始条件（例如timeToRun，startUpVariables等），或者 — 常规出口选项参数哪一个可选指定停止条件的该例程（例如timeToExpireBeforeRoutineStops，变量等）。

13.2.3 积极的回应消息

13.2.3.1 积极响应消息的定义

表381定义了肯定响应消息。

表381 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	RoutineControl响应SID	M	0x71	RCPR
#2	routineControlType	M	00-7F	RCTP_
#3 #4	routineIdentifier [] = [字节#1 (MSB) 字节 #2 (LSB)]	M M	0x00 – 0xFF 0x00 – 0xFF	RI_ B1 B2
#5	routineInfo	C_1	0x00 – 0xFF	RINF_
#6 : #n	routineStatusRecord[] = [routineStatus#1 : routineStatus# m]	U : U	0x00 – 0xFF : 0x00 – 0xFF	RSR_ RS_ : RS _

C_1RoutineInfo字节指定一个方案 (例如, StartRoutine, StopRoutine, RequestRoutineResults), 以允许通用外部测试设备处理任何例程。 即使routineStatusRecord的ISO / SAE定义大小等于“0”个数据字节, 该参数对于ISO / SAE规范 (例如ISO 27145-3, SAE J1979-DA, ISO 26021) 定义routineStatusRecord的任何例程。 对于routineStatusRecord完全由车辆制造商定义的例程, 此参数的支持是可选的。 该字节的定义应留给汽车制造商。

U如果车辆制造商为routineIdentifier (RID) 指定了RoutineStatusByte #m, 则只能包含在routineStatusRecord []中。

13.2.3.2 肯定回复消息数据参数定义

表382定义了肯定响应消息的数据参数。

表382 – 响应消息数据参数定义

定义
routineControlType 该参数是来自请求消息的子功能参数的比特6-0的回声。
routineIdentifier 该参数是来自请求消息的routineIdentifier的回显。
routineInfo RoutineInfo字节编码是车辆制造商特定的, 并且为车辆制造商提供了一种机制, 以便基于该返回值来支持通用外部测试设备处理所有实施的例程 (例如, 如果需要stopRoutine或requestRoutineResults)。
routineStatusRecord 该参数记录用于向客户提供: — 在例程开始之后关于服务器状态的附加信息 — 例程停止后的服务器状态的附加信息 (例如, 总运行时间, 停止前例程产生的结果等) 或 — 先前在服务器中停止的例程的结果 (退出状态信息)。

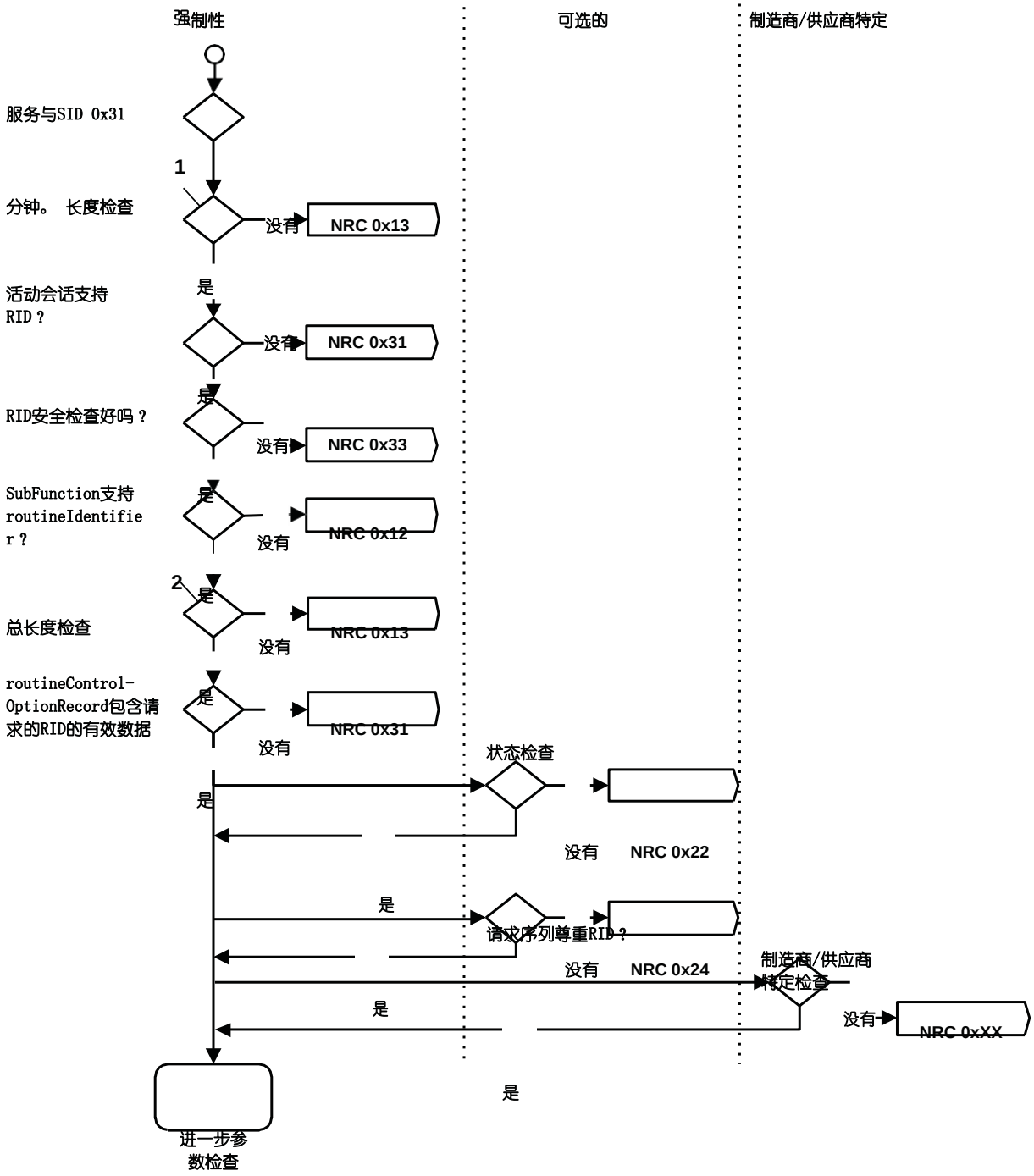
13.2.4 支持的否定响应代码 (NRC_)

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表383中。如果错误情况适用于服务器, 则应使用列出的否定响应。

表383 – 支持的否定响应代码

NRC	描述	助记符
0x12	**子functionNotSupported** 如果所请求的子功能通常不被支持，或者被请求的RoutineIdentifier不支持，则应发送该NRC。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果未满足请求RoutineControl的标准，则应返回此NRC。	CNC
0x24	**requestSequenceError** 这个NRC将被退回 • 该程序当前处于激活状态，并且在收到'startRoutine'子功能时不能重新启动（由汽车制造商决定是否可以在激活期间重新启动某个程序）， • 当收到'stopRoutine'子功能时，该程序当前不处于激活状态， • 当收到'requestRoutineResults'子函数时（例如，所请求的routineIdentifier从未启动过），常规结果不可用。	RSE
0x31	**requestOutOfRange** 如果出现下列情况，将返回NRC: — 服务器不支持请求的routineIdentifier， — 用户可选的routineControlOptionRecord包含请求的routineIdentifier的无效数据。	ROOR
0x33	**securityAccessDenied** 如果客户端使用有效的安全routineIdentifier发送请求并且服务器的安全功能当前处于活动状态，则应发送此NRC。	伤心
0x72	**GeneralProgrammingFailure** 如果服务器在执行访问服务器内部存储器的例程时检测到错误，则应返回此NRC。 例如，当例程擦除或编程永久存储器设备（例如闪存）中的某个存储器位置并且访问该存储器位置失败时。	GPF

评估顺序记录在图25中。



键

1 至少4个 (SI + SubFunction + RID参数)

2 1字节SI + 1字节SF + 2字节RID +第n个字节routineControlOptionRecord为特定RID所需

图25 - RoutineControl服务的NRC处理

13.2.5 消息流示例（s）RoutineControl

13.2.5.1 示例#1：子函数= startRoutine

本小节规定了测试条件，以便在服务器中启动一个例程，以便间歇性地（尽可能快地）测试所有输入和输出信号，同时技术人员将“摆动”被测系统的所有线束连接器。 routineIdentifier通过routineIdentifier 0x0201引用该例程。

测试条件：点火=开，发动机=关，车速= 0 [千phph]

通过将suppressPosRspMsgIndicationBit（子函数参数的第7位）设置为“FALSE”（'0'），客户端请求获得响应消息。

表384定义了RoutineControl请求消息流 - 示例#1。

表384 - RoutineControl请求消息流 - 示例#1

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RoutineControl请求SID	0x31	RC
#2	子函数= startRoutine, suppressPosRspMsgIndicationBit = FALSE	0x01	LEV_STR
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x01	RI_B2

表385定义了肯定响应消息流 - 示例#1。

表385 - 积极响应消息流 - 示例#1

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RoutineControl响应SID	0x71	RCPR
#2	routineControlType = startRoutine	0x01	STR
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x01	RI_B2
#5	routineStatusRecord [routineStatus#1] =车辆制造商特定的	0x32	RRS_

13.2.5.2 示例#2：子函数= stopRoutine

本条款规定了测试条件，用于停止服务器中的例程，该服务器在间歇期间连续测试（尽可能快）所有输入和输出信号，而技术人员将被“摆动”所测试系统的所有线束连接器。 routineIdentifier通过routineIdentifier 0x0201引用该例程。

测试条件：点火=开，发动机=关，车速= 0 [千phph]

通过将suppressPosRspMsgIndicationBit（子函数参数的第7位）设置为“FALSE”（'0'），客户端请求获得响应消息。

表386定义了RoutineControl请求消息流 - 示例#2。

表386 - RoutineControl请求消息流 - 示例#2

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RoutineControl请求SID	0x31	RC
#2	子函数= stopRoutine, suppressPosRspMsgIndicationBit = FALSE	0x02	SPR
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x01	RI_B2

表387定义了RoutineControl肯定响应消息流程 - 示例#2。

表387 - RoutineControl正响应消息流 - 示例#2

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	StopRoutine响应SID	0x71	RCPR
#2	routineControlType = stopRoutine	0x02	SPR
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x01	RI_B2
#5	routineStatusRecord [routineStatus#1] =车辆制造商特定的	0x30	RRS_

13.2.5.3 Example#3: sub-function = requestRoutineResults

此示例显示如何在例程完成后检索结果值。 例行程序不断测试（尽可能快）所有输入和输出信号的间歇性，同时技术人员在被测系统的所有线束接头处“摆动”。 引用此例程的routineIdentifier是0x0201。 测试条件：点火=开，发动机=关，车速= 0 [千phph]。

客户端请求通过将suppressPosRspMsgIndicationBit（子函数参数的第7位）设置为“FALSE”（'0'）来获得响应消息。

表388定义了RequestRoutineResults请求消息流 - 示例#3。

表388 - RequestRoutineResults请求消息流 - 示例#3

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RoutineControl请求SID	0x31	RC
#2	子函数= requestRoutineResults, suppressPosRspMsgIndicationBit = FALSE	0x03	存款准备金率
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x01	RI_B2

表389定义了RequestRoutineResults积极响应消息流 - 示例#3。

表389 - RequestRoutineResults正面响应消息流 - 示例#3

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RoutineControl响应SID	0x71	RCPR
#2	routineControlType = requestRoutineResults	0x03	存款准备金率
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x01	RI_B2
#5		0x30	RRS_
#6		0x33	RRS_
:		:	:
#n		0x8F	RRS_

13.2.5.4 Example#4: sub-function = startRoutine with routineControlOption

本条款规定了测试条件，以便在变速箱控制单元中启动例行程序，以便在特殊模式下校准特定档位的换档。 齿轮可以是#1至#20中的任何一种，并且该模式可以是长椅，独立式和车内式。 routineIdentifier通过routineIdentifier 0x0202引用该例程。

测试条件：点火=开，发动机=关，车速= 0 [千phph]。

通过将suppressPosRspMsgIndicationBit（子函数参数的第7位）设置为“FALSE”（'0'），客户端请求获得响应消息。

表390定义了RoutineControl请求消息流 - 示例#4。

表390 - RoutineControl请求消息流 - 示例#4

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RoutineControl请求SID	0x31	RC
#2	子函数= startRoutine, suppressPosRspMsgIndicationBit = FALSE	0x01	STR
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x02	RI_B2
#5	routineControlOption#1［选择的齿轮］=车辆制造商特定的	0x06	RCO_
#6	routineControlOption#2［测试条件］	0x01	RCO_

表391定义了RoutineControl肯定响应消息流程 - 示例#4。

表391 - RoutineControl肯定响应消息流 - 示例#4

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RoutineControl响应SID	0x71	RCPR
#2	routineControlType = startRoutine	0x01	STR
#3	routineIdentifier [byte#1] (MSB)	0x02	RI_B1
#4	routineIdentifier [byte#2] (LSB)	0x02	RI_B2
#5		0x32	RRS_
#6		0x33	RRS_
.		:	:
#n		0x8F	RRS_

14 上传下载功能单元

14.1 概观

表392定义了上传下载功能单元。

表392 – 上传下载功能单元

服务	描述
RequestDownload	客户端请求协商从客户端到服务器的数据传输。
RequestUpload	客户端请求协商从服务器到客户端的数据传输。
TransferData	客户端将数据传输到服务器（下载）或从服务器请求数据（上载）。
RequestTransferExit	客户端请求终止数据传输。
RequestFileTransfer	客户端请求协商服务器和客户端之间的文件传输。

14.2 请求下载（0x34）服务

14.2.1 服务说明

客户端使用requestDownload服务来启动从客户端到服务器的数据传输（下载）。

服务器收到requestDownload请求消息后，服务器应在发送肯定响应消息之前采取所有必要的操作来接收数据。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

14.2.2 请求消息

14.2.2.1 请求消息定义

表393定义了请求消息。

表393 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	请求下载请求SID	M	0x34	RD
#2	dataFormatIdentifier	M	0x00 – 0xFF	DFI_
#3	addressAndLengthFormatIdentifier	M	0x00 – 0xFF	ALFID
#4 : #(m-1)+4		M : C_1	0x00 – 0xFF : 0x00 – 0xFF	MA_ B1 : Bm
#N- (K-1) : #n		M : C_2	0x00 – 0xFF : 0x00 – 0xFF	MS_ B1 : Bk
C_1: The 存在 的 这个 参数 取决于 上 地址 长度 信息 参数 的 该 addressAndLengthFormatIdentifier				
C_2: The 存在 的 这个 参数 取决于 上 该 记忆 尺寸 长度 信息 的 该 addressAndLengthFormatIdentifier。				

14.2.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

14.2.2.3 请求消息数据参数定义

表394定义了请求消息的数据参数。

表394 – 请求消息数据参数定义

定义
dataFormatIdentifier 这个数据参数是一个字节值，每个半字节分开编码。 高半字节指定“compressionMethod”，低半字节指定“encryptingMethod”。 值0x00指定既不使用compressionMethod也不使用encryptingMethod。 0x00以外的值是车辆制造商特定的。
addressAndLengthFormatIdentifier 该参数是一个字节值，每个半字节分开编码（参见H.1的示例值）： — 位7 – 4: memorySize参数的长度（字节数） — 位3 – 0: memoryAddress参数的长度（字节数）
memoryAddress 参数memoryAddress是要写入数据的服务器内存的起始地址。 用于该地址的字节数由addressAndLengthFormatIdentifier的低半字节（bit 3-0）定义。 memoryAddress参数中的字节#m始终是服务器中引用地址的最低有效字节。 地址的最高有效字节可用作存储器标识符。 使用内存标识符的一个例子是具有16位寻址和内存地址重叠的双处理器服务器（当给定地址对任一处理器有效但是产生不同的物理内存设备或使用内部和外部闪存时）。 在这种情况下，可以将memoryAddress参数中另外未使用的字节指定为用于选择所需存储器设备的存储器标识符。 该功能的使用应该由车辆制造商/系统供应商定义。
memorySize 服务器应使用此参数来比较内存大小与传输数据服务期间传输的总数据量。 这增加了编程安全性。 用于此大小的字节数由addressAndLengthFormatIdentifier的高半字节（第7 – 4位）定义。 如果使用数据压缩，则无论内存大小是否表示压缩或未压缩大小，都是车辆制造商特定的。

14.2.3 积极的回应消息

14.2.3.1 积极响应消息的定义

表395定义了肯定响应消息。

表395 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	请求下载响应SID	M	0x74	RDPR
#2	lengthFormatIdentifier	M	0x00 – 0xF0	LFID
#3 : #n	maxNumberOfBlockLength = [字节#1 (MSB) : 字节#m]	M : M	0x00 – 0xFF : 0x00 – 0xFF	MNROB_ B1 : Bm

14.2.3.2 肯定回复消息数据参数定义

表396定义了肯定响应消息的数据参数。

表396 – 响应消息数据参数定义

定义
lengthFormatIdentifier 该参数是一个单字节值，每个半字节分别编码: — 位7 – 4: maxNumberOfBlockLength参数的长度（字节数）。 — 位3 – 0: 由文档保留，设置为'0'。 该参数的格式与请求消息中包含的addressAndLengthFormatIdentifier参数的格式兼容，但必须将下半字节设置为'0'。
maxNumberOfBlockLength requestDownload肯定响应消息使用此参数通知客户端有多少数据字节（maxNumberOfBlockLength）包含在来自客户端的每个TransferData请求消息中。 此长度反映了完整的消息长度，包括TransferData请求消息中存在的服务标识符和数据参数。 此参数允许客户端在开始将数据传输到服务器之前调整服务器的接收缓冲区大小。 要求服务器接受长度与报告的maxNumberOfBlockLength相等的transferData请求。 这是服务器特定的传输数据请求长度小于maxNumberOfBlockLength被接受（如果有的话）。 请注意，给定块中的最后一个transferData请求可能要求小于maxNumberOfBlockLength。 服务器不允许写入transferData消息中包含的其他数据字节（即填充字节）（压缩或未压缩格式），因为这会影响后续transferData请求数据的存储地址写出来。

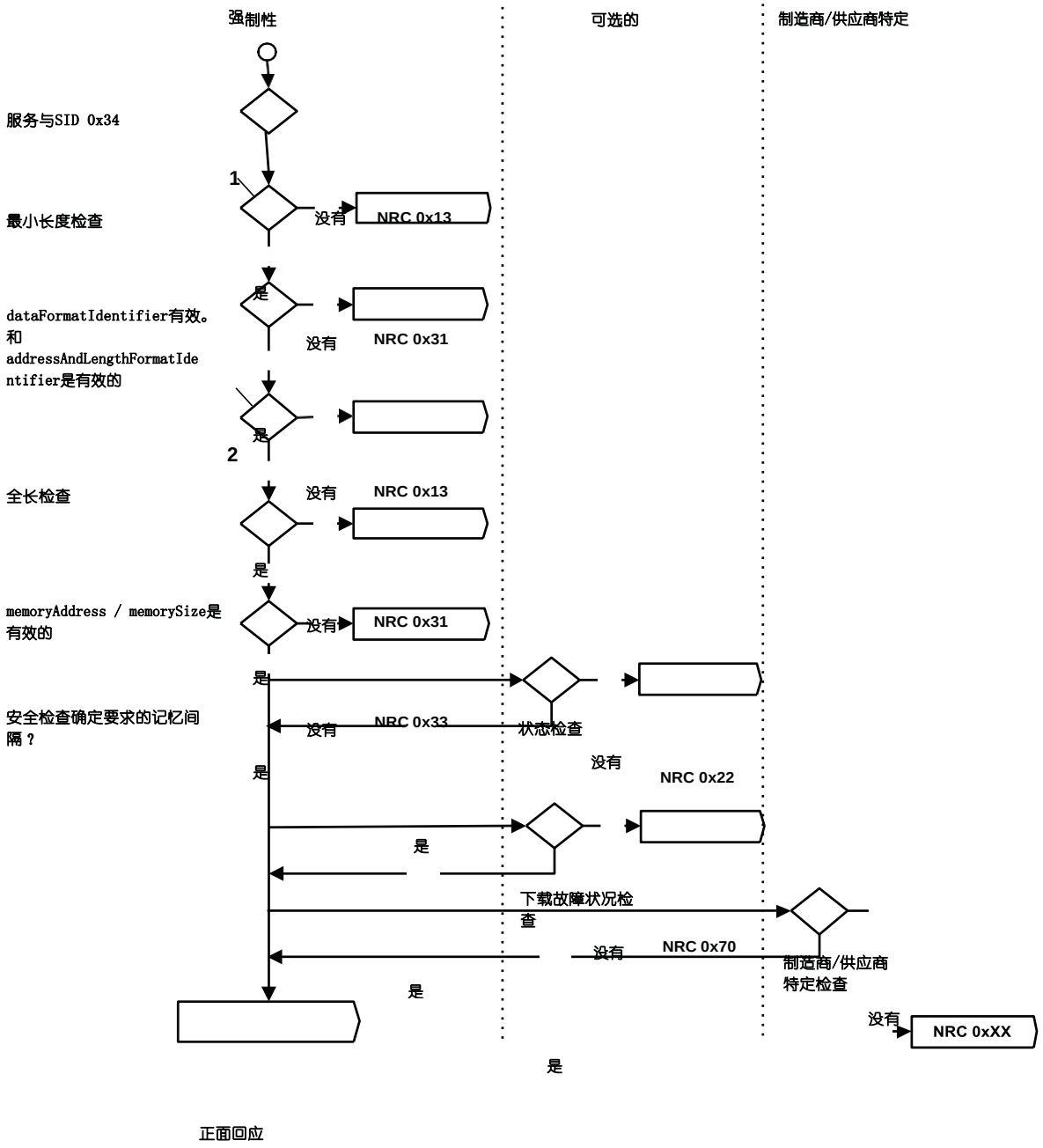
14.2.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表397中。如果错误情况适用于服务器，则应使用列出的否定响应。

表397 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果服务器在接收下载软件或校准模块的过程中收到对此服务的请求，则应返回此NRC。 如果在下载模块期间服务器和客户端之间的数据大小不匹配，则可能会发生这种情况。	CNC
0x31	**requestOutOfRange** 如果出现下列情况，将返回NRC: — 指定的dataFormatIdentifier无效。 — 指定的addressAndLengthFormatIdentifier无效。 — 指定的memoryAddress / memorySize无效。	ROOR
0x33	**securityAccessDenied** 如果服务器是安全的（对于支持SecurityAccess服务的服务器），则在收到对该服务的请求时应返回此NRC。	伤心
0x70	**uploadDownloadNotAccepted** 此NRC表示尝试下载到服务器的内存由于某些故障条件而无法完成。	UDNA

评估顺序记录在图26中。



键

1 至少5（SI + DFI_ + ALFID +最小\MA_ +最小\MS_）

2 长度可以通过addressAndLengthFormatIdentifier来计算

图26 - 请求下载服务的NRC处理

14.2.5 消息流示例请求下载

完整的消息流示例见14.5.5。

14.3 请求上传（0x35）服务

14.3.1 服务说明

客户端使用RequestUpload服务来启动从服务器到客户端的数据传输（上传）。

服务器收到requestUpload请求消息后，服务器应在发送肯定响应消息之前采取所有必要的动作发送数据。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

14.3.2 请求消息

14.3.2.1 请求消息定义

表398定义了请求消息。

表398 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	请求上传请求SID	M	0x35	RU
#2	dataFormatIdentifier	M	0x00 – 0xFF	DFI_
#3	addressAndLengthFormatIdentifier	M	0x00 – 0xFF	ALFID
#4 : #(m-1)+4		M : C_1	0x00 – 0xFF : 0x00 – 0xFF	MA_ B1 : Bm
#N- (K-1) : #n		M : C_2	0x00 – 0xFF : 0x00 – 0xFF	MS_ B1 : Bk
C_1: The 存在 的 这个 参数 取决于 上 地址 长度 信息 参数 的 该 addressAndLengthFormatIdentifier				
C_2: The 存在 的 这个 参数 取决于 上 该 记忆 尺寸 长度 信息 的 该 addressAndLengthFormatIdentifier。				

14.3.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

14.3.2.3 请求消息数据参数定义

表399定义了请求消息的数据参数。

表399 – 请求消息数据参数定义

定义
dataFormatIdentifier 这个数据参数是一个字节值，每个半字节分开编码。 高半字节指定“compressionMethod”，低半字节指定“encryptingMethod”。 值0x00指定既不使用compressionMethod也不使用encryptingMethod。 0x00以外的值是车辆制造商特定的。
addressAndLengthFormatIdentifier 该参数是一个字节值，每个半字节分开编码（参见H.1的示例值）： — 位7 – 4: memorySize参数的长度（字节数） — 位3 – 0: memoryAddress参数的长度（字节数）
memoryAddress 参数memoryAddress是要从中检索数据的服务器内存的起始地址。 用于该地址的字节数由addressAndLengthFormatIdentifier的低半字节（bit 3-0）定义。 memoryAddress参数中的字节#m始终是服务器中引用地址的最低有效字节。 地址的最高有效字节可用作存储器标识符。 使用内存标识符的一个例子是具有16位寻址和内存地址重叠的双处理器服务器（当给定地址对任一处理器有效但是产生不同的物理内存设备或使用内部和外部闪存时）。 在这种情况下，可以将memoryAddress参数中另外未使用的字节指定为用于选择所需存储器设备的存储器标识符。 该功能的使用应该由车辆制造商/系统供应商定义。
memorySize 服务器应使用此参数来比较内存大小与传输数据服务期间传输的总数据量。 这增加了编程安全性。 用于此大小的字节数由addressAndLengthFormatIdentifier的高半字节（第4位）定义。 如果使用数据压缩，则无论内存大小是否表示压缩或未压缩大小，都是车辆制造商特定的。

14.3.3 积极的回应消息

14.3.3.1 积极响应消息的定义

表400定义了肯定响应消息。

表400 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	请求上传响应SID	M	0x75	RUPR
#2	lengthFormatIdentifier	M	0x00 – 0xF0	LFID
#3 : #n	maxNumberOfBlockLength = [字节#1 (MSB) : 字节#m]	M : M	0x00 – 0xFF : 0x00 – 0xFF	MNROB_ B1 : Bm

14.3.3.2 肯定回复消息数据参数定义

表401定义了肯定响应消息的数据参数。

表格401-响应消息数据参数定义

定义
lengthFormatIdentifier 该参数是一个单字节值，每个半字节分别编码： — 位7 – 4：maxNumberOfBlockLength参数的长度（字节数）； — 位3-0：由文档保留，设置为0x0； 该参数的格式与请求消息中包含的addressAndLengthFormatIdentifier参数的格式兼容，但必须将低位半字节设置为0x0。
maxNumberOfBlockLength RequestUpload肯定响应消息使用此参数来通知客户端在来自服务器的每个TransferData肯定响应消息中应包含多少个数据字节。 该长度反映了完整的消息长度，包括服务标识符和TransferData位置响应消息中存在的数据参数。 此参数允许客户端在服务器开始将数据传输到客户端之前调整服务器的发送缓冲区大小。 客户端需要接受长度与报告的maxNumberOfBlockLength相等的transferData响应。 它是特定于服务器的传输数据响应长度小于maxNumberOfBlockLength（如果有）。 注：给定块中的最后一个transferData响应可能要求小于maxNumberOfBlockLength。

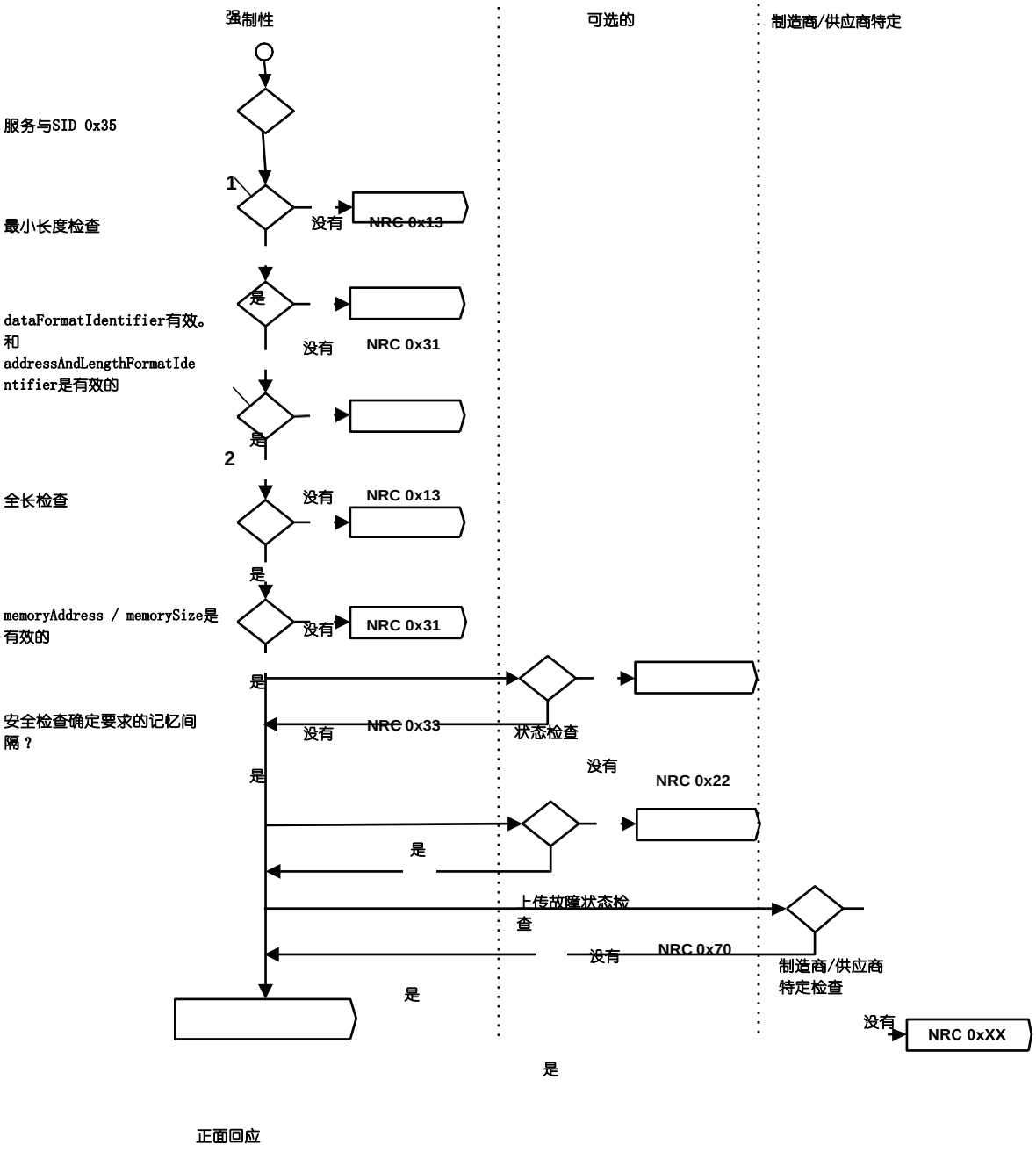
14.3.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 每个响应代码发生的情况记录在表402中。如果错误情况适用于服务器，则应使用列出的否定响应。

表402 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果requestUpload的标准不符合，则应返回此NRC。 如果服务器在requestUpload已处于活动状态但尚未完成时接收到对此服务的请求，则可能会发生这种情况。	CNC
0x31	**requestOutOfRange** 如果出现下列情况，将返回NRC： — 指定的dataFormatIdentifier无效； — 指定的addressAndLengthFormatIdentifier无效； — 指定的memoryAddress / memorySize无效；	ROOR
0x33	**securityAccessDenied** 如果服务器是安全的（对于支持SecurityAccess服务的服务器），则在收到对该服务的请求时应返回此NRC。	伤心
0x70	**uploadDownloadNotAccepted** 此NRC表示尝试上载到服务器的内存由于某些故障条件而无法完成。	UDNA

评估顺序记录在图27中。



键

1 至少5 (SI + DFI_ + ALFID +最小\MA_ +最小\MS_)

2 长度可以通过addressAndLengthFormatIdentifier来计算

图27 - RequestUpload服务的NRC处理

14.3.5 消息流示例RequestUpload

完整的消息流示例见14.5.5。

14.4 TransferData（0x36）服务

14.4.1 服务说明

TransferData服务用于客户端将数据从客户端传输到服务器（下载）或从服务器传输到客户端（上传）。

数据传输方向由前面的RequestDownload或RequestUpload服务定义。 如果客户端发起请求下载，则要下载的数据包含在TransferData请求消息中的参数transferRequestParameter中。 如果客户端发起请求上传，则要上传的数据包含在TransferData响应消息中的参数transferResponseParameter中。

TransferData服务请求包含一个blockSequenceCounter，以便在多个TransferData请求序列期间传输数据服务失败的情况下，改进错误处理。 当接收到RequestDownload（0x34）或RequestUpload（0x35）请求消息时，服务器的blockSequenceCounter应初始化为1。 这意味着RequestDownload（0x34）或RequestUpload（0x35）请求消息后面的第一个TransferData（0x36）请求消息以blockSequenceCounter开头。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

14.4.2 请求消息

14.4.2.1 请求消息定义

表格403定义了请求消息。

表403 - 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	TransferData请求SID	M	0x36	TD
#2	blockSequenceCounter	M	0x00 - 0xFF	BSC
#3 ： #n	transferRequestParameterRecord[] = [transferRequestParameter#1 ： transferRequestParameter#m]	C ： U	0x00 - 0xFF ： 0x00 - 0xFF	TRPR_ TRTP_ ： TRTP_
C =有条件的：如果正在下载，则此参数是强制性的。				

14.4.2.2 请求消息子函数参数$ Level（LEV_）定义

此服务不使用子功能参数。

14.4.2.3 请求消息数据参数定义

表404定义了请求消息的数据参数。

表404 – 请求消息数据参数定义

定义
blockSequenceCounter blockSequenceCounter参数值从0x01开始，第一个TransferData请求位于RequestDownload（0x34）或RequestUpload（0x35）服务之后。 对于每个后续TransferData请求，它的值都会加1。 在0xFF的值处，blockSequenceCounter翻转并在下一个TransferData请求消息的0x00处开始。 示例用例: — 如果TransferData请求下载数据在服务器中正确接收和处理，但肯定响应消息未到达客户端，则客户端将确定应用层超时并重复相同的请求（包括相同的blockSequenceCounter）。 服务器将收到重复的TransferData请求，并可以根据包含的BlockSequenceCounter确定该TransferData请求重复。 服务器会立即发送肯定响应消息，而不会再次将数据写入其内存。 — 如果在服务器中没有正确接收到下载数据的TransferData请求，则服务器不会发送肯定的响应消息。 客户端将确定应用程序层超时并将重复相同的请求（包括相同的blockSequenceCounter）。 服务器将收到重复的TransferData请求，并可以根据包含的BlockSequenceCounter确定这是一个新的TransferData。 服务器将处理服务并发送肯定响应消息。 — 如果TransferData请求上传数据在服务器中正确接收并处理，但肯定响应消息未到达客户端，则客户端将确定应用层超时并重复相同的请求（包括相同的blockSequenceCounter）。 服务器将收到重复的TransferData请求，并可以根据包含的BlockSequenceCounter确定该TransferData请求重复。 服务器会立即发送肯定响应消息，并立即访问之前提供的数据到其存储器中。 — 如果在服务器中未正确接收到传输数据上传请求，则服务器不会发送肯定的响应消息。 客户端将确定应用程序层超时并将重复相同的请求（包括相同的blockSequenceCounter）。 服务器将收到重复的TransferData请求，并可以根据包含的BlockSequenceCounter确定这是一个新的TransferData。 服务器将处理服务并发送肯定响应消息。
transferRequestParameterRecord 该参数记录包含服务器为支持数据传输所需的参数。 该参数的格式和长度是车辆制造商特定的。 例 对于下载，transferRequestParameterRecord包含要传输的数据。

14.4.3 积极的回应消息

14.4.3.1 积极响应消息的定义

表405定义了肯定响应消息。

表405 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	TransferData响应SID	M	0x76	TDPR
#2	blockSequenceCounter	M	0x00 – 0xFF	BSC
#3 : #n	transferResponseParameterRecord[] = [transferResponseParameter#1 : transferResponseParameter#m]	C : U	0x00 – 0xFF : 0x00 – 0xFF	TREPR_ TREP_ : TREP
C =有条件：如果上传正在进行，则此参数是强制性的。				

14.4.3.2 肯定回复消息数据参数定义

表406定义了肯定响应消息的数据参数。

表406 – 响应消息数据参数定义

定义
blockSequenceCounter 该参数是来自请求消息的blockSequenceCounter参数的回显。
transferResponseParameterRecord 该参数应包含客户端支持传输数据所需的参数。 该参数的格式和长度是车辆制造商特定的。 示例：对于下载，参数transferResponseParameterRecord可以包含服务器计算的校验和。 对于上传，参数transferResponseParameterRecord包含上传的数据。 对于下载，参数transferResponseParameterRecord不应重复transferRequestParameterRecord。

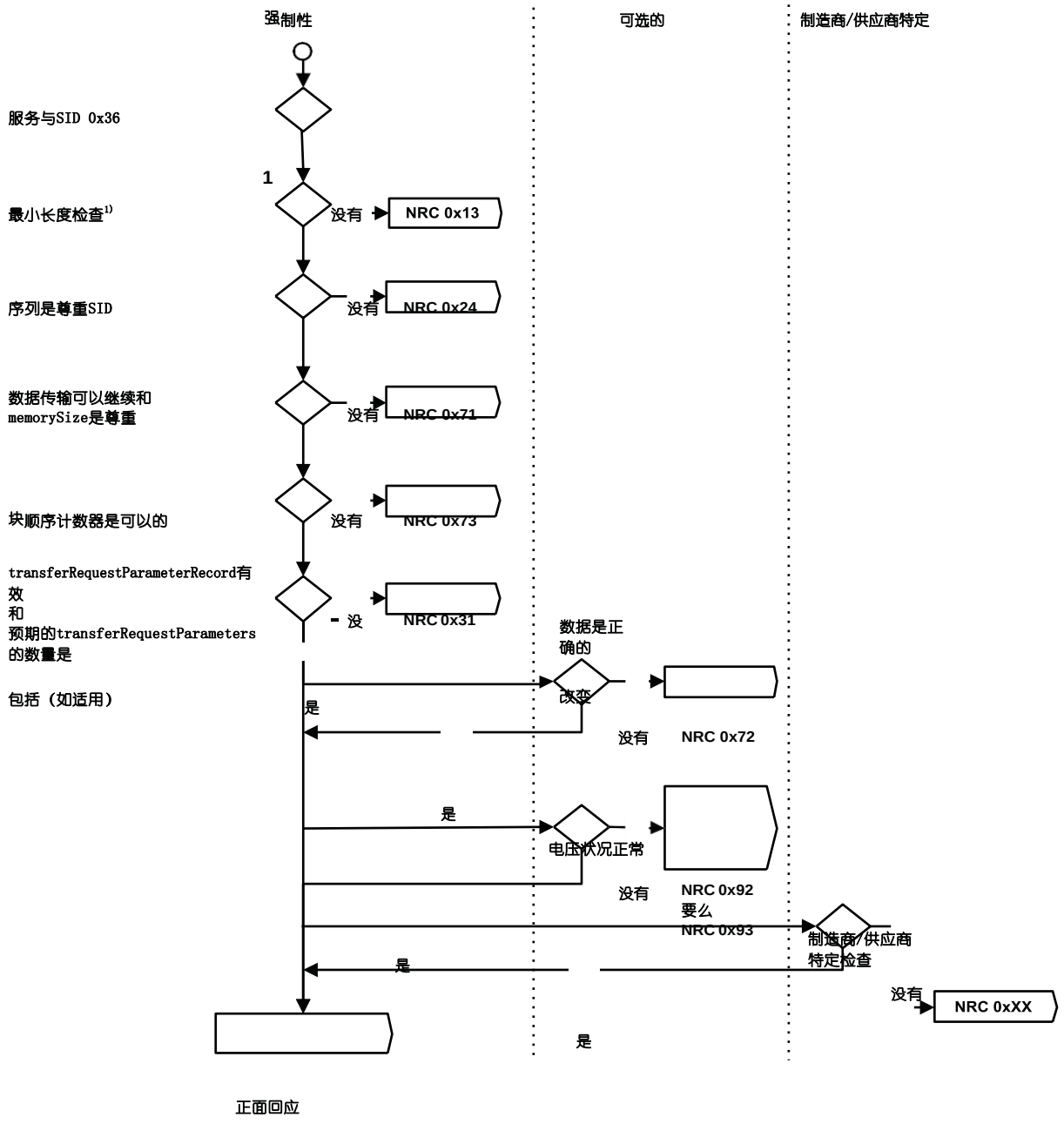
14.4.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 表407中记录了每个响应代码发生的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表407 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC（例如，消息长度不符合对requestDownload服务的肯定响应中返回的maxNumberOfBlockLength参数的要求）。	IMLOIF
0x24	**requestSequenceError** 服务器应该使用这个响应码： — 如果RequestDownload或RequestUpload服务在收到对该服务的请求时未处于活动状态； — 如果RequestDownload或RequestUpload服务处于活动状态，但服务器已经接收到由活动RequestDownlod或RequestUpload服务中的memorySize参数确定的所有数据； 注：重复带有blockSequenceCounter的TransferData请求消息必须等于前一个TransferData请求消息中包含的请求消息，服务器才能接受。	RSE
0x31	**requestOutOfRange** 如果出现下列情况，将返回NRC： — transferRequestParameterRecord包含附加的控制参数（例如附加地址信息），并且此控制信息无效。 — transferRequestParameterRecord与requestDownload或requestUpload服务参数maxNumberOfBlockLength不一致。 — transferRequestParameterRecord与服务器的内存对齐约束不一致。	ROOR
0x71	**transferDataSuspended** 如果下载模块长度不符合requestDownload服务的请求消息中发送的memorySize参数的要求，则应返回此NRC。	TDS
0x72	**generalProgrammingFailure** 如果服务器在下载数据期间擦除或编程永久存储器设备（例如闪存）中的存储器位置时检测到错误，则应返回该NRC。	GPF
0x73	**wrongBlockSequenceCounter** 如果服务器在blockSequenceCounter序列中检测到错误，则应该返回该NRC。 注：重复带有blockSequenceCounter的TransferData请求消息必须等于前一个TransferData请求消息中包含的请求消息，服务器才能接受。	WBSC
0x92 / 0x93	**voltageTooHigh / voltageTooLow** 如果在服务器的主电源引脚测量的电压超出了将数据下载到服务器的永久存储器（例如闪存）的可接受范围内，则应根据情况发送该返回代码。	VTH / VTL

评估顺序记录在图28中。



键

1　　如果请求上传正在进行（SI + BSC），则必须为2，
　　如果请求下载正在进行，则至少为3（SI + BSC +最小\TRPR_）

图28 - TransferData服务的NRC处理

14.4.5　消息流示例（s）TransferData

完整的消息流示例见14.5.5。

14.5 RequestTransferExit (0x37) 服务

14.5.1 服务说明

客户端使用此服务来终止客户端和服务器之间的数据传输（上传或下载）。

重要 - 服务器和客户端应符合7.5中规定的请求和响应消息行为。

14.5.2 请求消息

14.5.2.1 请求消息定义

表408定义了请求消息。

表408 - 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	RequestTransferExit请求SID	M	0x37	RTE
#2 : #n	transferRequestParameterRecord[] = [transferRequestParameter#1 : transferRequestParameter#m]	U : U	0x00 - 0xFF : 0x00 - 0xFF	TRPR_ TRTP_ : TRTP_

14.5.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

14.5.2.3 请求消息数据参数定义

表409定义了请求消息的数据参数。

表409-请求消息数据参数定义

定义
transferRequestParameterRecord 此参数记录包含服务器为支持数据传输所需的参数。 该参数的格式和长度是车辆制造商特定的。

14.5.3 积极的回应消息

14.5.3.1 积极响应消息的定义

表410定义了肯定响应消息。

表410 - 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	RequestTransferExit响应SID	M	0x77	RTEPR
#2 : #n	transferResponseParameterRecord[] = [transferResponseParameter#1 : transferResponseParameter#m]	U : U	0x00 - 0xFF : 0x00 - 0xFF	TREPR_ TREP_ : TREP

14.5.3.2 肯定回复消息数据参数定义

表格411定义了肯定响应消息的数据参数。

表411-响应消息数据参数定义

定义
transferResponseParameterRecord 该参数应包含客户端支持传输数据所需的参数。 该参数的格式和长度是车辆制造商特定的。

14.5.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 表412列出了每种负面响应代码出现的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表412 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应该返回NRC。	IMLOIF
0x24	**requestSequenceError** 如果出现下列情况，将返回NRC： — 收到该服务的请求时编程过程未完成； — RequestDownload或RequestUpload服务不活动；	RSE
0x31	**requestOutOfRange** 如果transferRequestParameterRecord包含无效数据，则应返回此NRC。	ROOR
0x72	**generalProgrammingFailure** 如果服务器在完成客户端和服务器之间的数据传输时（例如，通过完整性检查）检测到错误，则应返回此NRC。	GPF

评估顺序记录在图29中。

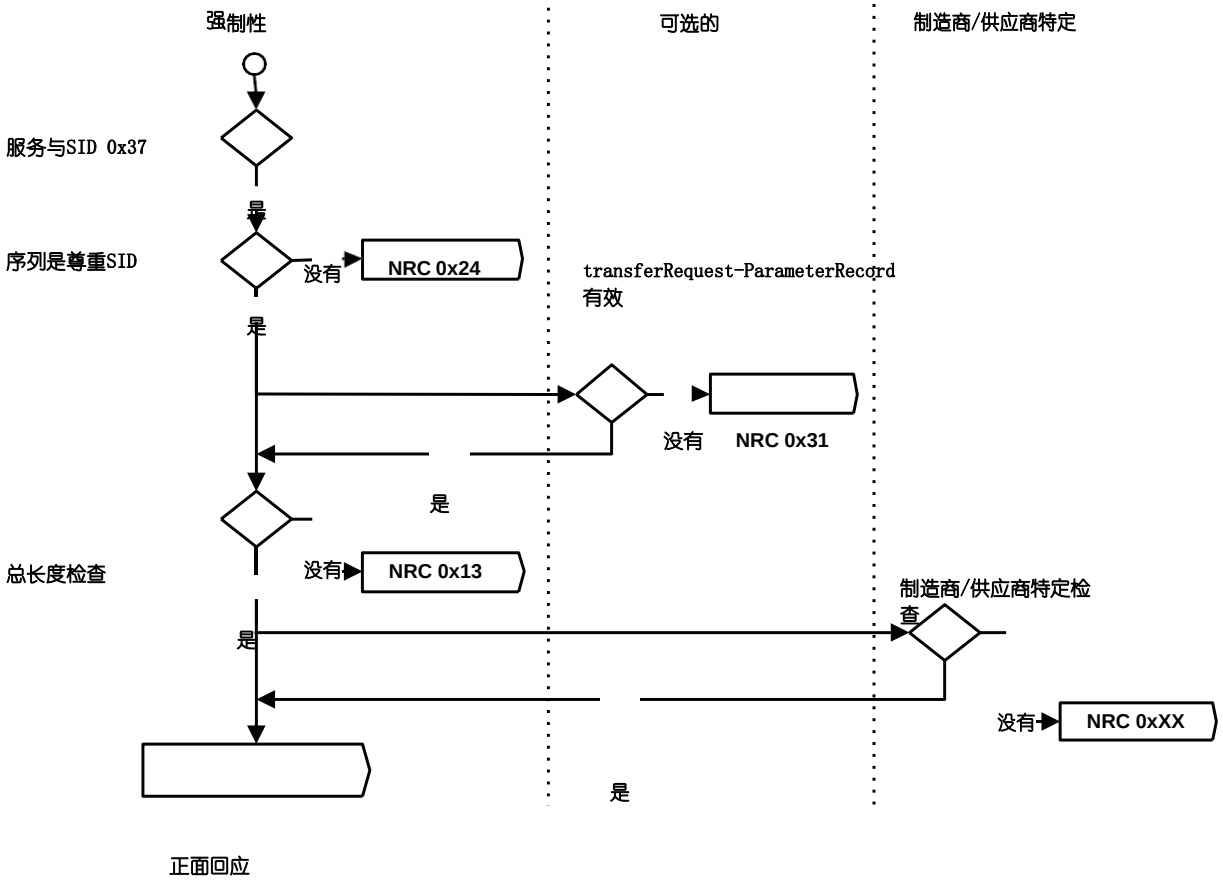


图29 – RequestTransferExit服务的NRC处理

14.5.5 用于下载/上传数据的消息流示例

14.5.5.1 将数据下载到服务器

14.5.5.1.1 假设

本条款规定了从客户端向服务器传输数据（下载）的条件。 该示例由三个步骤组成。

在1ST步骤中，客户端和服务器执行RequestDownload服务。 通过此服务，以下信息作为客户端与服务器之间的请求和肯定响应消息中的参数进行交换。

表413定义了transferRequestParameter值。

表413 – transferRequestParameter值的定义

数据参数名称	数据参数值 (s)	数据参数说明
memoryAddress (3字节)	0x602000	memoryAddress (开始) 将数据下载到
dataFormatIdentifier	0x11	dataFormatIdentifier: — compressionMethod = 0x1X — encryptingMethod = 0xX1

表413 – （续）

数据参数名称	数据参数值 (s)	数据参数说明
MemorySize (3字节)	0x00FFFF	MemorySize = (65 535字节) 服务器将使用此参数值与执行requestTransferExit服务期间传输的实际字节数进行比较。

表414定义了transferResponseParameter值。

表414 – transferResponseParameter值的定义

数据参数名称	数据参数值 (s)	数据参数说明
maximumNumberOfBlockLength	0x0081	maximumNumberOfBlockLength: (serviceId + BlockSequenceCounter (1字节) + 127个服务器数据字节= 129个数据字节)

在2ND步骤中，客户机将65 535字节的数据传输到闪存，从内存地址0x602000开始到服务器。

在3RD步骤中，客户端通过requestTransferExit服务终止向服务器的数据传输。 测试条件：点火=开，

发动机=关，车速= 0 [千phph]

假设，在这个例子中，服务器支持三字节的memoryAddress和三字节的MemorySize。 如果MemorySize包含未压缩的大小，则无法计算具有127个数据字节的TransferData服务的数量，因为压缩方法及其压缩比率未标准化。如果MemorySize包含压缩大小，则具有127个数据字节的TransferData服务的总数将为516，随后是具有三个字节的单个TransferData请求。 因此，假定最后一个TransferData请求消息包含一个等于0x05的blockSequenceCounter。

14.5.5.1.2 步骤#1：请求下载

表415定义了请求下载请求消息流程示例。

表415-请求下载请求消息流程示例

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	请求下载请求SID	0x34	RD
#2	dataFormatIdentifier	0x11	DFI
#3	addressAndLengthFormatIdentifier	0x33	ALFID
#4	memoryAddress [byte#1] (MSB)	0x60	MA_B1
#5	memoryAddress [byte#2]	0x20	MA_B2
#6	memoryAddress [byte#3] (LSB)	0x00	MA_B3
#7	MemorySize [byte#1] (MSB)	0x00	UCMS_B1
#8	MemorySize [字节#2]	为0xFF	UCMS_B2
#9	MemorySize [byte#3] (LSB)	为0xFF	UCMS_B3

表416 – 请求下载肯定响应消息流程示例

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	请求下载响应SID	0x74	RDPR
#2	LengthFormatIdentifier	0x20	LFID
#3	maxNumberOfBlockLength [byte#1] (MSB)	0x00	MNROB_B1
#4	maxNumberOfBlockLength [byte#2] (LSB)	0x81	MNROB_B1

14.5.5.1.3　步骤#2：传输数据

表417 – TransferData请求消息流示例

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	TransferData请求SID	0x36	TD
#2	blockSequenceCounter	0x01	BSC
#3 : #129	transferRequestParameterRecord [transferRequestParameter#1] = dataByte#3 : transferRequestParameterRecord [transferRequestParameter#127] = dataByte#129	0xXX : 0xXX	TRTP_1 : TRTP_127

表418- TransferData肯定响应消息流程示例

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	TransferData响应SID	0x76	TDPR
#2	blockSequenceCounter	0x01	BSC

⋮

表419 - TransferData请求消息流示例

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	TransferData请求SID	0x36	TD
#2	blockSequenceCounter	0x05	BSC
#3 ： #n+2	transferRequestParameterRecord [transferRequestParameter#1] = dataByte#3 ： transferRequestParameterRecord [transferRequestParameter#n-2] = dataByte#n	0xXX ： 0xXX	TRTP_1 ： TRTP_n-2

表420 - TransferData肯定响应消息流示例

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	TransferData响应SID	0x76	TDPR
#2	blockSequenceCounter	0x05	BSC

14.5.5.1.4　步骤#3：请求转移出口

表421 - RequestTransferExit请求消息流示例

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RequestTransferExit请求SID	0x37	RTE

表422 – RequestTransferExit肯定响应消息流程示例

消息方向		服务器 客户端	
消息类型		响应	
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RequestTransferExit响应SID	0x77	RTEPR

14.5.5.2 从服务器上传数据

本条款规定了从服务器向客户端传输数据（上传）的条件。 该示例由三个步骤组成。

在1ST步骤中，客户端和服务器执行requestUpload服务。 使用此服务，以下信息作为客户端与服务器之间的请求和肯定响应消息中的参数进行交换:

表423 – transferRequestParameter值的定义

数据参数名称	数据值（s）	数据参数说明
memoryAddress（3字节）	0x201000	memoryAddress（开始）从上传数据
dataFormatIdentifier	0x11	dataFormatIdentifier — compressionMethod = 0x1X — encryptingMethod = 0xX1
MemorySize（3字节）	0x0001FF	MemorySize =（511字节） 该参数值应表明应传送多少个数据字节，并由服务器用来比较执行requestTransferExit服务期间传送的实际字节数。

表424 – transferResponseParameter值的定义

数据参数名称	数据值（s）	数据参数说明
maximumNumberOfBlockLength	0x0081	maximumNumberOfBlockLength: （serviceId + BlockSequenceCounter（1个字节）+ 127个服务器数据字节 = 129个数据字节）

在2ND步骤中，服务器传输511个数据字节（4个transferData服务，包含129个服务器数据字节+ 1个ServiceId数据字节+ 1个blockSequenceCounter字节）和1个transferData服务（5个服务器数据字节+ 1个serviceId数据字节+ 1块序列计数器字节）来自外部RAM的数据字节，从服务器中的内存地址0x201000开始。

在3RD步骤中，客户端通过requestTransferExit服务终止与服务器的数据传输。 测试条件: 点火=开，

发动机=关，车速= 0［千phph］

假设，在这个例子中，服务器支持三字节的memoryAddress和三字节的MemorySize。 此外，假设服务器支持TransferData（0x36）服务中的blockSequenceCounter。

14.5.5.2.1 步骤#1：请求上传

表425 – 请求上传请求消息流示例

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	请求上传请求SID	0x35	RU
#2	dataFormatIdentifier	0x11	DFI
#3	addressAndLengthFormatIdentifier	0x33	ALFID
#4	memoryAddress [byte#1] (MSB)	0x20	MA_B1
#5	memoryAddress [byte#2]	0x10	MA_B2
#6	memoryAddress [byte#3] (LSB)	0x00	MA_B3
#7	MemorySize [byte#1] (MSB)	0x00	UCMS_B1
#8	MemorySize [字节#2]	0x01	UCMS_B2
#9	MemorySize [byte#3] (LSB)	为0xFF	UCMS_B3

表426-请求上传肯定响应消息流程示例

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	请求上传响应SID	0x75	RUPR
#2	lengthFormatIdentifier	0x20	LFID
#3	maxNumberOfBlockLength [byte#1] (MSB)	0x00	MNROB_B1
#4	maxNumberOfBlockLength [byte#2] (LSB)	0x81	MNROB_B1

14.5.5.2.2 步骤#2：传输数据

表427 – TransferData请求消息流示例

消息方向	客户端 服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	TransferData请求SID	0x36	TD
#2	blockSequenceCounter	0x01	BSC

表428 – TransferData肯定响应消息流示例

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	TransferData响应SID	0x76	TDPR
#2	blockSequenceCounter	0x01	BSC
#3 : #129		xx : xx	TREP_1 : TREP_127

⋮

表429 – TransferData请求消息流示例

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	TransferData请求SID	0x36	TD
#2	blockSequenceCounter	0x05	BSC

表430– TransferData肯定响应消息流程示例

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	TransferData响应SID	0x76	TDPR
#2	blockSequenceCounter	0x05	BSC
#3 : #5	transferResponseParameterRecord [transferResponseParameter#1] = dataByte3 : transferResponseParameterRecord [transferResponseParameter#3] = dataByte5	0xXX : 0xXX	TREP_1 : TREP_3

14.5.5.2.3　步骤#3：请求转移出口

表431 – RequestTransferExit请求消息流示例

消息方向	客户端　服务器		
消息类型	请求		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RequestTransferExit请求SID	0x37	RTE

表432 - RequestTransferExit肯定响应消息流示例

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RequestTransferExit响应SID	0x77	RTEPR

14.6 RequestFileTransfer (0x38) 服务

14.6.1 服务说明

客户端使用requestFileTransfer服务来启动从客户端到服务器或从服务器到客户端的文件数据传输（下载或上传）。 此外，该服务还具有检索有关文件系统信息的功能。

如果服务器实现数据存储的文件系统，则此服务旨在作为支持数据上载和下载功能的RequestDownload和RequestUpload服务的替代解决方案。 在配置文件系统的下载或上载过程时，应使用RequestFileTransfer服务替换RequestDownload或RequestUpload。 数据传输的实际数据传输和终止是通过使用RequestData或RequestUpload服务所使用的TransferData和RequestTransferExit来实现的。 该服务还包括删除服务器文件系统上的文件或目录的功能。 对于此用例，TransferData和RequestTransferExit服务不适用。

服务器接收到RequestFileTransfer请求消息后，服务器应在发送肯定响应消息之前采取所有必要的操作来接收或发送数据。

重要 – 服务器和客户端应符合7.5中规定的请求和响应消息行为。

14.6.2 请求消息

14.6.2.1 请求消息定义

表433定义了请求消息。

表433 – 请求消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	RequestFileTransfer请求SID	M	0x38	RFT
#2	操作模式	M	0x01 – 0x05	MOOP
#3 #4	filePathAndNameLength [字节#1 (MSB) 字节#2] (LSB)	M M	0x00 – 0xFF 0x00 – 0xFF	FPL_B1 FPL_B2
#5 : #5+n-1	filePathAndName = [字节#1 (MSB) : 字节#n]	M : C_1	0x00 – 0xFF : 0x00 – 0xFF	FP_B1 : FP_Bn
#5+n	dataFormatIdentifier	C_2	0x00 – 0xFF	DFI_
#5+n+1	fileSizeParameterLength	C_2	0x00 – 0xFF	FSL
#5+n+2 : #5 + N + 2 + k-1个	fileSizeUnCompressed= [字节#1 (MSB) : 字节#k]	C_2 : $C_{2,3}$	0x00 – 0xFF : 0x00 – 0xFF	FSUC_B1 : FSUC_Bk
#5 + N + 2 + K : #5 + N + 1 + 2k个	fileSizeCompressed= [字节#1 (MSB) : 字节#k]	C_2 : $C_{2,3}$	0x00 – 0xFF : 0x00 – 0xFF	FSC_B1 : FSC_Bk

C_1: 此消息参数的长度（字节数）由filePathAndNameLength参数定义。 C_2: 这些参数的存在取决于modeOfOperation参数。

C_3: 此消息参数的长度（字节数）由fileSizeParameterLength定义。

14.6.2.2 请求消息子函数参数$ Level (LEV_) 定义

此服务不使用子功能参数。

14.6.2.3 请求消息数据参数定义

表434定义了请求消息的数据参数。

表434-请求消息数据参数定义

定义
操作模式 此数据参数定义要应用于filePathAndName参数中指定的文件或目录的操作类型。 数据参数的值在附件G中定义。
filePathAndNameLength 定义参数filePath的字节长度。
filePathAndName 根据参数modeOfOperation参数定义应添加，删除，替换或读取文件的服务器的文件系统位置。 此外，此参数还包括应作为文件路径的一部分添加，删除，替换或读取的文件的文件名。 如果modeOfOperation参数等于0x05 (ReadDir)，则此参数指示要读取的目录。 该参数的每个字节应以ASCII格式编码。
dataFormatIdentifier 这个数据参数是一个字节值，每个半字节分开编码。 高半字节指定“compressionMethod”，低半字节指定“encryptingMethod”。值0x00指定既不使用compressionMethod也不使用encryptingMethod。0x00以外的值是车辆制造商特定的。 如果modeOfOperation参数等于0x02 (DeleteFile) 和0x05 (ReadDir)，则该参数不应包括在请求消息中。
fileSizeParameterLength 定义两个参数fileSizeUncompressed和fileSizeCompressed的长度 (以字节为单位)。 如果modeOfOperation参数等于0x02 (DeleteFile)，0x04 (ReadFile) 或0x05 (ReadDir)，则该参数不应包含在请求消息中。
fileSizeUncompressed 以字节为单位定义未压缩文件的大小。 如果modeOfOperation参数等于0x02 (DeleteFile)，0x04 (ReadFile) 或0x05 (ReadDir)，则该参数不应包括在请求消息中。
fileSizeCompressed 以字节为单位定义压缩文件的大小。 如果传输未压缩的文件，则应将此参数的所有字节设置为参数fileSizeUncompressed中使用的大小信息。 如果modeOfOperation参数等于0x02 (DeleteFile)，0x04 (ReadFile) 或0x05 (ReadDir)，则该参数不应包含在请求消息中。

14.6.3 积极的回应消息

14.6.3.1 积极响应消息的定义

表435定义了肯定响应消息。

表435 – 肯定响应消息定义

A_Data字节	参数名称	Cvt	字节值	助记符
#1	RequestFileTransfer响应SID	S	0x78	RRFT
#2	操作模式	M	0x01 – 0x05	MOOP
#3	lengthFormatIdentifier	C_1	0x00 – 0xFF	LFID
#4 : #4+(m-1)		$C_{1,2}$: $C_{1,2}$	0x00 – 0xFF : 0x00 – 0xFF	MNROB_ B1 : Bm
#4+m	dataFormatIdentifier	C_1	0x00 – 0xFF	DFI_
#4+m+1 #4+m+2		C_1 C_1	0x00 – 0xFF 0x00 – 0xFF	FSDIL_B1 FSDIL_B2
#4+m+3 : #4 + M + 3 + k-1个		$C_{1,3}$: $C_{1,3}$	0x00 – 0xFF : 0x00 – 0xFF	FSUDIL_B1 : FSUDIL_Bk
#4 + M + 3 + K : #4 + M + 3 + 2K-1		$C_{1,3}$: $C_{1,3}$	0x00 – 0xFF : 0x00 – 0xFF	FSC_B1 : FSC_Bk

C_1: 这些参数的存在取决于modeOfOperation参数。

C_2: 此消息参数的长度（字节数）由fileSizeOrDirInfoParameterLength参数定义

C_3: 此消息参数的长度（字节数）由lengthFormatIdentifier参数定义

14.6.3.2 肯定回复消息数据参数定义

表436定义了肯定响应消息的数据参数。

表436 – 响应消息数据参数定义

定义
操作模式 这是参数回应请求的值。
lengthFormatIdentifier 定义maxNumberOfBlockLength参数的长度（字节数）。 如果modeOfOperation参数等于0x02（DeleteFile），则该参数不应包括在响应消息中。

表436 – （续）

定义
maxNumberOfBlockLength requestFileTransfer肯定响应消息使用此参数通知客户端有多少数据字节（maxNumberOfBlockLength）包含在来自客户端的每个TransferData请求消息中，或者在上传数据时服务器将包含在TransferData肯定响应中的数据字节数。 该长度反映了完整的消息长度，包括TransferData请求消息或肯定响应消息中存在的服务标识符和数据参数。 此参数允许客户端在开始将数据传输到服务器之前调整服务器的接收缓冲区大小，或者指示在数据上传时每个TransferData肯定响应中将包含多少个数据字节。 要求服务器接受长度与报告的maxNumberOfBlockLength相等的transferData请求。 这是服务器特定的传输数据请求长度小于maxNumberOfBlockLength被接受（如果有的话）。 注：给定块中的最后一个transferData请求可能要求小于maxNumberOfBlockLength。 服务器不允许写入transferData消息中包含的其他数据字节（即填充字节）（压缩或未压缩格式），因为这会影响后续transferData请求数据的存储地址写出来。 如果modeOfOperation参数等于0x02（DeleteFile），则该参数不应包括在响应消息中。
dataFormatIdentifier 这是参数回应请求的值。 如果modeOfOperation参数等于0x02（DeleteFile），则该参数不应包括在响应消息中。） 如果modeOfOperation参数等于0x05（ReadDir），则此参数的值应等于0x00。
fileSizeOrDirInfoParameterLength 定义两个参数fileSizeUncompressedOrDirInfoLength和fileSizeCompressed的长度（以字节为单位）。 如果modeOfOperation参数等于0x01（AddFile），0x02（DeleteFile）或0x03（ReplaceFile），则该参数不应包括在响应消息中。
fileSizeUncompressedOrDirInfoLength 定义要上传的未压缩文件的大小或要读取的目录信息的长度（以字节为单位）。 如果modeOfOperation参数等于0x01（AddFile），0x02（DeleteFile）或0x03（ReplaceFile），则该参数不应包括在响应消息中。
fileSizeCompressed 以字节为单位定义压缩文件的大小。 如果modeOfOperation参数等于0x01（AddFile），0x02（DeleteFile，0x03（ReplaceFile））或0x05（ReadDir），则该参数不应包括在响应消息中。

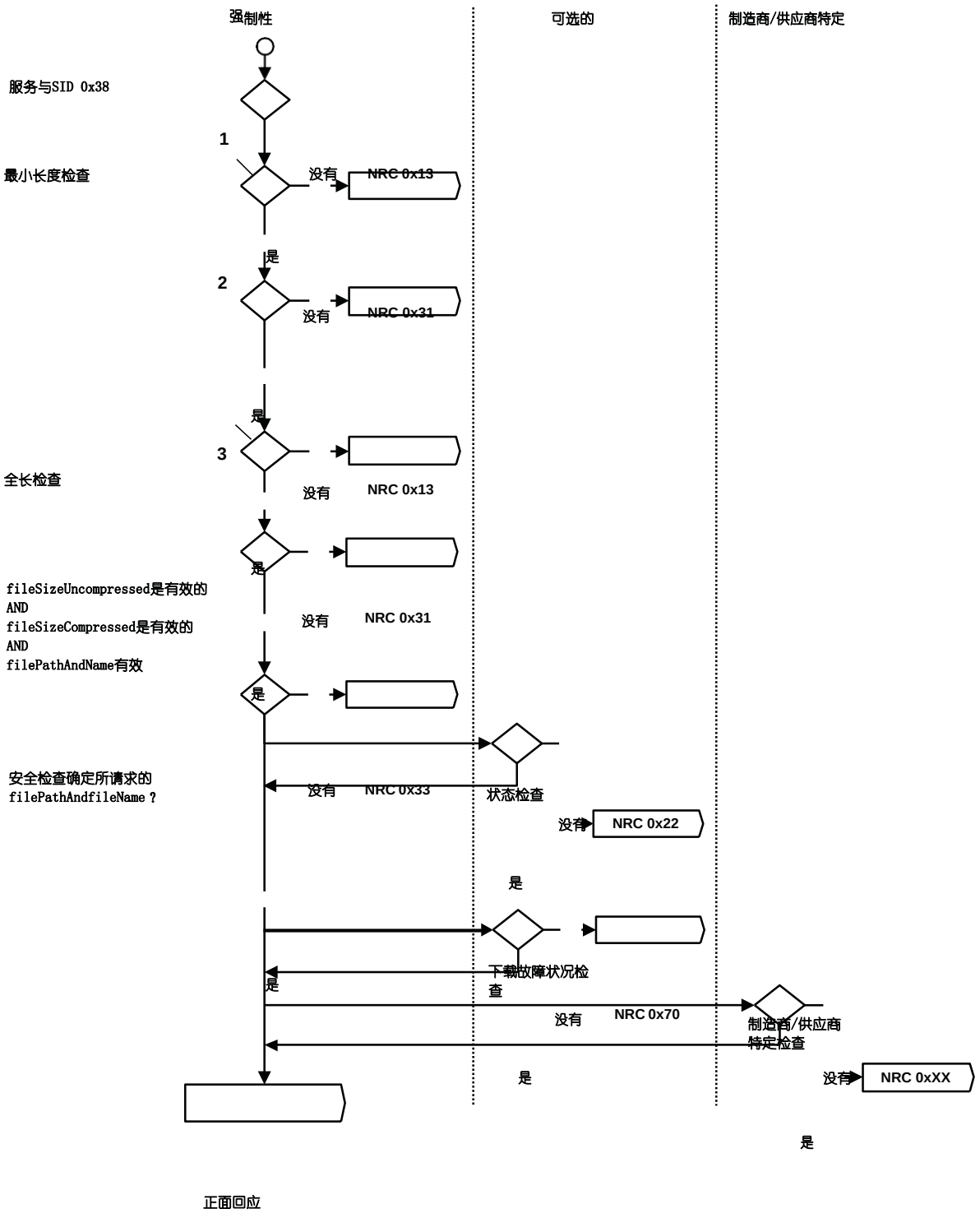
14.6.4 支持的否定响应代码（NRC_）

此服务应执行以下负面响应代码。 表437中记录了每个响应代码发生的情况。如果错误情况适用于服务器，则应使用列出的否定响应。

表437 – 支持的否定响应代码

NRC	描述	助记符
0x13	**incorrectMessageLengthOrInvalidFormat** 如果消息长度错误，则应发送NRC。	IMLOIF
0x22	**conditionsNotCorrect** 如果服务器在下载或上传数据或其他条件以便能够执行此服务的过程中未收到服务器接收到此服务的请求，则应返回此NRC。	CNC
0x31	**requestOutOfRange** 如果出现下列情况，将返回NRC: — 指定的dataFormatIdentifier无效 — 指定的modeOfOperation无效 — 指定的fileSizeParameterLength无效 — 指定的filePathAndNameLength无效 — 指定的fileSizeUncompressed无效 — 指定的fileSizeCompressed无效 — 指定的filePathAndName无效	ROOR
0x33	**securityAccessDenied** 如果服务器是安全的（对于支持SecurityAccess服务的服务器），则在收到对该服务的请求时应返回此NRC。	伤心
0x70	**uploadDownloadNotAccepted** 此NRC表示尝试下载到服务器的内存由于某些故障条件而无法完成。	UDNA

评估顺序记录在图30中。



键

1 最小长度：5字节（SI + MOOP + FPL_B1 + FPL_B2 + FP_B1）

2 消息参数的有效性检查取决于modeOfOperation参数

3 最大长度可以使用fileSizeParamterLength和filePathAndNameLength来计算

图30 - 响应评估序列requestFileTransfer

14.6.5 消息流示例RequestFileTransfer

14.6.5.1 假设

此子条款指定适用于此消息流示例的条件。

注：本示例仅限于requestFileTransfer请求和requestFileTransfer肯定响应的描述。 此上下文中transferData和requestTransferExit的用法与requestDownload或requestUpload中这些服务的用法相同，因此描述下载/上传顺序的示例也适用。

表438定义了消息参数值。

表438 – 定义RequestFileTransfer消息参数值

数据参数名称	数据参数值（s）	数据参数说明
操作模式	0x01	添加文件
filePathAndNameLength	0x001E	参数filePathAndName的长度是30。
filePathAndName	“d: \属于MapData \欧洲\ germanyl.yxz”	包含文件名的路径。
dataFormatIdentifier	0x11	compressionMethod = 0x1X; encryptingMethod = 0xX1
fileSizeParameterLength	0x02	两个文件大小参数的长度都是2个字节。
fileSizeUncompressed	0xC350	50千字节
fileSizeCompressed	0x7530	30千字节

14.6.5.2 请求文件传输

表439和表440显示了RequestFileTransfer请求和响应消息流的示例。

表439 – RequestFileTransfer请求消息示例

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RequestFileTransfer请求SID	0x38	RFT
#2	操作模式	0x01	MOOP
#3 #4	filePathAndNameLength [字节#1 (MSB) 字节 #2] (LSB)	0x00 0x1E	FPL_B1 FPL_B2

表439 – （续）

消息方向	服务器　客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
	filePathAndName = [		
#5	字节#1	0x44	FP_B1
#6	(MSB) 字节	0x3A	FP_B2
#7	#2字节#3字	0x5C	FP_B3
#8	节#4字节#5	0x6D	FP_B4
#9	字节#6字节	0x61	FP_B5
#10	#7字节#8字	0x70	FP_B6
#11	节#9字节#	0x64	FP_B7
#12	10字节#11字	0x61	FP_B8
#13	节#12字节#	0x74	FP_B9
#14	13字节#14字	0x61	FP_B10
#15	节#15字节#	0x5C	FP_B11
#16	16字节#17字	0x65	FP_B12
#17	节#18字节#	0x75	FP_B13
#18	19字节#20字	0x72	FP_B14
#19	节#21字节#	0x6F	FP_B15
#20	22字节#23字	0x70	FP_B16
#21	节#24字节#	0x65	FP_B17
#22	25字节#26字	0x5C	FP_B18
#23	节#27字节#	0x67	FP_B19
#24	28字节#29字	0x65	FP_B20
#25	节#30]	0x72	FP_B21
#26		0x6D	FP_B22
#27		0x61	FP_B23
#28		0x6E	FP_B24
#29		0x79	FP_B25
#30		0x31	FP_B26
#31		0x2E	FP_B27
#32		0x79	FP_B28
#33		0x78	FP_B29
#34		0x7A	FP_B30
#35	dataFormatIdentifier	0x11	DFI_
#36	fileSizeParameterLength	0x02	FSL
	fileSizeUnCompressed= [		
#37	字节#1 (MSB) 字节	0xC3	FSUC_B1
#38	#2]	0x50	FSUC_Bk
	fileSizeCompressed= [		
#39	字节#1 (MSB) 字节	0x75	FSC_B1
#40	#2]	0x30	FSC_Bk

表440 – RequestFileTransfer肯定响应请求消息示例

消息方向	服务器 客户端		
消息类型	响应		
A_Data字节	说明（所有值均为十六进制）	字节值	助记符
#1	RequestFileTransfer响应SID	0x78	RRFT
#2	操作模式	0x01	MOOP
#3	lengthFormatIdentifier	0x02	LFID
#4 #5	maxNumberOfBlockLength = [字节#1（MSB）字节 #m]	0xC3 0x50	MNROB_ B1 B2
#6	dataFormatIdentifier	0x11	DFI_

15 非易失性服务器内存编程过程

15.1 一般信息

本节定义了将一个或多个应用软件/数据模块以物理方式下载到非易失性服务器内存的框架。 定义的非易失性服务器内存编程序列地址：

a) 汽车制造商在编程过程中执行某些步骤时的特定需求，同时符合ISO 14229和本部分第2部分中规定的一般服务执行要求（如服务顺序和会话管理），

b) 为了支持具有连接的多个节点的网络，其使用正常的通信消息相互交互，

c) （点对点通信 – 服务器不支持功能诊断通信）或功能导向的车辆方法（点对点和点对多通信 – 服务器支持功能性诊断通信）。 一辆车只能支持上述车辆进近之一。

编程序列分为两个编程阶段。 所有步骤均基于以下类型进行分类：

— 标准化步骤：这种类型的步骤是强制性的。 客户端和服务器应按照规定行事。

— 可选/推荐的步骤：这种类型的步骤是可选的。 这些可选步骤需要使用特定的诊断服务标识符（如步骤中所述），并包含有关如何执行操作的建议。 在使用指定的功能的情况下，客户端和服务器应按照规定行事。

— 整车制造商的具体步骤：这种类型的步骤是可选的。 这些可选步骤的使用和内容（例如，使用的诊断服务标识符）由车辆制造商自行决定，并应符合ISO 14229-1和ISO 14229-2。

定义的步骤可以是：

— 功能上针对网络上的所有节点（功能导向的车辆方法，服务器支持功能诊断通信），或

— 物理地址给网络上的每个节点（物理导向的车辆方式）。

编程过程的两个编程阶段的每一步都将指定该步骤允许的寻址方法。 车辆制造商的具体步骤可以通过功能或物理地址（取决于OEM要求）。

图31描述了非易失性服务器内存编程过程概述。

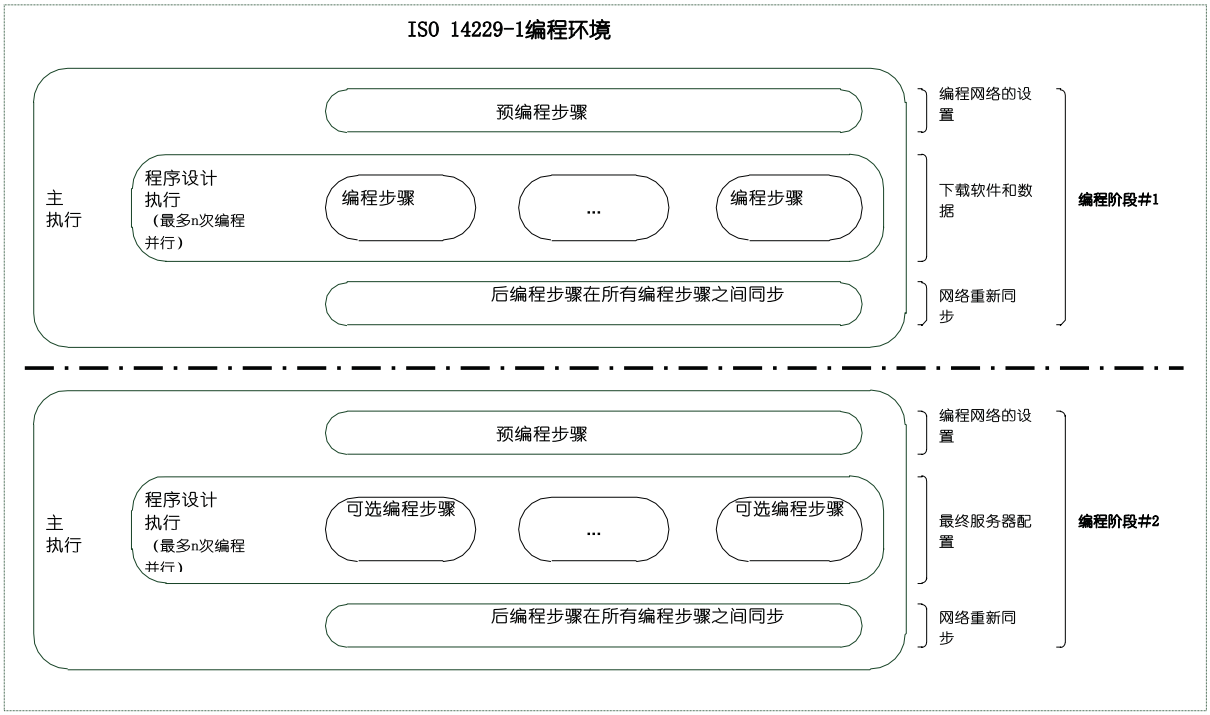


图31 – 非易失性服务器内存编程过程概述

客户端需要遵循的编程过程由两种不同类型的诊断服务执行组成：

— **主执行**：
在并行运行的多个编程步骤之间需要同步的所有步骤必须协调，因为它们旨在用于车辆宽泛的功能（例如，通常使用功能寻址）。 这是通过客户端的“主执行”实现的。 为各个编程阶段的“预编程步骤”和“后编程步骤”定义的步骤由客户端的“主执行”执行。 编程过程需要在各个“编程步骤”之间进行同步（例如，将车辆网络转换为允许对单个ECU进行编程的操作模式，或者在各个并行“编程步骤”达到点需要编程阶段的结论）。 主执行器必须将车辆维持在其已经转换到的操作模式中。

— **编程执行**：
在多个“编程步骤”之间不需要同步的所有步骤不需要由客户端协调并且可以并行运行，因此在执行这些步骤期间客户端不需要“主执行”。 各个ECU的“编程步骤”可由客户单独并行执行，直至完成并需要执行“编程后阶段”。 由“编程执行”控制的所有步骤都是面向ECU的步骤（例如，物理地址到要编程的ECU）。

a) **编程阶段#1 – 下载应用程序软件和/或应用程序数据**

1) 在编程阶段#1中，应用软件/数据被传送到服务器。

i) 可选的预编程步骤 – 设置车辆网络进行编程

阶段#1的预编程步骤是可选的，并且用于为一个或多个服务器的编程事件准备车辆网络。 该步骤提供了车辆制造商可以插入OEM车辆网络所需的特定操作（执行唤醒，确定通信参数，读取服务器标识数据等）的某些挂钩。

此步骤还包含增加波特率以提高下载性能的规定。 此功能的使用是可选的，并且只能在功能导向的车辆方法（服务器支持的功能性诊断通信）的情况下执行。

此步骤的请求消息可以是物理地址或功能地址。

2) 服务器编程步骤 – 下载应用程序软件和应用程序数据

阶段#1的服务器编程步骤用于编程一个或多个服务器（下载应用程序软件和/或应用程序数据和/或启动软件）。

在此步骤中，客户端只使用物理寻址，这允许并行或顺序编程多个节点。 在没有使用预编程步骤的情况下，具有子功能编程会话的DiagnosticSessionControl (0x10) 也可以使用功能性寻址来执行。

在此步骤结束时，重新编程的服务器的物理重置是可选的。 复位的使用导致实现编程阶段#2的要求，以便通过物理地清除重新编程的服务器中的DTC来最终结束编程事件，因为在该步骤期间的物理重置之后，重新编程的服务器（s）启用默认会话并执行其正常操作模式，而其余服务器仍然禁用正常通信。重新编程的服务器可能会设置DTC。

此外，应该认为重新编程的服务器可以激活一组新的诊断地址，该地址与执行编程事件时使用的地址不同（见15.3）。

如果重新编程的服务器没有改变其通信参数，或者客户端知道改变的通信参数，则在重新设置后，某些配置数据可以写入重新编程的服务器。

3) 后编程步骤 – 编程后的车辆网络重新同步

阶段#1的后期编程步骤结束编程阶段#1。 当每个重新编程的服务器的编程步骤完成时执行该步骤。

此步骤的请求消息可以是物理地址或功能地址。

车辆网络转换到其正常操作模式。 这可以通过使用ECUReset (0x11) 服务的重置或通过DiagnosticSessionControl (0x10) 服务显式转换到默认会话来完成。

b) **编程阶段#2 – 服务器配置（可选）**

1) 编程阶段#2是一个可选阶段，在该阶段中，客户端可以执行最终结束编程事件所需的进一步操作（写入VIN，触发防盗器学习例程等）。 例如，如果已经被重新编程的服务器被物理重置

在编程阶段＃1的服务器编程步骤期间，应在该服务器中清除DTC。

2) 执行此阶段时，下载的应用程序软件/应用程序数据在服务器中运行/激活，并且服务器提供其完整的诊断功能。

— 预编程步骤 - 设置车辆网络以进行服务器配置

阶段＃2的预编程步骤用于准备用于阶段＃2的编程步骤的车辆网络。 这一步骤是可选步骤，并提供车辆制造商可以插入OEM车辆网络所需特定操作（例如唤醒，确定通信参数）的某些挂钩。

这些步骤的请求消息可以是物理地或功能地解决。

— 编程步骤 - 最终服务器配置

编程步骤用于例如在服务器复位之后写入数据（例如VIN）。 这一步的内容是汽车制造商特

定的。

如果已经被重新编程的服务器在编程阶段＃1的服务器编程步骤结束时被物理重置，则在阶段＃1的编程步骤期间，DTC将在该服务器中被清除。 2。

这些步骤的请求消息在物理上得到解决。

— 后编程步骤 - 在最终服务器配置后重新同步车辆网络

编程后步骤结束编程阶段＃2。 当每个重新编程的服务器的编程步骤完成时执行该步骤。 车辆网络转换到其正常操作模式。

这一步可以是功能导向的（服务器支持功能性诊断通信）或物理导向的。

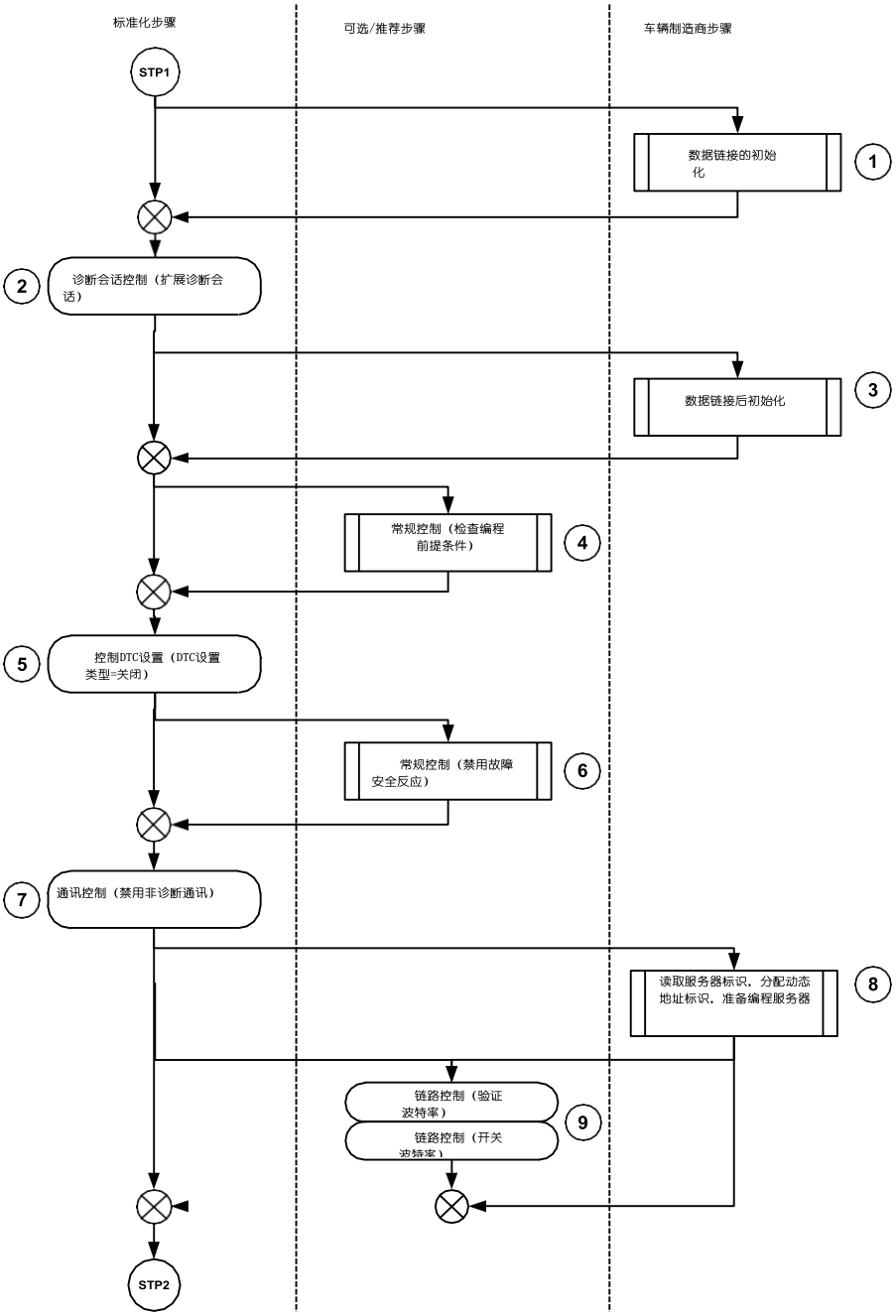
这些步骤的请求消息可以是物理地或功能地解决。

15.2 详细的编程顺序

15.2.1 编程阶段#1 – 下载应用程序软件和/或应用程序数据

15.2.1.1 阶段#1的预编程步骤 – 设置车辆网络进行编程

图32以图形方式描绘了预编程步骤中嵌入的功能。



键

1在数据链路上进行任何通信之前，应对网络进行初始化（例如，在网络上执行初始唤醒）。 唤醒方法和策略是车辆制造商特定的和可选的使用。 此外，该步骤允许确定服务器通信参数，例如由服务器使用的网络配置参数和服务器诊断地址。

2 为了能够禁用服务器之间的正常通信和设置DTC，需要在每个服务器中启动非默认会话，其中正常通信和DTC将被禁用。 这是通过一个sessionType等于extendedDiagnosticSession的DiagnosticSessionControl（0x10）服务实现的。该请求或者通过单个请求消息在功能上发送到所有服务器，或者在单独的请求消息中物理寻址到每个服务器（要求将物理寻址的TesterPresent（0x3E）请求消息发送到每个转换为非-defaultSession）。 汽车制造商是否需要响应消息。

3 在转换到扩展诊断会话之后，可以选择性地执行其他车辆制造商特定的数据链路初始化步骤。

示例车辆制造商特定的附加初始化步骤可以是发出请求，使网关设备在客户端无法通过诊断连接器直接访问的所有数据链路上执行唤醒。 只要非defaultSession在网关中保持活动状态，网关就会保持数据链路清醒。

4 这个可选的routineIdentifier（车辆制造商选择的编号）允许客户在尝试转换之前检查是否满足所有转换到编程会话的前置条件。

5 客户端使用DTCSettingType等于“off”的ControlDTCSetting（0x85）服务禁用每个服务器中的DTC设置。 该请求或者通过单个请求消息在功能上发送到所有服务器，或者在单独的请求消息中以物理地址发送到每个服务器。 汽车制造商是否需要响应消息。

6 这个可选的routineIdentifier（车辆制造商选择的编号）允许客户为了安全起见需要时启用或禁用ECU的故障安全响应。

7 客户端使用CommunicationControl（0x28）服务禁用发送和接收非诊断消息。 controlType参数和communicationType参数值是车辆制造商特定的（一个OEM可能会禁用传输，而另一个OEM可能会根据车辆制造商的特定需求禁用传输和接收）。 该请求或者通过单个请求消息在功能上发送到所有服务器，或者在单独的请求消息中以物理地址发送到每个服务器。 汽车制造商是否需要响应消息。

8 禁用正常通信后，可选车辆制造商的具体步骤如下，它允许以下内容。

— 读取要编程的服务器的状态（例如应用程序软件/编程的数据）。

— 从要编程的服务器读取服务器标识数据：

  — 鉴定（看到dataIdentifier定义）：应用软件识别，applicationDataIdentification，

  — 指纹（请参阅dataIdentifier定义）：applicationSoftwareFingerprint，applicationDataFingerprint，

— 通信配置，如动态分配“服务ECU”的地址标识符。

— 为即将到来的编程事件准备非可编程服务器，以便他们优化其数据链接硬件接受过滤，以便在不丢弃数据链接帧的情况下处理100％总线使用率（只接受功能请求地址标识符和它自己的物理请求地址标识符）。

9 增加编程事件的带宽是可选的，以便减少整体编程时间并获得额外的带宽以便能够并行编程多个服务器。 带有等于verifyBaudrateTransitionWithFixedMode 或 verifyBaudrateTransitionWithSpecificMode 的 linkControl 的LinkControl（0x87）服务通过responseRequired等于“yes”的单个请求消息以功能或物理地址发送到所有服务器。该服务用于验证相关数据链路上的模式转换是否可以执行。 此时不执行转换。 带有子功能transitionMode的第二个LinkControl（0x87）服务通过responseRequired等于“no”的单个请求消息在功能上发送到所有服务器。

一旦请求消息被成功传输，客户端和所有服务器就转换到先前验证的编程事件模式。 服务器必须在车辆制造商特定的时间窗口内转换个别数据链路特定模式。 在此期间加上安全余量，客户端不允许将任何请求消息传输到车辆网络（包括TesterPresent请求消息）。 转换成功执行后，请求的模式在服务器在非defaultSessions之间切换期间保持活动状态。 一旦服务器转换到defaultSession，它将重新启用它所连接的车辆链路的正常模式。

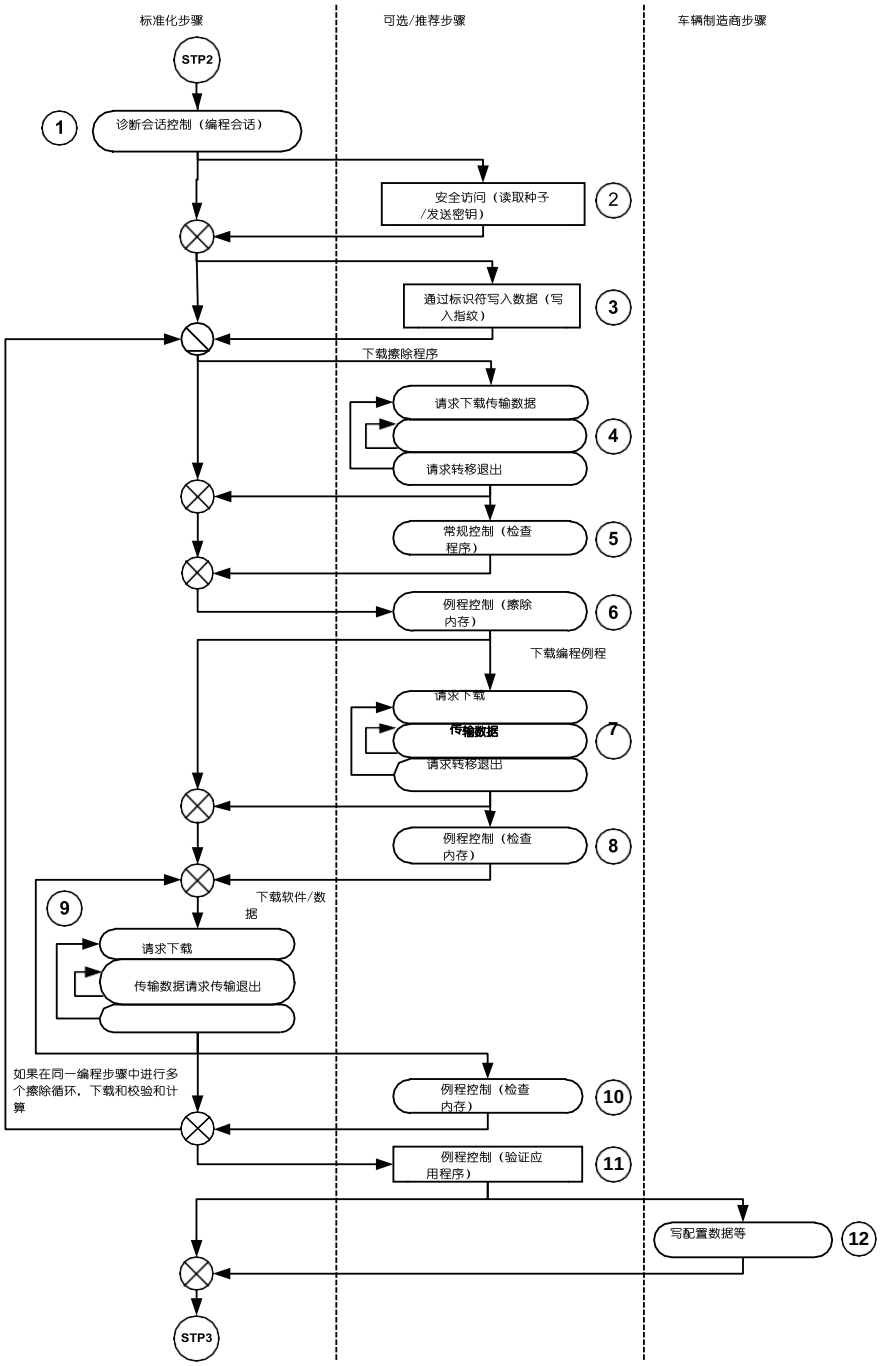
模式开关的使用要求在单个数据链路上的每个服务器中支持功能性诊断通信，该数据链路应该转换到相关的数据链路相关模式。

图32 – 阶段1的预编程步骤（STP1）

15.2.1.2 阶段1的编程步骤 – 下载应用程序软件和数据

在预编程步骤之后，执行一个或多个服务器的编程。 编程序列适用于单个服务器的编程事件，因此是物理导向的。 当多个服务器编程时，多个编程事件要么并行运行要么依次执行。

图33图示了嵌入在阶段#1的编程步骤中的功能。



键

1 编程事件通过服务器中的sessionType与programmingSession相同的DiagnosticSessionControl (0x10) 服务的物理/功能编址请求启动。 当服务器 (s)

接收请求，它/他们应分配编程所需的所有必要资源。 服务器是否开始执行不在启动软件的具体实现。

2 应该确保编程事件。 SecurityAccess (0x27) 服务对排放相关和安全系统是强制性的。 其他系统不需要实施这项服务。 ISO 14229的本部分规定了如何执行安全访问的方法。

3 在将任何数据（例如，应用软件）下载到ECU之前，车辆制造商特定于将“指纹”写入到服务器存储器中。 “指纹”标识修改服务器内存的人。 使用此选项时，应使用dataIdentifiers bootSoftwareFingerprint，applicationSoftwareFingerprint和applicationDataFingerprint写入指纹信息（请参阅dataIdentifier定义）。

4 如果服务器没有存储在永久存储器中的存储器擦除例程，则应执行存储器擦除例程的下载。 下载应遵循指定的顺序与RequestDownload (...)，TransferData和RequestTransferExit。

5 如果使用RoutineControl (0x31) 来检查内存擦除例程的下载是否成功，则它是车辆制造商特定的。 其他方法是在RequestTransferExit肯定响应消息中或通过包含对RequestTransferExit请求消息的适当否定响应代码的否定响应消息中提供结果。

6 存储器技术（例如闪存）需要时，服务器的存储器应该被擦除，以便允许应用软件/数据下载。 这是通过一个routineIdentifier实现的，使用RoutineControl (0x31) 服务来执行擦除例程。

7 如果服务器没有将存储器编程例程存储在永久存储器中，则应执行存储器编程例程的下载。 下载应遵循具有RequestDownload (0x34)，TransferData (0x36) 和RequestTransferExit (0x37) 的指定序列。 请注意，内存编程算法可能与内存擦除算法一起下载（请参阅脚注d）。

8 如果使用RoutineControl (0x31) 来检查内存程序例程的下载是否成功，则它是车辆制造商特定的。 其他方法是在RequestTransferExit肯定响应消息中或通过包含对RequestTransferExit请求消息的适当否定响应代码的否定响应消息中提供结果。

9 每个应用程序软件/数据的连续块下载到非易失性服务器存储器位置（完整的应用软件/数据模块或软件/数据模块的一部分）应始终遵循通用数据传输方法，使用以下服务序列:

— RequestDownload (0x34)；

— TransferData (0x36)；

— RequestTransferExit (0x37)。

单个应用程序软件/数据块可能需要多个TransferData (0x36) 请求消息才能完全传输（如果块的长度超过最大网络层缓冲区大小，则是这种情况）。

10 如果使用RoutineControl (0x31) 来检查内存的下载是否成功，则它是车辆制造商特定的。 其他方法是在RequestTransferExit肯定响应消息中或通过包含对RequestTransferExit请求消息的适当否定响应代码的否定响应消息中提供结果。

11 该可选的routineIdentifier（车辆制造商选择的编号）允许客户验证一旦所有应用软件/数据块/模块完全下载后，下载是否已成功执行。 此例程通常会触发服务器检查任何和所有重新编程依赖关系，并执行所有必要的操作以证明对非易失性存储器的下载和编程是成功和有效的（例如，校验和，签名，DTC，硬件/软件兼容性等）。 细节由车辆制造商决定。

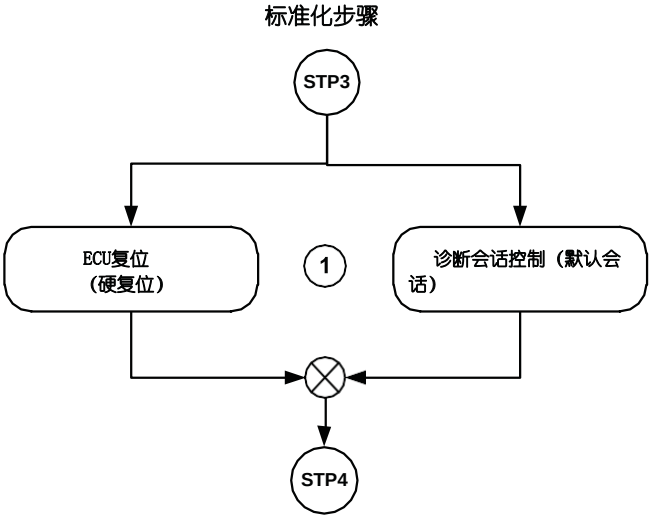
在下载应用程序软件/数据后，重置重新编程的服务器以启用下载的应用程序软件/数据是可选的。 应该认为重新编程的服务器可以激活一组新的诊断标识符，这与执行编程事件时使用的不同。 如果重新编程的服务器不改变其通信参数，或者编程环境知道改变的通信参数，则在重置之后，可以将某些配置数据写入重新编程的服务器。

12 在下载应用软件/数据之后，车辆制造商专门执行进一步的操作，例如将配置数据（例如VIN等）写入服务器。 这也取决于在用完启动软件时重新编程的服务器所支持的功能。

图33 - 阶段1的编程步骤（STP2）

15.2.1.3 阶段#1的后编程步骤 - 车辆网络的重新同步

图34以图形方式描绘了嵌入在阶段#1的后期编程步骤中的功能。



键

1客户端将ECUReset（0x11）服务请求消息传输到车辆网络，其中resetType等于hardReset或DiagnosticSessionControl（0x10），sessionType等于defaultSession。 这可以通过功能来解决或物理解决（取决于所支持的车辆方法）。此外，车辆制造商是否需要响应消息。

当波特率开关已经被执行时，这个步骤将在功能上执行，不需要响应消息，因为服务器执行波特率转换到它们的正常运行速度。

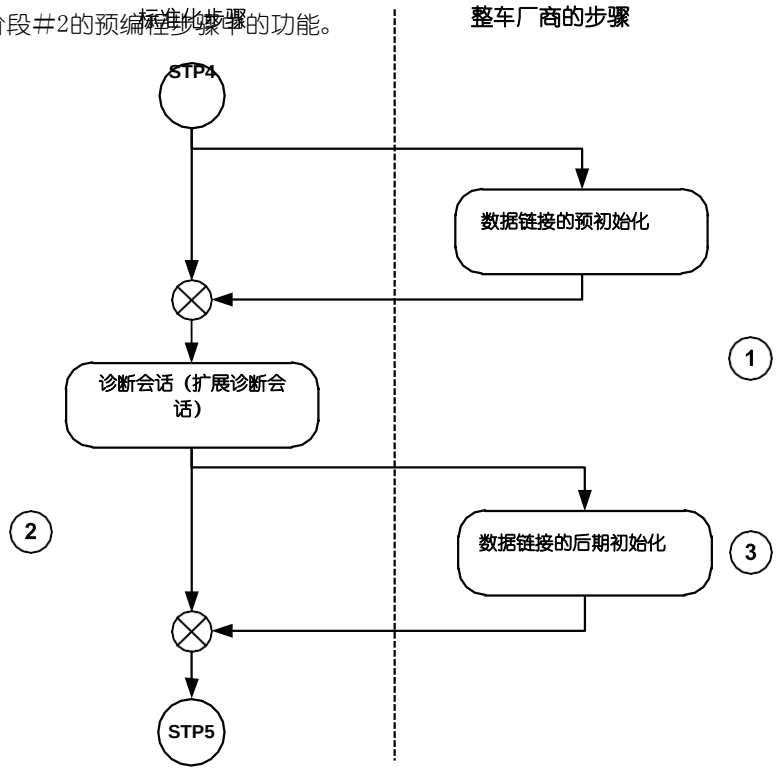
ECUReset（0x11）请求消息的接收导致服务器执行重置并启动defaultSession。

图34 - 阶段1的后编程步骤（STP3）

15.2.1.4 阶段#2的预编程步骤 - 服务器配置

阶段#2的预编程步骤是可选的，并且应在重新编程的服务器的软件重置之后需要执行特定操作时使用。 在阶段#1的编程步骤期间，当服务器没有提供所需的功能来最终结束编程事件时将会出现这种情况。

图35图示了嵌入在阶段#2的预编程步骤中的功能。



键

1 在数据链路上进行任何通信之前，应对网络进行初始化，这意味着应执行车辆网络的初始唤醒。 唤醒方法和策略是车辆制造商特定的和可选的使用。
此外，该步骤允许确定服务器通信参数，例如由服务器使用的网络配置参数服务器诊断地址和数据链路标识符。

2 为了能够在阶段#2的编程步骤中执行某些服务，应在编程事件结束所涉及的数据链路上的每个服务器中启动一个非defaultSession。 这是通过sessionType等于extendedDiagnosticSession的DiagnosticSessionControl (0x10)服务执行的。

3 在转换到扩展诊断会话之后，可以选择性地执行其他车辆制造商特定的数据链路初始化步骤。

示例车辆制造商特定的附加初始化步骤可以是发出请求，使网关设备在客户端无法通过诊断连接器直接访问的所有数据链路上执行唤醒。 只要非defaultSession在网关中保持活动状态，网关就会保持数据链路清醒。

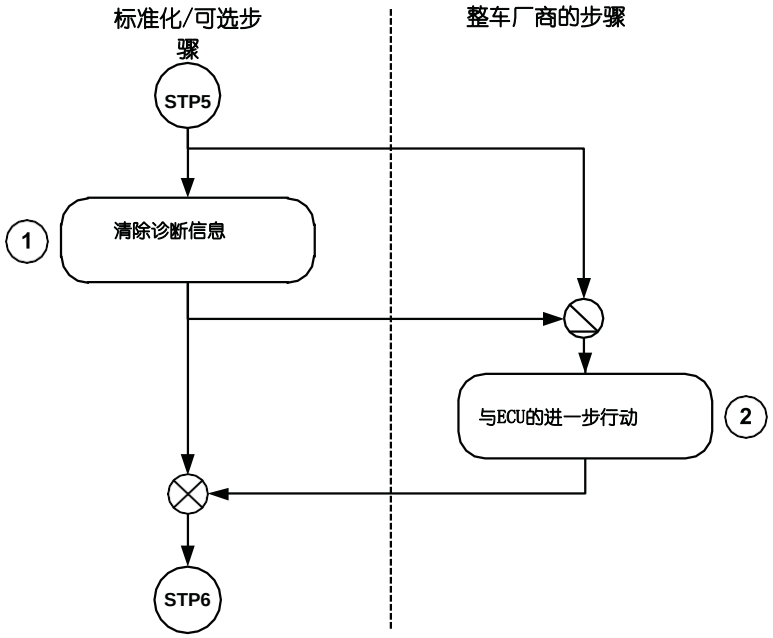
图35 – 阶段2的预编程步骤（STP4）

15.2.1.5 阶段#2的编程步骤 – 最终服务器配置

阶段#2的编程步骤是可选的，包含重置后（应用软件运行时）重新编程的服务器需要执行的任何操作，例如编写特定的标识信息。 如果在阶段#1的编程步骤期间服务器未提供执行启动软件所需的功能时，可能需要执行此步骤。

当多个服务器需要执行附加功能时，多个编程步骤可以并行运行或按顺序执行。

图36描述了阶段2（STP5）的编程步骤。



键

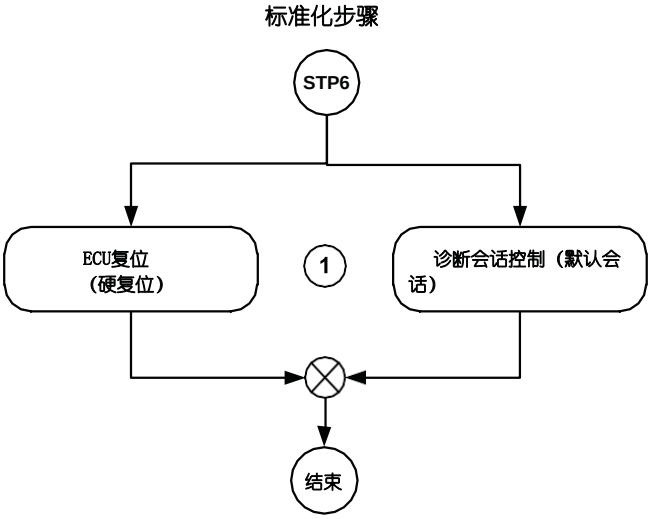
1 在编程阶段的编程步骤期间重新编程的服务器已经被重置的情况#1，则可能已经存储在重新编程的服务器中的任何诊断信息可以通过物理寻址的ClearDiagnosticInformation（0x14）服务来清除。

2 客户端执行任何需要的操作来结束与服务器的编程事件，例如写入配置数据（例如VIN）。

图36 - 阶段2的编程步骤（STP5）

15.2.1.6 阶段#2的后期编程步骤 - 车辆网络的重新同步

图37描述了阶段2的后期编程步骤（STP6）。



键

1客户端将ECUReset（0x11）服务请求消息传输到车辆网络，其中resetType等于hardReset或DiagnosticSessionControl（0x10），sessionType等于defaultSession。 这可以通过功能来解决或物理解决（取决于所支持的车辆方法）。此外，无论是否需要响应消息，它都是车辆制造商特有的。
当波特率开关已经被执行时，这个步骤将在功能上执行，不需要响应消息，因为服务器执行波特率转换到它们的正常运行速度。
ECUReset（0x11）请求消息的接收导致服务器执行重置并启动defaultSession。

图37 - 阶段2的后期编程步骤（STP6）

15.3 服务器重新编程要求

15.3.1 服务器支持编程的要求

在编程会话期间，服务器应将其物理I / O引脚（尽可能并且没有损坏服务器/车辆的风险并且没有安全危险的风险）默认为预定义的状态，从而最小化电流消耗。

15.3.1.1 引导软件的描述和要求

15.3.1.1.1 引导软件的一般要求

所有支持应用软件编程的可编程服务器应包含引导存储器分区中的引导软件。 支持引导软件的服务器通常会继续执行引导软件，直到完整的一组应用程序软件和应用程序数据被编程为止（例如，尽管没有编写100%的应用程序数据，某些服务器仍可能开始执行应用程序软件） 。

引导存储器分区应受到保护，防止意外删除，以致尝试修改应用程序数据或应用程序软件的失败不会妨碍服务器在失败尝试后恢复和编程的能力。 如果在编程过程中发生以下错误情况，服务器应能够恢复并重新编程：

a) 供电电源连接中断。

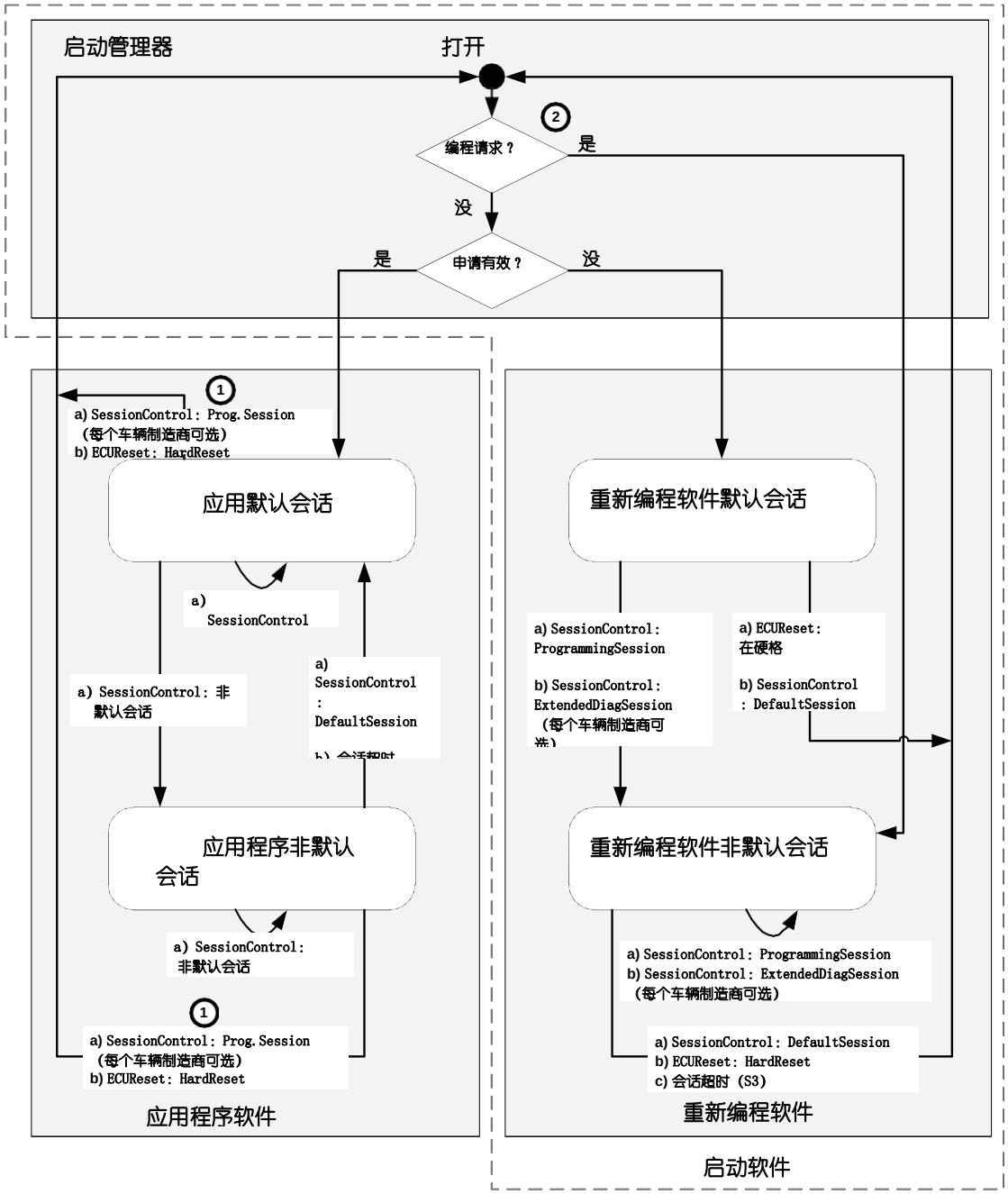
b) 失去地面连接。

c) 数据链路通信中断。

d) 过压或欠压条件。

引导软件可以通过硬件（例如，通过控制寄存器中防止某些扇区被擦除或写入的设置）或软件（例如编程例程中的地址范围限制）来保护。 建议引导软件不能通过用于修改应用程序软件和应用程序数据的相同编程擦除/写入例程进行修改。 编程引导软件作为编程过程的一部分可能是允许的，只要有一种机制来确保服务器在编程过程中不可能发生故障而无法恢复并且可以用后续编程事件。

引导软件驻留在引导内存分区中，是启动时服务器开始执行的软件。 一旦服务器被告知它将要被编程（例如，引用DiagnosticSessionControl服务和15.2.1.2中定义的编程过程），也会将程序控制转移到引导软件。 显示引导软件和应用程序软件之间的交互和转换的典型实现如图38所示。



键

1 某些实现可能有能力转换到重新编程软件编程会话，而无需进行全面的上电复位。

2 该检查可以用于两个目的。 一个是检查应用程序是否要求转换到重编程软件的编程会话。 另一种是通过其他条件替代进入重编程软件（例如，通过在小时间窗口扫描SessionControl编程会话请求）。

图38 – 应用程序和引导软件之间的典型交互和转换示例

15.3.1.1.2 引导软件诊断服务要求

表441到表443定义了可编程服务器启动软件的最低诊断服务要求。 列出的服务必须得到支持才能满足编程阶段#1期间执行非易失性服务器内存编程的要求。 这些表格利用为编程阶段#1定义的步骤（见15.2.1）。 步骤（1），（3）和（2）支持的服务
（8）应由车辆制造商定义。

表441 – 在阶段#1的预编程步骤期间支持启动软件诊断服务

服务	子功能/数据参数	序列号	备注
DiagnosticSessionControl (0x10)	sessionType = extendedDiagnosticSession (0x03)	(2)	强制性： 会话管理所需的（S3服务器超时，特别是在执行波特率转换和SecurityAccess服务时）。
CommunicationControl (0x28)	controlType = 特定于车辆制造商（禁用非诊断通信消息）	(7)	强制性： 服务器不需要执行任何特殊操作（非引导时非诊断消息被禁用），除了传送肯定响应消息。
RoutineControl (0x31)	routineIdentifier =车辆制造商特定的	(4), (6)	可选的： 如果支持编程预先条件或禁用故障安全响应，则为必需。
ControlDTCSetting (0x85)	DTCSettingType = off (0x02)	(5)	强制性： 服务器不需要执行任何特殊操作（DTC在引导用完时被禁用），除了传送肯定响应消息。
ReadDataByIdentifier (0x22)	dataIdentifier = 车辆制造商具体	(8)	可选的： 读取软件/数据识别数据时需要支持。
LinkControl (0x87)	linkControlType = verifyWithFixedBaudrate (0x01)，verifyWithSpecifcBaudrate (0x02)，transitionBaudrate (0x03)	(9)	可选的： 执行波特率开关时需要支持。

注意表441仅适用于车辆制造商支持阶段#1的预编程步骤。

表442 – 在阶段#1的编程步骤期间支持启动软件诊断服务

服务	子功能/数据参数	序列号	备注
DiagnosticSessionControl (0x10)	sessionType = programmingSession (0x02)	(1)	强制性： 为了与应用程序软件兼容，需要在客户端的编程应用程序中进行相同的处理。
SecurityAccess (0x27)	securityAccessType = requestSeed (0x01)，sendKey (0x02)	(2)	可选的： 需要得到盗窃，排放和安全相关系统的支持。
WriteDataByIdentifier (0x2E)	bootSoftwareFingerprint，appSoftwareFingerprint，appDataFingerprint，车辆制造商特定	(3)	可选的： 写入指纹和其他标识数据时需要。
请求下载 (0x34)	车辆制造商具体	(4), (7), (9)	
TransferData (0x36)	例程数据，应用程序软件或应用程序数据		
RequestTransferExit (0x37)	车辆制造商具体		
RoutineControl (0x31)	routineControlType = startRoutine (0x01) routineIdentifier = 参考序列步骤所需号码的详细信息	(5), (6), (8), (10), (11)	可选的： 如果任何序列步骤均由车辆制造商支持，则为必需。
ECUReset (0x11)	resetType = hardReset (0x01)	(12)	强制性： 在编程步骤结束时重新设置重新编程的服务器时需要。 已重新编程的服务器被强制执行重置以启动应用程序软件。
步骤（m）支持的服务应由车辆制造商定义。			

表443 – 在阶段#1的后编程步骤期间支持启动软件诊断服务

服务	子功能/数据参数	序列步骤	备注
ECUReset (0x11)	resetType = hardReset (0x01)	(1)	强制性： 已重新编程的服务器被强制执行重置以启动应用程序软件。

15.3.1.2 安全要求

所有具有发射，安全或盗窃相关功能的可编程服务器都应采用可通过SecurityAccess（0x27）服务访问的种子和密钥安全功能，以保护已编程的服务器免遭无意删除和未经授权的编程。 所有这些现场服务替换服务器应该在激活安全功能的情况下运送到现场（即，如果没有首先通过SecurityAccess服务获得访问权，编程工具就无法访问服务器）。

15.3.2 软件，数据识别和指纹

15.3.2.1 软件和数据识别

引导软件，应用程序软件和应用程序数据可以根据C.1通过dataIdentifiers标识。 用于bootSoftwareIdentification， applicationSoftwareIdentification和applicationDataIdentification的dataRecord的结构是汽车制造商特定的。

bootSoftwareIdentification，applicationSoftwareIdentification和applicationDataIdentification应该是下载到服务器中的每个模块的一部分；因此对定义的dataIdentifiers的任何写入操作都应该被服务器拒绝。

15.3.2.2 软件和数据指纹

指纹可唯一标识擦除和/或重新编程服务器软件/数据的编程工具。 如果服务器软件/数据在多个模块中分离，则指纹还可以识别哪个软件/数据模块被操作（例如，启动软件，应用软件和应用程序数据）。 如果支持，在进行任何软件/数据操作之前（例如擦除闪存之前），应将指纹写入服务器的非易失性存储器中。

引导软件，应用程序软件和应用程序数据指纹可以根据C.1通过dataIdentifiers进行标识。

bootSoftwareFingerprint，applicationSoftwareFingerprint和applicationDataFingerprint的dataRecord的结构是车辆制造商特定的。

15.3.3 服务器例程访问

例程用于执行非易失性内存访问，例如擦除非易失性内存并检查模块是否成功下载。

表444定义了用于非易失性存储器访问的标准化例程标识符。 编程序列中使用的其他常规识别号由车辆制造商指定。

表444 - 用于非易失性存储器访问的routineIdentifiers

字节值	描述	助记符
为0xFF00	**eraseMemory**	EM
	这个值应该用来启动服务器的内存擦除例程。 控制选项和状态记录格式应由ECU制定并由车辆制造商定义。	

15.4 非易失性服务器内存编程消息流示例

15.4.1 一般信息

以下示例显示单个服务器的非易失性服务器内存编程事件的CAN消息流量。 给定的消息流基于单个服务器和两个模块的传输，其中每个模块的长度为511个字节。 重新编程的服务器的网络层缓冲区大小为255个字节（在RequestDownload肯定响应消息中报告）。 编程示例使用ISO 15765-4中规定的11位OBD CAN标识符。 因此，所有帧必须用填充字节填充（DLC = 8）。 请求消息的所有CAN帧填充填充字节0x55。 响应消息的所有CAN帧都填充了0xAA的填充字节。

注意填充字节可以有任何值。

15.4.2 编程阶段#1 - 预编程步骤

见表445至表447。

表445 - StartDiagnosticSessionControl（extendedSession）

相对时间	CH#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
27.2174	1	7DF	FUNC。 请求	8	02 10 03 55 55 55 55 55	DSC 消息-SF
0,0001	1	7E8	响应	8	06 50 03 00 96 17 70 AA	DSC 消息-SF
0,0002	1	7E9	响应	8	06 50 03 00 96 17 70 AA	DSC 消息-SF

表446 - ControlDTCSetting（关闭）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
0,0505	1	7DF	FUNC。 请求	8	02 85 02 55 55 55 55 55	CDTCS 信息-SF
0,0001	1	7E8	响应	8	02 C5 02 AA AA AA AA AA	CDTCS 信息-SF
0,0001	1	7E9	响应	8	02 C5 02 AA AA AA AA AA	CDTCS 信息-SF

表447 - CommunicationControl（应用程序中的disableRxAndTx）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,0007	1	7DF	FUNC。 请求	8	03 28 03 01 55 55 55 55	CC 消息-SF
0,0001	1	7E8	响应	8	02 68 03 AA AA AA AA AA	CC 消息-SF
0,0001	1	7E9	响应	8	02 68 03 AA AA AA AA AA	CC 消息-SF
注意 在应用程序中使用子函数disableRxAndTx成功执行CommunicationControl之后，使用suppressPosRspMsgIndicationBit（子函数的第7位）函数寻址TesterPresent消息，=真（1）（无响应）发送约。 每隔2秒将所有服务器保持在该状态，以便不发送正常的通信消息。						

15.4.3 编程阶段#1 - 编程步骤

见表448至表463

表448 - DiagnosticSessionControl（programmingSession）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1.6964	1	7E0	物理学。 请求	8	02 10 02 55 55 55 55 55	DSC消息-SF
0,0012	1	7E8	响应	8	06 50 02 00 FA 0B B8 AA	DSC消息-SF
1,9987	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP消息-SF

表449 - SecurityAccess (requestSeed)

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,0000	1	7E0	物理学。 请求	8	02 27 01 55 55 55 55 55	SA 消息-SF
0,0008	1	7E8	响应	8	04 67 01 21 74 AA AA AA	SA 消息-SF
0,9989	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表450 - SecurityAccess (sendKey)

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,9998	1	7E0	物理学。 请求	8	04 27 02 47 11 55 55 55	SA 消息-SF
0,0002	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
0,0008	1	7E8	响应	8	02 67 02 AA AA AA AA AA	SA 消息-SF
1,9992	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表451 - RoutineControl (eraseMemory)

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
0.9995	1	7E0	物理学。 请求	8	04 31 01 FF 00 55 55 55	RC 消息-SF
0,0001	1	7E8	响应	8	03 7F 31 78 AA AA AA AA	NR w/ NRC78-SF
1,0004	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
1,9995	1	7E8	响应	8	03 7F 31 78 AA AA AA AA	NR w/ NRC78-SF
0,0005	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
2,0001	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
1,0002	1	7E8	响应	8	04 71 01 FF 00 AA AA AA	RC 消息-SF
0,9998	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表452 - 请求下载 - 模块#1

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,9989	1	7E0	物理学。 请求	8	10 09 34 00 33 00 19 68	RD 消息-FF
0,0001	1	7E8	响应	8	30 00 00 AA AA AA AA AA	流量控制
0,0010	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
0,0001	1	7E0	物理学。 请求	8	21 00 01 FF 55 55 55 55	RD 消息-CF
0,0012	1	7E8	响应	8	04 74 20 00 FF AA AA AA	RD 消息-SF
1,9987	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表453 - TransferData - 模块#1（块#1）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
0.9996	1	7E0	物理 请求学。	8	10 FF 36 01 02 03 04 05	TD 消息-FF)
0,0001	1	7E8	响应	8	30 00 00 AA AA AA AA AA	流量控制
0,0012	1	7E0	物理 请求学。	8	21 06 07 08 09 0A 0B 0C	TD 消息-CF
0,0010	1	7E0	物理 请求学。	8	22 0D 0E 0F 10 11 12 13	TD 消息-CF
0,0010	1	7E0	物理 请求学。	8	23 14 15 16 17 18 19 1A	TD 消息-CF
:	:	:	:	:	:	:
0,0010	1	7E0	物理 请求学。	8	23 F4 F5 F6 F7 F8 F9 FA	TD 消息-CF
0,0009	1	7E0	物理 请求学。	8	24 FB FC FD FE 55 55 55	TD 消息-CF
0,0011	1	7E8	响应	8	02 76 01 AA AA AA AA AA	TD 消息-SF
0,9630	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表454 - TransferData - 模块#1（块#2）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,9994	1	7E0	物理 请求学。	8	10 FF 36 02 02 03 04 05	TD 信息 (FF)
0,0001	1	7E8	响应	8	30 00 00 AA AA AA AA AA	流量控制
0,0012	1	7E0	物理 请求学。	8	21 06 07 08 09 0A 0B 0C	TD 信息 (CF)
0,0010	1	7E0	物理 请求学。	8	22 0D 0E 0F 10 11 12 13	TD 信息 (CF)
0,0010	1	7E0	物理 请求学。	8	23 14 15 16 17 18 19 1A	TD 信息 (CF)
:	:	:	:	:	:	:
0,0010	1	7E0	物理 请求学。	8	23 F4 F5 F6 F7 F8 F9 FA	TD 信息 (CF)
0,0009	1	7E0	物理 请求学。	8	24 FB FC FD FE 55 55 55	TD 信息 (CF)
0,0011	1	7E8	响应	8	02 76 02 AA AA AA AA AA	TD 信息
1,9633	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 信息

表455 - TransferData - 模块#1（块#3）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
0,9991	1	7E0	物理学。 请求	8	07 36 03 02 03 04 05 06	TD 消息-SF
0,0011	1	7E8	响应	8	02 76 03 AA AA AA AA AA	TD 消息-SF
0,9998	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表456 - RequestTransferExit - 模块#1

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,9999	1	7E0	物理学。 请求	8	01 37 55 55 55 55 55 55	RTE消息-SF
0,0002	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP消息-SF
0,0009	1	7E8	响应	8	01 77 AA AA AA AA AA AA	RTE消息-SF
1,9992	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP消息-SF

2,0001	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP消息-SF

表457 – 请求下载 – 模块#2

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,9995	1	7E0	物理学。 请求	8	10 09 34 00 33 00 1B 67	RD 消息-FF
0,0001	1	7E8	响应	8	30 00 00 AA AA AA AA AA	流量控制
0,0004	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
0,0007	1	7E0	物理学。 请求	8	21 00 01 FF 55 55 55 55	RD 消息-CF
0,0012	1	7E8	响应	8	04 74 20 00 FF AA AA AA	RD 消息-SF
1,9982	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表458 – TransferData – 模块#2（块#1）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,0002	1	7E0	物理 请求学。	8	10 FF 36 01 02 03 04 05	TD 消息-FF
0,0001	1	7E8	响应	8	30 00 00 AA AA AA AA AA	流量控制
0,0012	1	7E0	物理 请求学。	8	21 06 07 08 09 0A 0B 0C	TD 消息-CF
0,0010	1	7E0	物理 请求学。	8	22 0D 0E 0F 10 11 12 13	TD 消息-CF
0,0010	1	7E0	物理 请求学。	8	23 14 15 16 17 18 19 1A	TD 消息-CF
:	:	:	:	:	:	:
0,0010	1	7E0	物理 请求学。	8	23 F4 F5 F6 F7 F8 F9 FA	TD 消息-CF
0,0009	1	7E0	物理 请求学。	8	24 FB FC FD FE 55 55 55	TD 消息-CF
0,0011	1	7E8	响应	8	02 76 01 AA AA AA AA AA	TD 消息-SF
1,9626	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表459 – TransferData – 模块#2（块#2）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,9994	1	7E0	物理 请求学。	8	10 FF 36 02 02 03 04 05	TD 消息-FF
0,0001	1	7E8	响应	8	30 00 00 AA AA AA AA AA	流量控制
0,0012	1	7E0	物理 请求学。	8	21 06 07 08 09 0A 0B 0C	TD 消息-CF
0,0010	1	7E0	物理 请求学。	8	22 0D 0E 0F 10 11 12 13	TD 消息-CF
0,0010	1	7E0	物理 请求学。	8	23 14 15 16 17 18 19 1A	TD 消息-CF
:	:	:	:	:	:	:
0,0010	1	7E0	物理 请求学。	8	23 F4 F5 F6 F7 F8 F9 FA	TD 消息-CF
0,0009	1	7E0	物理 请求学。	8	24 FB FC FD FE 55 55 55	TD 消息-CF
0,0011	1	7E8	响应	8	02 76 02 AA AA AA AA AA	TD 消息-SF
1,9633	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表460-传输数据 – 模块#2（块#3）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
0,9996	1	7E0	物理学。 请求	8	07 36 03 02 03 04 05 06	TD 消息-FF

0,0011	1	7E8	响应	8	02 76 03 AA AA AA AA AA	TD 消息-SF
0,9993	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
2,0001	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表461 - RequestTransferExit - 模块#2

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
0,0002	1	7E0	物理学。 请求	8	01 37 55 55 55 55 55 55	RTE消息-SF
0,0011	1	7E8	响应	8	01 77 AA AA AA AA AA AA	RTE消息-SF
1,9987	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP消息-SF
2,0001	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP消息-SF

表462 - RoutineControl（验证应用程序）

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
1,0012	1	7E0	物理学。 请求	8	04 31 01 FF 01 55 55 55	RC 消息-SF
0,0001	1	7E8	响应	8	03 7F 31 78 AA AA AA AA	NR w/ NRC78-SF
0,9987	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
2,0001	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
0,0011	1	7E8	响应	8	03 7F 31 78 AA AA AA AA	NR w/ NRC78-SF
1,9990	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
1,0019	1	7E8	响应	8	04 71 01 FF 01 AA AA AA	RC 消息-SF
0,9982	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF
2,0001	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP 消息-SF

表463 - WriteDataByIdentifier - dataIdentifier = VIN

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
0,0004	1	7E0	物理学。 请求	8	10 14 2E F1 90 57 41 4C	WDBI消息-FF
0,0001	1	7E8	响应	8	30 00 00 AA AA AA AA AA	流量控制
0,0012	1	7E0	物理学。 请求	8	21 54 4F 4E 53 2D 57 45	WDBI消息-CF
0,0010	1	7E0	物理学。 请求	8	22 42 2E 43 4F 4D 20 20	WDBI消息-CF
0,0011	1	7E8	响应	8	03 6E F1 90 AA AA AA AA	WDBI消息-SF
1,9961	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP消息-SF
2,0001	1	7DF	FUNC。 请求	8	02 3E 80 55 55 55 55 55	TP消息-SF

15.4.4 编程阶段#1 - 后编程步骤

见表464。

表464 - ECUReset - hardReset

相对时间	CH。#	CAN ID	客户端请求/服务器响应	DLC	PCI和帧数据字节	注释
0,3946	1	7DF	FUNC。 请求	8	02 11 01 55 55 55 55 55	ER 消息-SF
0,0011	1	7E8	响应	8	02 51 01 AA AA AA AA AA	ER 消息-SF
0,0001	1	7E9	响应	8	02 51 01 AA AA AA AA AA	ER 消息-SF

附件A
(规范性附录)
全局参数定义

答1负面答复代码

表A.1定义了本标准中使用的所有负面响应代码。 每个诊断服务都指定适用的否定响应代码 服务器中的诊断服务实施也可以利用由车辆制造商定义的附加和适用的负面响应代码。

否定响应代码范围0x00 – 0xFF分为三个范围:

— 0x00: 服务器内部实现的positiveResponse参数值,

— 0x01 – 0x7F: 通信相关的否定响应码,

— 0x80 – 0xFF: 在服务器收到请求时不正确的特定条件的否定响应代码。 只要响应代码0x22 (conditionsNotCorrect) 被列为有效, 就可以使用这些响应代码, 以便更具体地报告为什么无法执行请求的操作。

表A.1 – 负面响应代码 (NRC) 的定义和值

字节值	负面响应码 (NRC) 的定义	助记符
0x00	**正面回应** NRC不得用于否定回复信息。 此positiveResponse参数值保留用于服务器内部实现。 参考7.5.5。	PR
0x01 – 0x0F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x10	**generalReject** 此NRC表示请求的操作已被服务器拒绝。 如果本文档中定义的否定响应代码均不符合实施需求, 则通用拒绝响应代码只能在服务器中实施。 本NRC决不会替代本文档中定义的响应代码。	GR
0x11	**serviceNotSupported** 此NRC表示请求的操作不会被采用, 因为服务器不支持所请求的服务。 如果客户端发送了一个服务标识符未知, 服务器不支持或被指定为响应服务标识符的请求消息, 服务器应发送此NRC。 因此, 此否定响应代码未显示在诊断服务支持的否定响应代码列表中, 因为此否定响应代码不适用于支持的服务。	SNS

表A.1 – （续）

字节值	负面响应码（NRC）的定义	助记符
0x12	**子functionNotSupported** 此NRC表示请求的操作不会被采用，因为服务器不支持请求消息的服务特定参数。 服务器应发送此NRC，以防客户端发送带有已知和支持的服务标识符但具有未知或不受支持的“子功能”的请求消息。	单频网
0x13	**incorrectMessageLengthOrInvalidFormat** 此NRC表示请求的操作不会被执行，因为接收到的请求消息的长度与指定服务的规定长度不匹配，或者参数的格式与指定服务的规定格式不匹配。	IMLOIF
0x14	**responseTooLong** 如果要生成的响应超过底层网络层可用的最大字节数，则应由服务器报告此NRC。 如果响应消息超过底层传输协议允许的最大大小，或者响应消息超出为此目的分配的服务器缓冲区大小，则可能会发生这种情况。 示例当请求多个DID并且响应中所有DID的组合超出了基础传输协议的限制时，可能会发生此问题。	RTL
0x15 – 0x20	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x21	**busyRepeatRequest** 此NRC表示服务器暂时无法执行请求的操作。 在这种情况下，客户端应该执行“相同的请求消息”或“另一个请求消息”的重复。 请求的重复应延迟相应实施文件规定的时间。 示例在多客户端环境中，一个客户端的诊断请求可能会暂时被NRC 0x21阻止，而另一个客户端则完成诊断任务。 如果服务器能够执行诊断任务但需要额外的时间来完成任务并准备响应，则应使用NRC 0x78而不是NRC 0x21。 该NRC通常由每个诊断服务提供支持，因为数据链接特定实施文档中未另行说明，因此它未列入诊断服务的适用响应代码列表中。	BRR
0x22	**conditionsNotCorrect** 此NRC表示由于未满足服务器先决条件而不会执行请求的操作。	CNC
0x23	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD

表A.1 – （续）

字节值	负面响应码（NRC）的定义	助记符
0x24	**requestSequenceError** 此NRC表示请求的操作不会被采用，因为服务器预计客户端发送的请求消息或消息的顺序不同。 当顺序敏感的请求以错误的顺序发出时，可能会发生这种情况。 示例成功的SecurityAccess服务在请求消息中指定一系列requestSeed和sendKey作为子功能。 如果客户端发送的序列不同，则服务器将发送否定响应消息，其中包含否定响应代码0x24 requestSequenceError。	RSE
0x25	**noResponseFromSubnetComponent** 此NRC表示服务器已收到请求，但请求的操作无法由服务器执行，因为提供所请求信息所需的子网组件在指定时间内没有响应。 noResponseFromSubnetComponent否定响应应由电子系统中的网关实施，该系统包含电子子网组件，并且不直接响应客户的请求。 网关可能会收到对子网组件的请求，然后从子网组件请求必要的信息。 如果子网组件未能响应，则服务器应使用此否定响应来通知客户有关子网组件的故障。 该NRC通常由每个诊断服务提供支持，因为数据链接特定实施文档中未另行说明，因此它未列入诊断服务的适用响应代码列表中。	NRFSC
0x26	**FailurePreventsExecutionOfRequestedAction** 此NRC表示请求的操作不会被执行，因为由DTC识别的故障条件（至少有一个TestFailed，Pending，Confirmed或TestFailedSinceLastClear的DTC状态位设置为1）已发生，并且此故障条件阻止服务器执行请求的操作。 例如，NRC可以指导技术人员阅读DTC以识别和解决问题。 注意这意味着用于访问DTC的诊断服务不应该实现这个NRC，因为外部测试工具可能会检查上述NRC，并在收到上述NRC时自动请求DTC。 除了数据链接特定实施文档中另有规定外，该NRC通常由每个诊断服务（上述服务除外）提供支持，因此它未列入诊断服务的适用响应代码列表中。	FPEORA
0x27 – 0x30	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x31	**requestOutOfRange** 此NRC表示请求的操作不会被采用，因为服务器检测到请求消息中包含一个试图替换超出其权限范围的值的参数（例如，当仅定义数据时尝试替换111的数据字节到100），或试图访问活动会话中不支持或不支持的dataIdentifier / routineIdentifer。 对于所有服务都应该实施NRC，这允许客户端通过服务器中的数据读取数据，写入数据或调整功能。	ROOR

表A.1 – （续）

字节值	负面响应码（NRC）的定义	助记符
0x32	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x33	**securityAccessDenied** 此NRC表示请求的操作不会被采用，因为服务器的安全策略尚未被客户满足。 如果发生以下情况之一，服务器应发送此NRC： — 服务器的测试条件不符合， — 该 需要 信息 序列 例如 DiagnosticSessionControl，securityAccess不符合， — 客户端发送了需要解锁服务器的请求消息。 除了强制使用本标准中适用服务中规定的否定响应代码之外，此否定响应代码还可用于任何需要安全且尚未授予执行所需服务的情况。	伤心
0x34	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x35	**无效的密钥** 此NRC表示服务器未授予安全访问权限，因为客户端发送的密钥与服务器内存中的密钥不匹配。 这被视为获得安全的尝试。 服务器应保持锁定状态并增加内部安全性AccessFailed计数器。	IK
0x36	**exceedNumberOfAttempts** 此NRC表示所请求的操作不会被采用，因为客户端尝试获取安全访问次数的次数超过了服务器的安全策略允许的次数。	ENOA
0x37	**requiredTimeDelayNotExpired** 此NRC表示将不会执行请求的操作，因为客户端的最新获得安全访问的尝试是在服务器的所需超时期限过去之前启动的。	RTDNE
0x38 – 0x4F	**reservedByExtendedDataLinkSecurityDocument** 扩展数据链接安全性保留了该值的范围。	RBEDLSD
0x50 – 0x6F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x70	**uploadDownloadNotAccepted** 此NRC表示尝试上传/下载到服务器的内存不能完成由于某些故障条件。	UDNA
0x71	**transferDataSuspended** 该NRC表示数据传输操作由于某种故障而中断。 激活的transferData序列应被中止。	TDS
0x72	**generalProgrammingFailure** 此NRC表示服务器在擦除或编程永久性存储器设备（例如闪存）中的存储器位置时检测到错误。	GPF

表A.1 – （续）

字节值	负面响应码（NRC）的定义	助记符
0x73	**wrongBlockSequenceCounter** 此NRC指示服务器在blockSequenceCounter值序列中检测到错误。 请注意，重复传输数据请求消息的blockSequenceCounter等于前一个TransferData请求消息中包含的请求消息应被服务器接受。	WBSC
0x74 – 0x77	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x78	**requestCorrectlyReceived-ResponsePending** 此NRC表示请求消息已正确接收，并且请求消息中的所有参数都是有效的，但要执行的操作尚未完成，并且服务器尚未准备好接收其他请求。 一旦所请求的服务完成，服务器应发送肯定响应消息或否定响应消息，其响应代码与此不同。 该NRC的否定响应消息可以由服务器重复，直到请求的服务完成并发送最终的响应消息。 此NRC可能会影响应用程序层计时参数值。 详细规范应包含在数据链接特定实现文档中。 如果服务器在完成所请求的诊断服务时无法从客户端收到进一步的请求消息，则只能在否定响应消息中使用此NRC。 使用此NRC时，服务器应始终发送一个最终响应（正数或负数），与suppressPosRspMsgIndicationBit值无关，或者针对功能性寻址请求上的NRC SNS，SFNS，SNSIAS，SFNSIAS和ROOR的响应抑制要求。 这种NRC可以使用的典型例子是当客户端发送请求消息时，其中包括要在服务器的闪存中编程或擦除的数据。 如果编程/擦除例程（通常在RAM外执行）在写入闪存时不能支持串行通信，服务器将发送一个否定响应消息和该响应代码。 该NRC通常由每个诊断服务提供支持，因为数据链接特定实施文档中未另行说明，因此它未列入诊断服务的适用响应代码列表中。	RCRRP
0x79 – 0x7D	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x7E	**子functionNotSupportedInActiveSession** 此NRC表示将不会执行请求的操作，因为服务器在当前活动的会话中不支持请求的子功能。 只有当已知请求的子功能在另一个会话中受支持时才能使用该NRC，否则应使用响应码SFNS（子功能不支持）（例如，执行引导软件的服务器通常不知道哪些子功能支持应用程序（反之亦然），因此可能需要使用NRC 0x12进行响应）。 如果在数据链路专用实施文档中没有另外说明，则每个诊断服务都应该使用子功能参数来支持该NRC，因此它不在诊断服务的适用响应代码列表中列出。	SFNSIAS

表A.1 – （续）

字节值	负面响应码（NRC）的定义	助记符
0x7F	**serviceNotSupportedInActiveSession** 此NRC表示请求的操作不会被执行，因为服务器在当前活动的会话中不支持所请求的服务。 只有当已知请求的服务在另一个会话中受支持时才能使用此NRC，否则应使用响应代码SNS（serviceNotSupported）（例如，执行引导软件的服务器通常不知道应用程序支持哪些服务（以及反之亦然），因此可能需要用NRC 0x11进行响应）。 该NRC通常由每个诊断服务提供支持，因为数据链接特定实施文档中未另行说明，因此它未列入诊断服务的适用响应代码列表中。	SNSIAS
0x80	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x81	**rpmTooHigh** 此NRC表示由于RPM的服务器必备条件未满足（当前RPM高于预编程的最大阈值），所请求的操作不会被执行。	RPMTH
0x82	**rpmTooLow** 此NRC表示由于RPM的服务器必备条件未满足（当前RPM低于预编程的最小阈值），因此不会执行请求的操作。	RPMTL
0x83	**engineIsRunning** 对于那些在发动机运转时无法启动的执行器测试，这个NRC是必需的。 这不同于RPM太高的负面响应，并且需要被允许。	EIR
0x84	**engineIsNotRunning** 除非发动机正在运转，否则这些执行器测试必须执行此NRC。 这不同于RPM太低的负面响应，并且需要被允许。	EINR
0x85	**engineRunTimeTooLow** 此NRC表示由于发动机运行时间的服务器必备条件未达到（当前发动机运行时间低于预先编程的限制），因此不会执行所请求的操作。	ERTTL
0x86	**temperatureTooHigh** 此NRC表示由于温度的服务器必备条件未得到满足（当前温度高于预编程的最大阈值），所请求的操作不会被采用。	TEMPTH
0x87	**temperatureTooLow** 此NRC表示不会执行请求的操作，因为温度的服务器必备条件未得到满足（当前温度低于预编程的最小阈值）。	TEMPTL
0x88	**vehicleSpeedTooHigh** 该NRC表示由于车速的服务器前提条件未得到满足（当前VS高于预编程的最大阈值），所请求的动作将不会被采用。	VSTH

表A.1 – （续）

字节值	负面响应码（NRC）的定义	助记符
0x89	**vehicleSpeedTooLow** 该NRC表示由于不符合车速的服务器前提条件（当前VS低于预编程的最小阈值），所请求的动作不会被采用。	VSTL
0x8A	油门/ PedalTooHigh 此NRC表示由于油门/踏板位置的服务器前提条件未满足（当前TP / APP高于预编程的最大阈值），因此不会执行所请求的操作。	TPTH
0x8B	油门/ PedalTooLow 此NRC表示由于油门/踏板位置的服务器必备条件未达到（当前TP / APP低于预编程的最小阈值），因此不会执行所请求的操作。	TPTL
0x8C	**transmissionRangeNotInNeutral** 此NRC表示不会执行请求的操作，因为处于空档的服务器必备条件未满足（当前变速器档位不是空档）。	TRNIN
0x8D	**transmissionRangeNotInGear** 此NRC表示由于未满足服务器必备条件（当前变速器档位不正常）而不会执行所请求的操作。	TRNIG
0x8E	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD
0x8F	brakeSwitch（es）NotClosed（制动踏板未按下或未应用） NRC表明，出于安全原因，这是某些测试开始前所必需的，并且必须在整个测试过程中保持不变。	BSNC
0x90	**shifterLeverNotInPark** NRC表明，出于安全原因，这是某些测试开始前所必需的，并且必须在整个测试过程中保持不变。	SLNIP
0x91	**torqueConverterClutchLocked** 此NRC表示由于变矩器离合器的服务器前提条件未满足（当前TCC状态高于预先设定的极限值或锁定），因此不会执行所请求的操作。	TCCL
0x92	**voltageTooHigh** 此NRC表示由于服务器（ECU）主引脚电压的服务器必备条件未满足（当前电压高于预编程的最大阈值），因此不会执行所请求的操作。	VTH
0x93	**voltageTooLow** 此NRC表示由于服务器（ECU）主引脚电压的服务器必备条件未满足（当前电压低于预编程的最大阈值），因此不会执行所请求的操作。	VTL
0x94 – 0xEF	**reservedForSpecificConditionsNotCorrect** 这个范围的值由本文件保留以备未来定义。	RFSCNC
0xF0 – 0xFE	**vehicleManufacturerSpecificConditionsNotCorrect** 这个值的范围是保留给车辆制造商的具体情况不正确的情况。	VMSCNC

表A.1 – （续）

字节值	负面响应码（NRC）的定义	助记符
为0xFF	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	ISOSAERESRVD

附件B
(规范性附录)

诊断和通信管理功能单元数据参数定义

B.1 通信类型参数定义

communicationType是一个1字节的值。 该字节的位编码的低半字节代表通信类型，可以通过CommunicationControl（0x28）服务进行控制。 例如，具有“11b”的比特组合（比特1-0）的通信类型是有效的并且禁用“正常通信消息”和“网络管理通信消息”消息。
communicationType 1字节值的高半字节定义了在接收到适当的CommunicationControl服务时，应该禁用/启用连接到接收节点的哪个子网。

表B.1定义了communicationType和subnetNumber字节。

表B.1 – 通信类型和子网号码字节的定义

位的编码	值	描述	Cvt	助记符
	0x0	**ISOSAEReserved**	M	
	0x1	**normalCommunicationMessages** 该值引用所有与应用程序相关的通信（多个车载服务器之间的交互信号交换）。	U	NCM
	0x2	**networkManagementCommunicationMessages** 该值引用所有与网络管理相关的通信。	U	NWMCM
	0x3	**networkManagementCommunicationMessages和 normalCommunicationMessages** 该值引用所有网络管理和与应用程序相关的通信。	U	NWMCM, NCM
2 – 3	0x0 – 0x3	**ISOSAEReserved**	M	ISOSAERESRVD
	0x0	**禁用/启用指定的通讯类型** 参见位0-1的编码。 在包含与所有连接网络通信的接收节点中。 这只会禁止节点进入连接网络的通信，而不会禁止网络中其他节点的通信（即接收节点不负责禁用网络中每个节点的通信）。	U	DISENSCT
	0x1 – 0xE	**禁用/启用由子网号码标识的特定子网**	U	DISENSSIVSN
	0xF	**禁用/启用接收请求的网络（接收节点（服务器））**	U	DENWRIRO

B.2 eventWindowTime参数定义

表B.2定义了eventWindowTime参数值。

表B.2 – eventWindowTime参数值的定义

字节值	描述	Cvt	助记符
0x00 – 0x01	**ISOSAEReserved** 该值由文档保留	M	ISOSAERESRVD
0x02	**infiniteTimeToResponse** 该值指定事件窗口应在无限的时间内保持活动状态（例如打开窗口直至关闭电源）。	U	ITTR
0x03 – 0x7F	**vehicleManufacturerSpecific** 这个值的范围保留给车辆制造商特定用途。 eventWindowTime参数的解析度由车辆制造商自行决定。	U	VMS
0x80 – 0xFF	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD

B.3 linkControlModeIdentifier参数定义

表B.3定义了linkControlModeIdentifier值。

表B.3 - linkControlModeIdentifier值的定义

字节值	描述	Cvt	助记符
0x00	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
0x01	**PC9600Baud** 此值指定9.6 KBaud的标准PC波特率。	U	PC9600
0x02	**PC19200Baud** 该值指定19.2 KBaud的标准PC波特率。	U	PC19200
0x03	**PC38400Baud** 该值指定38.4 KBaud的标准PC波特率。	U	PC38400
0x04	**PC57600Baud** 此值指定标准PC波特率为57.6 KBaud。	U	PC57600
0x05	**PC115200Baud** 该值指定115.2 KBaud的标准PC波特率。	U	PC115200
0x06 – 0x0F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD
0x10	**CAN125000Baud** 该值指定了125 KBaud的标准CAN波特率。	U	CAN125000
0x11	**CAN250000Baud** 该值指定250 KBaud的标准CAN波特率。	U	CAN250000
0x12	**CAN500000Baud** 该值指定500 KBaud的标准CAN波特率。	U	CAN500000
0x13	**CAN1000000Baud** 该值指定1 MBaud的标准CAN波特率。	U	CAN1000000
0x14 – 0x1F	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD
0x20	**ProgrammingSetup** 该值指定了网络的编程设置，可根据车辆网络要求进行参数设置。	U	PROGSU
0x21- 0xFF	**ISOSAEReserved** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD

B.4 nodeIdentificationNumber参数定义

nodeIdentificationNumber是一个2字节的值，它表示连接到车辆网络某处的节点的唯一标识号，其中同一节点可以连接到不同车线中的不同网络（例如，具有唯一节点地址的LIN节点是在一个模型中连接到网络A，而同一个节点连接到不同模型中的网络B）。 因此，nodeIdentificationNumber提供了一种机制，远程节点所连接的相关主节点将相关网络转换为某种诊断模式（例如，禁用LIN网络上的正常通信）。 只有检测到由nodeIdentificationNumber标识的相关节点连接的关联主节点才能执行请求的通信控制服务。

注意　　此参数仅在controlType值设置为0x04或0x05时可用。 个别参数将由车辆制造商定义。

表B.4定义了nodeIdentificationNumber值。

表B.4 - 定义nodeIdentificationNumber值

字节值	描述	Cvt	助记符
0x0000	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
0x0001 - 0xFFFF	**nodeIdentificationNumber** 这些值标识连接在车辆某处的总线系统上的节点。 只有在有效数字的情况下，接收ECU才能执行请求CommunicationControl功能。	U	NIN

附件C
(规范性附录)

数据传输功能单元数据参数定义

C.1 DID参数定义

参数dataIdentifier (DID) 逻辑上表示一个对象 (例如进气门位置) 或对象集合。 该参数应在服务器的内存中可用。 dataIdentifier值应存在固定内存中, 或者临时存储在RAM中, 如果由服务的dynamicDefineDataIdentifier动态定义的话。 通常, dataIdentifier能够用于许多诊断服务请求, 包括0x22 (readDataByIdentifier), 0x2E (writeDataByIdentifier) 和0x2F (inputOutputControlByIdentifier)。 dataIdentifier也用于各种诊断服务响应 (例如, 对服务0x19子功能readDTCSnapshotRecordByDTCNumber的肯定响应)。

重要 - 无论使用dataIdentifier与哪种服务一起使用, 它应始终在给定的ECU上代表同一事物 (即具有给定大小/含义/的给定对象)。

唯一不适用的情况是动态定义的dataIdentifiers, 因为它们不是在ECU中预定义的, 而是由客户端使用服务0x2C (dynamicsDefineDataIdentifier) 定义的。 DataIdentifier值在表C.1中定义。

表C.1 - DID数据参数定义

字节值	描述	Cvt	助记符
0x0000 - 0x00FF	**ISOSAEReserved** 这个值的范围应该被这个文件保留以备将来的定义。	M	ISOSAERESRVD
0x0100 - 0xA5FF	**VehicleManufacturerSpecific** 该值范围应用于在服务器内引用车辆制造商特定的记录数据标识符和输入/输出标识符。	U	VMS
0xA600 - 0xA7FF	**ReservedForLegislativeUse** 这个价值范围是为将来的立法要求而保留的。	M	RFLU
0xA800 - 0xACFF	**VehicleManufacturerSpecific** 该值范围应用于在服务器内引用车辆制造商特定的记录数据标识符和输入/输出标识符。	U	VMS
0xAD00 - 0xAFFF	**ReservedForLegislativeUse** 这个价值范围是为将来的立法要求而保留的。	M	RFLU
0xB000 - 0xB1FF	**VehicleManufacturerSpecific** 该值范围应用于在服务器内引用车辆制造商特定的记录数据标识符和输入/输出标识符。	U	VMS
0xB200 - 0xBFFF	**ReservedForLegislativeUse** 这个价值范围是为将来的立法要求而保留的。	M	RFLU

表C.1 – （续）

字节值	描述	Cvt	助记符
0xC000 – 0xC2FF	**VehicleManufacturerSpecific** 该值范围应用于在服务器内引用车辆制造商特定的记录数据标识符和输入/输出标识符。	U	VMS
0xC300 – 0xCEFF	**ReservedForLegislativeUse** 这个价值范围是为将来的立法要求而保留的。	M	RFLU
0xCF00 – 0xEFFF	**VehicleManufacturerSpecific** 该值范围应用于在服务器内引用车辆制造商特定的记录数据标识符和输入/输出标识符。	U	VMS
0xF000 – 0xF00F	**networkConfigurationDataForTractorTrailerApplicationData标识符** 该值应用于请求所有拖车系统的远程地址，而不受其功能的影响。	U	NCDFTTADID
0xF010 – 0xF0FF	**vehicleManufacturerSpecific** 该值范围应用于在服务器内引用车辆制造商特定的记录数据标识符和输入/输出标识符。	U	VMS
0xF100 – 0xF17F	**identificationOptionVehicleManufacturerSpecificDataIdentifier** 该值的范围应用于车辆制造商特定的服务器/车辆识别选项。	U	IDOPTVMSDID
0xF180	**BootSoftwareIdentificationDataIdentifier** 该值应用于参考车辆制造商特定的ECU启动软件标识记录。 记录数据的第一个数据字节应该是所报告的数字模块。 在numberOfModules之后，报告引导软件标识。 引导软件识别结构的格式应由ECU制定并由车辆制造商定义。	U	BSIDID
0xF181	**applicationSoftwareIdentificationDataIdentifier** 该值应用于参考车辆制造商特定的ECU应用软件编号。 记录数据的第一个数据字节应为所报告的数字模块。 在numberOfModules之后，报告应用程序软件标识。 应用软件识别结构的格式应由ECU制定并由车辆制造商定义。	U	ASIDID
0xF182	**applicationDataIdentificationDataIdentifier** 该值应用于参考车辆制造商特定的ECU应用数据识别记录。 记录数据的第一个数据字节应该是所报告的数字模块。 在numberOfModules之后，报告应用程序数据标识。 应用程序数据识别结构的格式应由ECU制定并由车辆制造商定义。	U	ADIDID
0xF183	**bootSoftwareFingerprintDataIdentifier** 该值应用于参考车辆制造商特定的ECU启动软件指纹识别记录。 记录数据内容和格式应由ECU制定并由车辆制造商定义。	U	BSFPDID

表C.1 – （续）

字节值	描述	Cvt	助记符
0xF184	**applicationSoftwareFingerprintDataIdentifier** 该值应用于参考车辆制造商特定的ECU应用软件指纹识别记录。 记录数据内容和格式应由ECU制定并由车辆制造商定义。	U	ASFPDID
0xF185	**applicationDataFingerprintDataIdentifier** 该值应用于参考车辆制造商特定的ECU应用数据指纹识别记录。 记录数据内容和格式应由ECU制定并由车辆制造商定义。	U	ADFPDID
0xF186	**ActiveDiagnosticSessionDataIdentifier** 该值应用于报告服务器中的活动诊断会话。 这些值由DiagnosticSessionControl服务中的diagnosticSessionType子函数参数定义。	U	ADSDID
0xF187	**vehicleManufacturerSparePartNumberDataIdentifier** 该值应用于参考车辆制造商备件号。 记录数据内容和格式应该是服务器特定的并由车辆制造商定义。	U	VMSPNDID
0xF188	**vehicleManufacturerECUSoftwareNumberDataIdentifier** 该值应用于参考车辆制造商的ECU（服务器）软件编号。 记录数据内容和格式应该是服务器特定的并由车辆制造商定义。	U	VMECUSNDID
0xF189	**vehicleManufacturerECUSoftwareVersionNumberDataIdentifier** 该值应用于参考车辆制造商的ECU（服务器）软件版本号。 记录数据内容和格式应该是服务器特定的并由车辆制造商定义。	U	VMECUSVNDID
0xF18A	**systemSupplierIdentifierDataIdentifier** 该值应用于引用系统供应商名称和地址信息。 记录数据内容和格式应由服务器专用并由系统供应商定义。	U	SSIDDID
0xF18B	**ECUManufacturingDateDataIdentifier** 该值应用于引用ECU（服务器）制造日期。 记录数据内容和格式应为无符号数字，ASCII或BCD，并应按年，月，日排序。	U	ECUMDDID
0xF18C	**ECUSerialNumberDataIdentifier** 这个值应该用来引用ECU（服务器）的序列号。 记录数据内容和格式应该是服务器特定的。	U	ECUSNDID
0xF18D	**supportedFunctionalUnitsDataIdentifier** 这个值应该用来请求服务器中实现的功能单元。	U	SFUDID
0xF18E	**VehicleManufacturerKitAssemblyPartNumberDataIdentifier** 当备件编号仅指定服务器（例如售后服务）时，此值应用于参考工具包的车辆制造商订单编号（作为生产整体购买的组装零件，例如驾驶舱）。 记录数据内容和格式应该是服务器特定的并由车辆制造商定义。	U	VMKAPNDID

表C.1 – （续）

字节值	描述	Cvt	助记符
0xF18F	**ISOSAEReservedStandardized** 这个范围的值应该被这个文件保留，以便将来定义标准化的服务器/车辆识别选项。	M	ISOSAERESRVD
0xF190	**VINDataIdentifier** 该值应用于参考VIN号码。 记录数据内容和格式应由车辆制造商指定。	U	VINDID
0xF191	**vehicleManufacturerECUHardwareNumberDataIdentifier** 读取服务时应使用该值，以引用车辆制造商特定的ECU（服务器）硬件编号。 记录数据内容和格式应该是服务器特定的并由车辆制造商定义。	U	VMECUHNDID
0xF192	**systemSupplierECUHardwareNumberDataIdentifier** 该值应用于引用系统供应商特定的ECU（服务器）硬件编号。 记录数据内容和格式应由服务器专用并由系统供应商定义。	U	SSECUHWNDID
0xF193	**systemSupplierECUHardwareVersionNumberDataIdentifier** 该值应用于引用系统供应商特定的ECU（服务器）硬件版本号。 记录数据内容和格式应由服务器专用并由系统供应商定义。	U	SSECUHWVNDID
0xF194	**systemSupplierECUSoftwareNumberDataIdentifier** 该值应用于引用系统供应商特定的ECU（服务器）软件编号。 记录数据内容和格式应由服务器专用并由系统供应商定义。	U	SSECUSWNDID
0xF195	**systemSupplierECUSoftwareVersionNumberDataIdentifier** 该值应用于引用系统供应商特定的ECU（服务器）软件版本号。 记录数据内容和格式应由服务器专用并由系统供应商定义。	U	SSECUSWVNDID
0xF196	**exhaustRegulationOrTypeApprovalNumberDataIdentifier** 该数值应用于排气管理或型式认证编号（适用于需要型式认证的系统）。记录数据内容和格式应该是服务器特定的并由车辆制造商定义。 有关任何适用要求，请参阅相关法规。	U	EROTANDID
0xF197	**systemNameOrEngineTypeDataIdentifier** 该值应用于引用系统名称或引擎类型。 记录数据内容和格式应该是服务器特定的并由车辆制造商定义。	U	SNOETDID
0xF198	**repairShopCodeOrTesterSerialNumberDataIdentifier** 该值应用于引用维修店代码或测试人员（客户端）的序列号（例如，指示最近使用的服务客户端使用的重新程序服务器内存）。 记录数据内容和格式应该是服务器特定的并由车辆制造商定义。	U	RSCOTSNDID
0xF199	**programmingDateDataIdentifier** 该值应用于引用服务器上次编程的日期。 记录数据内容和格式应为无符号数字，ASCII或BCD，并应按年，月，日排序。	U	PDDID

表C.1 – （续）

字节值	描述	Cvt	助记符
0xF19A	**calibrationRepairShopCodeOrCalibrationEquipmentSerialNumber DataIdentifier** 该值应用于引用维修店代码或客户端序列号（例如，指示客户端用来重新校准服务器的最新服务）。 记录数据内容和格式应该是服务器特定的并由车辆制造商定义。	U	CRSCOCESNDID
0xF19B	**calibrationDateDataIdentifier** 该值应用于引用服务器上次校准的日期。 记录数据内容和格式应为无符号数字，ASCII或BCD，并应按年，月，日排序。	U	CDDID
0xF19C	**calibrationEquipmentSoftwareNumberDataIdentifier** 该值应用于引用用于校准服务器的客户端内的软件版本。 记录数据内容和格式应该是服务器特定的并由车辆制造商定义。	U	CESWNDID
0xF19D	**ECUInstallationDateDataIdentifier** 这个值应该被用来引用ECU（服务器）安装在车辆上的日期。 记录数据内容和格式应为无符号数字，ASCII或BCD，并应按年，月，日排序。	U	EIDDID
0xF19E	**ODXFileDataIdentifier** 此值应用于引用服务器的ODX（开放式诊断数据交换）文件，用于解释和缩放服务器数据。	U	ODXFDID
0xF19F	**EntityDataIdentifier** 这个值应该被用来引用实体数据标识符来进行安全的数据传输。	U	EDID
0xF1A0 – 0xF1EF	**identificationOptionVehicleManufacturerSpecific** 该值的范围应用于车辆制造商特定的服务器/车辆识别选项。	U	IDOPTVMS
0xF1F0 – 0xF1FF	**identificationOptionSystemSupplierSpecific** 该值的范围应用于系统供应商特定的服务器/车辆系统识别选项。	U	IDOPTSSS
0xF200 – 0xF2FF	**periodicDataIdentifier** 这个值的范围应该用来引用定期的记录数据标识符。 这些可以是静态或动态定义的。	U	PDID
0xF300 – 0xF3FF	**DynamicallyDefinedDataIdentifier** 这个值的范围应该用于dynamicDefinedDataIdentifiers。	U	DDDDI
0xF400 – 0xF4FF	**OBDDataIdentifier** 这个值范围被保留用于ISO 15031-5中定义的OBD / EOBD PID。	M	OBDDID
0xF500 – 0xF5FF	**OBDDataIdentifier** 这个值的范围被保留来代表将来定义的OBD / EOBD PID。	M	OBDDID

表C.1 – （续）

字节值	描述	Cvt	助记符
0xF600 – 0xF6FF	**OBDMonitorDataIdentifier** 该值的范围保留用于ISO 15031-5中定义的OBD / EOBD车载监控结果值。	M	OBDMDID
0xF700 – 0xF7FF	**OBDMonitorDataIdentifier** 这个值的范围被保留以代表将来定义的OBD / EOBD车载监控结果值。	M	OBDMDID
0xF800 – 0xF8FF	**OBDInfoTypeDataIdentifier** 该值范围保留给ISO 15031-5中定义的OBD / EOBD信息类型值。	M	OBDINFTYPDID
0xF900 – 0xF9FF	**TachographDataIdentifier** 该值的范围为ISO 16844-7中定义的行驶记录仪DID保留。	M	TACHODID
0xFA00 – 0xFA0F	**AirbagDeploymentDataIdentifier** 这个数值范围保留给在ISO 26021-2中定义的机载烟火装置的使用寿命终止。	M	ADDID
0xFA10	**NumberOfEDRDevices** 该值应用于报告能够报告EDR数据的EDR设备的数量。	U	NOEDRD
0xFA11	**EDRIdentification** 该值应用于报告EDR识别数据。	U	EDRI
0xFA12	**EDRDeviceAddressInformation** 该值应用于根据ISO 26021-2中为dataIdentifier 0xFA02定义的格式报告EDR设备地址信息。	U	EDRDAI
0xFA13 – 0xFA18	**EDREntries** 该范围应用于报告单个EDR条目。 每个DID应代表一个EDR条目，其中0xFA13表示最新的EDR条目。	U	EDRES
0xFA19 – 0xFAFF	**SafetySystemDataIdentifier** 这个值的范围被保留来表示安全系统相关的DID。	M	SSDID
0xFB00 – 0xFCFF	**ReservedForLegislativeUse** 这个价值范围是为将来的立法要求而保留的。	M	RFLU
0xFD00 – 0xFEFF	**SystemSupplierSpecific** 这个值的范围应该用来引用服务器内系统供应商特定的记录数据标识符和输入/输出标识符。	U	SSS
为0xFF00	**UDSVersionDataIdentifier** 该值应用于引用在服务器中实现的UDS版本。 有关该DID的缩放情况，请参阅表C.11。	U	UDSVDID
0xFF01 – 0xFFFF	**ISOSAEReserved** 这个值的范围应该被这个文件保留以备将来的定义。	M	ISOSAERESRVD

C.2 scalingByte参数定义

参数scalingByte （SBYT）由一个字节（高低半字节）组成。 scalingByte高半字节定义了用于表示dataIdentifier（DID）的数据类型。 scalingByte低半字节定义了用于表示数据流中参数的字节数。

表C.2定义了scalingByte（高半字节）参数。

表C.2 – scalingByte（高半字节）参数定义

高半字节的编码	数据类型的描述	Cvt	助记符
0x0	**unSignedNumeric（1到4个字节）** 该编码使用通用的二进制加权方案来表示通过离散增量步长的平均值。 一个字节提供256个步骤；两个字节产生65 536个步骤等。	U	USN
0x1	**签名数字（1到4个字节）** 该编码使用二进制补码二进制加权方案来表示通过离散增量步长的平均值。 一个字节提供256个步骤；两个字节产生65 536个步骤等。	U	SN
0x2	**bitMappedReportedWithOutMask** 位映射编码使用单个位或小组位来表示状态。 有效性掩码用于指示特定应用程序每个位的有效性。 BitMappedReportedWithOutMask编码表示有效性掩码不是参数定义本身的一部分。 需要单独的scalingByteExtension（见C.3.1）来报告有效性掩码。	U	BMRWOM
0x3	**bitMappedReportedWithMask** 位映射编码使用单个位或小组位来表示状态。 BitMappedReportedWithMask编码表示包含的有效性掩码作为参数定义本身的一部分。 对于表示状态的每一位，都需要相应的掩码位作为参数定义的一部分。 掩码表示特定应用程序的每个位的有效性。 这种类型的位映射参数对于表示数据的每个状态字节包含一个有效性掩码字节。 由于有效性掩码是参数定义的一部分，因此不需要单独的scalingByteExtension。	U	BMRWM
0x4	**BinaryCodedDecimal** 常规二进制编码的十进制编码用于表示每个字节的两个数字。 高半字节用于表示最高有效数字（0-9），低半字节用于表示最低有效数字（0-9）。	U	BCD
0x5	**stateEncodedVariable（1字节）** 此编码使用二进制加权方案来表示多达256个不同的状态。 一个示例是代表点火开关状态的参数。 代码“00”，“01”，“02”和“03”可分别指示点火关闭，锁定，运行和启动。 该表示总是限制为一个字节。	U	SEV
0x6	**ASCII（每个缩放字节1到15个字节）** 常规ASCII编码用于表示最多128个标准字符，MSB =逻辑'0'。 另外128个自定义字符可以用MSB =逻辑'1'表示。	U	ASCII
0x7	**signedFloatingPoint** 浮点编码用于需要以浮点或科学记数法表示的数据。 标准IEEE格式应按照ANSI / IEEE标准754-1985使用。	U	SFP

表C.2 - （续）

高半字节的编码	数据类型的描述	Cvt	助记符
0x8	**包** 数据包包含多个数据值，通常相关，每个数据值都有独特的缩放比例 缩放信息不包括在各个值中。 见C.3.1。	U	P
0x9	**式** 一个公式用于从原始数据计算一个值。 公式标识符在定义公式标识符编码的表中指定。 见C.3.2。	U	F
0xA	**单元/格式** 单位和格式用于以更加用户友好的格式显示数据。 单位和格式标识符在定义formulaIdentifier编码的表中指定。 注：如果使用组合的单位和/或格式，例如mV，则每个单位/格式的一个scalingByte（和scalingData）应包含在readScalingDataByIdentifier正反应中。 见C.3.3。	U	U
0xB	**stateAndConnectionType（1字节）** 该编码特别用于输入和输出信号。 编码在数据字节中的信息指定高级物理布局，电平和功能状态。 建议将此选项用于数字输入和输出参数。 见C.3.4。	U	SACT
0xC – 0xF	**ISOSAEReserved** 本文档保留以备将来定义。	M	ISOSAERESRVD

表C.3定义了scalingByte（低半字节）参数。

表C.3 - scalingByte（低半字节）参数定义

低半字节的编码	数据类型的描述	Cvt	助记符
0x0 – 0xF	**numberOfBytesOfParameter** 该值的范围指定由参数标识符引用的数据流中的数据字节数。 参数的长度由缩放字节（s）定义，其总是在参数标识符（一个或多个字节）之前。 如果多个缩放字节遵循参数标识符，则由参数标识符引用的数据的长度是缩放字节中低半字节的内容的总和。 例如，VIN由单字节参数标识符标识，后跟两个缩放字节。 长度最多可计算17个数据字节。 两个低位半字节的内容可以具有总计为17个数据字节的值的任意组合。 注意对于高位半字节的scalingByte编码为公式或单位/格式，此值为0x0。	U	NROBOP

C.3 scalingByteExtension参数定义

C.3.1 scalingByteExtension for scalingByte bitibpedReportedWithOutMask的高半字节

参数scalingByteExtension (SBYE) 仅支持scalingByte参数，其中高半字节编码为公式，单位/格式或bitMappedReportedWithOutMask。

具有高位半字节的scalingByte编码为bitMappedReportedWithOutMask后面应接着代表位映射数据标识符的有效性掩码的scalingByteExtension字节。 每个字节应指示当前应用程序支持相应dataIdentifier字节的哪些位。

表C.4定义了bitMappedReportedWithOutMask的scalingByteExtension。

表C.4 - bitMappedReportedWithOutMask的scalingByteExtension

字节值	描述	Cvt
#1	dataIdentifier dataRecord#1有效性掩码	M
:	:	C1
#p	dataIdentifier dataRecord#p有效性掩码	C1
C1: 此参数的存在取决于请求信息的dataIdentifier的大小。 有效性掩码应具有与dataIdentifier具有dataRecords一样多的字节。		

C.3.2 scalingByteExtension为scalingByte公式的高半字节

参数scalingByteExtension (SBYE) 仅支持scalingByte参数，其中高半字节编码为公式，单位/格式或bitMappedReportedWithOutMask。

一个以字节高度编码为公式的scalingByte应该跟定义该公式的scalingByteExtension字节。scalingByteExtension由一个字节的formulaIdentifier和常量组成，如下表所述。

表C.5定义了公式的scalingByteExtension字节。

表C.5 - scalingByteExtension公式的字节数

字节值	描述	Cvt
#1	formulaIdentifier (请参阅定义formulaIdentifier编码的表以了解详细信息)	M
#2	C0高字节	M
#3	C0低字节	M
#4	C1高字节	U
#5	C1低字节	U
:	:	U
#2n+2	Cn高字节	U
#2n+3	Cn低字节	U

表C.6定义了公式标识符编码。

表C.6 – 公式标识符编码

字节值	描述	Cvt
0x00	y = C0 * x + C1	U
0x01	y = C0 * (x + C1)	U
0x02	y = C0 / (x + C1) + C2	U
0x03	y = x / C0 + C1	U
0x04	y = (x + C0) / C1	U
0x05	y = (x + C0) / C1 + C2	U
0x06	y = C0 * x	U
0x07	y = x / C0	U
0x08	y = x + C0	U
0x09	y = x * C0 / C1	U
0x0A – 0x7F	ISO / SAE保留	M
0x80 – 0xFF	车辆制造商特定	U

公式使用变量（y，x等）和常量（C0，C1，C2等）定义。 变量y是计算值。 其他变量按连续顺序是dataIdentifier引用的数据流的一部分。 每个常量表示为表C.7中定义的一个两字节实数。 两个字节的实数（C = M * 10E）包含一个12位有符号（2的补码）尾数（M）和一个4位有符号（2的补码）指数（E）。 尾数可以保持在-2 048至+2 047范围内的值，并且指数可以将数字缩放10^{-8}至10^{7}。 指数编码在两字节实数的高位字节的高半字节中。 尾数编码在高字节的低半字节和两字节实数的完整低字节中。

表C.7 – 两字节实数格式

高字节								低字节							
高啃				低半字节				高啃				低半字节			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
指数				尾数											

C.3.3 scalingByteExtension for scalingByte单位/格式的高半字节

参数scalingByteExtension（SBYE）仅支持scalingByte参数，其中高半字节编码为公式，单位/格式或bitMappedReportedWithOutMask。

一个高位半字节编码为单位/格式的scalingByte应该跟随一个定义单位/格式的scalingByteExtension字节。表C.8中定义了一个字节scalingByteExtension。 如果使用组合的单位和/或格式，例如mV，则每个单位/格式应包含一个scalingByte（和scalingByteExtension）。

表C.8 – 单位/格式scalingByteExtension编码

ScalingByteExtension字节#1	名称	符号	描述	Cvt
0x00	没有单位，没有前缀	---	---	U
0x01	仪表	m	长度	U
0x02	脚丫子	ft	长度	U
0x03	英寸	in	长度	U
0x04	码	yd	长度	U
0x05	英里（英语）	mi	长度	U
0x06	公克	g	块	U
0x07	吨（公制）	t	块	U
0x08	第二	s	时间	U
0x09	分钟	分	时间	U
0x0A	小时	h	时间	U
0x0B	天	d	时间	U
0x0C	年	y	时间	U
0x0D	安培	A	当前	U
0x0E	伏特	V	电压	U
0x0F	库仑	C	电荷	U
0x10	欧姆	W	抵抗性	U
0x11	法拉	F	电容	U
0x12	亨利	H	电感	U
0x13	西门子	S	电导	U
0x14	韦伯	Wb	磁通量	U
0x15	特斯拉	T	磁通密度	U
0x16	开	K	热力学温度	U
0x17	摄氏	°C	热力学温度	U
0x18	飞轮海	°F	热力学温度	U
0x19	坎德拉	光盘	发光强度	U
0x1A	弧度	拉德	平面角度	U
0x1B	度	°	平面角度	U
0x1C	赫兹	赫兹	频率	U
0x1D	焦耳	J	能源	U
0x1E	牛顿	N	力	U
0x1F	千克力	kp	力	U
0x20	磅力	磅	力	U
0x21	瓦	W	功率	U
0x22	马力（公制）	香港	功率	U
0x23	马力（英国和美国）	生命值	功率	U

表C.8 - （续）

ScalingByteExtension字节#1	名称	符号	描述	Cvt
0x24	帕斯卡尔	霸	压力	U
0x25	酒吧	酒吧	压力	U
0x26	大气层	自动取款机	压力	U
0x27	磅每平方英寸的力量	psi	压力	U
0x28	becqerel	Bq	放射性	U
0x29	流明	lm	光通量	U
0x2A	勒克司	lx	照度	U
0x2B	升	l	卷	U
0x2C	加仑（英国）	---	卷	U
0x2D	加仑（美国液化气）	---	卷	U
0x2E	立方英寸	cu in	卷	U
0x2F	米每秒	女士	速度	U
0x30	公里每小时	公里/小时	速度	U
0x31	英里每小时	英里	速度	U
0x32	每秒革命	rps	角速度	U
0x33	每分钟转数	转	角速度	U
0x34	计数	---	---	U
0x35	百分	%	---	U
0x36	毫克每搏	毫克/冲程	每发动机行程质量	U
0x37	米每平方秒	米/ SP^2p	促进	U
0x38	牛顿米	纳米	时刻（例如扭转时刻）	U
0x39	升每分钟	升/分钟	流	U
0x3A	瓦每平方米瓦/平方米强度	W / MP^2	强度	U
0x3B	酒吧每秒	酒吧/秒	压力变化	U
0x3C	每秒弧度	弧度/秒	角速度	U
0x3D	每平方秒的弧度	弧度/ SP^2p	角加速度	U
0x3E	千克每平方米	公斤/ MP^2p	---	U
0x3F	---	---	由文件保留	M
0x40	exa（前缀）	E	1018	U
0x41	皮塔（前缀）	P	1015	U
0x42	tera（前缀）	T	1012	U
0x43	千兆（前缀）	G	109	U
0x44	兆（前缀）	M	106	U
0x45	公斤（前缀）	k	103	U
0x46	hecto（前缀）	h	102	U

0x47	deca（前缀）	da	10	U

表C.8 – （续）

ScalingByteExtension字节#1	名称	符号	描述	Cvt
0x48	deci（前缀）	d	10-1	U
0x49	centi（前缀）	c	10-2	U
0x4A	毫（前缀）	m	10-3	U
0x4B	微（前缀）	m	10-6	U
0x4C	纳米（前缀）	n	10-9	U
0x4D	微微（前缀）	p	10-12	U
0x4E	毫微微（前缀）	f	10-15	U
0x4F	atto（前缀）	a	10-18	U
0x50	日期1	-	年月日	U
0x51	Date2	-	日月年	U
0x52	Date3	-	月日年	U
0x53	周	W	日历周	U
0x54	时间1	---	UTC时/分/秒	U
0x55	时间2	---	小时/分/秒	U
0x56	DateAndTime1	---	其次/分钟/小时/日/月/年	U
0x57	DateAndTime2	---	秒/分/小时/日/月/年/本地分钟偏移/本地小时偏移	U
0x58	DateAndTime3	---	秒/分/小时/月/日/年	U
0x59	DateAndTime4	---	秒/分/小时/月/日/年/本地分钟偏移/本地小时偏移	U
0x5A – 0xFF	---	---	ISO / SAE保留	M

C.3.4 scalingByteExtension为scalingByte stateAndConnectionType的高半位元组

具有高字节编码为stateAndConnectionType的scalingByte后面应跟随一个定义stateAndConnectionType的scalingByteExtension字节。 表C.9中定义了一个字节scalingByteExtension。 stateAndConnectionType编码专门用于输入和输出信号。 在scalingByteExtension数据字节中编码的是有关物理布局，电气水平和功能状态的信息。

表C.9 – 对scaleByte stateAndConnectionType的高半字节进行编码

位编码	值	与输入信号一起使用	与输出信号一起使用
0x0 – 0x2	0	状态：未激活	状态：未激活
	1	状态：有效，功能1	状态：有效，功能1
	2	状态：检测到错误	状态：检测到可信性错误
	3	状态：不可用	状态：不可用
	4	状态：激活，功能2（仅与3个状态组合）	状态：激活，功能2（仅与3个状态组合）
	5 – 7	保留的	保留的
0x3 – 0x4	0	低电平信号（地）	低电平信号（地）
	1	中层信号（地面与+之间）	中层信号（地面和地面之间） +)
	2	高电平信号（+）	高电平信号（+）
	3	由文件保留	由文件保留
0x5	0	输入信号	没有定义的
	1	没有定义的	输出信号
0x6 – 0x7	0	内部信号或通过CAN接口不能专门提供	内部信号或通过CAN专用于ECU连接器
	1	下拉电阻输入类型（2个状态）	低侧开关（2个状态）
	2	上拉电阻输入类型（2个状态）	高侧开关（2个状态）
	3	上拉和下拉电阻输入类型（3个状态）	低侧和高侧开关（3个状态）

C.4 transmissionMode参数定义

表C.10定义了transmissionMode参数。

表C.10 – transmissionMode参数定义

字节值	描述	Cvt	助记符
0x00	**ISOSAEReserved** 这个值应该被这个文件保留以备将来的定义。	M	ISOSAERESRVD
0x01	**sendAtSlowRate** 此参数指定服务器应响应请求消息（其中要发送的响应数量=maximumNumberOfResponsesToSend）以低速传输所请求的dataRecord信息。传输模式参数slow指定的重复率是车辆制造商特定的，并在服务器中预定义。	U	SASR
0x02	**sendAtMediumRate** 此参数指定服务器应响应请求消息（其中要发送的响应数量=maximumNumberOfResponsesToSend）以中等速率传输所请求的dataRecord信息。 由transmissionMode参数介质指定的重复率是车辆制造商特定的，并且在服务器中预先定义。	U	SAMR
0x03	**sendAtFastRate** 此参数指定服务器应响应请求消息（其中要发送的响应数量=maximumNumberOfResponsesToSend）以快速发送请求的dataRecord信息。 由transmissionMode参数fast指定的重复率是车辆制造商特定的，并且在服务器中预定义。	U	SAFR
0x04	**stopSending** 服务器停止发送周期性/重复发送的肯定响应消息。 请注意，如果transmissionMode = stopSending（否则，服务器操作可能未定义），maximumNumberOfResponsesToSend参数应设置为0x01。	M	SS
0x05 – 0xFF	**ISOSAEReserved** 这个值应该被这个文件保留以备将来的定义。	M	ISOSAERESRVD

C.5 UDS版本号的编码

表C.11定义了UDS版本号DID 0xFF00 – 4字节无符号值的编码。 本文档的规范发布版本是: 2.0.0.0。

表C.11 – UDS版本号DID的编码0xFF00 – 4字节无符号值

字节1 (MSB)	字节2	字节3	字节4 (LSB)
主要 (0..255)	次要 (0..255)	修订版 (0..255)	0

表C.12定义了V1.0.0.0和V2.0.0.0的两个示例。

表C.12 – ISO 14229-1的1ST和2ND版的DID 0xFF00值

字节1 (MSB)	字节2	字节3	字节4 (LSB)
版本1.0.0.0			
1	0	0	0
版本2.0.0.0			
2	0	0	0

附件D
(规范性附录)

存储的数据传输功能单元数据参数定义

D.1 groupOfDTC参数定义

表D.1提供了一组DTC定义。

表D.1 – groupOfDTC的定义和DTC号码的范围

字节值	描述	Cvt	助记符
0x000000 – 0x0000FF	这个价值范围是为将来的立法要求而保留的。	M	RFLU
	动力总成组：发动机和变速器	U	PG
	动力总成DTC	U	PDTC_
	机箱组	U	CG
	底盘DTC	U	CDTC_
	身体组	U	BG
	身体DTC	U	BDTC_
	网络通信集团	U	NCG
	网络通信DTC	U	NCDTC_
0xFFFF00 – 0xFFFFFE	低位字节应始终为表D.15中定义的FunctionalGroupIdentifier。 例如，值0xFFFF33应等于排放组，值0xFFFFD0应等于安全组。	M	ISOSAERESRVD
0xFFFFFF	所有团体（所有DTC）	M	AG

D.2 DTCStatusMask和statusOfDTC位定义

D.2.1 公约和定义

本小节定义了与ReadDTCInformation服务一起使用的DTCStatusMask / statusOfDTC参数的映射。 每个服务器都应遵守下表中定义的存储位堆栈DTC状态信息的约定。 比特字段的实际使用应在实施标准中定义。

TestFailed位的状态不应直接链接到与监视器状态相关的故障安全行为。 这意味着，为了触发与某个监视器的状态相关联的故障保护行为，需要保持单独的一组状态位。 车辆制造商应确定是否以及如何应用和实施DTC状态与故障安全相关监视器状态之间的任何同步机制。

以下是用于描述DTC状态位定义的定义列表。

— **测试：测试是一种车载诊断软件算法，通常在单个操作周期内确定组件或系统的故障状态。 一些测试仅在运行周期中运行一次。 其他测试可以运行每个程序循环，每隔几毫秒进行一次采样。 测试的最终结果代表完全成熟/合格的条件（即通过或失败）。 那**

意味着在特定时间内需要失败条件的测试或者在组件被认为失败之前进行额外真实性检查的评估，只有在满足所有成熟条件后才会返回"失败"条件。 每个DTC都与代表可检测故障症状的测试相关联。

— 测试样本：当满足测试运行标准时，测试样本代表单个DTC测试执行实例的“通过”或“失败”结果。 这代表单个样本，因此通常不是完全成熟/合格的条件。 对于支持DTC故障检测计数器的ECU，表示故障的测试样本将使DTC故障检测计数器增加一个特定量，并且代表通过的测试样本将使DTC故障检测计数器减少一定量。

— 完成：完成表示测试能够确定当前操作周期中是否存在故障或不存在（完整并不意味着失败）。

— 测试结果：测试运行或完成后，可能会向内部失败处理程序指示以下结果之一：

  — PreFailed：此状态可能被ECU中的测试用于指示测试目前正在成熟为失败状态。 该信息的一个用例是在制造中加速故障检测以优化工作流程，同时保持现场的容错性。

  — 失败：在测试运行到完成状态并且指示完全成熟的失败条件之后，此状态可用。

  — 已通过：在测试运行到完成并且指示系统或组件未失败后，此状态可用。

— 失败：失败是指组件或系统无法达到其预期功能。 如果检测到故障条件足够长时间，则发生故障，这意味着测试返回“失败”结果。 术语“故障”和“故障”是可以互换的。

— 监视器：监视器由一个或多个用于确定组件或系统正常运行的测试组成。

— 监视周期：监视周期是监视器运行完整的时间。 这是制造商定义的一组条件，在此期间可以运行显示器的测试。 监控周期可以在一个操作周期内执行多次，或者在几个操作周期内执行一次。

— 操作周期：操作周期定义监视器运行的开始和结束条件。 在一个操作周期内，几个监测周期可能已经完成（不管它们的测试结果如何）。 一个ECU可能支持几个操作周期。 对于车身和底盘ECU，制造商需要定义一个操作周期（例如，在开启和关闭ECU之间的时间或点火开关之间的时间）。 对于动力总成ECU，还有其他标准定义了一个运行周期。 与排放相关的动力总成ECU使用发动机运行或发动机关闭时间段来定义称为行驶周期的运行周期。 如果DTC状态位的复位条件与操作周期的开始相关联，则它也可能被认为是前一周期的结束（即，并不总是可以区分每个操作周期的开始与结束）。

注：对于与排放相关的监测仪，操作周期开始和结束的标准由立法确定。

— 待定：失败的待处理状态定义为在当前操作周期或上一次完成的操作周期中，此测试报告了“失败”结果的测试。 一旦测试报告了该故障的完整操作周期的“通过”条件，待处理状态就被重置。

— 确认阈值：确认的失败状态被定义为在测试已经结束的给定数量的操作周期中针对该测试报告“失败”的测试。 通常，对于非OBD使用情况，操作周期的阈值被定义为一个。 对于OBD使用情况，此阈值通常大于1。实现可以使用Trip Counter（见图D.9）作为触发器

将确认状态从0更改为1.跳闸计数器计算出现故障的操作循环次数（行驶循环次数）。 如果计数器达到阈值（例如2个驱动周期），则确认的位从0变为1。

— 老化阈值：DTC的老化被定义为对于给定数量的车辆制造商或规定的操作循环报告没有'失败'结果的测试，并且如果相应的循环根据是否相应循环触发递增老化计数器测试已经完成或没有在周期内完成。 实现可以使用一个老化计数器（参见图D.11）作为将确认状态从1更改为0并从非易失性存储器擦除DTC信息的触发器。 老化计数器计数满足前述标准的循环次数（例如，预热循环）。 如果此计数器达到阈值（例如，40个预热周期），则确认的位从1变为0。

— 驾驶循环：用于排放相关ECU的特定类型的操作循环。 有关更多详细信息，请参阅“OperationCycle”。在与排放相关的ECU中，应支持一个运行周期，这与法规定义的驾驶周期相同。

— 监控级启用条件：允许监控器运行并报告测试结果的标准/条件。

— DTC状态更新条件：允许监视器更新所有DTC状态位的条件（例如，controlDTCSetting DTCSettingType不等于'off'）。此一般条件适用于所有DTC状态位转换（即，如果此条件为假如图D.1至图D.8所示的转换不允许，除了复位由接收clearDiagnosticInformation命令（见9.9.1和11.2.1）触发的状态位）。

— DTC存储条件：由车辆制造商定义的条件，指示相关的DTC状态位和相关的DTC数据（例如DTC扩展或快照数据）是否更新并存储在非易失性存储器中。

D.2.2 伪代码数据字典

伪代码数据字典定义每个statusOfDTC位的伪代码定义中使用的变量。 表D.2定义了伪代码数据字典。

表D.2 – 伪代码数据字典

变量	描述
initializationFlag_TF initializationFlag_TFTOC initializationFlag_PDTC initializationFlag_CDTC initializationFlag_TNCSLC initializationFlag_TFSLC initializationFlag_TNCTOC initializationFlag_WIR	以下伪代码中使用的标志确保DTC状态位初始化操作仅执行一次。 至少，在第一次启动ECU之前，预计标志的默认值为FALSE。 变量应保持锁定为TRUE，直到ECU软件复位或执行任何其他此类车辆制造商特定的复位。 FALSE =初始化未执行TRUE =执行初始化
lastOperationCycle	存储变量用于记录最近完成的操作周期。 在以下伪代码给出的操作的相应初始化阶段期间，应将值赋予该变量。
currentOperationCycle	用于记录当前操作周期的存储变量。 在DTC状态位逻辑范围外持续更新。
failedOperationCycle	存储变量用于记录最近一次失败的操作周期。 在以下伪代码给出的操作的相应初始化阶段期间，应将值赋予该变量。
confirmStage	存储变量用于记录已确认的DTC状态位伪代码的操作阶段。

D.2.3 DTC状态位定义

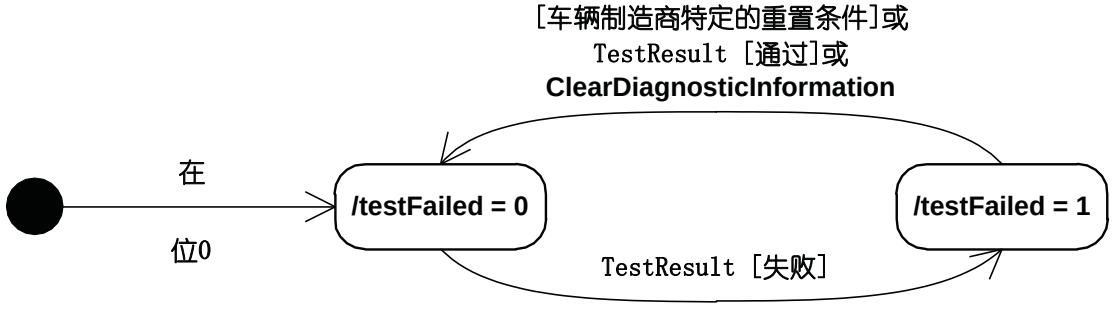
表D.3定义了DTC状态位'0' testFailed。

表D.3 - DTC状态位0 testFailed定义

位	描述	Cvt	助记符
	testFailed	U	TF
	该位应指示最近执行的测试的结果。 逻辑'1'表示最后一次测试失败，意味着失败完全成熟。 如果最近执行的测试的结果返回“通过”结果，意味着所有的非成熟标准已经满足，则重置为逻辑“0”。 附加的复位条件可以由车辆制造商/实施来定义		
	成功完成ClearDiagnosticInformation服务后的状态位	**逻辑'0'**	
	如果已经对ClearDiagnosticInformation进行了调用，则重置为逻辑“0”。		
	位状态定义: '0'=来自DTC测试的最新结果表明未检测到故障。 '1'=来自DTC测试的最新结果表明成熟的失败结果。		

#	伪代码操作
1	IF (initializationFlag_TF == FALSE) Set
2	initializationFlag_TF = TRUE Set
3	testFailed = 0
4	IF ((最新的测试结果==通过) 或 (ClearDiagnosticInformation
	requested == TRUE) 或
5	(车辆制造商/执行复位条件满足)) 设置testFailed = 0
6	ELSE IF (最近的测试结果== FAILED)
7	设置testFailed = 1

图D.1定义了DTC状态位'0' testFailed逻辑。



图D.1 - DTC状态位0 testFailed逻辑

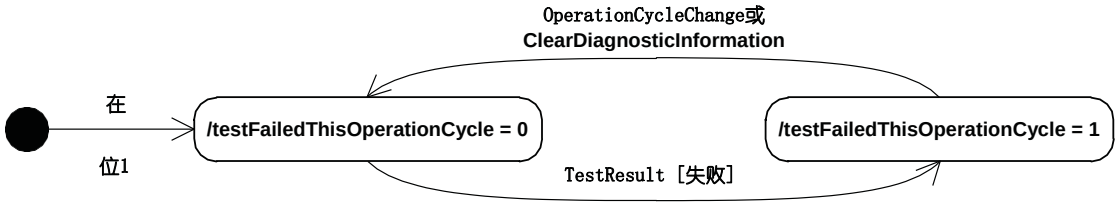
表D.4定义了DTC状态位'1' testFailedThisOperationCycle。

表D.4 - DTC状态位1 testFailedThisOperationCycle定义

<table>
<tr><th>位</th><th colspan="2">描述</th><th>Cvt</th><th>助记符</th></tr>
<tr><td rowspan="8"></td><td colspan="3">testFailedThisOperationCycle</td><td>U</td><td>TFTOC</td></tr>
<tr><td colspan="4">该位应指示诊断测试是否在当前操作周期的任何时间报告了testFailed结果（或者在当前操作周期期间以及在最后一次调用ClearDiagnosticInformation之后报告了testFailed结果）。 当启动一个新的操作循环时或在调用ClearDiagnosticInformation之后，重置为逻辑“0”。
如果该位设置为逻辑'1'，则在新的操作周期开始之前它应保持为'1'。</td></tr>
<tr><td colspan="3">成功ClearDiagnosticInformation服务后的位状态</td><td>逻辑'0'</td></tr>
<tr><td colspan="4">调用ClearDiagnosticInformation后重置为逻辑“0”。</td></tr>
<tr><td colspan="4">位状态定义:
'0'= testFailed: 在当前操作周期期间或在当前操作周期内调用ClearDiagnosticInformation之后未报告结果。
'1'= testFailed: 在当前操作周期中至少报告一次结果。</td></tr>
<tr><td>#</td><td colspan="3">伪代码操作</td></tr>
<tr><td>1
2
3
4
5

6
7
8
9</td><td colspan="3">IF (initializationFlag_TFTOC == FALSE) Set
initializationFlag_TFTOC = TRUE Set
testFailedThisOperationCycle = 0
设置lastOperationCycle = currentOperationCycle
IF ((currentOperationCycle! = lastOperationCycle) OR
(ClearDiagnosticInformation requested == TRUE))
设置lastOperationCycle = currentOperationCycle Set
testFailedThisOperationCycle = 0
ELSE IF (最近的测试结果== FAILED)
设置testFailedThisOperationCycle = 1</td></tr>
</table>

图D.2定义了DTC状态位'1' testFailedThisOperationCycle逻辑。



图D.2 - DTC状态位1 testFailedThisOperationCycle逻辑

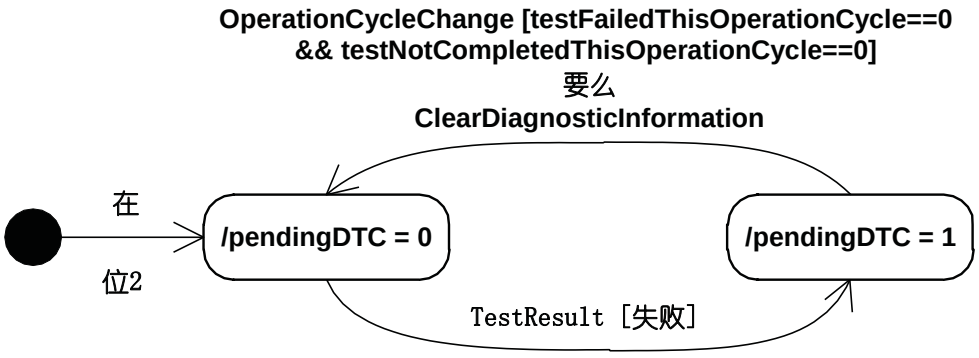
表D.5定义了DTC状态位'2' pendingDTC。

表D.5 - DTC状态位2 pendingDTC定义

位	描述	Cvt	助记符
	pendingDTC	U	PDTC
	该位应表明诊断测试是否在当前或上一次完成的操作周期中的任何时间报告了testFailed结果。 状态只有在测试运行并完成时才会更新。 设置pendingDTC位和TestFailedThisOperationCycle位的标准是相同的。 不同之处在于，在每个操作周期开始时，testFailedThisOperationCycle被清除，并且待操作的DTC位不会被清除，直到一个操作循环完成，测试至少经过一次并且从未失败。 如果测试在当前操作周期内未完成，状态位不应改变。 例如，如果监视器在确认的DTC置位后停止运行，则pendingDTC必须保持置1 ="1"。 对于OBD DTC，在第一个行驶循环中检测到故障后，需要存储未决的DTC。		
	成功ClearDiagnosticInformation服务后的位状态	**逻辑'0'**	
	调用ClearDiagnosticInformation后重置为逻辑 "0"。		
	位状态定义: '0'=在完成测试完成并且没有检测到故障的操作周期或者在调用ClearDiagnosticInformation服务时，该位应该被设置为0。 '1'=如果在当前操作周期内检测到故障，该位应设为1并锁存。		

#	伪代码操作
1	IF (initializationFlag_PDTC == FALSE) 设置
2	initializationFlag_PDTC = TRUE 设 置
3	pendingDTC = 0
4	IF (ClearDiagnosticInformation requested == TRUE) 设置
5	pendingDTC = 0
6	ELSE IF (最近的测试结果== FAILED) 设置pendingDTC = 1
7	ELSE IF ((currentOperationCycle == stop) AND
8	(testNotCompletedThisOperationCycle == 0) AND (testFailedThisOperationCycle == 0))
9	设置pendingDTC = 0

图D.3定义DTC状态位'2'，等待DTC逻辑。



图D.3 - DTC状态位2挂起DTC逻辑

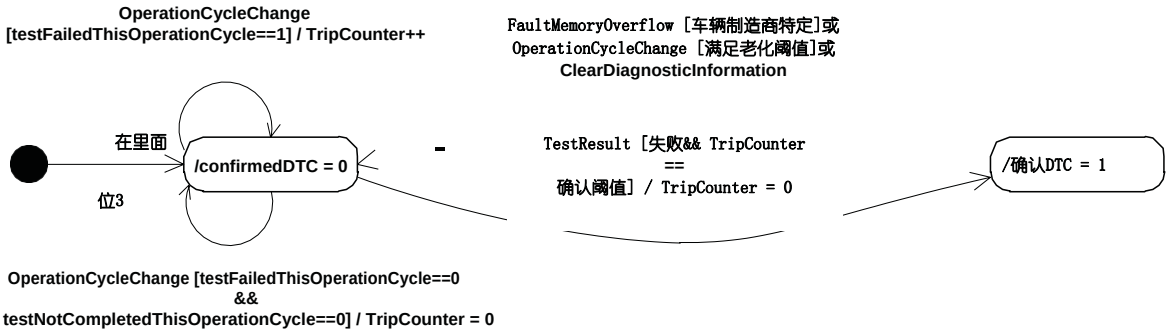
表D.6定义了DTC状态位'3'确认的DTC。

表D.6 – DTC状态位3确认了DTC定义

位	描述	Cvt	助记符
	confirmedDTC	M	CDTC
	该位应指示是否检测到故障的次数足以保证DTC需要存储在长期存储器中。 确认的DTC并不总是表明该请求时存在故障。 （testFailed可用于确定请求时是否存在故障）。 在呼叫ClearDiagnosticInformation或老化阈值已满足后（例如，发动机预热40次而没有检测到其他故障），重置为逻辑0。 此外，根据车辆制造商特定的故障存储器溢出要求，当与此DTC相关的故障记录被更新的DTC覆盖时，该位复位。 DTC确认阈值和老化阈值由车辆制造商定义或由车载诊断法规规定。		
	成功完成ClearDiagnosticInformation服务后的状态位	**逻辑'0'**	
	调用ClearDiagnosticInformation后重置为逻辑“0”。		
	位状态定义 '0'=自上次调用ClearDiagnosticInformation或DTC满足老化标准（或由于故障存储器溢出已清除DTC）后，从未确认过DTC。 '1'=自上次调用ClearDiagnosticInformation并且老化标准尚未得到满足以来，至少确认一次DTC。		

#	伪代码操作
1	IF (initializationFlag_CDTC == FALSE) Set
2	initializationFlag_CDTC = TRUE Set
3	confirmedDTC = 0
4	设置confirmStage = INITIAL_MONITOR IF
5	(confirmStage == INITIAL_MONITOR)
6	IF (确认阈值== TRUE) 设置确认DTC = 1
7	重置老化状态
8	设置confirmStage = AGING_MONITOR ELSE
9	设置确认DTC = 0
10	IF (confirmStage == AGING_MONITOR)
11	IF ((ClearDiagnosticInformation requested == TRUE) OR (老化阈值满=
12	TRUE))
13	设置确认DTC = 0
14	设置confirmStage = INITIAL_MONITOR
15	ELSE IF (最近的测试结果== FAILED) 重置老化状态
16	其他
17	根据情况更新老化状态
18	
19	

图D.4定义了DTC状态位'3'确认的DTC逻辑。



图D.4-DTC状态位3确认了DTC逻辑

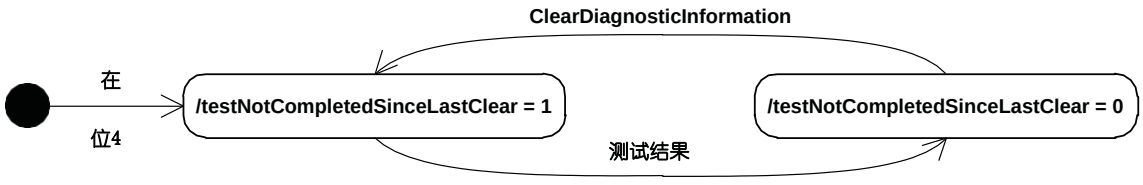
表D.7定义了DTC状态位'4'testNotCompletedSinceLastClear。

表D.7 - DTC状态位4 testNotCompletedSinceLastClear定义

位	描述	Cvt	助记符
	testNotCompletedSinceLastClear	U	TNCSLC
	该位应指示自上次调用ClearDiagnosticInformation以来DTC测试是否已运行并完成。 一（'1'）表示DTC测试尚未完成。 如果测试运行并通过，或者如果测试运行并失败（例如testFailedThisOperationCycle ='1'），则该位应置为'0'（并锁存）。		
	成功完成ClearDiagnosticInformation服务后的状态位	**逻辑'1'**	
	在调用ClearDiagnosticInformation之后重置为逻辑“1”。		
	位状态定义 '0'= DTC测试自上次诊断信息被清除后至少返回一次合格或不合格的测试结果。 '1'=自上次诊断信息被清除后，DTC测试尚未完成。		

#	伪代码操作
1	IF (initializationFlag_TNCSLC == FALSE) Set
2	initializationFlag_TNCSLC = TRUE Set
3	testNotCompletedSinceLastClear = 1
4	IF (ClearDiagnosticInformation requested = TRUE) Set
5	testNotCompletedSinceLastClear = 1
6	ELSE IF ((最近的测试结果=通过) 或 (最近的测试结果= FAILED))
7	设置testNotCompletedSinceLastClear = 0

图D.5定义了DTC状态位'4'testNotCompletedSinceLastClear逻辑。



图D.5 - DTC状态位4 testNotCompletedSinceLastClear逻辑

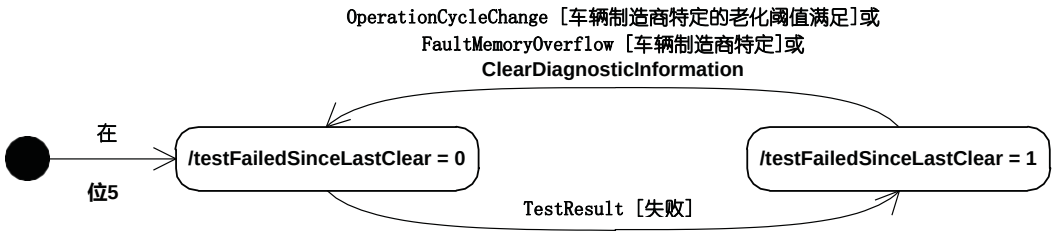
表D.8定义了DTC状态位'5' testFailedSinceLastClear。

表D.8 - DTC状态位5 testFailedSinceLastClear定义

位	描述	Cvt	助记符
	testFailedSinceLastClear	U	TFSLC
	该位应指示自上次调用ClearDiagnosticInformation（即，这是锁存的testFailedThisOperationCycle ='1'）以来DTC测试是否已完成，并且失败的结果已完成。 零（'0'）表示测试未运行或DTC测试运行并通过（但从未失败）。 如果测试运行并失败，那么该位应保持锁存在'1'。 车辆制造商有责任指定该位是否由老化标准复位或由于故障存储器溢出而复位。		
	成功完成ClearDiagnosticInformation服务后的状态位	**逻辑'0'**	
	调用ClearDiagnosticInformation后重置为逻辑“0”。		
	位状态定义 '0'=自上次诊断信息被清除后，DTC测试未显示失败结果。 如果在满足老化阈值或发生故障存储器溢出的情况下，该位也应重置为零（'0'），这是车辆制造商的责任。 '1'= DTC测试自上次诊断信息被清除后至少返回一次失败结果。		

#	伪代码操作
1	IF （initializationFlag_TFSLC == FALSE） Set
2	initializationFlag_TFSLC = TRUE Set
3	testFailedSinceLastClear = 0
4	IF (ClearDiagnosticInformation requested == TRUE) /* 可选的: OR (老化阈值满= TRUE)
5	/* 可选的: OR (溢出条件满足== TRUE) Set
6	testFailedSinceLastClear = 0
7	ELSE IF (最近的测试结果== FAILED) 设置testFailedSinceLastClear = 1

图D.6定义了DTC状态位'5' testFailedSinceLastClear逻辑。



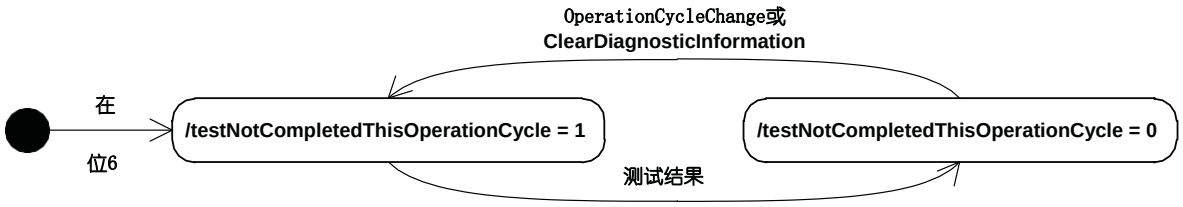
图D.6 - DTC状态位5 testFailedSinceLastClear逻辑

表D.9定义了DTC状态位'6' testNotCompletedThisOperationCycle。

表D.9 - DTC状态位6 testNotCompletedThisOperationCycle定义

<table>
<tr><th>位</th><th colspan="2">描述</th><th>Cvt</th><th>助记符</th></tr>
<tr><td rowspan="9"></td><td colspan="2">testNotCompletedThisOperationCycle</td><td>U</td><td>TNCTOC</td></tr>
<tr><td colspan="4">该位应指示DTC测试是否在当前操作周期内运行并完成（或者在上次调用ClearDiagnosticInformation之后的当前操作周期内完成）。
逻辑"1"表示DTC测试在当前操作周期内未完成。 如果测试运行并通过或失败，则该位应设置（并锁存）为0，直到开始新的运行周期。</td></tr>
<tr><td colspan="3">成功完成ClearDiagnosticInformation服务后的状态位</td><td>逻辑'1'</td></tr>
<tr><td colspan="4">在调用ClearDiagnosticInformation之后重置为逻辑 "1"。</td></tr>
<tr><td colspan="4">位状态定义
'0'= DTC测试在当前驾驶循环期间（或自上一次诊断信息在当前操作循环期间被清除）返回通过的或testFailedThisOperationCycle ='1'的结果。
'1'= DTC测试未运行完成此操作周期（或自上次诊断信息被清除此操作周期以来）。</td></tr>
<tr><td>#</td><td colspan="3">伪代码操作</td></tr>
<tr><td>1
2
3
4
5
6
7
8
9
10
11</td><td colspan="3">IF (initializationFlag_TNCTOC == FALSE) 设置
initializationFlag_TNCTOC = TRUE
设置testNotCompletedThisOperationCycle = 1
设置lastOperationCycle = currentOperationCycle IF
(ClearDiagnosticInformation requested == TRUE)
设置testNotCompletedThisOperationCycle = 1
ELSE IF (currentOperationCycle! = lastOperationCycle) 设置
lastOperationCycle = currentOperationCycle
设置testNotCompletedThisOperationCycle = 1 ELSE IF
((最近的测试结果==通过) 或
(最近的测试结果== FAILED))
设置testNotCompletedThisOperationCycle = 0</td></tr>
</table>

图D.7定义了DTC状态位'6' testNotCompletedThisOperationCycle逻辑。



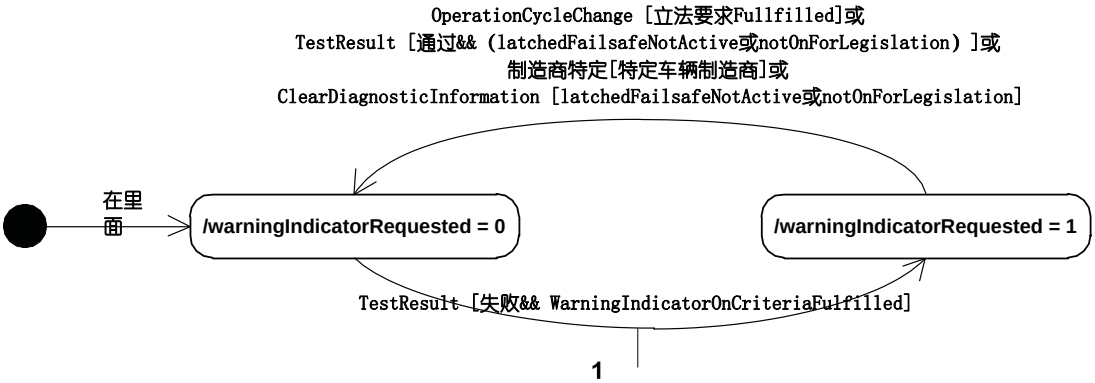
图D.7 - DTC状态位6 testNotCompletedThisOperationCycle逻辑

表D.10定义了请求的DTC状态位'7' WarningIndicator。

表D.10 – DTC状态位7 WarningIndicator请求的定义

<table>
<tr><th>位</th><th>描述</th><th>Cvt</th><th>助记符</th></tr>
<tr><td rowspan="8"></td><td>warningIndicatorRequested</td><td>U</td><td>世界投资报告</td></tr>
<tr><td colspan="3">该位应报告与特定DTC相关的任何警告指示器的状态。 警告输出可能由指示灯，显示的文字信息等组成。如果特定DTC没有警告指示，则此状态应默认为逻辑'0'状态。
启动警告指示器的条件应由车辆制造商/实施者定义，但是如果给定DTC的警告指示灯亮起，则确认的DTC也应设置为'1'（以下所述情况除外）。</td></tr>
<tr><td>成功完成ClearDiagnosticInformation服务后的状态位</td><td colspan="2">逻辑'0'</td></tr>
<tr><td colspan="3">调用ClearDiagnosticInformation后重置为逻辑“0”。 某些ECU可能锁定与当前操作周期中特定的确认故障相关的故障安全策略。 如果在调用ClearDiagnosticInformation之后由于该锁存故障安全而仍然要求警告指示器，则该位不应被清除为逻辑0。 相反，该位应保持逻辑“1”，直到故障安全策略不再有效（例如，测试完成并通过）。 额外的复位条件应由车辆制造商/实施者定义。</td></tr>
<tr><td colspan="3">位状态定义
'0'=服务器没有请求warningIndicator被激活'1'=服务器正
在请求warningIndicator被激活</td></tr>
<tr><td colspan="3">
<table>
<tr><th>#</th><th>伪代码操作</th></tr>
<tr><td>1</td><td>IF（initializationFlag_WIR == FALSE）设置</td></tr>
<tr><td>2</td><td>initializationFlag_WIR = TRUE 设 置</td></tr>
<tr><td>3</td><td>warningIndicatorRequested = 0</td></tr>
<tr><td>4</td><td>IF（（（（ClearDiagnosticInformation requested == TRUE）或（TestResult == Passed）
OR（车辆制造商或特定实施警告指示符禁用标准得到满足））
和（（由于特定DTC的锁定故障安全，不需要警告指示）或（警告指示未通过立法要求）））
设置warningIndicatorRequested = 0</td></tr>
<tr><td>5</td><td>ELSE IF（（（TestResult == Failed）AND（警告指示符存在于特定的DTC中）</td></tr>
<tr><td>6</td><td>AND（（确认DTC == 1）
要么 （车辆制造商或特定实施警告指示符使能标准得到满足）））
设置warningIndicatorRequested = 1</td></tr>
<tr><td>7</td><td></td></tr>
</table>
</td></tr>
</table>

图D.8定义了DTC状态位'7' WarningIndicator所要求的逻辑。



键

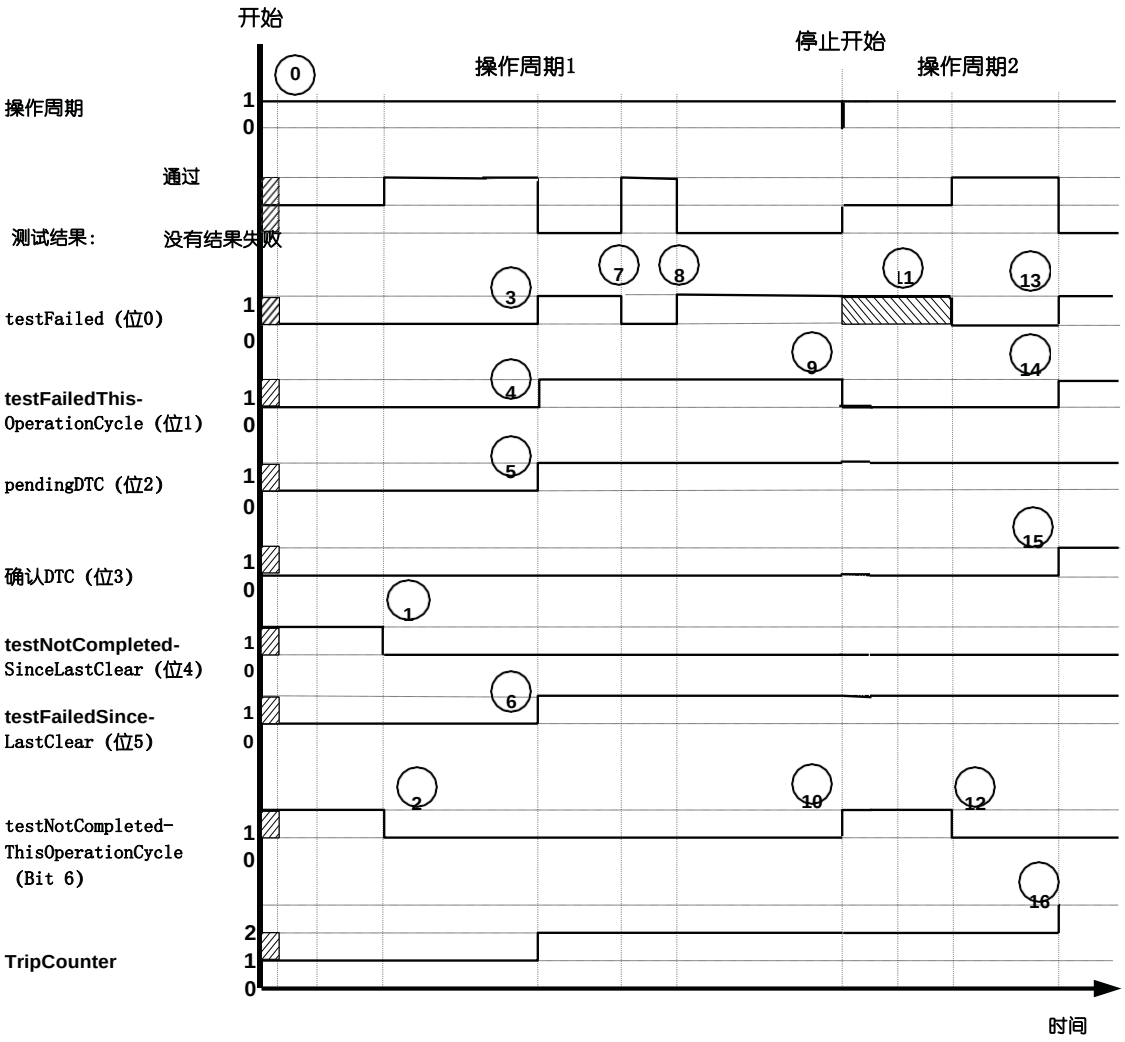
1 WarningIndicatorOnCriteriaFulfilled =警告指示符存在于特定的DTC和 (确认的DTC = 1或车辆制造商或特定于实施的警告指示器使能标准得到满足)

图D.8 - DTC状态位7 WarningIndicator请求的逻辑

D.2.4 DTC状态位操作示例

本示例概述了两个操作循环中排放相关的OBD DTC中DTC状态位的操作。 该图显示了两个操作循环排放相关的OBD DTC的处理情况。 处理也可应用于非排放相关的OBD DTC，并在此处显示用于一般信息目的。

图D.9定义了两个操作循环排放相关的OBD DTC的示例。



键

▨	未定义的位状态制造商特定的位
▧	状态
0	ClearDiagnosticInformation接收到DTC状态字节的初始化
1, 2	相关的诊断监视器报告有足够数量的满足DTC通过标准Æ testNotCompleted位（4和6）的通过测试样本从1变为0，表示监视器已运行至完成并且自上次清除以来已达到DTC就绪状态操作周期1
3, 4, 5, 6	相关诊断监视器报告有足够数量的失败测试样本满足DTC失败标准Æ testFailed，testFailedThisMonitoringCycle，pendingDTC和testFailedSinceLastClear位从0变为1，表示已检测到故障但故障尚未确认超过2个操作周期
7	相关诊断监视器报告满足DTC通过标准的足够数量的合格测试样本 Æ testFailed位从1变为0，表示故障当前未激活
8	相关诊断监视器报告了足够数量的满足DTC失败标准Æ的失败测试样本 testFailed位从0变为1，表示在运行周期1中重复检测到故障

9,10操作循环1结束并且操作循环2开始，testFailed此操作循环从1变为0并且testNotCompleteThisOperationCycle从0变为1；如果在运行周期的最后或在开始新的周期之后执行该复位，则是制造商特定的

11 在新的操作周期开始后（制造商特定是否通过从操作周期1到操作周期2的转换保留了testFailed状态），相关的诊断监视器报告了足够数量的满足DTC的结果满足DTC通过标准Æ testFailed位转换为0

12 在新的操作周期开始后，相关的诊断监视器报告有足够数量的满足DTC通过的测试样本通过标准Æ testNotCompleteThisOperationCycle位从1变为0，表示在新操作周期内监视器至少运行一次

相关诊断监视器报告有足够数量的失败测试样本满足DTC失败标准Æ testFailed，testFailedThisMonitoringCycle位从0变为1，表示在新操作周期内检测到故障

15 确认的DTC位从0变为1，表示在上一个操作周期内检测到的相关故障仍然存在

16 根据图D.4，在DTC状态变为确认DTC时，TripCounter尖峰到'2'，然后立即重置为'0'。

图D.9 - 两个操作循环排放相关的OBD DTC示例

D.3 DTC严重性和类定义

D.3.1 DTC严重性和类字节定义

本小节定义了与ReadDTCInformation服务一起使用的DTCSeverityMask / DTCSeverity参数的映射。 每个服务器应遵守表D.11中定义的用于存储比特压缩DTC严重性信息的约定。

DTCSeverityMask / DTCSeverity字节包含DTC严重性和DTC类别信息。 DTCSeverityMask / DTCSeverity字节以表D.11中定义的1字节值报告。 1字节值的可选高3位（位7-5）用于表示DTC严重性信息。 如果服务器不支持那些位应设置为“0”。 1字节值的强制低位5位（位4-0）用于表示DTC类别信息。

表D.11 - DTCSeverityMask / DTCSeverity字节定义

DTCSeverity字节							
位7	位6	位5	位4	位3	位2	位1	位0
DTC严重性信息（可选）			DTC类别信息				

D.3.2 DTC严重性位定义

DTC严重性位定义定义位状态以报告系统（例如车辆）操作员采取的建议措施。 表D.12定义了DTC严重性状态位。

表D.12 - DTC严重性位定义（位7-5）

位	描述	Cvt	助记符
5	**maintenanceOnly** 0 =无维护只有严重性1 =维护只有 严重性 该值表示故障请求仅维护。	M	MO
6	**checkAtNextHalt** 0 =不检查AtNextHalt 1 = checkAtNextHalt 该值表示在下次停车时需要检查车辆的故障。	M	CHKANH
7	**checkImmediately** 0 =不检查立即1 =检查立即 该值表示故障需要立即检查车辆。	M	CHKI

D.3.3 DTC类定义

DTC类别定义适用于符合WWH-OBD GTR的OBD系统。 A类，B1，B2或C类是排放相关DTC的属性。 根据WWH-OBD GTR的要求，这些属性表征故障对排放或OBD系统监控能力的影响。

注：包含在一个诊断请求中的DTC类信息被允许有多个位设置为1，以请求多个DTC类的信息。 诊断响应中包含的DTC类别信息只能有一个位设置为1.表D.13定义了GTR DTC类别定义（位4-0）。

表D.13 - GTR DTC类别定义（位4-0）

比特值	描述	Cvt	助记符
0	**DTCClass_0** DTCClass 0未分类。 如果DTCSeverity包含在响应消息中但未报告DTC类信息，例如SAE J2012-DA和ISO 14229-1中定义的传统DTC，则应使用此类。 位= 0：对于报告的DTC禁用DTCClass_0。 位= 1：为报 告的DTC启用DTCClass_0。	M	DTCCLASS_0
1	**DTCClass_1** DTCClass_1与GTR模块B类A定义相匹配。 假定超过相关的OBD阈值限值（OTL）时，故障应标识为A类。 可以接受的是，当这类故障发生时，排放量可能不会超过OTL。 位= 0：报告的DTC禁用DTCClass_1。 位= 1：为报告的 DTC启用DTCClass_1。	M	DTCCLASS_1

表D.13 – （续）

比特值	描述	Cvt	助记符
2	**DTCClass_2** DTCClass_2与GTR模块B类B1定义相匹配。 如果情况存在，可能导致排放超过OTL但不能估计对排放的确切影响，因此根据情况的实际排放量可能高于或低于OTL的情况下，故障应标识为B1类。 B1类故障应包括限制OBD系统监控A类或B1类故障能力的故障。 位= 0：对报告的DTC禁用DTCClass_2。 位= 1：为报告的DTC启用DTCClass_2。	M	DTCCLASS_2
3	**DTCClass_3** DTCClass_3与GTR模块B类B2定义相匹配。 当存在假定会影响排放但未达到超过OTL水平的情况时，故障应标识为B2类。 限制OBD系统监控B2类故障能力的故障应分类为B1或B2类。 位= 0：对报告的DTC禁用DTCClass_3。 位= 1：为报告的DTC启用DTCClass_3。	M	DTCCLASS_3
4	**DTCClass_4** DTCClass_4与GTR模块B类C定义相匹配。 当存在环境时，如果监测到故障，则认为故障影响排放，但达到不超过规定排放限值的水平时，故障应标识为C类。 限制OBD系统执行C类故障监测的能力的故障应划分为B1或B2类。 位= 0：报告的DTC禁用DTCClass_4。 位= 1：为报告的DTC启用DTCClass_4。	M	DTCCLASS_4

D.4 DTCFormatIdentifier定义

该参数值定义了服务器报告的DTC的格式。 给定的服务器只能支持一个DTCFormatIdentifier。

表D.14 - DTC格式标识符 (DTCFID_) 的定义

字节值	描述	Cvt	助记符
0x00	**SAE_J2012-DA_DTCFormat_00** 该参数值标识ISO 15031-6规范中定义的服务器报告的DTC格式。	M	J2012-DADTCF00
0x01	**ISO_14229-1_DTCFormat** 该参数值通过参数DTCAndStatusRecord标识服务器报告的DTC格式，如本表中所定义。	M	14229-1DTCF
0x02	**SAE_J1939-73_DTCFormat** 该参数值标识SAE J1939-73中定义的服务器报告的DTC格式。	M	J1939-73DTCF
0x03	**ISO_11992-4_DTCFormat** 该参数值标识ISO 11992-4规范中定义的服务器报告的DTC格式。	M	11992-4DTCF
0x04	**SAE_J2012-DA_DTCFormat_04** 该参数值标识ISO 27145-2规范中定义的服务器报告的DTC格式。	M	J2012-DADTCF04
0x05 - 0xFF	ISO / SAE保留	M	ISOSAERESRVD
	该值由本文档保留以备将来定义。		

D.5 FunctionalGroupIdentifier定义

FunctionalGroupIdentifier指定不同的功能系统组。 该标识符用于区分由多个不同服务器组成的电气架构内不同功能系统组之间由测试设备发送的命令。 如果服务器已经实施了排放系统的软件以及在I / M测试期间可能检查的其他系统，重要的是仅报告所请求的功能系统组的DTC信息。 排放I / M测试不应该失败，因为另一个功能系统组（例如安全系统组）存储了DTC信息。

FunctionalGroupIdentifier指定一个功能系统组，用于:

— 请求Unified Diagnostic Services版本号来识别协议，

— 从车辆请求DTC状态信息，以及

— 清除车辆中的DTC信息。

主要目的是能够报告/清除特定于功能系统组的DTC信息。 ECU可能是多个功能系统组的一部分，例如排放系统，制动系统等。如果在排放检查和维护（I / M）测试期间报告制动系统的DTC，则车辆不应使排放I / M因为作为排放功能系统一部分的ECU也报告制动功能系统DTC。

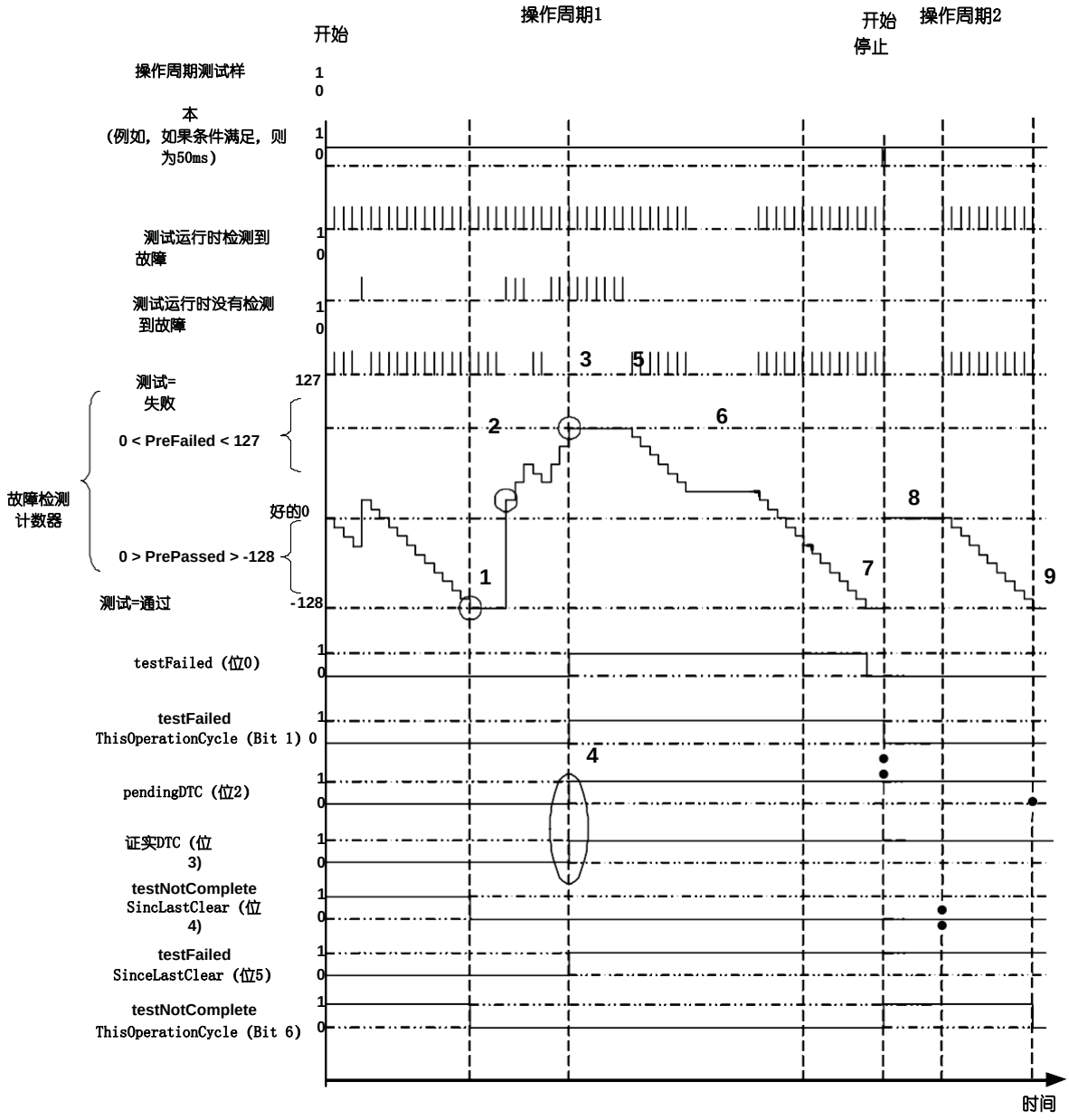
表D.15定义了FunctionalGroupIdentifiers。

表D.15 - FunctionalGroupIdentifiers (FGID_) 的定义

字节值	描述	Cvt	助记符
0x00 - 0x32	**ISO / SAE保留** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD
0x33	**排放系统组** 此值标识服务器中的排放系统。	M	EMSYSGRP
0x34 - 0xCF	**ISO / SAE保留** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD
0xD0	**安全系统组** 该值标识服务器中的安全系统。	M	SAFESYSGRP
0xD1 - 0xDF	**立法制度小组** 这个值的范围是为了将来的定义而保留给本文件所需要的立法组标识符。	M	LEGSYSGRP
0xE0 - 0xFD	**ISO / SAE保留** 这个范围的值由本文件保留以备未来定义。	M	ISOSAERESRVD
0xFE的	**VOBD系统** 该值标识VOBD系统设备。 根据所实施的VOBD策略，只有网关，专用VOBD ECU或任何其他具有VOBD功能的ECU（例如发动机控制器）才会作出响应。	M	VOBDSYSGRP
为0xFF	**所有功能系统组** 此值标识服务器中此表中列出的所有功能系统组。	M	ALLFCTSYSGRP

D.6 DTCFaultDetectionCounter操作实现示例

图D.10显示了非排放相关服务器的DTC故障检测计数器操作。



键

1 当故障检测计数器达到最小值（-128）或最大值（127）时测试完成，因此testNotCompleteSinceLastClear和testNotCompleteThisOperationCycle位从1变为0。

2 如果测试的一个测试样本返回失败结果，则始终会导致故障检测计数器增加到0以上（确保测试完成后的失败检测时间不会加倍）

3 故障检测计数器达到最大值（127），表明故障状态已经完全成熟； 测试报告失败的结果，因此testFailed，testFailedThisOperationCycle和testFailedSinceLastClear位从0更改为1。

4 ConfirmedDTC位与pendingDTC位同时置位（从0变为1），因为此示例适用于确认阈值为1的非排放相关服务器/ ECU。

5 如果一个测试的一个测试样本返回一个通过的结果，则它是制造商特定的，它总是导致故障检测计数器从0开始递减（确保测试完成后通过的检测时间不会加倍）。

6 与测试相关的监视器不运行，因为监视器级别启用条件未满足，因此会生成测试样本结果。 当监视器启用条件再次满足时，故障检测计数器是否被重置为0是制造商特定的。

7 计数器在当前操作周期中再次达到其最小值（-128），因此testFailed位从1变为0。

8 在新的操作周期开始后，与测试相关的显示器尚未启用； 因此DTC状态位不会改变，除了与操作周期开始相关的位。这些位最迟在新的操作周期开始时重置。

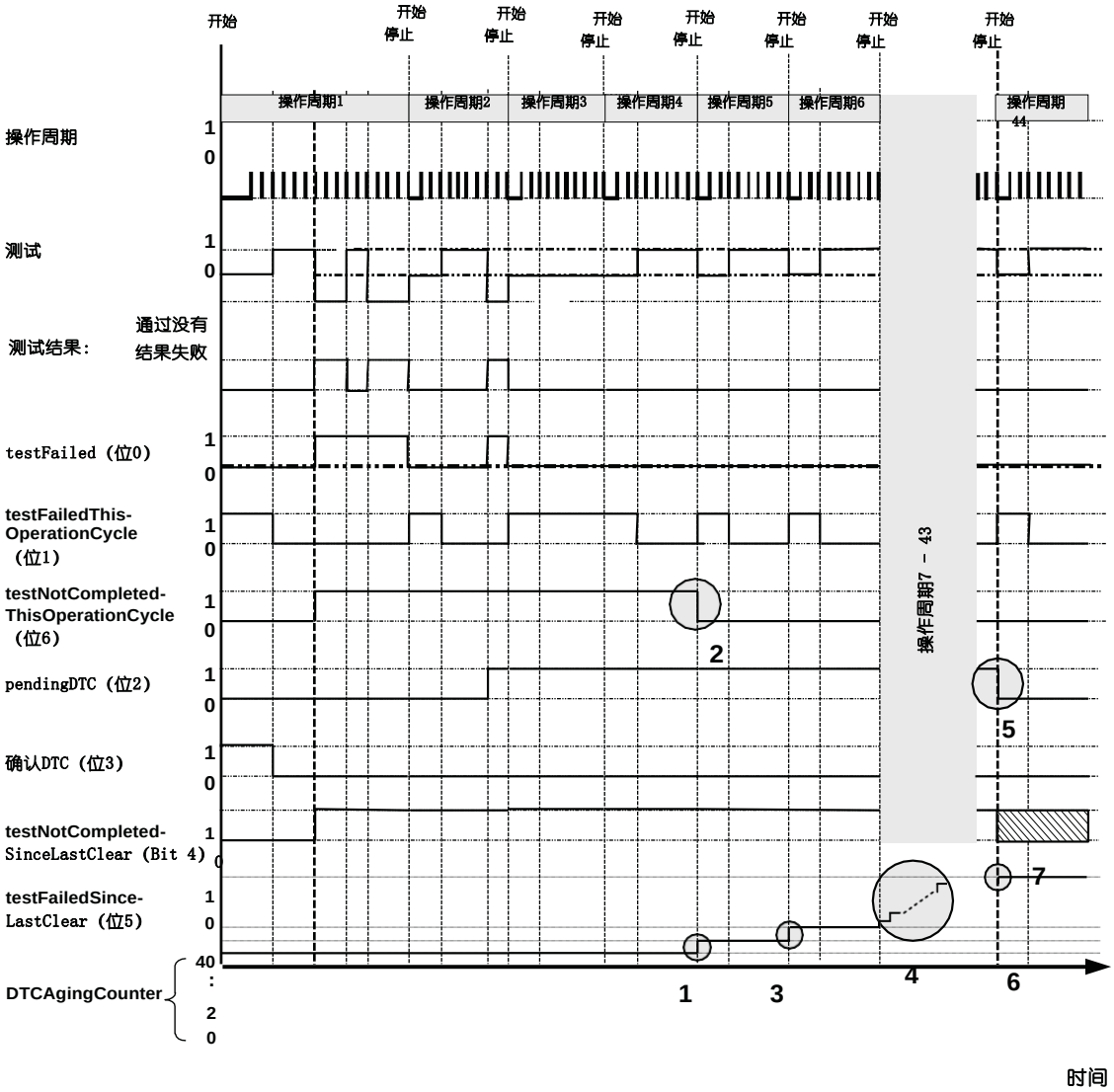
9 在新的操作周期开始后，计数器达到最小值（-128），因此testNotCompleteThisOperationCycle位从1变为0。

图D.10 - 无排放相关服务器的DTCFaultDetectionCounter操作示例

D.7 DTCAgingCounter示例

这个例子提供了一个DTCAgingCounter操作的概述，它计算自故障最新失败以来的驱动周期数。

图D.11定义了DTCAgingCounter示例。



键

1 DTCAgingCounter在完成测试未失败的操作周期后递增

2 在测试完成并且没有失败的操作周期之后，pendingDTC被设置为零。 如果ECU不支持断电顺序（即在关闭点火装置时立即关闭），它将无法检测到操作周期的结束。 因此，在下一个操作周期开始时将pendingDTC位设置为零也是有效的

3 DTCAgingCounter在完成测试未失败的操作周期后递增

4 DTCAgingCounter继续增加，因为在这些操作周期中测试没有失败

5 当老化标准完全满足时，确认的DTC被设置为零（例如，DTCAgingCounter达到特定值）

6 DTCAgingCounter达到最大值（例如40），此时确认的DTC位被清除

7 车辆制造商有责任指定testFailedSinceLastClear位是否由老化标准或由于故障存储器溢出而复位

图D.11 - DTCAgingCounter示例

附件E
(规范性附录)

输入输出控制功能单元数据参数定义

E.1 InputOutputControlParameter定义

表E.1定义了inputOutputControlParameter。

表E.1 - inputOutputControlParameter定义

字节值	描述	Cvt	助记符
0x00	**returnControlToECU** 该值应指示服务器客户端不再控制有关dataIdentifier所引用的输入信号，内部参数和/或输出信号。 请求中的controlState字节的详细信息：0字节 正面响应中的controlState字节的详细信息：等于dataIdentifier的dataRecord的大小和格式	U	RCTECU
0x01	**重置为默认** 该值应向服务器指示，请求将dataIdentifier所引用的输入信号，内部参数和/或输出信号重置为其默认状态。 请求中的controlState字节的详细信息：0字节 controlState字节的详细信息为正数。 响应：等于dataIdentifier的dataRecord的大小和格式	U	RTD
0x02	**freezeCurrentState** 该值应指示服务器，要求冻结输入信号的当前状态，内部参数和/或由dataIdentifier引用的输出信号。 请求中的controlState字节的详细信息：0字节 controlState字节的详细信息为正数。 响应：等于dataIdentifier的dataRecord的大小和格式	U	FCS
0x03	**shortTermAdjustment** 该值应向服务器表明，要求将RAM中dataIdentifier所引用的输入信号，内部参数和/或受控输出信号调整为controlOption中包含的值（s）参数（例如，将怠速空气控制阀设置为特定步号，将阀的脉冲宽度设置为特定值/占空比）。 请求中的controlState字节的详细信息：等于dataIdentifier的dataRecord的大小和格式 controlState字节在pos中的详细信息。 响应：等于dataIdentifier的dataRecord的大小和格式	U	STA
0x04 - 0xFF	**ISOSAEReserved** 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

附件F
(规范性附录)

常规功能单元数据参数定义

F.1 RoutineIdentifier (RID) 定义

表F.1定义了routineIdentifier。

表F.1 – 例行标识符定义

字节值	描述	Cvt	助记符
0x0000 – 0x00FF	**ISOSAEReserved** 这个值应该被这个文件保留以备将来的定义。	M	ISOSAERESRVD
0x0100 - 0x01FF	**TachographTestIds** 该值的范围被保留以表示行驶记录仪测试结果值。	U	TACHORI_
0x0200 - 0xDFFF	**vehicleManufacturerSpecific** 这个值的范围保留给车辆制造商特定用途。	U	VMS_
0xE000 0xE1FF	**OBDTestIds** 这个值的范围被保留以表示OBD / EOBD测试结果值。	U	OBDRI_
0xE200	**DeployLoopRoutineID** 这个值应该用来启动先前选择的点火回路的部署。	U	DLRI_
0xE201 – 0xE2FF	**SafetySystemRoutineIDs** 这个范围的值应该被这个文件保留，以便将来定义由安全相关系统实现的例程。	M	SASRI_
0xE300 - 0xEFFF	**ISOSAEReserved** 这个值应该被这个文件保留以备将来的定义。	M	ISOSAERESRVD
0xF000 - 0xFEFF	**systemSupplierSpecific** 这个值的范围保留给系统供应商特定用途。	U	SSS_
为0xFF00	**eraseMemory** 该值应该用于启动服务器的内存擦除例程。 控制选项和状态记录格式应由ECU制定并由车辆制造商定义。	U	EM_
0xFF01	**checkProgrammingDependencies** 这个值应该用来检查服务器的内存编程依赖性。 控制选项和状态记录格式应由ECU制定并由车辆制造商定义。	U	CPD_
0xFF02	**eraseMirrorMemoryDTCs** 该值应该用于擦除服务器的镜像内存DTC。	U	EMMDTC_
0xFF03 - 0xFFFF	**ISOSAEReserved** 这个值应该被这个文件保留以备将来的定义。	M	ISOSAERESRVD

附件G
(规范性附录)

上传和下载功能单元数据参数

G.1 modeOfOperation值的定义

RequestFileTransfer请求消息包含modeOfOperation参数。 这些值在表G.1中定义。

表G.1 - modeOfOperation值的定义

字节值	描述	Cvt	助记符
0x00	ISO / SAE保留 该值由本文档保留以备将来定义。	M	ISOSAERESRVD
0x01	添加文件 该值将用于添加filePathAndName参数中定义的文件（下载）。	U	添加文件
0x02	删除文件 该值应该用于删除filePathAndName参数中定义的文件。	U	DELFILE
0x03	**ReplaceFile** 该值应用于替换filePathAndName参数中定义的文件（下载）。 如果文件没有存储在文件的位置上，应该添加。	U	REPLFILE
0x04	ReadFile的 该值应该用于读取filePathAndName参数定义位置处的文件（上载）。	U	RDFILE
0x05	**ReadDir** 该值应用于读取filePathAndName参数中定义的目录。 该值暗示该请求不包含fileName。	U	RDDIR
0x06 - 0xFF	ISO / SAE保留 该值由本文档保留以备将来定义。	M	ISOSAERESRVD

附件H
(资料)

addressAndLengthFormatIdentifier参数值的示例

H.1 addressAndLengthFormatIdentifier示例值

表H.1包含addressAndLengthFormatIdentifier的高半字节和低半字节值组合的例子。 需要考虑以下几点:

— 对于“可管理的内存大小”或“内存地址范围”，标记为“不适用”的值不允许使用，并且必须由服务器通过否定响应消息拒绝。

— 此参数允许使用适用的“可管理的内存大小”和“内存地址范围”。

表H.1 – addressAndLengthFormatIdentifier示例

字节值	描述			
	位7-4（高半字节）的 memorySize字节数		位3-0（低半字节）的 memoryAddress字节数	
	用于memorySize参数的字节	可管理的大小	用于memoryAddress参数的字节	可寻址的存储器
0x00	不适用	不适用	不适用	不适用
0x01	不适用	不适用	1	256字节 - 1
0x02	不适用	不适用	2	64 KB - 1
0x03	不适用	不适用	3	16 MB - 1
0x04	不适用	不适用	4	4 GB - 1
0x05	不适用	不适用	5	1,024 GB - 1
0x06 – 0x0F	:	:	:	:
0x10	1	256字节	不适用	不适用
0x11	1	256字节	1	256字节 - 1
0x12	1	256字节	2	64 KB - 1
0x13	1	256字节	3	16 MB - 1
0x14	1	256字节	4	4 GB - 1
0x15	1	256字节	5	1,024 GB - 1
0x16 – 0x1F	:	:	:	:
0x20	2	64 KB	不适用	不适用
0x21	2	64 KB	1	256字节 - 1
0x22	2	64 KB	2	64 KB - 1
0x23	2	64 KB	3	16 MB - 1
0x24	2	64 KB	4	4 GB - 1

表H.1 – （续）

字节值	描述			
	位7-4（高半字节）的 memorySize字节数		位3-0（低半字节）的 memoryAddress字节数	
	用于memorySize参数的字节	可管理的大小	用于memoryAddress参数的字节	可寻址的存储器
0x25	2	64 KB	5	1,024 GB - 1
0x26 – 0x2F	:	:	:	:
0x30	3	16 MB	不适用	不适用
0x31	3	16 MB	1	256字节 - 1
0x32	3	16 MB	2	64 KB - 1
0x33	3	16 MB	3	16 MB - 1
0x34	3	16 MB	4	4 GB - 1
0x35	3	16 MB	5	1,024 GB - 1
0x36 – 0x3F	:	:	:	:
0x40	4	4 GB	不适用	不适用
0x41	4	4 GB	1	256字节 - 1
0x42	4	4 GB	2	64 KB - 1
0x43	4	4 GB	3	16 MB - 1
0x44	4	4 GB	4	4 GB - 1
0x45	4	4 GB	5	1,024 GB - 1
0x46 -0xFF	:	:	:	:

附件一
(规范性附录)

安全访问状态图

I.1 一般

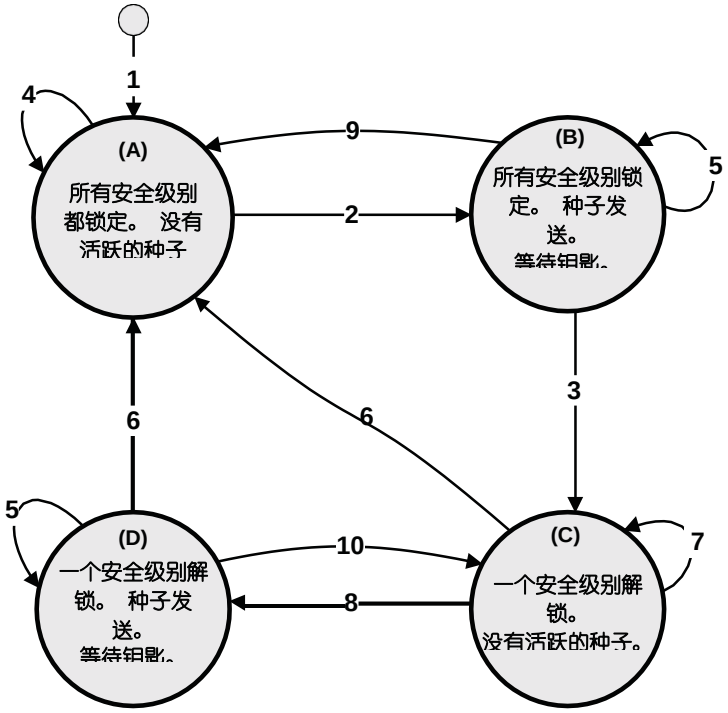
本附件的目的是根据状态图和状态转换条件和动作定义来描述ECU中的SecurityAccess服务处理。 以下是定义的基础:

— 为了在状态之间和状态之内定义单个转换，使用析取范式。

— “析取范式”的定义:“如果陈述是由一个或多个析取物组成的析取（OR序列），它是一个或多个文字的连接（AND）

I.2 基于分离正常形式的状态转换定义

图I.1以图形方式描述了SecurityAccess处理的状态图。 给定的数字引用状态转换条件以及在转换中要执行的操作。



键

(A) 所有安全级别都锁定。 没有活跃的种子。
(B) 所有安全级别锁定。 种子发送。 等待钥匙。
(C) 一个安全级别解锁。 没有活跃的种子。
(D) 一个安全级别解锁。 种子发送。 等待钥匙。 1 .. 10
见表I.2

图I.1 - SecurityAccess状态图

状态图考虑到一般会话处理在会话管理层内的适当位置完成（参见ISO 14229-2），因此不需要在状态图中考虑。

状态转换定义使用了一些可以根据车辆制造商特定要求设置的参数。 Delay_Timer和Att_Cnt参数的支持是可选的，由汽车制造商决定。 通常，对于较长的种子/密钥长度（例如，16字节及以上），这些参数的支持不再那么重要。

表I.1定义了状态转换 – 参数。

表I.1 – 状态转换 – 参数

名称	描述
延时定时器	如果支持，则此值表示安全访问尝试之间所需的最短时间。 此外，这是延迟计时器是否会在每次开机/启动时调用，这是汽车制造商的具体情况。 标准用例对于延迟时间将具有固定值，但客户特定用例也可能具有变量值（例如，该值取决于存储在其中的错误访问尝试的次数非易失性存储器）。 注意服务器可以选择为每个安全级别实施单独的计时器，或者为所有级别使用单个计时器。
Att_Cnt	如果支持，则此值表示插入延迟时间（Start_Delay）之前的错误安全访问尝试次数。 实施时，每个安全级别都需要计数器。
Static_Seed	这表示一个布尔值，其中true表示根据表I.2在某些条件下种子被存储并重新用于种子请求的种子请求。 值为false表示每次接收到新的种子请求时都使用随机种子。 如果不支持Delay_Timer和Att_Cnt，则应始终使用随机种子。
xx	这表示服务器收到的最后一个requestSeed securityAccessType。
yy	这表示服务器接收的当前sendKey securityAccessType。

传说：

和，或者 逻辑运算
斜体 可选，客户特定的“==”平等（比较运算符）
"=" 赋值运算符
“<>” 不等
“<” 少于
“>” 比...更棒
“+” 数学加法
“-” 数学减法
"++" 增量运算符（变量++与变量=变量+ 1相同）

表I.2包含完整的状态转换定义。

表I.2 – 状态转换 – 析取范式表示

没有。	手术		条件	行动
1			ECU应用程序的启动/重新启动（例如，ECU复位，电源循环，键循环，睡眠→唤醒转换等）。	初始化Att_Cnt（如果适用）。 启动Delay_Timer[a]（如果启动时需要）。
			接收到SecurityAccess requestSeed。 消息长度正常[b]。 *可选的前提条件已满足。* 延迟过期（如果适用）。	
			接收到SecurityAccess的sendKey 子函数：yy == xx + 1[c]。 消息长度正常。 键确定。	
			接收到SecurityAccess requestSeed。 消息长度NOK。	
			接收到SecurityAccess的sendKey。	传输否定响应NRC 0x24。
			接收到SecurityAccess requestSeed。 *消息长度正常。* *未满足可选的前提条件*[d]。	
			接收到SecurityAccess requestSeed。 消息长度正常。 延迟未过期（如适用）。 *可选的前提条件已满足。*	
			根据一般否定响应处理，SecurityAccess请求会产生一般的否定响应代码（例如，支持的最小长度，子功能）（请参阅第7.5节）。	传输7.5节定义的否定响应代码。
			接收到SecurityAccess requestSeed。 Static_Seed == False。	
			接收到SecurityAccess requestSeed。 Static_Seed == True。 请求的securityAccessType具有活动的存储种子。	
			接收到SecurityAccess requestSeed。 Static_Seed == True。 请求的securityAccessType没有存储活动的种子（不同于以前的securityAccessType）。	

表I.2 - （续）

没有。	手术		条件	行动
6			DiagnosticSessionControl接受或发生会话超时。	开始适当的诊断会话。 锁定ECU。
			接收到SecurityAccess的sendKey。	传输否定响应NRC 0x24。
			接收到SecurityAccess requestSeed。	
			请求的级别解锁。	
			根据一般否定响应处理，SecurityAccess请求会产生一般的否定响应代码（例如，支持的最小长度，子功能）（请参阅第7.5节）。	传输7.5节定义的否定响应代码。
			接收到SecurityAccess requestSeed。	
			请求的级别没有解锁。	
			消息长度正常。	
			可选的前提条件已满足。	
			延迟过期（如果适用）。	
			接收到SecurityAccess的sendKey。	
			子函数：yy == xx + 1。	
			消息长度正常。	
			关键NOK。	
			（ATT_CNT + 1） < Att_Cnt_Limit（如果适用）。	
			接收到SecurityAccess的sendKey。	
			子函数：yy == xx + 1。	
			消息长度正常。	
			关键NOK。	
			（Att_Cnt + 1）> = Att_Cnt_Limit。	
			接收到SecurityAccess的sendKey。	
			子函数：yy <> xx + 1。	
			接收到SecurityAccess的sendKey。	
			子函数：yy == xx + 1。	
			消息长度NOK。	
		和	DiagnosticSessionControl接受或发生会话超时。	开始适当的诊断会话。
			根据一般否定响应处理，SecurityAccess请求会产生一般的否定响应代码（例如，支持的最小长度，子功能）（请参阅第7.5节）。	传输7.5节定义的否定响应代码。

表I.2 - （续）

没有。	手术		条件	行动
			接收到SecurityAccess requestSeed。 请求的级别解锁。	
			接收到SecurityAccess的sendKey。 子功能：yy == xx + 1[h]。 消息长度正常。 键确定。	
			接收到SecurityAccess的sendKey。 子函数：yy == xx + 1。 消息长度正常。 关键NOK。 （Att_Cnt < Att_Cnt_Limit（if + 1）适用）。	
			接收到SecurityAccess的sendKey。 子函数：yy == xx + 1。 消息长度正常。 关键NOK。 （Att_Cnt + 1）> = Att_Cnt_Limit。	
			接收到SecurityAccess的sendKey。 子函数：yy <> xx + 1。	
			接收到SecurityAccess的sendKey。 子函数：yy == xx + 1。 消息长度NOK。	
			根据一般否定响应处理，SecurityAccess请求会产生一般的否定响应代码（例如，支持的最小长度，子功能）（请参阅第7.5节）。	传输7.5节定义的否定响应代码。

[a] 默认使用案例的延迟时间将具有固定值，但对于客户特定的使用案例，如果值存储在非易失性存储器中，则该值取决于错误访问尝试次数。

[b] 由于长度取决于子功能（即，requestSeed的长度与sendKey消息的长度不同），因此只能在评估子功能后才能进行确切的长度检查。在一般服务评估过程中检查最小长度。

[c] sendKey子函数（yy）必须是预期的securityAccessType（活动的存储种子用于相应的requestSeed securityAccessType，即sendKey securityAccessType - 1）。

[d] 可以检查客户特定的前提条件（例如，在此驾驶循环中写入的指纹，发动机不运转，车辆不移动）。

[e] 一旦给定的安全级别被解锁，即使在接收到针对不同安全级别的种子请求之后，它也将保持未锁定状态，直到新的安全级别完全解锁或由于其他原因退出安全访问（例如DiagnosticSessionControl已接受或会话超时发生）。

[f] 对于错误访问尝试的计数器将随着每个有效的格式化而增加，但是无效的密钥值将被增加，并且在接收到有效密钥的情况下将被设置为零。 如果需要启动延迟或延迟时间（甚至可能延迟时间取决于此计数器的值），则可能是客户特定的用例将此计数器存储在非易失性存储器中，以便能够在复位后进行判断。 如果接收到有效的格式化密钥，则丢弃存储的种子。

g 一旦给定的安全级别被解锁，即使在接收到针对不同安全级别的种子请求之后，它仍将保持未锁定状态，直到新的安全性完全解锁或由于其他原因退出安全访问（例如，接收到diagnosticSessionControl）。

h sendKey子函数必须是预期的访问类型（活动的存储种子用于sendKey accessType - 1）。

i 对于错误访问尝试的计数器将随着每个有效的格式化而增加，但是无效的密钥值将被增加，并且在接收到有效密钥的情况下将被设置为零。 如果需要启动延迟或延迟时间（甚至可能延迟时间取决于此计数器的值），则可能是客户特定的用例将此计数器存储在非易失性存储器中，以便能够在复位后进行判断。 如果接收到有效的格式化密钥，则丢弃存储的种子。

j 对于错误访问尝试的计数器将随着每个有效的格式化而增加，但是无效的密钥值将被增加，并且在接收到有效密钥的情况下将被设置为零。 如果需要启动延迟或延迟时间（甚至可能延迟时间取决于此计数器的值），则可能是客户特定的用例将此计数器存储在非易失性存储器中，以便能够在复位后进行判断。 如果接收到有效的格式化密钥，则丢弃存储的种子。

注意必须考虑的是，当通过多个连接OR来定义状态转换时，每个连接都应用一个动作，使得一次只有一个连接的连接成为真，并强制进行状态转换只对所定义的特定状态转换执行一个动作（例如，仅传送单个否定响应）。

附件J.
（资料）

推荐实施多个客户端环境

J.1 介绍

本附件旨在解决越来越多的使用案例，其中诊断车辆拓扑结构通过将单个诊断客户端（外部测试设备）和多个服务器（车辆中的ECU）添加到基本诊断拓扑中的一个或多个板载诊断客户端而得到扩展， 。

本文档和此处的规范性参考文献不限制服务器可以支持的诊断通信通道的数量。 这种用于多客户端处理的服务器实现的设计需要考虑到存在规定和限制，这些规定和限制迫使某些诊断客户端以比其他优先级更高的优先级来服务，例如，以满足现有的立法OBD要求。 在这种情况下，车辆系统设计需要确保并行客户端请求可以由相应的服务器处理。

这种情况的一个例子是内部数据记录器，该记录器与外部连接到诊断连接器的OBD扫描工具并行连接到服务器。

总体车辆设计要考虑这种并行处理客户端请求的情况（例如网关仲裁机制），或者各个服务器必须实施新的策略以将可用资源分配给不同的客户端。 在服务器中，协议实现或可用资源是唯一的，一次只能由一个客户端访问。

本附录仅介绍服务器级别的实现。 车辆制造商有责任选择最适合其个人需求的机构。

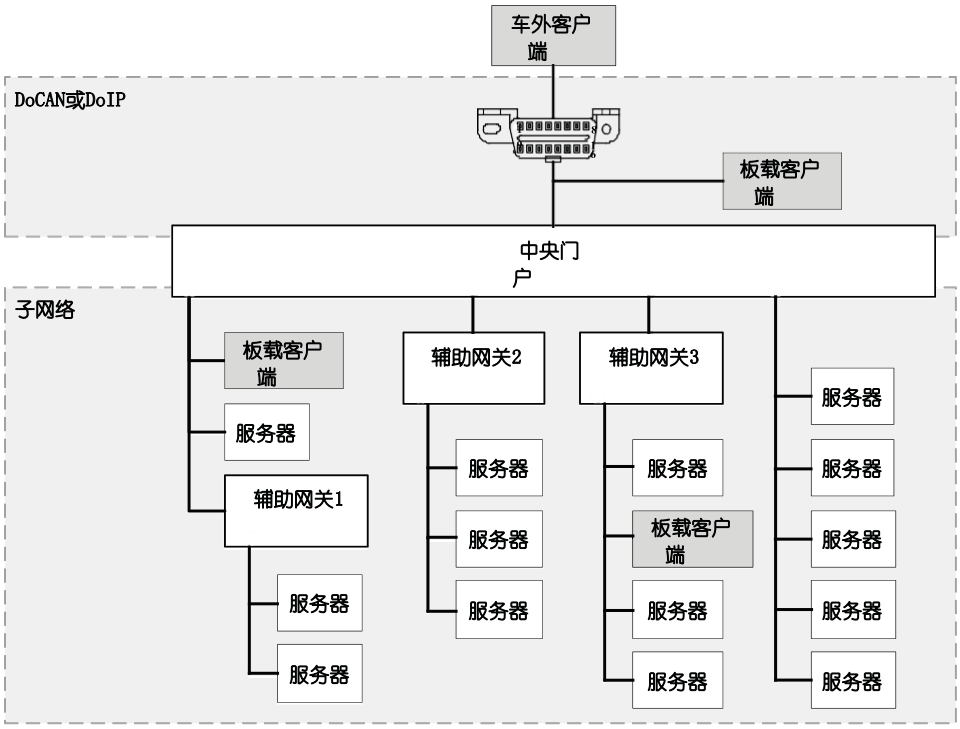
J.2 实施的具体限制

必须为每个通信参与者分配一个唯一的地址信息，以允许检测不同的客户端，然后可以使用这些客户端来限制功能或分配优先级。

如果车辆制造商的设计没有为某些对等协议实体使用唯一的地址信息，则本附录中描述的实施不适用。 在这种情况下，车辆设计需要确保处理多个客户的所选方法符合立法要求。

J.3 与系统设计相关的用例

图J.1显示了存在多个客户端的车辆拓扑结构示例。



图J.1 – 车载客户端的车辆拓扑示例

本文档中描述的实现旨在实现图J.1中总结的用例。 下表中标记为'N / A'的所有用例情景均不作为本标准的一部分进行描述。 强烈建议避免这种情况。 本附录中规定的实施和设计规则并非旨在支持超出表J.1中定义的方案的OBD通信要求。

表J.1 – 需要解决多个客户端场景的用例（UC）矩阵

测验设备 正在使用 额外 测验设备	非车载客户 （车辆外部测试设备）		机载客户端 （车辆内部测试设备）		
	OBD扫描（工具）测试设备	OEM服务测试设备	车载客户端1	车载客户端2	机载客户端n
OBD扫描（工具）测试设备	不存在	N/A	A（UC 1）	A（UC 1）	A（UC 1）
OEM服务测试设备	N/A	不存在	X（UC 2）	X（UC 2）	X（UC 2）
车载客户端1	T（UC 3）	X（UC 4）	不存在	X（UC 5）	X（UC 5）
车载客户端2	T（UC 3）	X（UC 4）	X（UC5）	不存在	X（UC 5）
…	…	…	…	…	…
机载客户端n	T（UC3）	X（UC 4）	X（UC5）	X（UC5）	不存在
T：使用的测试设备具有比其他测试工具更高的特性 答：附加测试设备的优先级高于使用中的测试工具X：车辆制造商特定（相同或不同优先级）					

当提及术语“正在使用的测试设备”时，需要区分服务器和客户端的角度，如下所示：

— 从服务器角度来看，如果当前正在处理请求或非默认会话处于活动状态，则会考虑使用测试工具

— 从客户的角度来看，如果尚未收到预期响应，则认为测试工具正在使用中，P3客户端尚未过期或非默认会话处于活动状态

当提到“附加测试设备”一词时，适用下列定义：

— 如果使用其他工具，则在此情况下测试设备被视为“附加”（参考使用中测试工具的定义）

当提及术语“OBD扫描（工具）测试设备”时，以下定义适用：

— 根据SAE J1978 / ISO 15031-4，车载诊断（OBD）规定要求轿车和轻型，中型和重型卡车支持与车外测试设备通信最少的诊断信息。 如果与测试设备（例如，手持式扫描工具，基于PC的诊断计算机等）的通信不能按照适当的标准所定义的那样进行，则车辆被认为是不合规的。

当提及术语’OEM服务测试设备’时，以下定义适用：

— OEM指定的测试设备，满足OEM要求并使用专有地址信息。
OEM服务测试设备可以使用标准化部件，即SAE J2534在应用程序和服务工具硬件之间进行通信，但与车辆的通信使用OEM专有信息。

当提及“机载客户”一词时，以下定义适用：

— 可以包括至少一个诊断客户端部分但也可以包括诊断服务器部分的ECU。 客户端部分具有将诊断服务请求发送到车辆中的其他服务器的功能。 一个示例可以是可集成到还包括其他功能的ECU中的远程信息处理网关。 如果OEM服务工具连接到服务器，则Telematics网关将充当服务器

诊断连接器并且请求来自远程信息处理网关的数据，但远程信息处理网关本身也充当从车辆中的其他服务器请求数据的客户端。

J.4 用例评估：

表J.2旨在指导系统设计人员应该选择什么样的概念。

表J.2 – 评估多个客户使用案例

用例#	伪并行的概念	优先的概念
	优点： — 无需停止协议即可处理两种类型的测试设备 — 所有的客户端都会被服务器通知（最坏的情况是NRC 0x21，通常是正面的响应） — 如果OBD扫描（工具）测试设备客户端永久连接（第三方工具），则可以并行处理OBD和非OBD请求	**优点：** — 基于专用的优先级假设进行处理：OBD比机载客户端具有更高的灵活性 — 不需要资源管理 — 由于正在进行的非OBD响应将被停止，因此专用于OBD响应的定时行为 — 只需要一个单一的缓冲区
	缺点： — 需要资源管理 — 非OBD请求可能会被拒绝或甚至没有回应 — OBD II：如果物理OBD CAN ID用于UDS概念不起作用	**缺点：** — 客户端（车载测试设备）请求只能在当前未处理OBD请求时处理 — 使应用程序能够“杀死”来自其他客户端的持续请求 — 如果永久连接，则取决于请求频率，板载客户端是否仍然能够收集板载数据。
	优点： — 如果默认会话处于活动状态且协议参数相同，则基于到达时间处理客户端请求，而不停止协议 — 当服务器正忙于处理不同的请求或处于由不同客户端请求的非默认会话时，客户端被NRC 0x21通知	**优点：** — 基于专用优先级进行处理 — 允许优先考虑不同的客户
	缺点： — 如果客户端不请求非默认会话，则可以进行并行处理 — 客户端在进行数据检索时应始终请求默认会话	**缺点：** — 低prio客户端不知道它不会被送达的事实 — 仅通过超时检测（$P2_{最大}$超时） — 使应用程序能够“杀死”来自其他客户端的持续请求

J.5 多个客户端服务器级别实施

J.5.1 诊断协议的定义

在这种情况下，诊断通信协议是取决于地址信息（例如，协议缓冲区大小，会话时间，支持的服务，安全级别）的特定参数值的汇编。

协议由对等协议实体之间建立的通信路径来标识。 每个对等协议实体具有恰好一个唯一的物理地址，以及0..n个功能地址（对于（一个或多个）服务器）由相应的N_AI标识。

注1 这意味着一个地址不能用于不同的协议。

笔记2： 正如ISO 14229的本部分所定义的那样，在一个特定的ECU中一次只有一个诊断会话状态和一个安全级别状态是活动的并且在所有活动协议上共享。

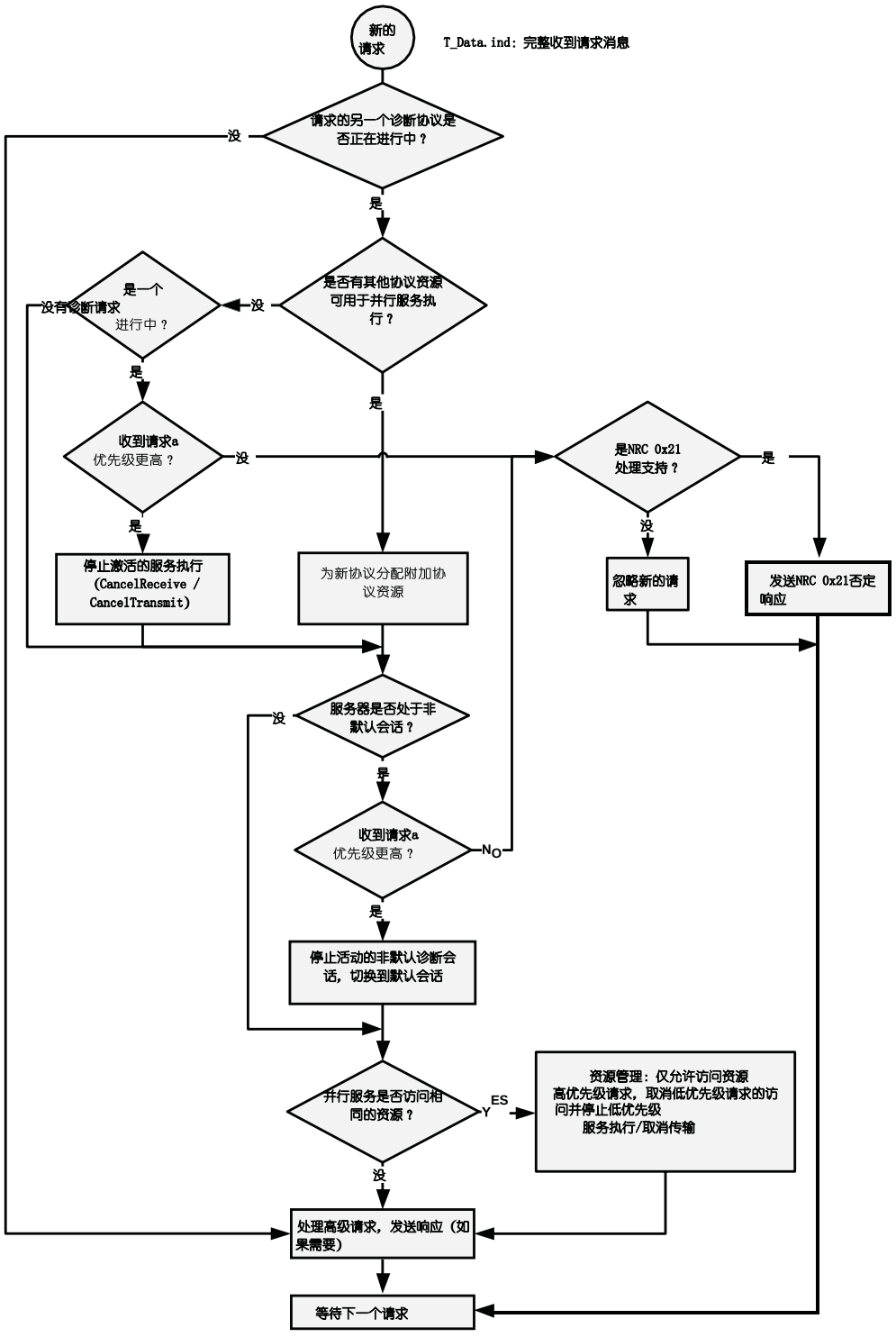
J.5.2 假设

一个协议可以有专有协议资源，也可以有多个协议可以共享一个协议资源。

OBD扫描（工具）测试设备客户端地址具有最高的优先级，或者为该地址分配专用协议资源，确保法规要求得以实现。

J.5.3 多个客户端处理流程

如果服务器在服务器级上实现多个客户机处理，则实施应遵循图J.2所示的流程图。



键

1　　溢出原因：第二个请求的临时接收缓冲区限制为一帧。

图J.2 – 多客户端处理流程

参考书目

[1] ISO 4092：1988 / Cor.1：1991，道路车辆 – 汽车诊断系统 – 词汇 – *技术勘误1*

[2] ISO / IEC 7498-1，信息技术 – 开放系统互连 – 基本参考模型：*基本模型*

[3] ISO / TR 8509：1987，信息处理系统 – 开放系统互连 – 服务惯例

[4] ISO / IEC 10731，信息技术 – 开放系统互连 – 基本参考模型 – *定义OSI服务的公约*

[5] ISO 11992-4，道路车辆 – 牵引车和牵引车之间电气连接的数字信息交换 – 第4部分：诊断

[6] ISO 14229-3，道路车辆 – 统一诊断服务（UDS） – 第3部分：CAN实施方面的统一诊断服务（UDSonCAN）

[7] ISO 14229-4，道路车辆 – 统一诊断服务（UDS） – 第4部分：FlexRay实施方面的统一诊断服务（UDSonFR）

[8] ISO 14229-5，道路车辆 – 统一诊断服务（UDS） – 第5部分：因特网协议实施的统一诊断服务（UDSonIP）[1)]

[9] ISO 14229-6，道路车辆 – 统一诊断服务（UDS） – 第6部分：K线实施的统一诊断服务（UDSonK-Line）

[10] ISO 14229-7，道路车辆 – 统一诊断服务（UDS） – 第7部分：本地互连网络实施（UDSonLIN）的统一诊断服务[2)]

[11] ISO 15031-2，道路车辆 – 用于排放相关诊断的车辆和外部设备之间的通信 – 第2部分：关于术语，定义，缩写和首字母缩略词的指导

[12] ISO 15031-6，道路车辆 – 用于排放相关诊断的车辆和外部设备之间的通信 – 第6部分：诊断故障代码定义

[13] ISO 15765-4，道路车辆 – 控制器区域网络（DoCAN）上的诊断通信 – *第4部分：排放相关系统的要求*

[14] ISO 22901-1，道路车辆 – 开放式诊断数据交换（ODX） – 第1部分：数据模型规范

[15] ISO 26021-2，道路车辆 – 机载烟火装置的报废激活 – 第2部分：*通信要求*

[16] ISO 27145-2，道路车辆 – 实施全球统一车载诊断（WWH-OBD）通信要求 – 第2部分：通用数据字典

---

1) 待出版。

2) 在准备之中。

[17] ISO 27145-3，道路车辆 - 实施全球统一车载诊断（WWH-OBD）通信要求 - 第3部分：通用消息字典

[18] SAE J1939：2011，串行控制和通信重型车辆网络 - 顶级文件

[19] SAE J1939-73：2010，应用层 - 诊断

[20] ANSI / IEEE标准754-1985，二进制浮点运算的IEEE标准

ICS 43.180

价格基于392页