ICS 25.040
N 10

中华人民共和国国家标准

GB/T 26790.4—2020

工业无线网络 WIA 规范 第 4 部分：WIA-FA 协议一致性测试规范

Industrial wireless networks WIA specification—Part 4：WIA-FA protocol conformance test specification

2020-07-21 发布　　2021-02-01 实施

国家市场监督管理总局
国家标准化管理委员会　发布

目　　次

前　言

GB/T 26790《工业无线网络 WIA 规范》已发布及计划发布以下 8 部分：

——第 1 部分：用于过程自动化的 WIA 系统结构与通信规范；

——第 2 部分：用于工厂自动化的 WIA 系统结构与通信规范；

——第 3 部分：WIA-PA 协议一致性测试规范；

——第 4 部分：WIA-FA 协议一致性测试规范；

——第 5 部分：WIA-PA 互操作性测试规范；

——第 6 部分：WIA-FA 互操作性测试规范；

——第 7 部分：WIA-PA 产品通用条件；

——第 8 部分：WIA 行业规范。

本部分为 GB/T 26790 的第 4 部分。

本部分按照 GB/T 1.1—2009 给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本部分由中国机械工业联合会提出。

本部分由全国工业过程测量控制和自动化标准化技术委员会(SAC/TC 124)归口。

本部分起草单位：中国科学院沈阳自动化研究所、机械工业仪器仪表综合技术经济研究所、北京科技大学。

本部分主要起草人：梁炜、张思超、万亚东、刘丹、王恺、胡男、齐悦、谢素芬、王沁。

工业无线网络 WIA 规范
第 4 部分：WIA-FA 协议一致性测试规范

1 范围

GB/T 26790 的本部分给出了 WIA-FA 一致性测试系统结构、现场设备测试集、接入设备测试集和网关设备测试集。

本部分适用于基于 GB/T 26790.2—2015 的无线网络设备的协议一致性测试。

2 规范性引用文件

下列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，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 26790.2—2015 工业无线网络 WIA 规范 第 2 部分：用于工厂自动化的 WIA 系统结构与通信规范

3 术语和定义

GB/T 26790.2—2015 界定的术语和定义适用于本文件。为了便于使用，以下重复列出了 GB/T 26790.2—2015 中的某些术语和定义。

3.1

接入设备 access device

安装在工业现场，负责将现场设备上的传感器数据、告警及网络管理相关信息转发到网关设备，或将网关设备的控制信号、管理信息和配置信息转发给现场设备。

[GB/T 26790.2—2015，定义 3.1.2]

3.2

信标 beacon

在 WIA-FA 网络中由接入设备广播的帧。

注：新的现场设备在加入 WIA-FA 网络前首先要监听信标。

[GB/T 26790.2—2015，定义 3.1.7]

3.3

现场设备 field device

安装在工业现场，连接传感器和执行器，负责发送现场数据和接收控制命令的 WIA-FA 设备。

[GB/T 26790.2—2015，定义 3.1.12]

3.4

网关设备 gateway device

连接 WIA-FA 网络与其他网络的设备。

[GB/T 26790.2—2015，定义 3.1.13]

3.5

超帧　superframe

一组周期性重复出现的信道和时隙集合。

注：超帧中时隙的数目决定了超帧循环的频率。

[GB/T 26790.2—2015，定义 3.1.30]

3.6

时隙　timeslot

在 WIA-FA 网络中交换数据所采用的基本时间单位。

注：WIA-FA 网络中的时隙长度是可配置的。

[GB/T 26790.2—2015，定义 3.1.31]

4　缩略语

下列缩略语适用于本文件。

AD：接入设备(Access Device)

AL：应用层(Application Layer)

AMPB：应用子层发布者状态机(ASL State Machine of Publisher)

AMRS：应用子层报告源点状态机(ASL State Machine of Report Source)

ASL：应用子层(Application Sub-layer)

ASN：绝对时隙号(Absolute Slot Number)

CCM*：增强的密码段链接消息验证编码协议计数器(Extension of counter with cipher block chaining message authentication code)

C/S：客户机/服务器(Client/Server)

DLL：数据链路层(Data Link Layer)

FCS：帧校验序列(Frame Check Sequence)

FD：现场设备(Field Device)

GACK：ACK 组(Group ACK)

GW：网关设备(Gateway Device)

HMAC：Hash 加密消息认证码(keyed-Hash Message Authentication Code)

KEDB：广播数据加密密钥(Broadcast Data Encryption Key)

KEDU：单播数据加密密钥(Unicast Data Encryption Key)

KEK：密钥加密的密钥(Key Encryption Key)

KJ：加入密钥(Join Key)

KS：共享密钥(Shared Key)

MIB：管理信息库(Mangement Information Base)

MIC：消息完整性代码(Message Integrity Code)

NACK：否定应答(Negative Acknowledgement)

P/S：发布者/预订者(Publisher/Subscriber)

R/S：报告源点/报告汇点(Report source/report Sink)

UAO：用户应用对象(User Application Object)

UAP：用户应用进程(User Application Process)

VCR：虚拟通信关系(Virtual Communication Relationship)

WIA-FA：用于工厂自动化的工业无线网络(Wireless Network for Industrial Automation-Factory

Automation)

5 WIA-FA 一致性测试系统结构

5.1 测试环境概述

WIA-FA 一致性测试的测试环境由测试系统和被测设备构成，其中被测设备分为三类：现场设备、接入设备和网关设备。为保证测试的通信质量，宜测试系统和被测设备的最大距离为 5 m，且测试环境中无其他同频无线干扰。三类设备的测试环境分别如图 1、图 2、图 3 所示。

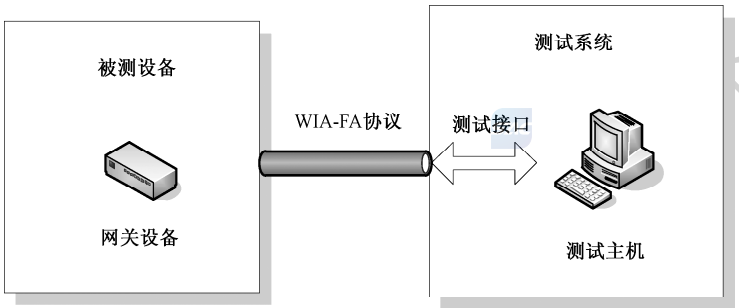

图 1 网关设备测试环境

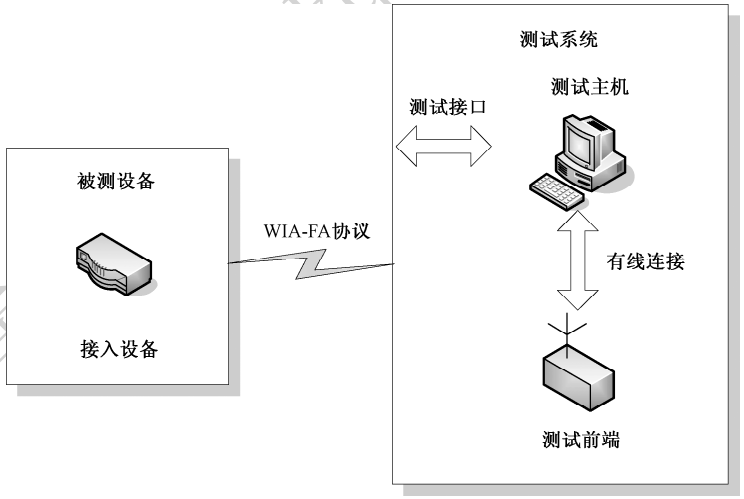

图 2 接入设备测试环境

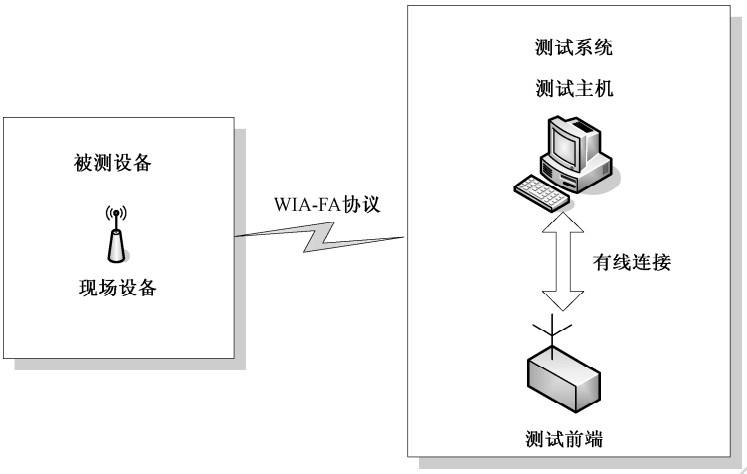

图 3　现场设备测试环境

其中：

——测试主机：按照规定的测试用例生成并发出相应的数据包或命令包，根据接收到的被测设备的数据包或命令包来判断被测设备行为是否正确。

——测试前端：与测试主机通过逻辑接口连接，作为测试主机与被测设备通信的无线收发设备。

——测试接口：连接被测接入设备或网关设备，测试有线服务。

——被测设备：接收测试主机发出的数据包或命令包并作出相应的响应，或者按照协议规定向测试系统主动发出请求。当被测设备是网关时，被测方应提供配套的上位机软件。

本部分规定测试系统和被测设备使用的参考信道为 6。

本部分规定测试系统和被测设备使用的参考网络 ID 为{0xaa}。

本部分规定测试中使用的加入密钥 KJ 为{0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7 0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf}，共享密钥 KS 为{0xa0 0xa1 0xa2 0xa3 0xa4 0xa5 0xa6 0xa7 0xa8 0xa9 0xaa 0xab 0xac 0xad 0xae 0xaf}。

本部分规定测试中使用加密算法为 CCM* 加密算法。

本部分规定测试前应已知被测设备的时间同步精度及时隙结构。

本部分规定测试中使用以下参考短地址：

——网关设备短地址：0x01；

——接入设备短地址：0x02；

——现场设备短地址：0x03；

——聚合测试中另一个现场设备短地址：0x04。

5.2　测试过程

WIA-FA 一致性测试包含以下过程：

——测试系统启动；

——被测设备启动；

——被测设备或测试系统加入网络；

——测试系统和被测设备执行测试用例；

——显示测试结果。

5.3 测试名称说明

WIA-FA 一致性测试用例可分为正向(positive)测试和反向(negative)测试,此外,有些测试用例用于测试可选功能:

——正向测试:测试系统发送正确的服务请求,检查被测设备是否返回指示服务成功执行的正响应;

——反向测试:测试系统发送带错误数据的服务请求,检查被测设备是否返回指示服务未成功执行的负响应以及检查负响应中的错误代码来判断被测设备是否正确操作;

——可选测试:如果被测设备具备该功能,则测试系统执行此测试用例;否则,测试系统将不执行此测试用例。

5.4 测试伪代码说明

5.4.1 函数说明

测试用例伪代码中使用的函数说明如表 1 所示。

表 1 函数说明

序号	函数名称	函数含义
1	Compare(Msgstring1, Msgstring2)	返回两个数据包的比较结果,如果一致返回 SUCCESS,否则返回 FAIL
2	wiredVerify(RcvPacket,RefPacket)	比较有线测试中的两个数据包是否一致 result = compare(packet, RefPacket) return result

3	Verify(RcvPacket, RefPacket, SecLevel = 0, Key=NULL)	比较无线测试中的两个数据包是否一致
		if(SecLevel == 0 ‖ 1)
		result = compare(RcvPacket, RefPacket);
		else
		packet = SecIn(RcvPacket, SecLevel ,Key)
		result = compare(packet, RefPacket)
		return result

4	Printscreen(MsgString)	在屏幕打印字符串 MsgString
5	wiredSend(SendPacket)	通过有线发送一个数据包

6	Send(SendPacket, SecLevel, Key)	发送帧或包
		if (SecLevel == 0 ‖ 1)
		send_packet(SendPacket);
		else
		send_packet(SecOut(SendPacket,SecLevel,Key))

7	Receive(RcvPacket)	接收帧或包,返回一个指向接数据的指针
8	Max(data1,data2)	取 data1 与 data2 的最大值
9	Length(Packet)	取 Packet 的实际长度,以字节为单位

表 1（续）

序号	函数名称	函数含义

10	SecIn(frame, SecLevel, Key)	对接收到的安全帧根据据 SecLevel 参数，利用 Key 进行解密和/或 MIC 校验，返回一个解密和/或 MIC 校验后的数据包指针
		if (SecLevel== 0 ‖ 1)
		newframe = frame;
		else if(SecLevel == 2 ‖ 6)
		check = MIC32_check(frame, Key);
		else if(SecLevel == 3 ‖ 7)
		check = MIC64_check(frame, Key);
		else if (SecLevel == 4 ‖ 8)
		check = MIC128_check(frame, Key);
		if(SecLevel == 5 ‖ 6 ‖ 7 ‖ 8)
		newframe = decrypt(frame, Key);
		if (check == FALSE)
		return NULL;
		else
		return newframe;

11	SecOut(frame, SecLevel,Key=NULL)	对要发送的帧根据 SecLevel 参数，利用 Key 进行加密和/或 MIC 计算，返回一个加密和/或 MIC 填充后的数据包指针
		if(SecLevel == 5 ‖ 6 ‖ 7 ‖ 8)
		newframe = encrypt(frame, Key);
		if (SecLevel== 0 ‖ 1)
		newframe = frame;
		else if(SecLevel == 2 ‖ 6)
		newframe = MIC32_fill(newframe, Key);
		else if(SecLevel == 3 ‖ 7)
		newframe = MIC64_fill(newframe, Key);
		else if (SecLevel == 4 ‖ 8)
		newframe = MIC128_fill(newframe, Key);
		return newframe;

其中，SecIn(frame, Key)和 SecOut(frame, Key)指定为 CCM* 模式(参见 IEEE 802.15.4-2011 附录 B)。

5.4.2 语句说明

测试用例伪代码中使用的语句说明如表 2 所示。

表 2　语句说明

序号	语句名称	表达形式	用法说明

1	IF 语句	IF(表达式)
		{
		语句 1
		}
			如果表达式为真,则执行语句 1

2	IF…ELSE 语句	IF(表达式)
		{
		　语句 1
		}
		ELSE
		{
		　语句 2
		}
			如果表达式为真,则执行语句 1,否则执行语句 2

3	IF…ELSE IF…ELSE 语句	IF(表达式 1)
		{
		　语句 1
		}
		ELSE IF(表达式 2)
		{
		　语句 2
		}
		ELSE
		{
		　语句 3
		}
			如果表达式 1 为真,则执行语句 1;如果表达式 2 为真,则执行语句 2;否则执行语句 3

4	WHILE 语句	WHILE(表达式) for Duration	如果未超时,则判断表达式,若表达式为真,则继续等待;若表达式为假,则语句结束;如果超时,则语句结束
5	FOR	FOR(表达式)	

6	BREAK 语句	WHILE(表达式) for Duration
		{
		　语句 1
		　BREAK;
		}
		语句 2
			跳出当前 WHILE 循环执行语句 2

5.4.3　变量说明

测试用例伪代码中使用的变量说明如表 3 所示。

表 3　变量说明

序号	符号表示	变量含义	取值范围	缺省值
1	TimeSlot	时隙长度	—	1ms
2	TestCaseResult	字符串变量,用来表示测试结果	—	N/A
3	MaxScanTime	网络发现的最大扫描时间	大于或等于 32×TimeSlot	32×TimeSlot
4	DevStaRptCycle	设备状态报告周期	大于或等于 32×TimeSlot	32×TimeSlot
5	ChaStaRptCycle	设备信道状态报告周期	大于或等于 32×TimeSlot	32×TimeSlot
6	SuperframeCycle	设备超帧周期	大于或等于 32×TimeSlot	32×TimeSlot
7	Count	计数值	从 0 开始的整数	3
8	CycleError	周期偏差,即实际周期与标准周期的最大差值	大于或等于 TimeSlot	TimeSlot
9	MaxRetryTime	最大重传等待时间	大于或等于 TimeSlot	TimeSlot
10	MaxWaitTime	响应最大等待时间	大于或等于 32×TimeSlot	32×TimeSlot
11	RefPacket	参考包格式	—	N/A
12	SendPacket	发送数据包	—	—
13	RcvPacket	接收数据包	—	—

5.4.4　常量说明

测试用例伪代码中使用的常量说明如表 4 所示。

表 4　常量说明

序号	符号表示	变量含义	取值
1	SUCCESS	返回值为真	1
2	FAILED	返回值为非真	0

5.4.5　组成部分说明

测试用例伪代码的组成部分说明如表 5 所示。

表 5　组成部分说明

序号	所属部分	中文名	作用
1	TEST BODY	测试主体	测试行为的伪代码描述
2	TEST RESULT	测试结果	TEST RESULT 测试结果包括 SUCCESS(测试通过)和 FAILED(测试未通过)

5.4.6　符号使用说明

测试用例伪代码符号使用说明如表 6 所示。

表 6　符号使用说明

序号	符号	符号名称	说明
1	==	等于运算符	判断运算符两侧相等为真,不等为假
2	! =	不等于运算符	判断运算符两侧相等为假,不等为真
3	&&	与运算符	判断运算符两侧同时为真,则为真;不同时为真,则为假
4	&	按位与	二进制按位与
5	=	赋值运算符	将运算符右侧变量赋值给左侧变量
6	“ ”	字符串	引号内表示字符串
7	( )	圆括号	括号具有最高运算优先级

5.4.7　包表示法规则

测试用例伪代码包表示法规则如表 7 所示。

表 7　包表示法规则

包表示	含义
Packet.all	表示包的全部字段
Packet.type	表示包的类型
Packet.header	表示包头
Packet.payload	表示包的载荷
Packet.*	表示包中的任意一个字段

表 7 中的 Packet 为 WIA-FA 标准(见 GB/T 26790.2—2015)中定义的包/帧,具体如表 8 所示。

表 8　测试中的包或帧

测试中的包或帧	WIA-FA 中的帧或包
Beacon	信标帧(Beacon)
JoiningRequest	加入请求
JoiningResponse	加入响应
AttributeSettingRequest	远程配置属性请求
AttributeSettingResponse	远程配置属性响应
AttributeGettingRequest	远程读属性请求
AttributeGettingResponse	远程读属性响应
TimeSynchronizeRequest	双向时间同步请求
TimeSynchronizeResponse	双向时间同步响应
DeviceConditionReport	设备状态报告
ChannelConditionReport	信道状态报告
LeavingRequest	离开请求
LeavingResponse	离开响应
NACK	否定应答帧

表 8（续）

测试中的包或帧	WIA-FA 中的帧或包
GACK	GACK
Data	数据包
KeyEstablishRequest	密钥分发请求
KeyEstablishResponse	密钥分发响应
KeyUpdateRequest	密钥更新请求
KeyUpdateResponse	密钥更新响应
WiredADJoinRequest	有线接入设备加入请求
WiredADJoinResponse	有线接入设备加入响应
WiredIndicatingNACK	有线网关设备指示接入设备发送 NACK
WiredIndicatingGACK	网关设备有线指示接入设备发送 GACK
WiredDataRequest	有线数据发送请求
WiredDataIndication	有线数据发送指示
WiredFDJoinConfirm	有线设备加入指示
WiredFDJoinResponse	有线设备加入响应
WiredDeviceStatusIndication	有线设备状态报告指示
WiredChannelConditionIndication	有线信道状况报告指示
WiredAttributeGettingRequest	有线远程读属性请求
WiredAttributeGettingConfirm	有线远程读属性证实
WiredAttributeSettingRequest	有线远程配置属性请求
WiredAttributeSettingConfirm	有线远程配置属性证实
WiredLeaveRequest	有线设备离开请求
WiredLeaveResponse	有线设备离开响应
UAPReadRequest(AttributeName)	UAP 读 AttributeName 请求
UAPReadResponse(AttributeName)(+)	UAP 读正响应
UAPReadResponse(AttributeName)(-)	UAP 读负响应
UAPWriteRequest(AttributeName)	UAP 写 AttributeName 请求
UAPWriteResponse(AttributeName)(+)	UAP 写正响应
UAPWriteResponse(AttributeName)(-)	UAP 写负响应
UAPPublishRequestMessage	UAP 发布请求
UAPReportRequestMessage	UAP 报告请求
UAOReportAckResonse(+)	UAP 报告确认正响应
UAOReportAckResonse(-)	UAP 报告确认负响应
ASLData(DstAddr, ServiceID, UAP_ID, Priority, AsduLength, Asdu)	ASL 请求发送数据包
ASLDataResponse(PacketControl, UAP_ID, Asdulength, Asdu)	ASL 收到的数据响应包
SecAlarmRequest	安全告警请求包
WiredSecAlarmRequest	有线安全告警请求包

5.5 参考数据包符号说明

测试用例参考数据包符号说明如表 9 所示。

表 9 参考数据包符号说明

符号表示	符号含义
0x	十六进制表示法
0b	二进制表示法[a]
*	保留位(默认值为 0)
(n)	n 表示字节数
…	省略字节
[a] 二进制数据编写顺序为 b7,b6…b0, WIA-FA 设备采用大端模式。	

6 现场设备测试集

6.1 加入过程测试集

6.1.1 现场设备加入网络(正向测试)[FD-JOIN001]

该测试用例测试现场设备能否正确加入 WIA-FA 网络。

测试过程为:

a) 测试系统向被测设备发送信标;

b) 被测设备接收到信标后,向测试系统发送加入请求;

c) 测试系统将接收的加入请求报文与期望的报文进行比对,如果比对匹配,则测试通过,并向被测设备返回加入响应。

具体时序如图 4 所示,具体测试说明如表 10 所示。

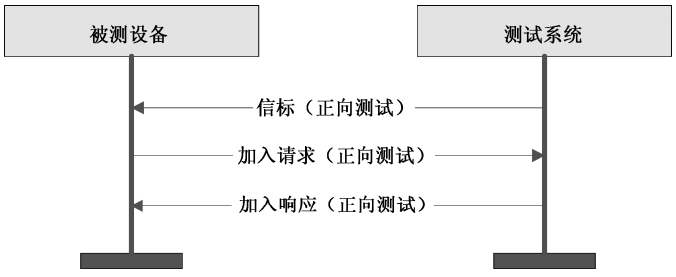

图 4 现场设备加入网络(正向测试)时序图

表 10 现场设备加入网络(正向测试)说明

用例名称	现场设备加入网络(正向测试)[FD-JOIN001]
被测设备	现场设备
依赖测试条件 SecLevel=0	测试系统已登记被测设备的长地址 测试系统已完成被测设备的网络 ID 配置 被测设备未加入测试系统网络 KS=NULL

表 10（续）

依赖测试条件 SecLevel＝1	测试系统已登记被测设备的长地址 测试系统已完成被测设备的网络 ID 和 SecLevel 设置 测试系统已完成被测设备的加入密钥 KJ 配置 被测设备未加入测试系统网络 KS＝NULL
依赖测试条件 SecLevel＝2～8	测试系统已登记被测设备的长地址 测试系统已完成被测设备的网络 ID 和 SecLevel 设置 测试系统已完成被测设备的加入密钥 KJ 和共享密钥 KS 配置 被测设备未加入测试系统网络

测试用例伪代码描述	TEST BODY：
	SendPacket1 = Beacon；
	SendPacket2 = JoinResponse；
	RefPacket =JoinRequest；
	Send(SendPacket1, SecLevel, KS)；
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxScanTime；
	IF(RcvPacket.type == RefPacket.type)
	{
	TestCaseResult == SUCCESS
	IF(SecLevel ! =0)
	IF(Compare(RcvPacket.secMaterial,
	0xFFFFFFFF&(HMAC(KJ,RcvPacket.PhyAddress))! =SUCCESS)
	{
	TestCaseResult =FAILED；
	}
	IF (TestCaseResult == SUCCESS&&Verify(RcvPacket.all, RefPacket.all, SecLevel, KS) == SUCCESS)
	{
	Send(SendPacket2, SecLevel, KS)；
	Printscreen(“JoiningNetwork Test Success!”)；
	TestCaseResult =SUCCESS；
	}
	ELSE
	{
	Printscreen(“JoinRequest Payload error!”)；
	TestCaseResult =FAILED；
	}
	}
	ELSE
	{
	Printscreen(“No JoinRequest received!”)；
	TestCaseResult = FAILED；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

表 10（续）

参考数据包 SecLevel=0	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x00 0x11 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Join Response 帧：0x06 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x02 | 0x00 | 0x03 | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(8)|序列号(2)|帧长度(2)|加入状态(1)|分配的短地址(1)|FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Join Request 帧：0x05 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x00 | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(8)|序列号(2)|帧长度(2)|FCS(2)
参考数据包 SecLevel=1，5	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x00 0x11 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x?? 0x??…0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Join Response 帧：0x06 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x02 | 0x00 | 0x03 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(8)|序列号(2)|帧长度(2)|加入状态(1)|分配的短地址(1)| FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Join Request 帧：0x05 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x08 | 0x?? … 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(8)|序列号(2)|帧长度(2)|安全材料(8)| FCS(2)

表 10（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80	0xaa	0xff	0x?? 0x??	0x00 0x11	0x?? 0x??	0x?? 0x??	0x?? 0x??	0x?? 0x??	0x??	0x?? … 0x??	0x??…0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)| MIC(n)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Join Response 帧：0x06	0xaa	0x?? … 0x??	0x?? 0x??	0x00 0x02	0x00	0x03	0x??…0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(8)|序列号(2)|帧长度(2)|加入状态(1)|分配的短地址(1)| MIC(n)| FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Join Request 帧：0x05	0xaa	0x?? … 0x??	0x?? 0x??	0x00 0x08	0x?? … 0x??	0x??…0x??	0x?? 0x?? 帧域说明：帧控制(1)	网络 ID(1)	源地址(8)	序列号(2)	帧长度(2)	安全材料(8)	MIC(n)	FCS(2)

6.1.2 现场设备加入网络（反向测试）[FD-JOIN002]

该测试用例测试现场设备能否加入 WIA-FA 网络。

测试过程为：

a） 测试系统向被测设备发送信标。

b） 被测设备接收到信标后，向测试系统发送加入请求。

c） 测试系统将接收的加入请求报文与期望的报文进行比对，如果比对匹配，则向被测设备返回错误的加入响应，具体情况如下：

1） 网络 ID 不匹配；

2） 认证失败（SecLevel 不为 0 时）；

3） 网络规模超限。

参考数据包中具体载荷内容如表 11 所示。

表 11 现场设备加入网络（反向测试）参考数据包载荷

测试内容	测试系统应发加入响应数据载荷
网络 ID 不匹配	0x01	0x03
	加入状态(1)|分配的短地址(1)

表 11（续）

测试内容	测试系统应发加入响应数据载荷
认证失败	0x02 | 0x03
	加入状态(1)|分配的短地址(1)
网络规模超限	0x03 | 0x03
	加入状态(1)|分配的短地址(1)

d） 测试系统向被测设备发送远程配置属性（超帧）请求，被测设备接收到请求后，不向测试系统返回远程配置属性（超帧）响应，则测试通过。

该测试用例用于现场设备入网的三种反向测试，测试体应循环执行，具体时序如图 5 所示，具体测试说明如所示，具体测试说明如表 12 所示。

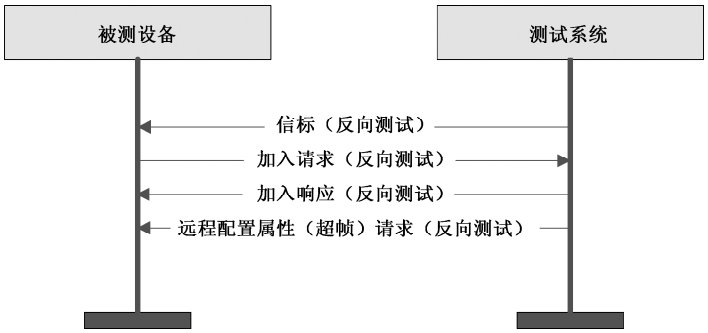

图 5 现场设备加入网络（反向测试）时序图

表 12 现场设备加入网络（反向测试）说明

用例名称	现场设备加入网络（反向测试）[FD-JOIN002]
被测设备	现场设备
依赖测试条件 SecLevel＝0	测试系统已登记被测设备的长地址 测试系统已完成被测设备的网络 ID 配置 被测设备未加入测试系统网络 KS＝NULL
依赖测试条件 SecLevel＝1	测试系统已登记被测设备的长地址 测试系统已完成被测设备的网络 ID 和 SecLevel 设置 测试系统已完成被测设备的加入密钥 KJ 配置 被测设备未加入测试系统网络 KS＝NULL
依赖测试条件 SecLevel＝2～8	测试系统已登记被测设备的长地址 测试系统已完成被测设备的网络 ID 和 SecLevel 设置 测试系统已完成被测设备的加入密钥 KJ 和共享密钥 KS 配置 被测设备未加入测试系统网络

表 12（续）

测试用例伪代码描述	TEST BODY:
	SendPacket1 = Beacon;
	SendPacket2 = JoinResponse;
	SendPacket3 = AttributeSettingRequest(Superframe);
	RefPacket1 =JoinRequest;
	RefPacket2 =AttributeSettingResponse;
	Send(SendPacket1, SecLevel, KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxScanTime;
	IF(RcvPacket .type == RefPacket1.type)
	{
	TestCaseResult == SUCCESS
	IF (SecLevel ! =0)
	IF (Compare (RcvPacket. secMaterial, 0xFFFFFFFF& (HMAC (KJ, RcvPacket. PhyAddress))! =SUCCESS)
	{
	TestCaseResult =FAILED;
	}
	IF (TestCaseResult ==SUCCESS &&Verify(RcvPacket.all, RefPacket1.all, SecLevel, KS) ! = SUCCESS)
	{
	Printscreen("JoinRequest Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket2, SecLevel, KS);
	Printscreen("JoinRequest Payload correct!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No JoinRequest received!");
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	Send(SendPacket3, SecLevel, KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket .type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all, SecLevel, KS) == SUCCESS)
	{
	Printscreen("JoiningNetwork Test Failed!");
	TestCaseResult =FAILED;
	}
	}
	ELSE

表 12（续）

测试用例伪代码描述	{
	Printscreen(“JoiningNetwork Test Success!”);
	TestCaseResult = SUCCESS;
	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0	测试系统应发数据包： 数据包 1： 协议层:DLL 帧名称:Beacon 帧：0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x00 0x11 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Join Response 帧：0x06 | 0xaa | 0x?? ... 0x?? | 0x?? 0x?? | 0x00 0x02 | 0x?? 0x03 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(8)|序列号(2)|帧长度(2)|载荷(2)|短地址(1)|FCS(2) 数据包 3： 协议层：DLL 帧名称:Attribute Setting Request 帧:0x8f | 0xaa | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x80 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层:DLL 帧名称：Join Request 帧：0x05 | 0xaa | 0x?? ... 0x?? | 0x?? 0x?? | 0x00 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(8)|序列号(2)|帧长度(2)|FCS(2) 数据包 2： 协议层:DLL

表 12（续）

参考数据包 SecLevel=0	帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x80 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2)
参考数据包 SecLevel=1,5	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x00 0x11 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Join Response 帧：0x06 | 0xaa | 0x?? ... 0x?? | 0x?? 0x?? | 0x00 0x02 | 0x?? 0x03 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(8)|序列号(2)|帧长度(2)|载荷(2)| FCS(2) 数据包 3： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f | 0xaa | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x80 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)| FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Join Request 帧：0x05 | 0xaa | 0x?? ... 0x?? | 0x?? 0x?? | 0x00 0x08 | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(8)|序列号(2)|帧长度(2)|安全材料(8)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x80 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2)

表 12（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x00 0x11 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x??… 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|MIC(n)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Join Response 帧：0x06 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x02 | 0x?? 0x03 | 0x??…0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(8)|序列号(2)|帧长度(2)|载荷(2)|MIC(n)|FCS(2) 数据包 3： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f | 0xaa | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x80 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x?? | 0x??… 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(n)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Join Request 帧：0x05 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x08 | 0x?? … 0x?? | 0x??…0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(8)|序列号(2)|帧长度(2)|安全材料(8)|MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x80 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x??… 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|MIC(n)| FCS(2)

6.1.3 双向时间同步测试[FD-JOIN003]

该测试用例测试现场设备能否正确进行双向时间同步。

测试过程为：

a） 测试系统向被测设备发送信标；

b） 被测设备接收到信标后，向测试系统发送双向时间同步请求；

c） 测试系统将接收的双向时间同步请求报文与期望的报文进行比对，如果比对匹配，则测试通过，并向被测设备返回双向时间同步响应。

具体时序如图 6 所示，具体测试说明如表 13 所示。

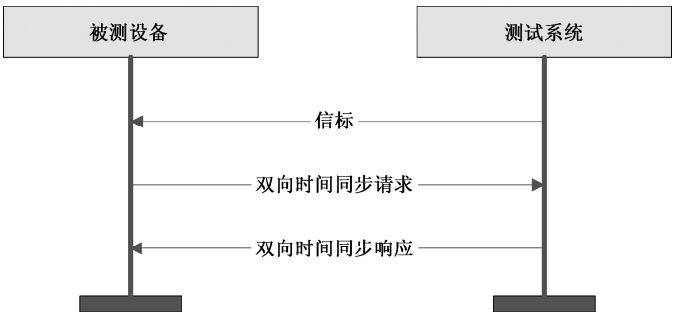

图 6　双向时间同步测试时序图

表 13　双向时间同步测试说明

用例名称	双向时间同步测试[FD-JOIN003]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已安全加入测试系统网络 测试系统已完成对被测设备 MIB 中的 TwoWayTimeSyn 参数配置 KS=NULL
依赖测试条件 SecLevel=2～8	被测设备已安全加入测试系统网络 测试系统已完成对被测设备 MIB 中的 TwoWayTimeSyn 参数配置

测试用例伪代码描述	TEST BODY:
	SendPacket1 = Beacon;
	SendPacket2 = TimeSynchronizeResponse;
	RefPacket =TimeSynchronizeRequest;
	Send(SendPacket1, SecLevel, KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxScanTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all, SecLevel, KS) ! = SUCCESS)
	{
	Printscreen("Time Synchronize Request Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket2, SecLevel, KS);
	Printscreen("Time Synchronize Request Test Success!");

表 13（续）

测试用例伪代码描述	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No Time Synchronize Request received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包: 数据包 1: 协议层:DLL 帧名称:Beacon 帧: 0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x00 0x11 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|FCS(2) 数据包 2: 协议层: DLL 帧名称: Time Synchronize Response 帧: 0x8c | 0xaa | 0x03 | 0x?? 0x?? | 0x000x10| 0x?? ... 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|现场设备发送时刻时间值(8)|接入设备接收时刻时间值(8)|FCS(2)
	测试系统应收数据包: 协议层:DLL 帧名称: Time Synchronize Request 帧: 0x8b | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x08| 0x?? ... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|现场设备发送时刻时间值(8)|FCS(2)
参考数据包 SecLevel=2,3,4,6,7,8	测试系统应发数据包: 数据包 1: 协议层:DLL 帧名称:Beacon 帧: 0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x00 0x11 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? ... 0x?? | 0x??... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|MIC(n)| FCS(2)

表 13（续）

参考数据包 SecLevel＝2，3，4，6，7，8	数据包 2： 协议层：DLL 帧名称：Time Synchronize Response 帧：0x8c	0xaa	0x03	0x?? 0x??	0x000x10| 0x?? … 0x??	0x?? … 0x??	0x??… 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|现场设备发送时刻时间值(8)|接入设备接收时刻时间值(8)| MIC(n)| FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Time Synchronize Request 帧：0x8b	0xaa	0x03	0x?? 0x??	0x00 0x08| 0x?? … 0x??	0x??… 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|现场设备发送时刻时间值(8)| MIC(n)| FCS(2)

6.1.4 超帧资源分配测试[FD-JOIN004]

该测试用例测试现场设备能否正确响应远程配置属性(超帧)请求。

测试过程为：

a) 被测设备加入网络后，测试系统向被测设备发送远程配置属性(超帧)请求；

b) 被测设备接收到请求后，向测试系统返回远程配置属性(超帧)响应；

c) 测试系统将接收的远程配置属性(超帧)响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 7 所示，具体测试说明如表 14 所示。

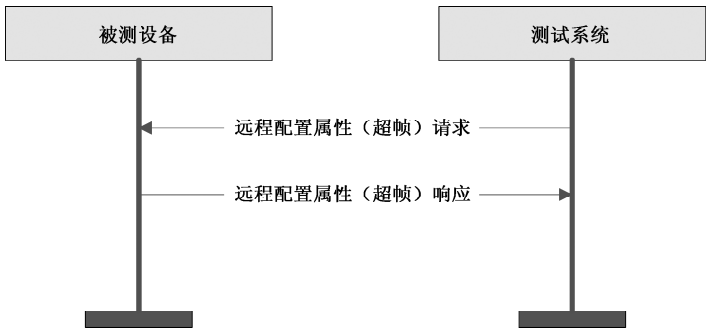

图 7 超帧资源分配测试时序图

表 14 超帧资源分配测试说明

用例名称	超帧资源分配测试[FD-JOIN004]
被测设备	现场设备
依赖测试条件 SecLevel＝0，1	被测设备已安全加入测试系统网络 KS＝NULL

表 14（续）

依赖测试条件 SecLevel＝2～8	被测设备已安全加入测试系统网络

测试用例伪代码描述	TEST BODY：
	SendPacket = AttributeSettingRequest(Superframe)；
	RefPacket =AttributeSettingResponse；
	Send(SendPacket，SecLevel，KS)；
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxScanTime；
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all，RefPacket.all，SecLevel，KS) ! = SUCCESS)
	{
	Printscreen(“Attribute Setting Response Payload error!”)；
	TestCaseResult =FAILED；
	}
	ELSE
	{
	Printscreen(“SuperframeResource Distributing Test Success!”)；
	TestCaseResult =SUCCESS；
	}
	}
	ELSE
	{
	Printscreen(“No Attribute Setting Response received!”)；
	TestCaseResult = FAILED；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

参考数据包 SecLevel = 0，1，5	测试系统应发数据包： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x80 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x80 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2)

表 14（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x00	0x80	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(n)| FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x00	0x80	0x??	0x?? 0x??	0x?? 0x??	0x00| 0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|MIC(n)| FCS(2)

6.1.5 链路资源分配测试[FD-JOIN005]

该测试用例测试现场设备能否正确响应远程配置属性(链路)请求。

测试过程为：

a) 被测设备加入网络后，测试系统向被测设备发送远程配置属性(链路)请求；

b) 被测设备接收到请求后，向测试系统返回远程配置属性(链路)响应；

c) 测试系统将接收的远程配置属性(链路)响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 8 所示，具体测试说明如表 15 所示。

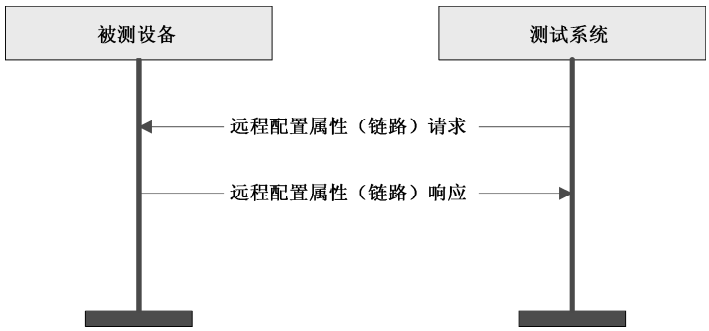

图 8 链路资源分配测试时序图

表 15　链路资源分配测试说明

用例名称	链路资源分配测试[FD-JOIN005]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已安全加入测试系统网络 KS=NULL
依赖测试条件 SecLevel=2～8	被测设备已安全加入测试系统网络

测试用例伪代码描述	TEST BODY:
	SendPacket =AttributeSettingRequest(Link);
	RefPacket =AttributeSettingResponse;
	Send(SendPacket, SecLevel, KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all, SecLevel, KS) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Response Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Link Resource Distributing Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Attribute Setting Response received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包: 协议层: DLL 帧名称:Attribute Setting Request 帧:0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x81 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明:帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|FCS(2)

表 15（续）

参考数据包 SecLevel = 0，1，5	测试系统应收数据包： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x81 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x81 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(n)| FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x81 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|MIC(n)| FCS(2)

6.1.6 读现场设备 UAO 测试（正向测试）[FD-JOIN006]

该测试用例测试现场设备能否正确响应 AL 的读（READ）服务。读现场设备 UAO 测试包括读现场设备设备表中的 NumOfSupUAO 属性以及读现场设备的 SupUAOList 属性。

测试过程为：

a） 被测设备加入网络后，测试系统向被测设备发送读（READ）请求，具体读取内容依次为：

1） DeviceList 中的 NumOfSupUAO 值；

2） SupUAOList 值。

b） 被测设备接收到请求后，向测试系统返回读（READ）正响应。

c） 测试系统将接收的读（READ）正响应与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 9 所示，具体测试说明如表 16 所示。

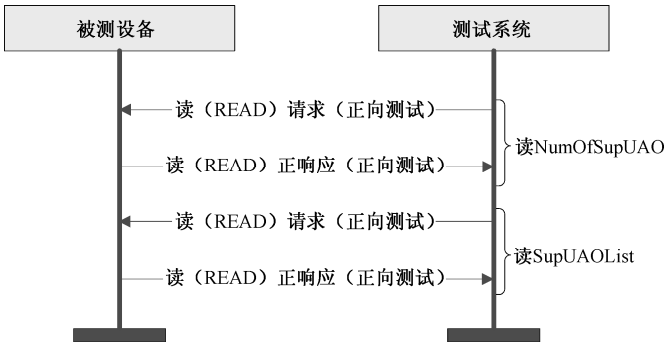

图 9　读现场设备 UAO 测试(正向测试)时序图

表 16　读现场设备 UAO 测试(正向测试)说明

用例名称	读现场设备 UAO 测试(正向测试)[FD-JOIN006]
被测设备	现场设备
依赖测试条件	被测设备已加入测试系统网络

测试用例伪代码描述	TEST BODY:
	SendPacket 1 = UAPReadRequest(NumOfSupUAO);
	SendPacket 2 = UAPReadRequest(SupUAOList);
	RefPacket 1 = UAPReadResponse(NumOfSupUAO);
	RefPacket 2 = UAPReadResponse(SupUAOList);
	Send(ASLDataRequest(DstAddr, 0x01, 0x00, 0x04, 0x05, SendPacket1));
	WHILE(Receive(ASLDataResponse).PacketControl & 0x1F ! = 0x04) for LossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F== 0x04)
	{
	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("ReadNumOfSupUAOPositiveResponse Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("ReadNumOfSupUAOPositiveTest Success!");
	Send(ASLDataRequest(DstAddr, 0x01, 0x00, 0x04, 0x05, SendPacket2));
	WHILE(Receive(ASLDataResponse).PacketControl & 0x1F ! = 0x04) for LossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F== 0x04)
	{
	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket2.all) ! = SUCCESS)
	{
	Printscreen("ReadSupUAOListPositiveResponse Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("ReadSupUAOListPositiveTest Success!");
	TestCaseResult =SUCCESS;

表 16（续）

测试用例伪代码描述	}
	}
	}
	}
	ELSE
	{
	Printscreen(“NoReadPositive Response received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发包： 数据包 1： 协议层：UAP 包名称 1:UAPReadRequest(NumOfSupUAO) 包：0x00 0x00 | 0x83 | 0x?? | 0x04 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1) 包名称 2：UAPReadRequest(SupUAOList) 包：0x00 0x00 | 0x86 | 0x00 | 0xff 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1) 数据包 2： 协议层：ASL 包名称:ASLDataRequest(DstAddr，ServiceID，UAP_ID，Priority，AsduLength，Asdu) 包 1：0x04 | 0x00 | 0x00 0x05 | 0x00 0x00 | 0x83 | 0x?? | 0x04 包 2：0x04 | 0x00 | 0x00 0x05 | 0x00 0x00 | 0x86 | 0x00 | 0xff 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|UAO 标识符(2)| 属性标识符(1)|存储索引 (1)|成员标识符(1)
	测试系统应收包： 数据包 1： 协议层:ASL 包名称 1：ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包 1:0x14 | 0x00 | 0x00 0x02 | 0x?? 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(2) 包 2：0x14 | 0x00 | 0x00 0x?? | 0x??... 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(n) 数据包 2： 协议层:UAP 包名称 1：UAPReadResponse(NumOfSupUAO)(+) 包：0x?? 0x?? 帧域说明:数据(2) 包名称 2：UAPReadResponse(SupUAOList) (+) 包：0x??... 0x?? 帧域说明:数据(n)

6.1.7 读现场设备 UAO 测试(反向测试)[FD-JOIN007]

该测试用例测试现场设备能否响应 AL 的读(READ)服务。读现场设备 UAO 测试包括读现场设备设备表中的 NumOfSupUAO 属性以及读现场设备的 SupUAOList 属性。

测试过程为:

a) 被测设备加入网络后,测试系统向被测设备发送错误的读(READ)请求,具体读取内容包括 DeviceList 中的 NumOfSupUAO 值以及 SupUAOList 值,错误类型包括:
   1) 服务超时;(本地控制即可,这个代码没有意义)
   2) 服务不被支持;
   3) UAO 不存在;
   4) 属性不存在;
   5) 存储索引不存在;
   6) 成员不存在;
   7) 长度太长。(本地控制即可,这个代码没有意义)

b) 被测设备接收到请求后,向测试系统返回携带对应错误代码的读(READ)负响应。

c) 测试系统将接收的读(READ)负响应与期望的报文进行比对,如果比对匹配,则测试通过。

具体时序如图 10 所示,具体测试说明如表 17 所示。

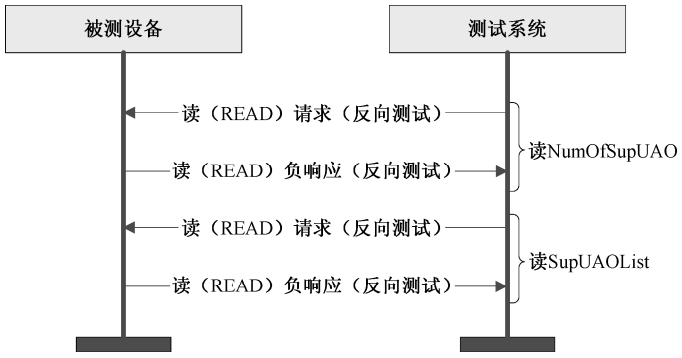

图 10 读现场设备 UAO 测试(反向测试)时序图

表 17 读现场设备 UAO 测试(反向测试)说明

用例名称	读现场设备 UAO 测试(反向测试)[FD-JOIN007]
被测设备	现场设备
依赖测试条件	被测设备已加入测试系统网络

测试用例伪代码描述	TEST BODY:
	SendPacket 1 = UAPReadRequest(NumOfSupUAO);
	SendPacket 2 = UAPReadRequest(SupUAOList);
	RefPacket 1 = UAPReadResponse(NumOfSupUAO)(+);
	RefPacket 2 = UAPReadResponse(SupUAOList)(+);
	Send(ASLDataRequest(DstAddr, 0x01, 0x00, 0x04, 0x05,SendPacket1));
	WHILE(Receive(ASLDataResponse).PacketControl & 0x1F ! = 0x0C) for LossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F==0x0C)

表 17 （续）

测试用例伪代码描述	{
	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("ReadNumOfSupUAONegativeResponse Payload error!");
	IF Receive(ASLDataResponse). Asdu.ErrorCode= 1
	TestCaseResult ="SERVICE_EXPIRATION";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 2
	TestCaseResult ="SERVICE_NOT_SUPPORTED";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 3
	TestCaseResult ="UAO_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 4
	TestCaseResult ="ATTRIBUTE_NOTE_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 5
	TestCaseResult ="STOREINDEX_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 6
	TestCaseResult ="MEMBER_NOTE_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 7
	TestCaseResult ="LENGTH_TOO_LARGE";
	ELSE
	TestCaseResult ="OTHERS"
	}
	ELSE
	{
	Printscreen("ReadNumOfSupUAONegativeTest Success!");
	Send(ASLDataRequest(DstAddr, 0x01, 0x00, 0x04, 0x05, SendPacket2));
	WHILE (Receive(ASLDataResponse).PacketControl&0x1F! =0x0C) forLossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F== 0x0C)
	{
	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket2.all) ! = SUCCESS)
	{
	Printscreen("ReadSupUAOListNegativeResponse Payload error!");
	IF Receive(ASLDataResponse). Asdu.ErrorCode= 1
	TestCaseResult ="SERVICE_EXPIRATION";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 2
	TestCaseResult ="SERVICE_NOT_SUPPORTED";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 3
	TestCaseResult ="UAO_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 4
	TestCaseResult ="ATTRIBUTE_NOTE_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 5
	TestCaseResult ="STOREINDEX_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 6
	TestCaseResult ="MEMBER_NOTE_EXISTENT";

表 17 （续）

测试用例伪代码描述	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 7
	TestCaseResult =“LENGTH_TOO_LARGE”;
	ELSE
	TestCaseResult =“OTHERS”
	}
	ELSE
	{
	Printscreen(“ReadSupUAOListNegative Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	}
	}
	ELSE
	{
	Printscreen(“NoReadResponse received!”);
	TestCaseResult = “SERVICE_EXPIRATION”;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or SERVICE_EXPIRATION or SERVICE_NOT_SUPPORTED or UAO_NOT_EXISTENT or ATTRIBUTE_NOTE_EXISTENT or STOREINDEX_NOT_EXISTENT or MEMBER_NOTE_EXISTENT or LENGTH_TOO_LARGE or OTHERS or FAILED

参考数据包	测试系统应发包： 数据包 1： 协议层：UAP 包名称 1:UAPReadRequest(NumOfSupUAO) 包： 0x00 0x00 | 0x83 | 0x?? | 0x04(无错误) 0x00 0x01 | 0x83 | 0x?? | 0x04(UAO 不存在) 0x00 0x00 | 0xff | 0x?? | 0x04(属性不存在) 0x00 0x00 | 0x83 | 0xff | 0x04 (存储索引不存在) 0x00 0x00 | 0x83 | 0x?? |0xfe(成员不存在) 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)成员标识符(1) 包名称 2:UAPReadRequest(SupUAOList) 包： 0x00 0x00 | 0x86 | 0x00 | 0xff(无错误) 0x00 0x01 | 0x86 | 0x00 | 0xff (UAO 不存在) 0x00 0x00 | 0xff | 0x00 | 0xff (属性不存在) 0x00 0x00 | 0x86 | 0xff | 0xff (存储索引不存在) 0x00 0x00 | 0x86 | 0x00 |0xfe (成员不存在) 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)成员标识符(1)

表 17 （续）

参考数据包	数据包 2： 协议层：ASL 包名称：ASLDataRequest(DstAddr，ServiceID，UAP_ID，Priority，AsduLength，Asdu) 包 1： 0x04 | 0x00 | 0x00 0x05 | 0x00 0x00 | 0x83 | 0x?? | 0x04 包 2： 0x04 | 0x00 | 0x00 0x05 | 0x00 0x01 | 0xff | 0xff |0xfe 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)
	测试系统应收包： 数据包 1： 协议层：ASL 包名称 1：ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包 1：0x0C | 0x00 | 0x00 0x02 | 0x?? 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(2) 包 2：0x0C | 0x00 | 0x00 0x02 | 0x?? 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(n) 数据包 2： 协议层：UAP 包名称 1：UAPReadResponse(NumOfSupUAO) (-) 包：0x?? 0x?? 帧域说明：数据(2) 包名称 2：UAPReadResponse(SupUAOList) (-) 包：0x??... 0x?? 帧域说明：数据(n)

6.1.8 配置现场设备 UAO 测试（正向测试）[FD-JOIN008]

该测试用例测试现场设备能否正确响应 AL 的写（WRITE）服务。配置现场设备 UAO 测试包括写现场设备设备表中的 NumOfCfgUAO 属性以及写现场设备的 CfgUAOList 属性。

测试过程为：

a） 网关设备读完现场设备 UAO 后，测试系统向被测设备发送写（WRITE）请求，具体配置内容包括：

1） DeviceList 中的 NumOfCfgUAO 值；

2） CfgUAOList 值。

b） 被测设备接收到请求后，向测试系统返回写（WRITE）正响应。

c） 测试系统将接收的写（WRITE）正响应与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 11 所示，具体测试说明如表 18 所示。

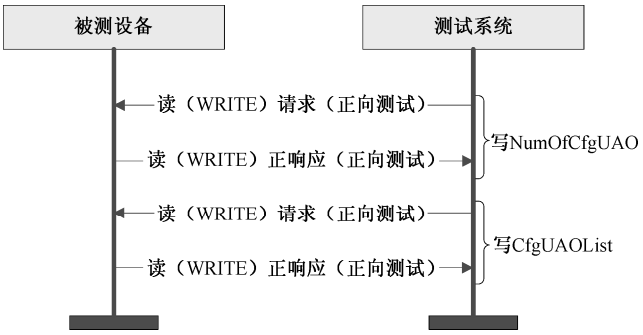

图 11　配置现场设备 UAO 测试(正向测试)时序图

表 18　配置现场设备 UAO 测试(正向测试)说明

用例名称	配置现场设备 UAO 测试(正向测试)[FD-JOIN008]
被测设备	现场设备
依赖测试条件	被测设备已通过读现场设备 UAO 测试(正向测试)

测试用例伪代码描述	TEST BODY:
	SendPacket 1 = UAPWriteRequest(NumOfCfgUAO);
	SendPacket 2 = UAPWriteRequest(CfgUAOList) ;
	RefPacket 1 = UAPWriteResponse(NumOfCfgUAO);
	RefPacket 2 = UAPWriteResponse(CfgUAOList);
	Send(ASLDataRequest(DstAddr, 0x02, 0x00, 0x04, 0x07, SendPacket1));
	WHILE(Receive(ASLDataResponse).PacketControl&0x1F! =0x02)forLossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F== 0x02)
	{
	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("WriteNumOfCfgUAORequest Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("WriteNumOfCfgUAORequest Test Success!");
	Send(ASLDataRequest(DstAddr, 0x02, 0x00, 0x04, Length(SendPacket2), SendPacket2));
	WHILE (Receive(ASLDataResponse).PacketControl & 0x1F ! = 0x02) for LossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F== 0x02)
	{
	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket2.all) ! = SUCCESS)
	{
	Printscreen("WriteCfgUAOListTest Error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Write CfgUAOList Test Success!");
	TestCaseResult =SUCCESS;

表 18（续）

测试用例伪代码描述	}
	}
	}
	}
	ELSE
	{
	Printscreen(“NoWrite CfgUAOList Request received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESSor FAILED

参考数据包	测试系统应发包： 数据包 1： 协议层：UAP 包名称 1:UAPWriteRequest(NumOfCfgUAO) 包：0x00 0x00 | 0x83 | 0x?? | 0x?? | 0x?? 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)数据(2) 包名称 2:UAPWriteRequest(CfgUAOList) 包：0x00 0x00 | 0x87 | 0xff | 0xff | 0x?? ... 0x?? 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)载荷(n) 数据包 2： 协议层：ASL 包名称:ASLDataRequest(DstAddr, ServiceID, UAP_ID, Priority, AsduLength, Asdu) 包 1:0x02 | 0x00 | 0x00 0x07 | 0x00 0x00 | 0x83 | 0x?? | 0x?? | 0x?? 包 2:0x02 | 0x00 | 0x00 0x?? | 0x00 0x00 | 0x87 | 0xff | 0xff | 0x?? ... 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)载荷(n)
	测试系统应收包： 数据包 1： 协议层：ASL 包名称 1:ASLDataResponse(PacketControl, UAP_ID, Asdulength, Asdu) 包 1:0x22 | 0x00 | 0x00 0x02 | 0x?? 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(2) 包 2:0x22 | 0x00 | 0x00 0x02 | 0x?? 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(2) 数据包 2： 协议层：UAP 包名称 1:UAPWriteResponse(NumOfCfgUAO)(+) 包：0x?? 0x?? 帧域说明:数据(2) 包名称 2:UAPWriteResponse(CfgUAOList)(+) 包：0x?? 0x?? 帧域说明:数据(2)

6.1.9 配置现场设备 UAO 测试(反向测试)[FD-JOIN009]

该测试用例测试现场设备能否响应 AL 的写(WRITE)服务。配置现场设备 UAO 测试包括写现场设备设备表中的 NumOfCfgUAO 属性以及写现场设备的 CfgUAOList 属性。

测试过程为:

a) 被测设备加入网络后,测试系统向被测设备发送错误的读(WRITE)请求,具体写内容包括 DeviceList 中的 NumOfCfgUAO 值以及 CfgUAOList 值,错误类型包括:
   1) 服务超时;(本地控制即可,这个代码没有意义)
   2) 服务不被支持;
   3) UAO 不存在;
   4) 属性不存在;
   5) 存储索引不存在;
   6) 成员不存在;
   7) 长度太长;(本地控制即可,这个代码没有意义)
   8) 值超出范围;
   9) 其他。

b) 被测设备接收到请求后,向测试系统返回携带对应错误代码的写(WRITE)负响应。

c) 测试系统将接收的写(WRITE)负响应与期望的报文进行比对,如果比对匹配,则测试通过。

具体时序如图 12 所示,具体测试说明如表 19 所示。

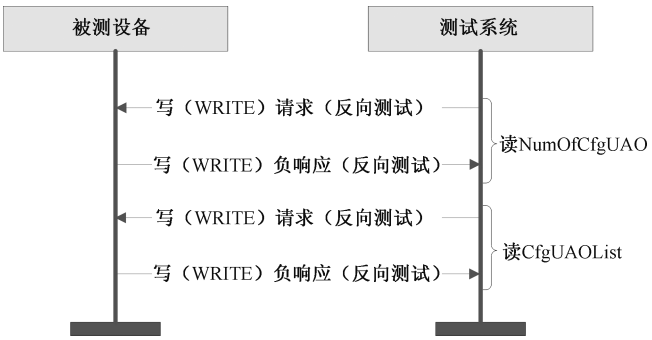

图 12 配置现场设备 UAO 测试(反向测试)时序图

表 19 配置现场设备 UAO 测试(反向测试)说明

用例名称	配置现场设备 UAO 测试(反向测试)[FD-JOIN009]
被测设备	现场设备
依赖测试条件	被测设备已通过读现场设备 UAO 测试(正向测试)

测试用例伪代码描述	TEST BODY:
	SendPacket 1 = UAPReadRequest(NumOfSupUAO);
	SendPacket 2 = UAPReadRequest(SupUAOList);
	RefPacket 1 = UAPReadResponse(NumOfSupUAO);
	RefPacket 2 = UAPReadResponse(SupUAOList);
	end(ASLDataRequest(DstAddr, 0x02, 0x00, 0x04, 0x07, SendPacket1));
	WHILE(Receive(ASLDataResponse).PacketControl&0x1F! =0x02)forLossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F== 0x02)
	{

表 19（续）

测试用例伪代码描述	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket1.all)! = SUCCESS)
	{
	Printscreen("WriteNumOfCfgUAONegative Response Payload error!");
	IF Receive(ASLDataResponse). Asdu.ErrorCode= 1
	TestCaseResult ="SERVICE_EXPIRATION";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 2
	TestCaseResult ="SERVICE_NOT_SUPPORTED";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 3
	TestCaseResult ="UAO_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 4
	TestCaseResult ="ATTRIBUTE_NOTE_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 5
	TestCaseResult ="STOREINDEX_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 6
	TestCaseResult ="MEMBER_NOTE_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 7
	TestCaseResult ="LENGTH_NOT_MATCH";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 8
	TestCaseResult ="VALUE_EXCEED_SCOPE";
	ELSE
	TestCaseResult ="OTHERS"
	}
	ELSE
	{
	Printscreen("WriteNumOfCfgUAONegative Test Success!");
	Send(ASLDataRequest(DstAddr, 0x02, 0x00, 0x04, Length(SendPacket2), SendPacket2));
	WHILE(Receive(ASLDataResponse).PacketControl&0x1F ! = 0x02)for
	LossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F== 0x02)
	{
	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket2.all)! = SUCCESS)
	{
	Printscreen("WriteCfgUAOListNegative Response Payload error!");
	IF Receive(ASLDataResponse). Asdu.ErrorCode= 1
	TestCaseResult ="SERVICE_EXPIRATION";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 2
	TestCaseResult ="SERVICE_NOT_SUPPORTED";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 3
	TestCaseResult ="UAO_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 4
	TestCaseResult ="ATTRIBUTE_NOTE_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 5
	TestCaseResult ="STOREINDEX_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 6
	TestCaseResult ="MEMBER_NOTE_EXISTENT";

表 19（续）

测试用例伪代码描述	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 7
	TestCaseResult =“LENGTH_NOT_MATCH”;
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 8
	TestCaseResult =“VALUE_EXCEED_SCOPE”;
	ELSE
	TestCaseResult =“OTHERS”
	}
	}
	ELSE
	{
	Printscreen(“Write CfgUAOListNegative Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	}
	ELSE
	{
	Printscreen(“NoWriteResponse received!”);
	TestCaseResult = “SERVICE_EXPIRATION”;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or SERVICE_EXPIRATION or SERVICE_NOT_SUPPORTED or UAO_NOT_EXISTENT or ATTRIBUTE_NOTE_EXISTENT or STOREINDEX_ NOT _ EXISTENT or MEMBER _ NOTE _ EXISTENT or LENGTH _ NOT _ MATCH or VALUE_EXCEED_SCOPE or OTHERS or FAILED

参考数据包	测试系统应发包： 数据包 1： 协议层：UAP 包名称 1:UAPWriteRequest(NumOfCfgUAO) 包： 0x00 0x01 | 0x83 | 0x?? | 0x?? | 0x?? （UAO 不存在） 0x00 0x00 | 0xff | 0x?? | 0x?? | 0x?? （属性不存在） 0x00 0x00 | 0x83 | 0xff | 0x?? | 0x?? (存储索引不存在） 0x00 0x00 | 0x83 | 0x?? | 0xfe | 0x?? （成员不存在） 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)数据(2) 包名称 2:UAPWriteRequest(CfgUAOList) 包： 0x00 0x01 | 0x87 | 0xff | 0xff | 0x?? ... 0x?? (UAO 不存在） 0x00 0x00 | 0xff | 0xff | 0xff | 0x?? ... 0x?? (属性不存在） 0x00 0x00 | 0x87 | 0x00 | 0xff | 0x?? ... 0x?? (存储索引不存在） 0x00 0x00 | 0x87 | 0xff | 0xfe | 0x?? ... 0x?? (成员不存在） 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)数据(n) 数据包 2： 协议层：ASL

表 19(续)

参考数据包	包名称:ASLDataRequest(DstAddr, ServiceID, UAP_ID, Priority, AsduLength, Asdu) 包 1: 0x02 | 0x00 | 0x00 0x07 | 0x00 0x00 | 0x?? | 0x?? | 0x?? | 0x?? 帧域说明:包控制(1)|UAP 标识符(1)|载荷长度(2)|UAO 标识符(2)| 属性标识符(1)|存储索引(1)|成员标识符(1)载荷(2) 包 2: 0x02 | 0x00 | 0x00 0x?? | 0x00 0x00 | 0x?? | 0x?? | 0x?? | 0x?? ... 0x?? 帧域说明:包控制(1)|UAP 标识符(1)|载荷长度(2)|UAO 标识符(2)| 属性标识符(1)|存储索引(1)|成员标识符(1)载荷(n)
	测试系统应收包: 数据包 1: 协议层:ASL 包名称 1:ASLDataResponse(PacketControl, UAP_ID, Asdulength, Asdu) 包 1:0x0A| 0x00 | 0x00 0x02 | 0x?? 0x?? 帧域说明:包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(2) 包 2:0x0A | 0x00 | 0x00 0x02 | 0x?? 0x?? 帧域说明:包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(2) 数据包 2: 协议层:UAP 包名称 1:UAPWriteResponse(NumOfCfgUAO)(-) 包:0x?? 0x?? 帧域说明:数据(2) 包名称 2:UAPWriteResponse(CfgUAOList)(-) 包:0x?? 0x?? 帧域说明:数据(2)|

6.1.10 配置现场设备 VCR 测试(正向测试)[FD-JOIN010]

该测试用例测试现场设备能否正确响应 AL 的写(WRITE)服务。配置现场设备 VCR 测试包括写现场设备 VCRList 属性。

测试过程为:

a) 网关设备配置完现场设备 UAO 后,测试系统向被测设备发送写(WRITE)请求,具体配置内容为现场设备的 VCRList 属性;

b) 被测设备接收到请求后,向测试系统返回写(WRITE)正响应;

c) 测试系统将接收的写(WRITE)正响应与期望的报文进行比对,如果比对匹配,则测试通过。

具体时序如图 13 所示,具体测试说明如表 20 所示。

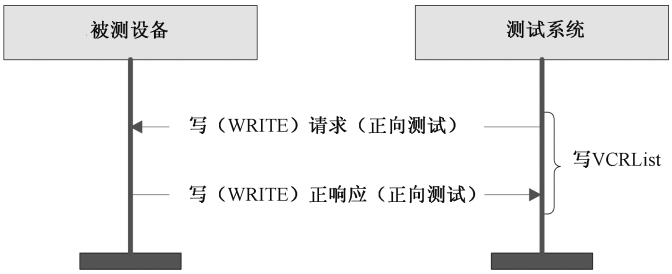

图 13　配置现场设备 VCR 测试(正向测试)时序图

表 20　配置现场设备 VCR 测试(正向测试)说明

用例名称	配置现场设备 VCR 测试(正向测试)[FD-JOIN010]
被测设备	现场设备
依赖测试条件	被测设备已通过配置现场设备 UAO 测试(正向测试和反向测试)

测试用例伪代码描述	TEST BODY:
	SendPacket = UAPWriteRequest(VCRList);
	RefPacket = UAPWriteResponse(VCRList);
	Send(ASLDataRequest(DstAddr, 0x02, 0x00, 0x04, Length(SendPacket), SendPacket));
	WHILE(Receive(ASLDataResponse).PacketControl&0x1F! =0x02)forLossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F== 0x02)
	{
	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WriteVCRListRequest Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("WriteVCRListRequest Test Success!");
	}
	}
	ELSE
	{
	Printscreen("NoWrite VCRList Request received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发包: 数据包 1: 协议层: UAP 包名称:UAPWriteRequest(VCRList) 包: 0x00 0x00 | 0x85| 0xff | 0xff | 0x?? ... 0x??

表 20（续）

参考数据包	帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)载荷(n) 数据包 2： 协议层：ASL 包名称：ASLDataRequest(DstAddr，ServiceID，UAP_ID，Priority，AsduLength，Asdu) 包 2：0x02	0x00	0x00 0x??	0x00 0x00	0x85| 0xff	0xff	0x?? … 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|UAO 标识符(2)| 属性标识符(1)|存储索引(1)|成员标识符(1)载荷(n) 测试系统应收包： 数据包 1： 协议层：ASL 包名称：ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包：0x22	0x00	0x00 0x02	0x?? 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(2) 数据包 2： 协议层：UAP 包名称：UAPWriteResponse(VCRList)(+) 包：0x?? 0x?? 帧域说明：数据(2)

6.1.11 配置现场设备 VCR 测试(反向测试)[FD-JOIN011]

该测试用例测试现场设备能否响应 AL 的写(WRITE)服务。配置现场设备 VCR 测试包括写现场设备 VCRList 属性。

测试过程为：

a) 被测设备加入网络后，测试系统向被测设备发送错误的读(WRITE)请求，具体写内容为现场设备的 VCRList 属性，错误类型包括：
   1) 服务超时；(本地控制即可，这个代码没有意义)
   2) 服务不被支持；
   3) UAO 不存在；
   4) 属性不存在；
   5) 存储索引不存在；
   6) 成员不存在；
   7) 长度太长；(本地控制即可，这个代码没有意义)
   8) 值超出范围；
   9) 其他。

b) 被测设备接收到请求后，向测试系统返回携带对应错误代码的写(WRITE)负响应。

c) 测试系统将接收的写(WRITE)负响应与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 14 所示，具体测试说明如表 21 所示。

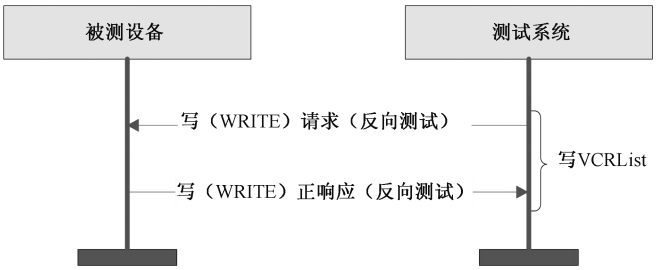

图 14　配置现场设备 VCR 测试(反向测试)时序图

表 21　配置现场设备 VCR 测试(反向测试)说明

用例名称	配置现场设备 VCR 测试(反向测试)[FD-JOIN011]
被测设备	现场设备
依赖测试条件	被测设备已通过配置现场设备 UAO 测试(正向测试和反向测试)

测试用例伪代码描述	TEST BODY:
	SendPacket = UAPWriteRequest(VCRList);
	RefPacket = UAPWriteResponse(VCRList);
	Send(ASLDataRequest(DstAddr, 0x02, 0x00, 0x04, Length(SendPacket), SendPacket));
	WHILE(Receive(ASLDataResponse).PacketControl&0x1F! =0x02)forLossConnectDuration;
	IF(Receive(ASLDataResponse).PacketControl & 0x1F== 0x02)
	{
	IF (Verify(Receive(ASLDataResponse). Asdu,RefPacket.all)! = SUCCESS)
	{
	Printscreen("Write VCRListRequest Payload error!");
	IF Receive(ASLDataResponse). Asdu.ErrorCode= 1
	TestCaseResult ="SERVICE_EXPIRATION";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 2
	TestCaseResult ="SERVICE_NOT_SUPPORTED";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 3
	TestCaseResult ="UAO_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 4
	TestCaseResult ="ATTRIBUTE_NOTE_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 5
	TestCaseResult ="STOREINDEX_NOT_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 6
	TestCaseResult ="MEMBER_NOTE_EXISTENT";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 7
	TestCaseResult ="LENGTH_NOT_MATCH";
	ELSE IF Receive(ASLDataResponse). Asdu.ErrorCode= 8
	TestCaseResult ="VALUE_EXCEED_SCOPE";
	ELSE
	TestCaseResult ="OTHERS"
	}
	ELSE
	{
	Printscreen("Write VCRListRequest Test Success!");

表 21（续）

测试用例代码描述	}
	}
	ELSE
	{
	Printscreen(“NoWrite VCRList Request received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESSor SERVICE_EXPIRATION or SERVICE_NOT_SUPPORTED or UAO_NOT_EXISTENT or ATTRIBUTE_NOTE_EXISTENT or STOREINDEX_NOT_EXISTENT or MEMBER_NOTE_EXISTENT or LENGTH_NOT_MATCH or VALUE_EXCEED_SCOPE or OTHERS or FAILED

参考数据包	测试系统应发包： 数据包 1： 协议层：UAP 包名称:UAPWriteRequest(VCRList) 包： 0x00 0x01 | 0x85 | 0xff | 0xff | 0x?? ... 0x??（UAO 不存在） 0x00 0x00 | 0xff | 0xff | 0xff | 0x?? ... 0x??（属性不存在） 0x00 0x00 | 0x85| 0x00 | 0xff | 0x?? ... 0x??（存储索引不存在） 0x00 0x00 | 0x85| 0xff | 0xfe | 0x?? ... 0x??（成员不存在） 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)载荷(n) 数据包 2： 协议层：ASL 包名称:ASLDataRequest(DstAddr，ServiceID，UAP_ID，Priority，AsduLength，Asdu) 包： 0x02 | 0x00 | 0x00 0x?? | 0x00 0x01 | 0x85| 0xff | 0xff | 0x?? ... 0x??（UAO 不存在） 0x02 | 0x00 | 0x00 0x?? | 0x00 0x00 | 0xff | 0xff | 0xff | 0x?? ... 0x??（属性不存在） 0x02 | 0x00 | 0x00 0x?? | 0x00 0x00 | 0x85| 0x00 | 0xff | 0x?? ... 0x??（存储索引不存在） 0x02 | 0x00 | 0x00 0x?? | 0x00 0x00 | 0x85| 0xff | 0xfe | 0x?? ... 0x??（成员不存在） 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)载荷(n)
	测试系统应收包： 数据包 1： 协议层：ASL 包名称:ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包：0x22 | 0x00 | 0x00 0x02 | 0x?? 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(2) 数据包 2： 协议层：UAP 包名称:UAPWriteResponse(VCRList) (-) 包：0x?? 0x?? 帧域说明:数据(2)

6.1.12 KEK 密钥分发测试(正向测试)[FD-JOIN012]

该测试用例用于测试现场设备能否正确响应 KEK 密钥分发请求。

测试过程为:

a) 被测设备安全加入网络后,测试系统向被测设备发送密钥分发请求;

b) 被测设备接收到密钥分发请求后,向测试系统返回密钥分发响应;

c) 测试系统将接收到的密钥分发响应报文与期望的报文进行比对,如果比对匹配,则测试通过。

具体时序如图 15 所示,具体测试说明如表 22 所示。

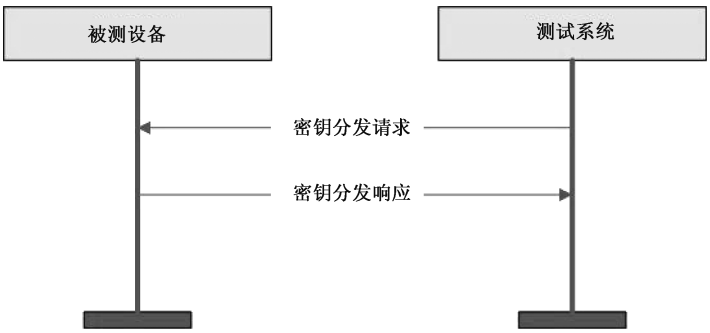

图 15 KEK 密钥分发测试(正向测试)时序图

表 22 KEK 密钥分发测试(正向测试)说明

用例名称	KEK 密钥分发测试(正向测试)[FD-JOIN012]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 SecLevel =2～8

测试用例伪代码描述	TEST BODY;
	SendPacket = KeyEstablishRequest(KEK);
	RefPacket =KeyEstablishResponse;
	Send(SendPacket,SecLevel,KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KS)! = SUCCESS)
	{
	Printscreen("Key Establish Response Payload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	Printscreen("KEK Establish Test Success!");
	TestCaseResult ="SUCCESS";
	}
	}
	ELSE
	{

表 22（续）

测试用例伪代码描述	Printscreen(“No Key Establish Response received!”); TestCaseResult = “FAILED”; } Printscreen(TestCaseResult); TEST RESULT: SUCCESS or FAILED
参考数据包 SecLevel=5	测试系统应发数据包: 协议层: DLL 帧名称:Key Establish Request with KEK 帧: 0x91 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x02 | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2)
	测试系统应收数据包: 协议层:DLL 帧名称:Key EstablishResponse 帧: 0x92 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00 | 0x?? 0x?? 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)|FCS(2)
参考数据包 SecLevel=2,3,4,6,7,8	测试系统应发数据包: 协议层: DLL 帧名称:Key Establish Request with KEK 帧: 0x91 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x02 | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|MIC(n)| FCS(2)
	测试系统应收数据包: 协议层:DLL 帧名称:Key EstablishResponse 帧: 0x92 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00 | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)| MIC(n)| FCS(2)

6.1.13 KEK 密钥分发测试(反向测试)[FD-JOIN013]

该测试用例用于测试现场设备能否正确响应 KEK 密钥分发请求。

测试过程为:

a) 被测设备安全加入网络后,测试系统向被测设备发送一个错误的 KEK 密钥;

b) 被测设备接收到密钥分发请求后,向测试系统返回密钥分发响应失败;

c) 测试系统向被测设备发送一个密钥属性读取请求;

d) 被测设备接收到属性读取请求后，向测试系统返回一个失败的属性读取响应；

e) 测试系统将接收到的密钥分发响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 16 所示，具体测试说明如表 23 所示。

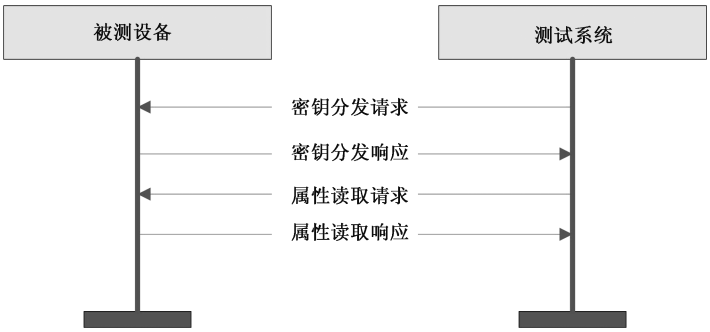

图 16 KEK 密钥分发测试(反向测试)时序图

表 23 KEK 密钥分发测试(反向测试)说明

用例名称	KEK 密钥分发测试(反向测试)[FD-JOIN013]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 SecLevel =2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = KeyEstablishRequest(KEK);
	SendPacket2 = AttributeGettingRequest(KEK);
	RefPacket1 = KeyEstablishResponse;
	RefPacket2 = AttributeGettingResponse;
	Send(SendPacket1,SecLevel,KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket1.all, RefPacket1.all,SecLevel, KS) ! = SUCCESS)
	{
	Printscreen("Key Establish Response Payload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	Printscreen("KEK Establish Response Payload Correct!");
	TestCaseResult ="SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Key Establish Response received!");
	TestCaseResult = "FAILED";}
	IF(TestCaseResult == "SUCCESS")
	{

表 23（续）

<table>
<tr><td>测试用例伪代码描述</td><td>
Send(SendPacket2,SecLevel,KS);
WHILE(Receive(RcvPacket).type ! = RefPacket2.type)for MaxWaitTime;
IF(RcvPacket .type == RefPacket2.type)
{
IF (Verify(RcvPacket.all, RefPacket2.all,SecLevel,KS)== SUCCESS)
{
Printscreen(“KEK Establish Test Success!”);
TestCaseResult = “SUCCESS”; }
}
ELSE
{
TestCaseResult =“FAILED”;
Printscreen(“KEK EstablishTest Failed!”);
}
}
ELSE
{
TestCaseResult = “FAILED”;
}
Printscreen(TestCaseResult);

TEST RESULT:
SUCCESS or FAILED
</td></tr>
<tr><td rowspan="2">参考数据包
SecLevel=5</td><td>
测试系统应发数据包：
数据包 1：
协议层：DLL
帧名称:Key Establish Request with KEK
帧：0x91 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x02 | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 0x?? 0x??
帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2)
数据包 2：
协议层：DLL
帧名称：Attribute Getting Request(KEK)
帧:0x8e | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x84 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 0x??
帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)| FCS(2)
</td></tr>
<tr><td>
测试系统应收数据包：
数据包 1：
协议层:DLL
帧名称:Key EstablishResponse
帧：0x92 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x01 | 0x?? 0x?? 0x?? 0x??
帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态
</td></tr>
</table>

表 23（续）

参考数据包 SecLevel＝5	(1)	FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8d	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x84	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)	网络 ID(1)	源地址(1)	序列号(2)	帧长度(2)	执行结果(1)	属性标识符(1)	属性成员标识符(1)	多个属性值的第一个存储索引(2)属性数目(2)	属性值(n)	FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Key Establish Request with KEK 帧：0x91	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x02	0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|MIC(n)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Request(KEK) 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x84	0x??	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|MIC(n)| FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Key EstablishResponse 帧：0x92	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x01	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)|MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8d| 0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x84	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(n)|FCS(2)

6.1.14 KEDU 密钥分发测试（正向测试）[FD-JOIN014]

该测试用例用于测试现场设备能否正确响应 KEDU 密钥分发请求。

测试过程为：

a） 被测设备安全加入网络后，测试系统向被测设备发送密钥分发请求；

b） 被测设备接收到密钥分发请求后，向测试系统返回密钥分发响应；

c） 测试系统将接收到的密钥分发响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 17 所示，具体测试说明如表 24 所示。

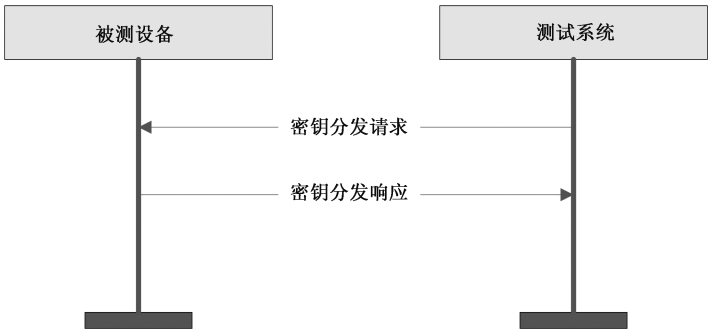

图 17 KEDU 密钥分发测试（正向测试）时序图

表 24 KEDU 密钥分发测试（正向测试）说明

用例名称	KEDU 密钥分发测试（正向测试）[FD-JOIN014]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 SecLevel ＝2～8

测试用例伪代码描述	TEST BODY：
	SendPacket ＝ KeyEstablishRequest(KEDU)；
	RefPacket ＝KeyEstablishResponse；
	Send(SendPacket,SecLevel,KS)；
	WHILE(Receive(RcvPacket).type ！＝ RefPacket.type)for MaxWaitTime；
	IF(RcvPacket.type ＝＝ RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KS)！＝ SUCCESS)
	{
	Printscreen(“Key Establish Response Payload error!”)；
	TestCaseResult ＝“FAILED”；
	}
	ELSE
	{
	Printscreen(“KEDU Establish Test Success!”)；
	TestCaseResult ＝“SUCCESS”；
	}
	}
	ELSE
	{
	Printscreen(“No Key Establish Response received!”)；
	TestCaseResult ＝ “FAILED”；
	}

表 24（续）

测试用例伪代码描述	Printscreen(TestCaseResult)； TEST RESULT： SUCCESS or FAILED
参考数据包 SecLevel＝5	测试系统应发数据包： 协议层：DLL 帧名称：Key Establish Request with KEDU 帧：0x91 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x03 | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Key EstablishResponse 帧：0x92 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00 | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)|FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 协议层：DLL 帧名称：Key Establish Request with KEDU 帧：0x91 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x03 | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|MIC(n)| FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Key EstablishResponse 帧：0x92 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00 | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)|MIC(n)| FCS(2)

6.1.15 KEDU 密钥分发测试（反向测试）[FD-JOIN015]

该测试用例用于测试现场设备能否正确响应 KEDU 密钥分发请求。

测试过程为：

a） 被测设备安全加入网络后，测试系统向被测设备发送一个错误的 KEDU 密钥材料；

b） 被测设备接收到密钥分发请求后，向测试系统返回密钥分发响应失败；

c） 测试系统向被测设备发送一个密钥属性读取请求；

d） 被测设备接收到属性读取请求后，向测试系统返回一个失败的属性读取响应；

e） 测试系统将接收到的密钥分发响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 18 所示，具体测试说明如表 25 所示。

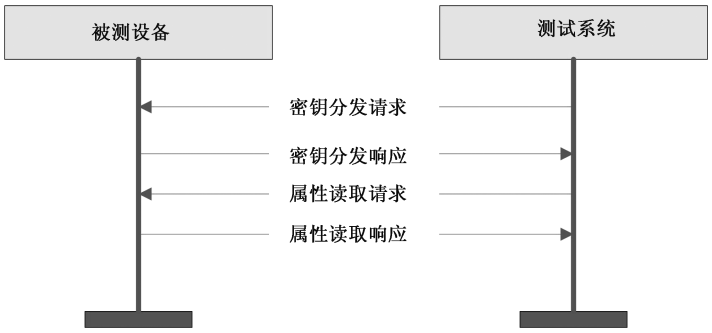

图 18　KEDU 密钥分发测试(反向测试)时序图

表 25　KEDU 密钥分发测试(反向测试)说明

用例名称	KEDU 密钥分发测试(反向测试)[FD-JOIN015]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 SecLevel =2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = KeyEstablishRequest(KEDU);
	SendPacket2 = AttributeGettingRequest(KEDU);
	RefPacket1 =KeyEstablishResponse;
	RefPacket2 = AttributeGettingResponse;
	Send(SendPacket1,SecLevel,KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type)for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket1.all, RefPacket1.all,SecLevel,KS)! = SUCCESS)
	{
	Printscreen("Key Establish Response Payload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	Printscreen("KEDU Establish Response Payload Correct!");
	TestCaseResult ="SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Key Establish Response received!");
	TestCaseResult = "FAILED";
	}
	IF(TestCaseResult == "SUCCESS")
	{
	Send(SendPacket2,SecLevel,KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type)for MaxWaitTime;

表 25（续）

测试用例伪代码描述	IF(RcvPacket .type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all,SecLevel,KS)== SUCCESS)
	{
	Printscreen(“KEDU Establish Test Success!”);
	TestCaseResult = “SUCCESS”;
	}
	}
	ELSE
	{
	Printscreen(“KEDU EstablishTest Failed!”);
	TestCaseResult =“FAILED”;
	}
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=5	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称:Key Establish Request with KEDU 帧：0x91 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x03 | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Request(KEDU) 帧:0x8e | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x84 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)| FCS(2)
	测试系统应收数据包： 数据包 1： 协议层:DLL 帧名称:Key EstablishResponse 帧：0x92 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x01 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response

表 25（续）

参考数据包 SecLevel=5	帧：0x8d	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x84	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)	网络 ID(1)	源地址(1)	序列号(2)	帧长度(2)	执行结果(1)	属性标识符(1)	属性成员标识符(1)	多个属性值的第一个存储索引(2)	属性数目(2)	属性值(n)	FCS(2)
参考数据包 SecLevel=2,3,4,6,7,8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Key Establish Request with KEDU 帧：0x91	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x03	0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 0x?? 0x??	0x?? …0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|MIC(n)| FCS(2)FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Request(KEDU) 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x84	0x??	0x?? 0x??	0x?? 0x??	0x?? …0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)| MIC(n)| FCS(2)FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Key EstablishResponse 帧：0x92	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x01	0x?? …0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)| MIC(n)| FCS(2)FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8d| 0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x84	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x?? …0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)| MIC(n)| FCS(2)

6.1.16 KEDB 密钥分发测试(正向测试)[FD-JOIN016]

该测试用例用于测试现场设备能否正确响应 KEDB 密钥分发请求。

测试过程为：

a) 被测设备安全加入网络后，测试系统向被测设备发送密钥分发请求；

b) 被测设备接收到密钥分发请求后，向测试系统返回密钥分发响应；

c) 测试系统将接收到的密钥分发响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 19 所示，具体测试说明如表 26 所示。

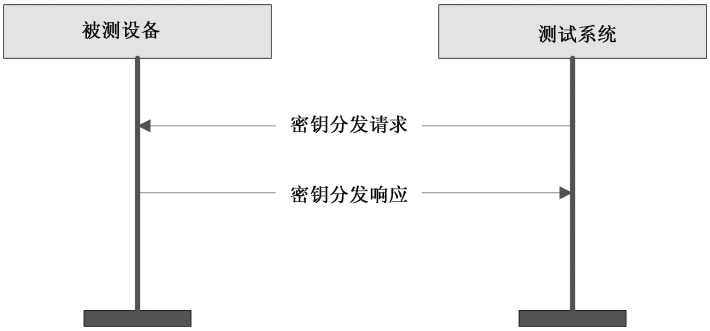

图 19　KEDB 密钥分发测试(正向测试)时序图

表 26　KEDB 密钥分发测试(正向测试)说明

用例名称	KEDB 密钥分发测试(正向测试)[FD-JOIN016]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 SecLevel =2～8

测试用例伪代码描述	TEST BODY:
	SendPacket = KeyEstablishRequest(KEDB);
	RefPacket =KeyEstablishResponse;
	Send(SendPacket, SecLevel,KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type)for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all, SecLevel,KS)! = SUCCESS)
	{
	Printscreen("Key Establish Response Payload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	Printscreen("KEDB Establish Test Success!");
	TestCaseResult ="SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Key Establish Response received!");
	TestCaseResult = "FAILED";
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 26（续）

参考数据包 SecLevel=5	测试系统应发数据包： 协议层：DLL 帧名称：Key Establish Request with KEDB 帧：0x91 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x04 | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Key EstablishResponse 帧：0x92 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)|FCS(2)
参考数据包 SecLevel=2,3,4,6,7,8	测试系统应发数据包： 协议层：DLL 帧名称：Key Establish Request with KEDB 帧：0x91 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x04 | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? …0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|MIC(n)|FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Key EstablishResponse 帧：0x92 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00 | 0x?? …0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)|MIC(n)| FCS(2)

6.1.17 KEDB 密钥分发测试（反向测试）[FD-JOIN017]

该测试用例用于测试现场设备能否正确响应 KEDB 密钥分发请求。

测试过程为：

a） 被测设备安全加入网络后，测试系统向被测设备发送一个错误的 KEDB 密钥；

b） 被测设备接收到密钥分发请求后，向测试系统返回密钥分发响应失败；

c） 测试系统向被测设备发送一个密钥属性读取请求；

d） 被测设备接收到属性读取请求后，向测试系统返回一个失败的属性读取响应；

e） 测试系统将接收到的密钥分发响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 20 所示，具体测试说明如表 27 所示。

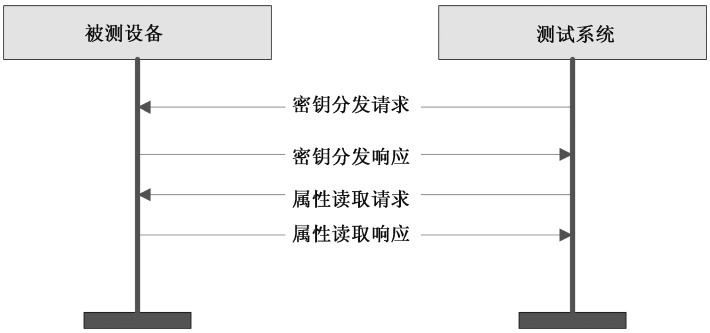

图 20　KEDB 密钥分发测试(反向测试)时序图

表 27　KEDB 密钥分发测试(反向测试)说明

用例名称	KEDB 密钥分发测试(反向测试)[FD-JOIN017]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 SecLevel =2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = KeyEstablishRequest(KEDB);
	SendPacket2 = AttributeGettingRequest(KEDB);
	RefPacket1 =KeyEstablishResponse;
	RefPacket2 = AttributeGettingResponse;
	Send(SendPacket1,SecLevel,KS);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type)for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket1.all, RefPacket1.all,SecLevel,KS)! = SUCCESS)
	{
	Printscreen("Key Establish Response Payload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	Printscreen("KEDB Establish Response Payload Correct!");
	TestCaseResult ="SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Key Establish Response received!");
	TestCaseResult = "FAILED";
	}
	IF(TestCaseResult == "SUCCESS")
	{
	Send(SendPacket2,SecLevel,KS);

表 27（续）

<table>
<tr><td>测试用例伪代码描述</td><td>

```
    WHILE(Receive(RcvPacket).type ! = RefPacket2.type)for MaxWaitTime;
    IF(RcvPacket .type == RefPacket2.type)
    {
        IF (Verify(RcvPacket.all, RefPacket2.all,SecLevel,KS)== SUCCESS)
        {
            Printscreen( "KEDB Establish Test Success!");
            TestCaseResult = "SUCCESS";
        }
    }
    ELSE
    {
        Printscreen("KEDB EstablishTest Failed!");
        TestCaseResult ="FAILED";
    }
}
Printscreen(TestCaseResult);

TEST RESULT:
    SUCCESS or FAILED
```

</td></tr>
<tr><td>参考数据包
SecLevel=5</td><td>测试系统应发数据包：
数据包 1：
协议层：DLL
帧名称：Key Establish Request with KEDU
帧：0x91 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x04 | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x??
帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2)
数据包 2：
协议层：DLL
帧名称：Attribute Getting Request(KEDU)
帧：0x8e | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x84 | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x??
帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|FCS(2)</td></tr>
<tr><td></td><td>测试系统应收数据包：
数据包 1：
协议层：DLL
帧名称：Key EstablishResponse
帧：0x92 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x01 | 0x?? 0x??
帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|状态(1)|FCS(2)
数据包 2：</td></tr>
</table>

表 27（续）

参考数据包 SecLevel＝5	协议层：DLL 帧名称：Attribute Getting Response 帧：0x8d｜0xaa｜0x03｜0x?? 0x??｜0x?? 0x??｜0x01｜0x84｜0x??｜0x?? 0x??｜0x?? 0x??｜0x?? ... 0x??｜0x?? 0x?? 帧域说明：帧控制(1)｜网络 ID(1)｜源地址(1)｜序列号(2)｜帧长度(2)｜执行结果(1)｜属性标识符(1)｜属性成员标识符(1)｜多个属性值的第一个存储索引(2)｜属性数目(2)｜属性值(n)｜FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Key Establish Request with KEDU 帧：0x91｜0xaa｜0x03｜0x?? 0x??｜0x00 0x1D｜0x?? 0x??｜0x04｜0x?? 0x?? 0x?? 0x?? 0x?? 0x??｜0x??... 0x??｜0x?? 0x?? 0x?? 0x??｜0x?? …0x??｜0x?? 0x?? 帧域说明：帧控制(1)｜网络 ID(1)｜目的地址(1)｜序列号(2)｜帧长度(2)｜密钥 ID(2)｜密钥类型(1)｜密钥激活时隙(6)｜密钥值(16)｜密钥 MIC(4)｜MIC(n)｜ FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Request(KEDU) 帧：0x8e｜0xaa｜0x03｜0x?? 0x??｜0x?? 0x??｜0x84｜0x??｜0x?? 0x??｜0x?? 0x??｜0x?? …0x??｜0x?? 0x?? 帧域说明：帧控制(1)｜网络 ID(1)｜目的地址(1)｜序列号(2)｜帧长度(2)｜属性标识符(1)｜属性成员标识符(1)｜多个属性值的第一个存储索引(2)｜属性数目(2)｜MIC(n)｜ FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Key EstablishResponse 帧：0x92｜0xaa｜0x03｜0x?? 0x??｜0x00 0x03｜0x?? 0x??｜0x01｜0x?? …0x??｜0x?? 0x?? 帧域说明：帧控制(1)｜网络 ID(1)｜源地址(1)｜序列号(2)｜帧长度(2)｜密钥 ID(2)｜状态(1)｜MIC(n)｜ FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8d｜0xaa｜0x03｜0x?? 0x??｜0x?? 0x??｜0x01｜0x84｜0x??｜0x?? 0x??｜0x?? 0x??｜0x?? ... 0x??｜0x?? …0x??｜0x?? 0x?? 帧域说明：帧控制(1)｜网络 ID(1)｜源地址(1)｜序列号(2)｜帧长度(2)｜执行结果(1)｜属性标识符(1)｜属性成员标识符(1)｜多个属性值的第一个存储索引(2)｜属性数目(2)｜属性值(n)｜MIC(n)｜FCS(2)

6.2 运行过程测试集

6.2.1 数据传输测试[FD-RUN001]

该测试用例测试现场设备能否周期性发送数据帧。

测试过程为：

a） 被测设备加入网络后，向测试系统发送数据帧；

b） 在 4 个超帧时间内，测试系统应至少接收到 3 个数据帧，被测设备发出数据帧的周期的最大误差为 1×TimeSlot；

c） 测试系统将接收的数据帧报文与期望的报文进行比对，如果比对匹配且周期误差不超过最大误差，则测试通过。

具体时序如图 21 所示，具体测试说明如表 28 所示。

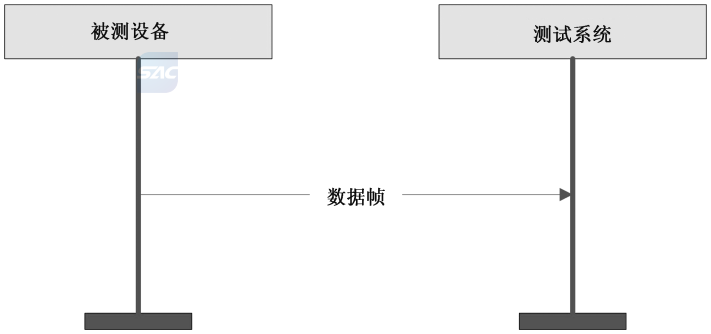

图 21 数据传输测试时序图

表 28 数据传输测试说明

用例名称	数据传输测试[FD-RUN001]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) KEDU 已启用

测试用例伪代码描述	TEST BODY：
	RefPacket = Data；
	Count = 0；
	WHILE(1)|for 4 * SuperframeCycle
	{
	IF(Receive(RcvPacket).type == RefPacket.type)
	{
	Count++；
	IF (Verify(RcvPacket.all, RefPacket.all, SecLevel, KEDU)! =SUCCESS)
	{
	Printscreen ("DataPayload error!")；
	TestCaseResult = FAILED；
	}
	}
	}

表 28（续）

测试用例伪代码描述	IF(Count < 3)
	{
	Printscreen ("Not EnoughData received!");
	TestCaseResult = FAILED;
	}
	ELSE IF(CycleError >1* TimeSlot)
	{
	Printscreen ("Data Cycle beyonds tolerance!");
	TestCaseResult = FAILED;
	}
	ELSE
	{
	Printscreen("Data Transmission Test Success!");
	TestCaseResult = SUCCESS;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包： 无
	测试系统应收数据包： 协议层:DLL 帧名称：Data 帧：0x81 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|载荷(n)|FCS(2)
参考数据包 SecLevel=2,3,4,6,7,8	测试系统应发数据包： 无
	测试系统应收数据包： 协议层:DLL 帧名称：Data 帧：0x81 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x?? … 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|载荷(n)| MIC(n)| FCS(2)

6.2.2 设备状态报告测试[FD-RUN002]

该测试用例测试现场设备能否周期性发送设备状态报告。

测试过程为：

a) 被测设备加入网络后，向测试系统发送设备状态报告；

b) 在 4 个 DevStaRptCycle 时间内，测试系统应至少接收到 3 个设备状态报告，被测设备发出设备状态报告的周期的最大误差为 max(10%×DevStaRptCycle，1×TimeSlot)；

c) 测试系统将接收的设备状态报告报文与期望的报文进行比对，如果比对匹配且周期误差不超过最大误差，则测试通过。

具体时序如图 22 所示，具体测试说明如表 29 所示。

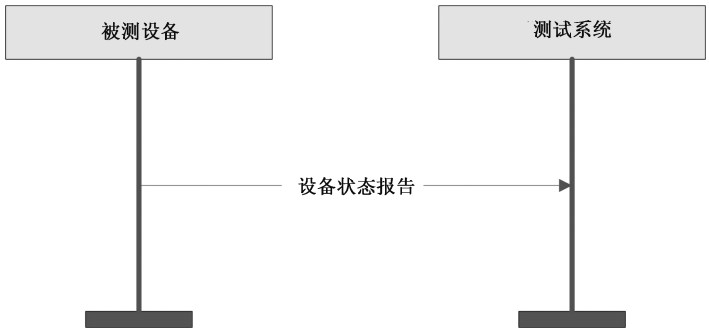

图 22 设备状态报告测试时序图

表 29 设备状态报告测试说明

用例名称	设备状态报告测试[FD-RUN002]
被测设备	现场设备
依赖测试条件 SecLevel=0，1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 测试系统已完成对被测设备 MIB 中的 DevStaRptCycle 参数配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) 测试系统已完成对被测设备 MIB 中的 DevStaRptCycle 参数配置 KEDU 已启用

测试用例伪代码描述	TEST BODY：
	RefPacket = DeviceConditionReport；
	Count = 0；
	WHILE(1)|for 4 * DevStaRptCycle
	{
	IF(Receive(RcvPacket).type == RefPacket.type)
	{
	Count++；
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KEDU)! = SUCCESS)
	{
	Printscreen ("Device Condition ReportPayload error!")；
	TestCaseResult = FAILED；
	}
	}
	}
	IF(Count < 3)
	{

表 29（续）

测试用例伪代码描述	Printscreen（“No EnoughDevice Condition Report received!”）；
	TestCaseResult = FAILED；
	}
	ELSE IF(CycleError ＞Max(10％* DevStaRptCycle，1* TimeSlot))
	{
	Printscreen（“Device Condition Report Cycle beyonds tolerance!”）；
	TestCaseResult = FAILED；
	}
	ELSE
	{
	Printscreen(“Device Condition Report Test Success!”)；
	TestCaseResult = SUCCESS；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

参考数据包 SecLevel = 0，1，5	测试系统应发数据包： 无
	测试系统应收数据包： 协议层：DLL 帧名称：Device Condition Report 帧：0x89 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x01 | 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|设备状态信息(1)|FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 无
	测试系统应收数据包： 协议层：DLL 帧名称：Device Condition Report 帧：0x89 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x01 | 0x?? | 0x?? …0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|设备状态信息(1)|MCS(n)|FCS(2)

6.2.3 信道状况报告测试[FD-RUN003]

该测试用例测试现场设备能否周期性发送信道状况报告。

测试过程为：

a） 被测设备加入网络后，向测试系统发送信道状况报告；

b） 在 4 个 ChaStaRptCycle 时间内，测试系统应至少接收到 3 个信道状况报告，被测设备发出信道状况报告的周期的最大误差为 max(10％×ChaStaRptCycle，1×TimeSlot)；

c） 测试系统将接收的信道状况报告报文与期望的报文进行比对，如果比对匹配且周期误差不超

过最大误差，则测试通过。

具体时序如图 23 所示，具体测试说明如表 30 所示。

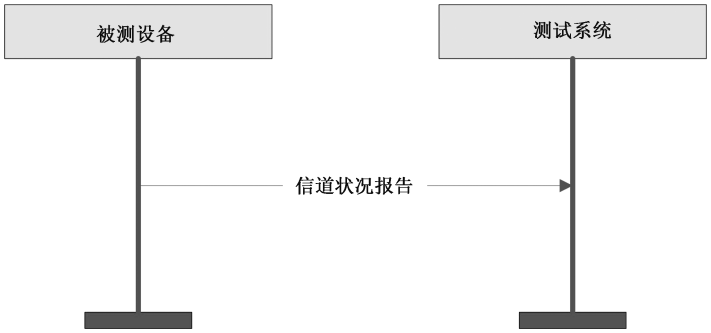

图 23　信道状况报告测试时序图

表 30　信道状况报告测试说明

用例名称	信道状况报告测试[FD-RUN003]
被测设备	现场设备
依赖测试条件 SecLevel=0，1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 测试系统已完成对被测设备 MIB 中的 ChaStaRptCycle 参数配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) 测试系统已完成对被测设备 MIB 中的 ChaStaRptCycle 参数配置 KEDU 已启用

测试用例伪代码描述	TEST BODY:
	RefPacket = ChannelConditionReport;
	Count = 0;
	WHILE(1)|for 4 * ChaStaRptCycle
	{
	IF(Receive(RcvPacket).type == RefPacket.type)
	{
	Count++;
	IF (Verify(RcvPacket.all, RefPacket.all, SecLevel,KEDU)! = SUCCESS)
	{
	Printscreen("Channel Condition Report Payload error!");
	TestCaseResult =FAILED;
	}
	}
	}
	IF(Count < 3)
	{
	Printscreen ("Not EnoughChannel Condition Report received!");
	TestCaseResult = FAILED;

表 30（续）

测试用例伪代码描述	}
	ELSE IF(CycleError ＞Max(10％＊ChaStaRptCycle,1＊TimeSlot))
	{
	Printscreen(“Channel Condition Report Cycle beyonds tolerance!”);
	TestCaseResult = FAILED;
	}
	ELSE
	{
	Printscreen(“Channel Condition ReportTest Success!”);
	TestCaseResult = SUCCESS;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包： 无
	测试系统应收数据包： 协议层：DLL 帧名称:Channel Condition Report 帧：0x8a	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|信道状况信息(n)|FCS(2)
参考数据包 SecLevel＝2,3,4,6,7,8	测试系统应发数据包： 无
	测试系统应收数据包： 协议层：DLL 帧名称:Channel Condition Report 帧：0x8a	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x??	0x?? ... 0x??	0x?? …0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|信道列表数量(1)|信道状况信息列表(n)| MIC(n)| FCS(2)

6.2.4 远程读属性测试(正向测试)[FD-RUN004]

该测试用例测试现场设备能否正确响应远程读属性请求。

测试过程为：

a) 被测设备加入网络后，测试系统向被测设备发送远程读属性请求，具体情况如下：

1) 读非结构化属性；

2) 读结构体属性的一个成员；

3) 读结构体属性的全部成员；

4) 读结构体列表属性一个成员的一个记录；

5) 读结构体列表属性一个成员的多个记录；

6） 读结构体列表属性一个成员的全部记录；
7） 读结构体列表属性全部成员的一个记录；
8） 读结构体列表属性全部成员的多个记录；
9） 读结构体列表属性全部成员的全部记录。

参考数据包中具体载荷内容如表 31 所示。

表 31 远程读属性测试(正向测试)参考数据包载荷

测试内容	应发数据载荷	应收数据载荷
读非结构化属性	0x?? | 0xff | 0x?? 0x?? | 0x?? 0x??	0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x??
	属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性一个成员的一个记录	0x?? | 0x?? | 0x?? 0x?? | 0x00 0x01	0x00 | 0x?? | 0x?? (不等于 0xff)| 0x?? 0x?? | 0x00 0x01 | 0x?? … 0x??
	属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性一个成员的多个记录	0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x??	0x00 | 0x?? | 0x?? (不等于 0xff)| 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x??
	属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性一个成员的全部记录	0x?? | 0x?? | 0x?? 0x?? | 0x00 0x00	0x00 | 0x?? | 0x?? (不等于 0xff)| 0x?? 0x?? | 0x00 0x00 | 0x?? … 0x??
	属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性全部成员的一个记录	0x?? | 0xff | 0x?? 0x?? | 0x00 0x01	0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x01 | 0x?? … 0x??
	属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性全部成员的多个记录	0x?? | 0xff | 0x?? 0x?? | 0x?? 0x??	0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x??
	属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)

表 31（续）

测试内容	应发数据载荷	应收数据载荷
读结构化属性全部成员的全部记录	0x?? | 0xff | 0x?? 0x?? | 0x00 0x00	0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x00 | 0x?? … 0x??
	属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)

b） 被测设备接收到请求后，向测试系统返回远程读属性响应；

c） 测试系统将接收的远程读属性响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

该测试用例用于读被测设备信息库中的所有属性，测试体应循环执行，具体时序如图 24 所示，具体测试说明如表 32 所示。

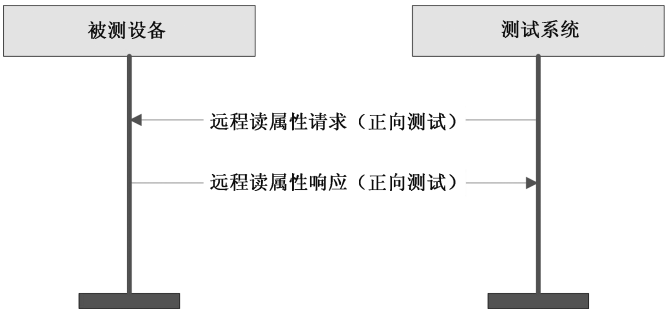

图 24 远程读属性测试(正向测试)时序图

表 32 远程读属性测试(正向测试)说明

用例名称	远程读属性测试(正向测试)[FD-RUN004]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = AttributeGettingRequest;
	RefPacket = AttributeGettingResponse;
	Send(SendPacket,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type)for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KEDU)! = SUCCESS)
	{
	Printscreen("Attribute Getting Response Payload error!");

表 32（续）

测试用例伪代码描述	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Attribute Getting Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("NoAttribute Getting Response received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESSor FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包: 协议层: DLL 帧名称:Attribute Getting Request 帧:0x8d | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x06 | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(6)|FCS(2)
	测试系统应收数据包: 协议层:DLL 帧名称:Attribute GettingResponse 帧: 0x8e | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|FCS(2)
参考数据包 SecLevel=2,3,4,6,7,8	测试系统应发数据包: 协议层: DLL 帧名称:Attribute Getting Request 帧:0x8d | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x06 | 0x?? ... 0x?? | 0x??...0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(6)| MIC(n)|FCS(2)
	测试系统应收数据包: 协议层:DLL 帧名称:Attribute GettingResponse 帧:0x8e | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x00 | 0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x??... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(n)|FCS(2)

6.2.5 远程读属性测试(反向测试)[FD-RUN005]

该测试用例测试现场设备能否响应远程读属性请求。

测试过程为:

a) 被测设备加入网络后,测试系统向被测设备发送错误的远程读属性请求,具体情况如下:

1) 读不存在的属性;

2) 读结构体属性不存在的成员;

3) 读结构体列表属性不存在的记录。

b) 被测设备接收到请求后,向测试系统返回远程读属性响应;

c) 测试系统将接收的远程读属性响应报文与期望的报文进行比对,如果比对匹配,则测试通过。

该测试用例用于读多个属性的反向测试,测试体应循环执行,具体时序如图 25 所示,具体测试说明如表 33 所示。

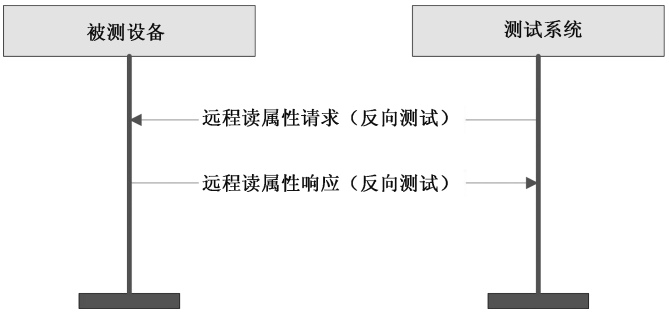

图 25 远程读属性测试(反向测试)时序图

表 33 远程读属性测试(反向测试)说明

用例名称	远程读属性测试(反向测试)[FD-RUN005]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2~8	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = AttributeGettingRequest;
	RefPacket = AttributeGettingResponse;
	Send(SendPacket,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type)for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KEDU)! = SUCCESS)
	{
	Printscreen("Attribute Getting Response Payload error!");

表 33（续）

测试用例伪代码描述	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Attribute Getting Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("NoAttribute Getting Response received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESSor FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包: 协议层: DLL 帧名称:Attribute Getting Request 帧:0x8d | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x06 | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(6)|FCS(2)
	测试系统应收数据包: 协议层:DLL 帧名称:Attribute GettingResponse 帧: 0x8e | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x01 | 0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|FCS(2)
参考数据包 SecLevel=2,3,4,6,7,8	测试系统应发数据包: 协议层: DLL 帧名称:Attribute Getting Request 帧:0x8d | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x06 | 0x?? ... 0x?? | 0x?? … 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)序|列号(2)|帧长度(2)|载荷(6)| MIC(n)|FCS(2)
	测试系统应收数据包: 协议层:DLL 帧名称:Attribute GettingResponse 帧:0x8e | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x01| 0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x??... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(4)|FCS(2)

6.2.6 远程配置属性测试(正向测试)[FD-RUN006]

该测试用例测试现场设备能否正确响应远程配置属性请求。

测试过程为:

a) 被测设备加入网络后,测试系统向被测设备发送远程配置属性请求,具体情况如下:

1) 配置非结构化属性;
2) 配置结构化属性一个成员的一个记录;
3) 配置结构化属性一个成员的多个记录;
4) 配置结构化属性一个成员的全部记录;
5) 配置结构化属性全部成员的一个记录;
6) 配置结构化属性全部成员的多个记录;
7) 配置结构化属性全部成员的全部记录。

参考数据包中具体载荷内容如表 34 所示。

表 34 远程配置属性测试(正向测试)参考数据包载荷

测试内容	应发数据载荷	应收数据载荷
配置非结构化属性	0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x??	0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x00
	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性一个成员的一个记录	0x?? | 0x?? | 0x?? | 0x?? 0x?? | 0x00 0x01 | 0x?? ... 0x??	0x?? | 0x?? | 0x?? (不等于 0xff)| 0x?? 0x?? | 0x00 0x01 | 0x00
	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性一个成员的多个记录	0x?? | 0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x??	0x?? | 0x?? | 0x?? (不等于 0xff)| 0x?? 0x?? | 0x?? 0x?? | 0x00
	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性一个成员的全部记录	0x?? | 0x?? | 0x?? | 0x?? 0x?? | 0x00 0x00 | 0x?? ... 0x??	0x?? | 0x?? | 0x?? (不等于 0xff)| 0x?? 0x?? | 0x00 0x00 | 0x00
	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性全部成员的一个记录	0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x01 | 0x?? ... 0x??	0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x01 | 0x00
	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)

表 34（续）

测试内容	应发数据载荷	应收数据载荷
配置结构化属性全部成员的多个记录	0x??｜0x??｜0xff｜0x?? 0x??｜0x?? 0x??｜0x?? ... 0x??	0x??｜0x??｜0xff｜0x?? 0x??｜0x?? 0x??｜0x00
	远程属性操作(1)｜属性标识符(1)｜属性成员标识符(1)｜多个属性值的第一个存储索引(2)｜属性数目(2)｜属性值(n)	远程属性操作(1)｜属性标识符(1)｜属性成员标识符(1)｜多个属性值的第一个存储索引(2)｜属性数目(2)｜执行结果(1)
配置结构化属性全部成员的全部记录	0x??｜0x??｜0xff｜0x?? 0x??｜0x000x00｜0x?? ... 0x??	0x??｜0x??｜0xff｜0x?? 0x??｜0x000x00｜0x00
	远程属性操作(1)｜属性标识符(1)｜属性成员标识符(1)｜多个属性值的第一个存储索引(2)｜属性数目(2)｜属性值(n)	远程属性操作(1)｜属性标识符(1)｜属性成员标识符(1)｜多个属性值的第一个存储索引(2)｜属性数目(1)｜执行结果(1)

b） 被测设备接收到请求后，向测试系统返回远程配置属性响应；

c） 测试系统接收到远程配置属性响应后，再向被测设备发出远程读属性请求，并将被测设备返回的远程读属性响应值与之前的属性配置值相比较，如果一致，则测试通过。

该测试用例用于配置被测设备信息库中的所有属性，测试体应循环执行，具体时序如图 26 所示，具体测试说明如表 35 所示。

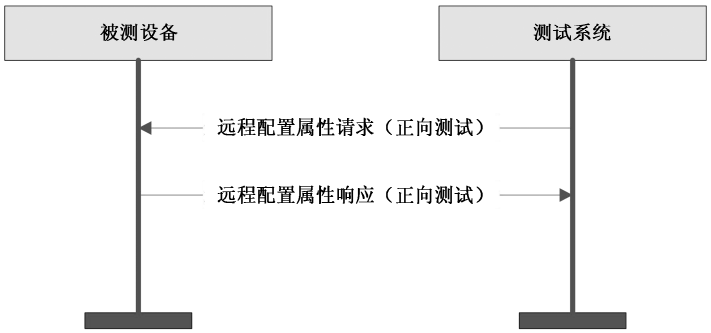

图 26 远程配置属性测试(正向测试)时序图

表 35 远程配置属性测试(正向测试)说明

用例名称	远程配置属性测试(正向测试)[FD-RUN006]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 测试系统已完成对被测设备的属性获取 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) 测试系统已完成对被测设备的属性获取 KEDU 已启用

表 35（续）

测试用例伪代码描述	TEST BODY：
	SendPacket = AttributeSettingRequest；
	RefPacket = AttributeSettingResponse；
	Send(SendPacket,SecLevel,KEDU)；
	WHILE(Receive(RcvPackettype ! = RefPacket.type) for MaxWaitTime；
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KEDU).) ! = SUCCESS)；
	{
	Printscreen(“Attribute Setting Response Payload error!”)；
	TestCaseResult =FAILED；
	}
	ELSE
	{
	Printscreen(“Attribute Getting Test Success!”)；
	TestCaseResult =SUCCESS；
	}
	}
	ELSE
	{
	Printscreen(“No Attribute Setting Response received!”)；
	TestCaseResult = FAILED；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

参考数据包 SecLevel = 0，1，5	测试系统应发数据包： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? … 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(n)|FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? |0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2)

表 35（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x?? … 0x??	0x??… 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(n)|MIC(n)|FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x?? |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? |0x00| 0x??…0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)| MIC(n)| FCS(2)

6.2.7 远程配置属性测试（反向测试）[FD-RUN007]

该测试用例测试现场设备能否响应远程配置属性请求。

测试过程为：

a） 被测设备加入网络后，测试系统向被测设备发送错误的远程配置属性请求，具体情况如下：

1） 配置不存在的属性；

2） 配置不存在的成员；

3） 配置错误的属性值；

4） 配置不可写的属性参数；

5） 访问不存在的记录。

b） 被测设备接收到请求后，向测试系统返回远程配置属性响应。

c） 测试系统将接收的远程配置属性响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

该测试用例用于配置多个属性的反向测试，测试体应循环执行，具体时序如图 27 所示，具体测试说明如表 36 所示。

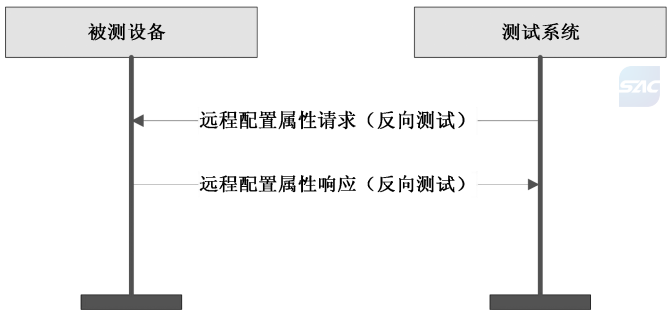

图 27 远程配置属性测试（反向测试）时序图

表 36　远程配置属性测试(反向测试)说明

用例名称	远程配置属性测试(反向测试)[FD-RUN007]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 测试系统已完成对被测设备的属性获取 KEDU=NULL
依赖测试条件 SecLevel=1～8	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) 测试系统已完成对被测设备的属性获取 KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = AttributeSettingRequest;
	RefPacket = AttributeSettingResponse;
	Send(SendPacket, SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all, SecLevel,KEDU) ! = SUCCESS);
	{
	Printscreen("Attribute Setting Response Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Attribute Getting Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Attribute Setting Response received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包: 协议层: DLL 帧名称:Attribute Setting Request 帧:0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? ... 0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(n)|FCS(2)

表 36（续）

参考数据包 SecLevel = 0，1，5	测试系统应收数据包： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? |0x01 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2)
参考数据包 SecLevel = 2，3，4，6，7，8	测试系统应发数据包： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? … 0x?? | 0x??… 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(n)| MIC(n)| FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? |0x01| 0x??… 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)| MIC(n)|FCS(2)

6.2.8 基于 NACK 重传测试[FD-RUN008]

该测试用例测试现场设备能否正确响应 NACK 进行重传。

测试过程为：

a) 被测设备加入网络后，向测试系统发送设备状态报告；

b) 测试系统接收到设备状态报告后，向被测设备发送包含被测设备短地址的 NACK；

c) 测试系统等待被测设备在重传时隙重传设备状态报告，并将接收的设备状态报告报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 28 所示，具体测试说明如表 37 所示。

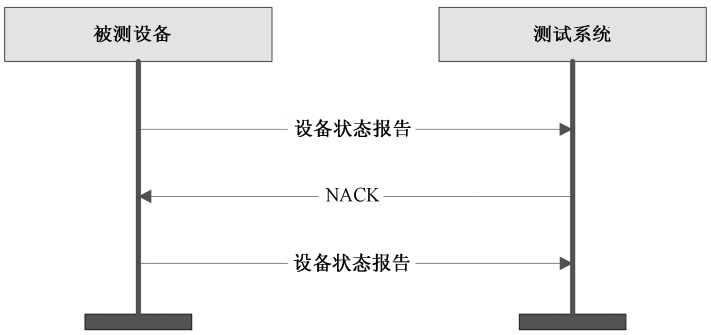

图 28 基于 NACK 重传测试时序图

表 37　基于 NACK 重传测试说明

用例名称	基于 NACK 重传测试[FD-RUN008]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 测试系统已完成对被测设备 MIB 中的 DevStaRptCycle 参数配置 KEDB=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) 测试系统已完成对被测设备 MIB 中的 DevStaRptCycle 参数配置 KEDB 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = NACK;
	RefPacket = DeviceConditionReport;
	WHILE(1)|for 2 * DevStaRptCycle;
	IF(RcvPacket.type == RefPacket.type)
	{
	Send(SendPacket,SecLevel,KEDB);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxRetryTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KEDB) ! = SUCCESS)
	{
	Printscreen("Retrying Device Condition Report Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Retrying basad on NACK Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Retrying Device Condition Report received!");
	TestCaseResult = FAILED;
	}
	}
	ELSE
	{
	Printscreen("No Device Condition Report received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 37（续）

参考数据包 SecLevel = 0，1，5	测试系统应发数据包： 协议层：DLL 帧名称：NACK 帧：0x84 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x02 |0x01 | 0x03| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|重传设备数量(1)|设备短地址列表(1)|FCS(2)|
	测试系统应收数据包： 协议层：DLL 帧名称：Device Condition Report 帧：0x89 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? … 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|设备状态信息(n)|FCS(2)
参考数据包 SecLevel = 2，3，4，6，7，8	测试系统应发数据包： 协议层：DLL 帧名称：NACK 帧：0x84 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x02 |0x01 | 0x03| 0x??… 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|重传设备数量(1)|设备短地址列表(1)| MIC(n)|FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Device Condition Report 帧：0x89 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x01 | 0x?? | 0x??…0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|设备状态信息(1)|MIC(n)| FCS(2)

6.2.9 基于 GACK 重传测试[FD-RUN009]

该测试用例测试现场设备能否正确响应 GACK 进行重传。

测试过程为：

a） 被测设备加入网络后，测试系统向被测设备发送远程读属性请求；

b） 被测设备接收到请求后，向测试系统返回远程读属性响应；

c） 测试系统接收到响应后，向被测设备发送包含被测设备短地址的 GACK；

d） 测试系统等待被测设备在重传时隙重传远程读属性响应，并将接收的远程读属性响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 29 所示，具体测试说明如表 38 所示。

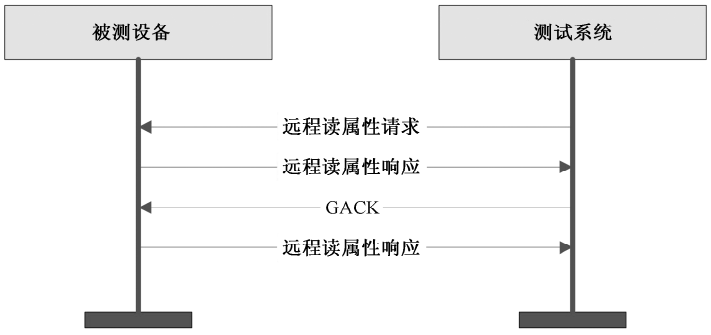

图 29 基于 GACK 重传测试时序图

表 38　基于 GACK 重传测试说明

用例名称	基于 GACK 重传测试[FD-RUN009]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 KEDB=NULL,KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) KEDB,KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket1 = AttributeGettingRequest;
	SendPacket2 = GACK;
	RefPacket =AttributeGettingResponse;
	Send(SendPacket1,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	Send(SendPacket2,SecLevel,KEDB);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxRetryTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Retrying Attribute Getting Response Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Retrying basad on GACK Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("NoRetryingAttribute Getting Response received!");
	TestCaseResult = FAILED;
	}
	}
	ELSE
	{
	Printscreen("NoAttribute Getting Response received!");
	TestCaseResult = FAILED;
	}

表 38（续）

测试用例伪代码描述	Printscreen(TestCaseResult)； TEST RESULT： SUCCESS or FAILED
参考数据包 SecLevel = 0，1，5	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Getting Request 帧：0x8d	0xaa	0x03	0x?? 0x??	0x00 0x06 |0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(6)|FCS(2) 数据包 2： 协议层：DLL 帧名称：GACK 帧：0x83	0xaa	0x03	0x?? 0x??	0x00 0x04 |0x01	0x03| 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|设备数(1)|设备短地址(1)|帧序列号(2)|FCS(2)
	测试系统应收数据包： 协议层：DLL 帧名称：Attribute GettingResponse 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? |0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|FCS(2)
参考数据包 SecLevel = 2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Getting Request 帧：0x8d	0xaa	0x03	0x?? 0x??	0x00 0x06 |0x?? ... 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(6)| MIC(n)| FCS(2) 数据包 2： 协议层：DLL 帧名称：GACK 帧：0x83	0xaa	0x03	0x?? 0x??	0x00 0x01| 0x01	0x00| 0x04| 0x?? 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|设备数(1)|地址模式(1)|设备短地址(1)|帧序列号(2)|MIC(n)| FCS(2)

表 38（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x?? |0x?? |0x?? 0x?? |0x?? 0x??	0x?? ... 0x??	0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(n)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? |0x?? ... 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性(n)|MIC(n)|FCS(2)

6.2.10 P/S 通信测试[FD-RUN010]

该测试用例测试现场设备能否周期性发送输入数据。

测试过程为：

a） 被测设备配置 VCR 结束后，其 AL 运行 AMPB 状态机，向测试系统发送输入数据；

b） 在 4 个超帧时间内，测试系统应至少接收到 3 个数据，被测设备发出数据的周期的最大误差为 1×TimeSlot；

c） 测试系统将接收的报文与期望的报文进行比对，如果比对匹配且周期误差不超过最大误差，则测试通过。

具体时序如图 30 所示，具体测试说明如表 39 所示。

图 30 P/S 通信测试时序图

表 39　P/S 通信测试说明

用例名称	P/S 通信测试[FD-RUN010]
被测设备	现场设备
依赖测试条件	被测设备已被配置 VCR

测试用例伪代码描述	TEST BODY:
	RefPacket = UAPPublishRequestMessage;
	Count = 0;
	WHILE(1)|for 4 * SuperframeCycle
	{
	IF(Receive(RcvPacket). DLLframeHeader.frameControl. frameType ==0b00001 &&Receive (RcvPacket).ASLHeader. PacketControl. MessageType ==0b011&&Receive(RcvPacket).ASLHeader. PacketControl. ServiceIdentifier ==0b00)
	{
	Count++;
	IF (Verify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen ("Publish Request Message error!");
	TestCaseResult = FAILED;
	}
	}
	}
	IF(Count < 3)
	{
	Printscreen ("Not EnoughData received!");
	TestCaseResult = FAILED;
	}
	ELSE IF(CycleError >1 * TimeSlot)
	{
	Printscreen ("Data Cycle beyonds tolerance!");
	TestCaseResult = FAILED;
	}
	ELSE
	{
	Printscreen("Publish Request Message Transmission Test Success!");
	TestCaseResult = SUCCESS;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 39（续）

参考数据包	测试系统应发数据包： 无
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：DLLData 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? … 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)|DLL 载荷(n)|FCS(2) 数据包 2： 协议层：ASL 包名称：ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包：0x06 | 0x?? | 0x?? 0x?? | 0x?? … 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(n) 数据包 3： 协议层：UAP 消息名称：UAPPublishRequestMessage 消息：0x?? … 0x?? 帧域说明：数据(n)

6.2.11 R/S 通信测试(非证实服务)[FD-RUN011]

该测试用例测试现场设备能否正确发送告警。

测试过程为：

a) 被测设备配置 VCR 结束后，其 AL 运行 AMRS 状态机，向测试系统发送告警；

b) 在 AlarmRptDur 时间内，测试系统应至少接收到 1 个告警；

c) 测试系统将接收的告警报文与期望报文进行比对，如果比对匹配，且告警未失效，则测试通过。

具体时序如图 31 所示，具体测试说明如表 40 所示。

图 31 R/S 通信测试(非证实服务)时序图

表 40　R/S 通信测试(非证实服务)说明

用例名称	R/S 通信测试(非证实服务)[FD-RUN011]
被测设备	现场设备
依赖测试条件	被测设备已被配置 VCR

测试用例伪代码描述	TEST BODY:
	RefPacket= UAPReportRequestMessage;
	WHILE(1)|for AlarmRptDur
	{
	IF(Receive(RcvPacket). DLLframeHeader.frameControl. frameType ==0b00001
	&&Receive(RcvPacket).ASLHeader. PacketControl. MessageType
	==0b100&&Receive(RcvPacket).ASLHeader. PacketControl. ServiceIdentifier ==0b00)
	{
	IF (Verify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen ("Report Request Message error!");
	TestCaseResult = FAILED;
	}
	}
	}
	ELSE IF(CycleError >1* TimeSlot)
	{
	Printscreen ("Data Cycle beyonds tolerance!");
	TestCaseResult = FAILED;
	}
	ELSE
	{
	Printscreen("Report Request Message Transmission Test Success!");
	TestCaseResult = SUCCESS;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 无
	测试系统应收数据包: 数据包 1: 协议层:DLL 帧名称: DLLData 帧: 0x90| 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x0b| 0x01 0x?? 0x07| 0x?? ... 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)|DLL 载荷(11)| FCS(2) 数据包 2: 协议层: ASL

表 40（续）

参考数据包	包名称：ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包：0x01	0x??	0x00 0x07| 0x?? … 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(7) 协议层：UAP 消息名称：UAPReportRequestMessage 消息：0x?? 0x??	0x?? 0x?? 0x?? 0x??	0x?? 帧域说明：UAO 标识符(2)|事件(4)|附件信息(1)

6.2.12 R/S 通信测试(证实服务)(正向测试)[FD-RUN012]

该测试用例测试现场设备能否正确发送报告确认正响应。

测试过程为：

a） 测试系统向被测设备发送正确的报告确认请求；

b） 被测设备 AL 运行 AMRS 状态机，向测试系统发送报告确认正响应；

c） 在 AlarmRptDur 时间内，测试系统应至少接收到 1 个报告确认正响应，被测设备发出报告确认正响应的周期的最大误差为 1×TimeSlot；

d） 测试系统将接收的报告确认正响应与期望报文进行比对，如果比对匹配，则测试通过。

具体时序如图 32 所示，具体测试说明如表 41 所示。

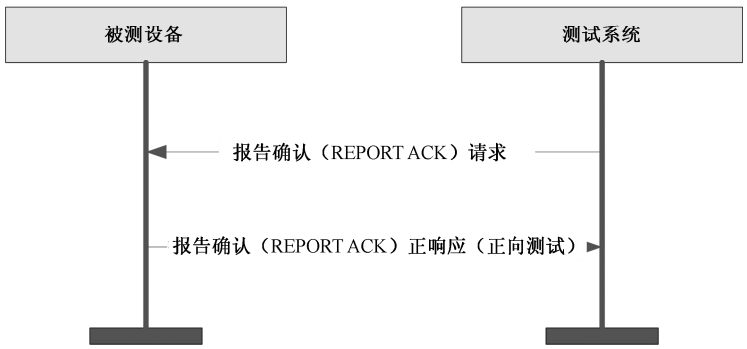

图 32 R/S 通信测试(证实服务)(正向测试)时序图

表 41 R/S 通信测试(证实服务)(正向测试)说明

用例名称	R/S 通信测试(证实服务)(正向测试)[FD-RUN012]
被测设备	现场设备
依赖测试条件	被测设备已被配置 VCR

测试用例伪代码描述	TEST BODY：
	RefPacket = UAOReportAckResonse (+)；
	WHILE(1)|for AlarmRptDur
	{
	IF(Receive(RcvPacket). DLLframeHeader.frameControl. frameType ==0b00001
	&&Receive(RcvPacket).ASLHeader. PacketControl. MessageType
	==0b101&&Receive(RcvPacket).ASLHeader. PacketControl. ServiceIdentifier ==0b01)

表 41（续）

测试用例伪代码描述	{
	IF (Verify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen ("Report Ack Response (+)Message error!");
	TestCaseResult = FAILED;
	}
	}
	}
	ELSE IF(CycleError >1* TimeSlot)
	{
	Printscreen ("Report Ack Data Cycle beyonds tolerance!");
	TestCaseResult = FAILED;
	}
	ELSE
	{
	Printscreen("Report Ack Response (+)Message Transmission Test Success!");
	TestCaseResult = SUCCESS;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 数据包 1： 协议层：UAP 消息名称:UAPReportAckRequestMessage 消息：0x?? 0x?? | 0x?? 0x01 帧域说明：UAO 标识符(2)确认事件(2) 数据包 2： 协议层：ASL 包名称:ASLDataResponse(PacketControl, UAP_ID, Asdulength, Asdu) 包：0x05 | 0x?? | 0x00 0x04| 0x?? 0x?? 0x?? 0x01 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(4) 数据包 3： 协议层:DLL 帧名称:DLLData 帧：0x90| 0xaa | 0x?? | 0x?? 0x?? | 0x00 0x08| 0x05 0x?? 0x00 0x040x?? 0x?? 0x?? 0x01| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)|DLL 载荷(8)|FCS(2)

表 41（续）

参考数据包	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：DLLData 帧：0x90| 0xaa	0x??	0x?? 0x??	0x00 0x06	0x15 0x?? 0x00 0x02 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)|DLL 载荷(6)|FCS(2) 数据包 2： 协议层：ASL 包名称：ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包：0x15	0x??	0x00 0x02| 0x?? 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(2) 数据包 3： 协议层：UAP 消息名称：UAPReportAckResponseMessage(+) 消息：0x?? 0x?? 帧域说明：UAO 标识符(2)

6.2.13 R/S 通信测试(证实服务)(反向测试)[FD-RUN013]

该测试用例测试现场设备能否正确发送报告确认负响应。

测试过程为：

a) 测试系统向被测设备发送报告确认请求；

b) 被测设备 AL 运行 AMRS 状态机，向测试系统发送报告确认负响应；

c) 在 AlarmRptDur 时间内，测试系统应至少接收到 1 个报告确认负响应，被测设备发出报告确认的周期的最大误差为 1×TimeSlot；

d) 测试系统将接收的报告确认负响应与期望报文进行比对，如果比对匹配，则测试通过。

具体时序如图 33 所示，具体测试说明如表 42 所示。

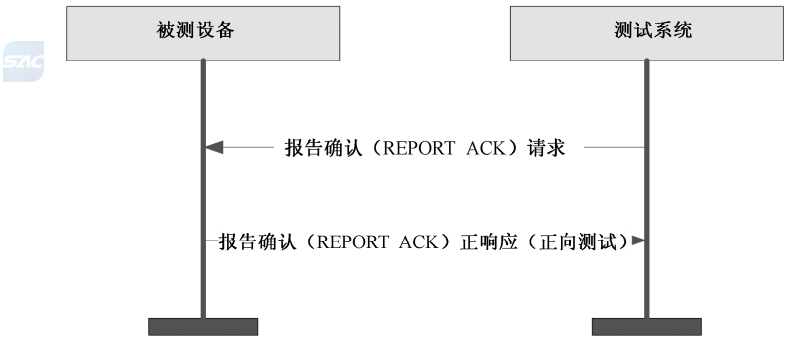

图 33 R/S 通信测试(证实服务)(反向测试)时序图

表 42　R/S 通信测试(证实服务)(反向测试)说明

用例名称	R/S 通信测试(证实服务)(反向测试)[FD-RUN013]
被测设备	现场设备
依赖测试条件	被测设备已被配置 VCR

测试用例伪代码描述	TEST BODY:
	RefPacket = UAOReportAckResonse (-);
	WHILE(1)|for AlarmRptDur
	{
	IF(Receive(RcvPacket). DLLframeHeader.frameControl. frameType ==0b00001 &&Receive (RcvPacket).ASLHeader. PacketControl. MessageType ==0b101&&Receive(RcvPacket).ASL-Header. PacketControl. ServiceIdentifier ==0b10)
	{
	IF (Verify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen ("Report Ack Response (-)Message error!");
	TestCaseResult = FAILED;
	}
	}
	}
	ELSE IF(CycleError >1* TimeSlot)
	{
	Printscreen ("Report Ack Data Cycle beyonds tolerance!");
	TestCaseResult = FAILED;
	}
	ELSE
	{
	Printscreen("Report Ack Response (-)Message Transmission Test Success!");
	TestCaseResult = SUCCESS;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 数据包 1: 协议层: UAP 消息名称:UAPReportAckRequestMessage 消息: 0x?? 0x?? | 0x?? 0x01 帧域说明: UAO 标识符(2)确认事件(2) 数据包 2: 协议层: ASL 包名称:ASLDataResponse(PacketControl, UAP_ID, Asdulength, Asdu) 包: 0x05 | 0x?? | 0x00 0x04| 0x?? 0x?? 0x?? 0x01

表 42（续）

	帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(4) 数据包 3： 协议层:DLL 帧名称:DLLData 帧：0x90| 0xaa	0x??	0x?? 0x??	0x00 0x08| 0x05 0x?? 0x00 0x04 0x?? 0x?? 0x?? 0x01| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)|DLL 载荷(8)|FCS(2)
参考数据包	测试系统应收数据包： 数据包 1： 协议层:DLL 帧名称:DLLData 帧：0x90| 0xaa	0x??	0x?? 0x??	0x00 0x08| 0x0e 0x?? 0x00 0x04 0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)|DLL 载荷(6)|FCS(2) 数据包 2： 协议层：ASL 包名称:ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包：0x0e	0x??	0x00 0x04| 0x?? 0x?? 0x?? 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(4) 数据包 3： 协议层：UAP 消息名称：UAPReportAckResponseMessage(-) 消息：0x?? 0x?? 0x?? 0x?? 帧域说明：UAO 标识符(4)

6.2.14 密钥更新测试(正向测试)[FD-RUN014]

该测试用例用于测试启用安全的现场设备能否正确进行密钥更新。

测试过程为：

a) 测试系统向被测设备发送用于 KEDU 更新的密钥更新请求包，更新现场设备与接入设备通信的 KEDU 并将 KEDU 的 ActiveTime 设置为当前 ASN+SuperframeCycle×3；

b) 当被测设备接收到密钥更新请求后，向测试系统返回密钥更新响应；

c) 测试系统等待 SuperframeCycle×3 后，向被测设备发送属性获取请求，且属性获取请求用更新的 KEDU 进行加密；

d) 当被测设备接收到属性获取请求后，向测试系统返回属性获取响应；

e) 测试系统将接收到的报文与期望的报文进行比对，如果比对正确，则认为测试通过。

具体时序图如图 34 所示，具体测试说明如表 43 所示。

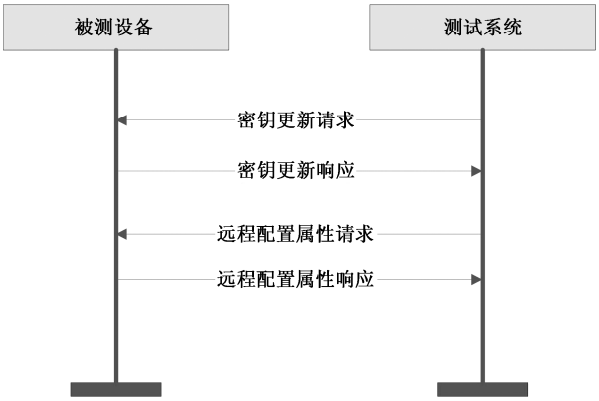

图 34　密钥更新测试(正向测试)时序图

表 43　密钥更新测试(正向测试)说明

用例名称	密钥更新测试(正向测试)[FD-RUN014]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) SecLevel=2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = KeyUpdateRequest;
	SendPacket2 = AttributeGettingRequest;
	RefPacket1 = KeyUpdateResponse;
	RefPacket2 = AttributeGettingResponse;
	Send(SendPacket1,SecLevel, KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel, KEDU) ! = SUCCESS)
	{
	Printscreen("Key Update Response Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	WHILE(1)|for 3 * SuperframeCycle
	Send(SendPacket2, ,SecLevel, KEDU_new);
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all, SecLevel,
	KEDU_new) ! = SUCCESS)
	{
	Printscreen("Attribute Getting Response Payload error!");

表 43（续）

测试用例伪代码描述	TestCaseResult = “FAILED”;
	}
	ELSE
	{
	Printscreen(“Key Update Test success!”);
	TestCaseResult = “SUCCESS”;
	}
	}
	ELSE
	{
	Printscreen(“No Attribute Getting Response received!”);
	TestCaseResult = “FAILED”;
	}
	}
	ELSE
	{
	Printscreen(“No Key Update Response received!”);
	TestCaseResult = “FAILED”;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=5	测试系统应发数据包： 数据包 1： 协议层:DLL 帧名称：Key Update Request 帧：0x93 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x?? | 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2) 数据包 2： 协议层:DLL 帧名称：Attribute Getting Request 帧：0x8d | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层:DLL 帧名称：Key Update Response 帧：0x94 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00 | 0x?? 0x??

表 43（续）

参考数据包 SecLevel=5	帧域说明：帧控制(1)	网络 ID(1)	源地址(1)	序列号(2)	帧长度(2)	密钥 ID(2)	密钥状态(1)	FCS(2) 数据包 2： 协议层:DLL 帧名称：Attribute Getting Response 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x00	0x??	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)	网络 ID(1)	目的地址(1)	序列号(2)	帧长度(2)	状态(1)	属性标识符(1)	属性成员标识符(1)	多个属性值的第一个存储索引(2)	属性数目(2)	属性值(n)	FCS(2)
参考数据包 SecLevel=2,3,4,6,7,8	测试系统应发数据包： 数据包 1： 协议层:DLL 帧名称：Key Update Request 帧：0x93	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x??	0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)| MIC(n)| FCS(2) 数据包 2： 协议层:DLL 帧名称：Attribute Getting Request 帧：0x8d	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x??	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)| MIC(n)| FCS(2)
	测试系统应收数据包： 数据包 1： 协议层:DLL 帧名称：Key Update Response 帧：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x00	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|MIC(n)| FCS(2) 数据包 2： 协议层:DLL 帧名称：Attribute Getting Response 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x00	0x??	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|状态(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)| MIC(n)| FCS(2)

6.2.15 密钥更新测试(反向测试)[FD-RUN015]

该测试用例用于测试启用安全的现场设备能否正确进行密钥更新。

测试过程为:

a) 测试系统向被测设备发送带有错误 MIC 的 KEDU 密钥更新请求包,更新现场设备与接入设备通信的 KEDU 并将 KEDU 的 ActiveTime 设置为当前 ASN+SuperframeCycle×3;

b) 当被测设备接收到密钥更新请求后,向测试系统返回密钥更新响应;

c) 测试系统等待 SuperframeCycle×3 后,向被测设备发送属性获取请求,且属性获取请求用更新的 KEDU 进行加密;

d) 当被测设备接收到属性获取请求后,向测试系统返回属性获取响应;

e) 测试系统将接收到的报文与期望的报文进行比对,如果比对正确,则认为测试通过。

具体时序图如图 35 所示,具体测试说明如表 44 所示。

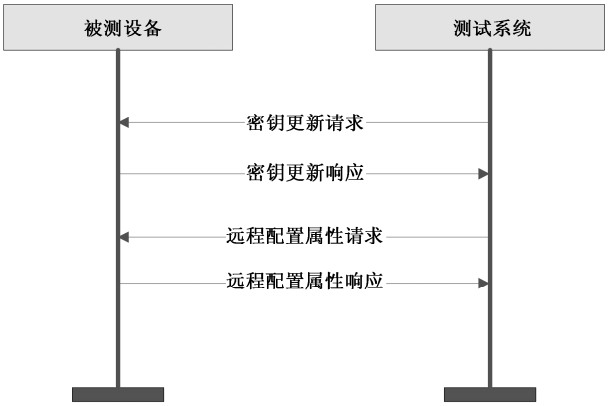

图 35 密钥更新测试(反向测试)时序图

表 44 密钥更新测试(反向测试)说明

用例名称	密钥更新测试(反向测试)[FD-RUN015]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥等) SecLevel=2~8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = KeyUpdateRequest;
	SendPacket2 = AttributeGettingRequest;
	RefPacket1 = KeyUpdateResponse;
	RefPacket2 = AttributeGettingResponse;
	Send(SendPacket1,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Key Update Response Payload error!");

表 44（续）

测试用例伪代码描述	TestCaseResult = “FAILED”;
	}
	ELSE
	{
	WHILE(1)|for 3 * SuperframeCycle
	Send(SendPacket2, SecLevel, KEDU_new);
	WHILE(Receive(RcvPacket)).type! =RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all,SecLevel,
	KEDU_new) ! = SUCCESS)
	{
	Printscreen(“Attribute Getting Response Payload error!”);
	TestCaseResult = “FAILED”;
	}
	ELSE
	{
	Printscreen(“Key Update Test success!”);
	TestCaseResult = “SUCCESS”;
	}
	}
	ELSE
	{
	Printscreen(“No Attribute Getting Response received!”);
	TestCaseResult = “FAILED”;
	}
	}
	ELSE
	{
	Printscreen(“No Key Update Response received!”);
	TestCaseResult = “FAILED”;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=5	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Key Update Request 帧：0x93	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x??	0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2)

表 44（续）

参考数据包 SecLevel＝5	数据包 2： 协议层：DLL 帧名称：Attribute Getting Request 帧：0x8d	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x??	0x??	0x?? 0x??	0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)	网络 ID(1)	目的地址(1)	序列号(2)	帧长度(2)	属性标识符(1)	属性成员标识符(1)	多个属性值的第一个存储索引(2)	属性数目(2)	FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Key Update Response 帧：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x01| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x??	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|状态(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Key Update Request 帧：0x93	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x??	0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Request 帧：0x8d	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x??	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|MIC(n)| FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Key Update Response 帧：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x01	0x??... 0x??	0x?? 0x??

表 44（续）

参考数据包 SecLevel＝2，3，4，6，7，8	帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|MIC(n)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x??	0x??	0x?? 0x??	0x?? 0x??	0x?? ... 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|状态(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(n)| FCS(2)

6.2.16 KEK 攻击告警测试[FD-RUN016]

该测试用例用于测试启用安全的现场设备在 KEK 受到威胁时，能否正确向网关进行安全告警。

测试过程为：

a) 被测设备安全加入网络后，测试系统向被测设备发送属性配置请求，设置被测设备的 MaxKeyAttackedCnt 属性为 1；

b) 当被测设备接收到属性配置请求后，向测试系统返回属性配置响应；

c) 测试系统将接收到的属性配置响应报文与期望的报文进行比对，如果比对匹配，发送 KEDU 密钥更新请求命令包，并赋予 KEDU 错误的 MIC；

d) 当被测设备接收到错误的密钥更新请求命令包后，在 AlarmRptDur 时间内，发送安全告警包；

e) 测试系统收到被测设备的安全告警包后，向被测设备发送 KEK 密钥更新请求包；

f) 被测设备向测试系统发送 KEK 密钥更新响应包；

g) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 36 所示，具体测试说明如表 45 所示。

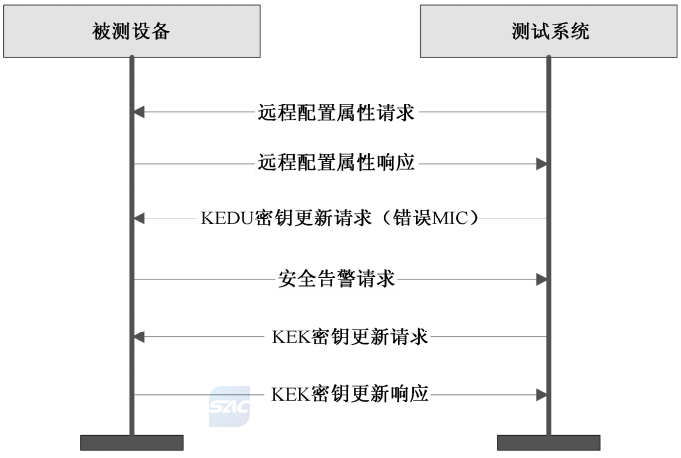

图 36 KEK 攻击告警测试时序图

表 45　KEK 攻击告警测试说明

用例名称	KEK 攻击告警测试[FD-RUN016]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥) SecLevel=2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = AttributeSettingRequest(MaxKeyAttackedCnt);
	SendPacket2 = KeyUpdateRequest(KEK with wrong MIC);
	SendPacket3 = KeyUpdataRequest(KEK);
	RefPacket1 = AttributeSettingResponse;
	RefPacket2 = SecurityAlarmRequest;
	RefPacket3 = KeyUpdateResponse(KEK);
	Send(SendPacket1,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Response Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Send(SendPacket2,KEDU,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type! =RefPacket2.type)for 2* AlarmRptDur;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Security Attack Alarm Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Printscreen("Security Attack Alarm Received successful!");
	TestCaseResult = "SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Security Attack Alarm Request received!");
	TestCaseResult = "FAILED";
	}
	}

表 45（续）

测试用例伪代码描述	}
	ELSE
	{
	Printscreen(“No Attribute Setting Response received!”);
	TestCaseResult = “FAILED”;
	}
	IF (TestCaseResult == SUCCESS)
	{
	Send(SendPacket3,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket3.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket3.type)
	{
	IF (Verify(RcvPacket.all, RefPacket3.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen(“Key Updata Response Payload error!”);
	TestCaseResult = “FAILED”;
	}
	ELSE
	{
	Printscreen(“Key Update finished, KEK Attack Alarm Test success!”);
	TestCaseResult = “SUCCESS”;
	}
	}
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=5	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x13	0x**	0x** 0x**	0x00 0x00	0x01	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Key Update Request with wrong MIC of KeyMaterial 包：0x93	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x02| 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x11 0x11 0x11 0x11| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2) 数据包 3：

表 45（续）

参考数据包 SecLevel＝5	协议层：DLL 帧名称：Key Update Request 包：0x93 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x02| 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x02 | 0x13 | 0x** | 0x** 0x** | 0x00 0x00| 0x00| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Security Attack Alarm Request 帧：0x95 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x01 | 0x?? 0x?? |0b******？1| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)|FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x02 | 0x13 | 0x** | 0x** 0x** | 0x00 0x00 | 0x01 | 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)| MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Key Update Request with wrong MIC of KeyMaterial 包：0x93 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x02| 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x11 0x11 0x11 0x11| 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)| MIC(n)| FCS(2) 数据包 3：

表 45（续）

	协议层：DLL 帧名称：Key Update Request 包：0x93 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x02| 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)| MIC(n)| FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x02 | 0x13 | 0x** | 0x** 0x** | 0x00 0x00| 0x00| 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)| MIC(n)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Security Attack Alarm Request 帧：0x95 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x01 | 0x?? 0x?? |0b******? 1| 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)| MIC(n)| FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00 | 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)| MIC(n)| FCS(2)

6.2.17 KEDU 攻击告警测试[FD-RUN017]

该测试用例用于测试启用安全的现场设备在 KEDU 受到威胁时，能否正确向网关进行安全告警。测试过程为：

a) 被测设备安全加入网络后，测试系统向被测设备发送属性配置请求，设置被测设备的 MaxKeyAttackedCnt 属性为 1；

b) 当被测设备接收到属性配置请求后，向测试系统返回属性配置响应；

c) 测试系统将接收到的属性配置响应报文与期望的报文进行比对，如果比对匹配，发送加密数据包，并赋予错误的 MIC；

d) 当被测设备接收到错误的数据包后，在 AlarmRptDur 时间内，发送安全告警包；

e) 测试系统收到被测设备的安全告警包后，向被测设备发送 KEDU 密钥更新请求包；

f) 被测系统向测试系统发送 KEDU 密钥更新响应包；

g) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 37 所示，具体测试说明如表 46 所示。

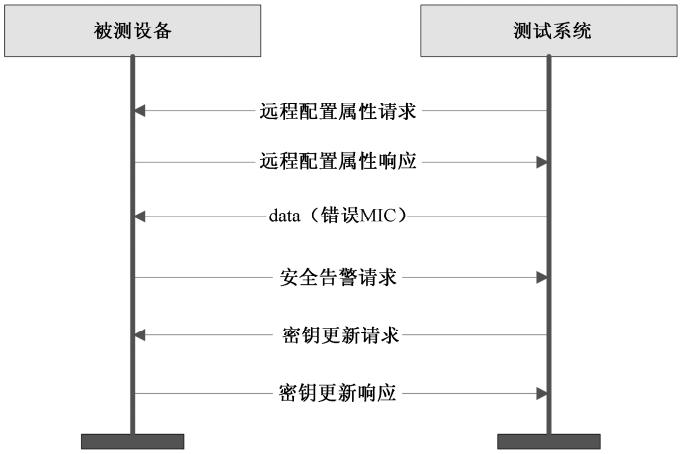

图 37　KEDU 攻击告警测试时序图

表 46　KEDU 攻击告警测试说明

用例名称	KEDU 攻击告警测试 [FD-RUN017]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥) SecLevel=2,3,4,6,7,8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = AttributeSettingRequest(MaxKeyAttackedCnt);
	SendPacket2 = Data (with wrong MIC);
	SendPacket3 = KeyUpdataRequest(KEDU);
	RefPacket1 = AttributeSettingResponse;
	RefPacket2 = SecurityAlarmRequest;
	RefPacket3 = KeyUpdateResponse(KEDU);
	Send(SendPacket1,SecLevel,KEDU);
	WHILE(Receive(RcvPacket). type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Response Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Send(SendPacket2, ,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type! =RefPacket2.type)for 2 * AlarmRptDur;
	IF(RcvPacket.type == RefPacket2.type)
	{

表 46（续）

测试用例伪代码描述	IF (Verify(RcvPacket.all, RefPacket2.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Security Attack Alarm Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Printscreen("Security Attack Alarm Received successful!");
	TestCaseResult = "SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Security Attack Alarm Request received!");
	TestCaseResult = "FAILED";
	}
	}
	}
	ELSE
	{
	Printscreen("No Attribute Setting Response received!");
	TestCaseResult = "FAILED";
	}
	IF (TestCaseResult == SUCCESS)
	{
	Send(SendPacket3,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket3.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket3.type)
	{
	IF (Verify(RcvPacket.all, RefPacket3.all, SecLevel, KEDU) ! = SUCCESS)
	{
	Printscreen("Key Updata Response Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Printscreen("Key Update finished, Security Attack Alarm Test success!");
	TestCaseResult = "SUCCESS";
	}
	}
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 46（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x13	0x**	0x** 0x**	0x00 0x00	0x01	0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)| MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：data eycrypted by KEDU with wrong MIC 帧：0x81	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x?? … 0x?? |0x11 … 0x11| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|载荷(n)| MIC(n)| FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Request 包：0x93	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x03| 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??… 0x??	0x?? 0x?? 0x?? 0x??	0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|MIC(n)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x13	0x**	0x** 0x**	0x00 0x00| 0x00| 0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)| MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Security Attack Alarm Request 帧：0x95	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x?? 0x?? |0b******? 1| 0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)| MIC(n)|FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x00	0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|MIC(n)|FCS(2)

6.2.18 KEDB 攻击告警测试[FD-RUN018]

该测试用例用于测试启用安全的现场设备在 KEDB 受到威胁时，能否正确向网关进行安全告警。测试过程为：

a) 被测设备安全加入网络后，测试系统向被测设备发送属性配置请求，设置被测设备的 MaxKeyAttackedCnt 属性为 1；

b) 当被测设备接收到属性配置请求后，向测试系统返回属性配置响应；

c) 测试系统将接收到的属性配置响应报文与期望的报文进行比对，如果比对匹配，发送加密广播数据包，并赋予错误的 MIC；

d) 当被测设备接收到广播的加密数据报后，在 AlarmRptDur 时间内，发送安全告警包；

e) 测试系统收到被测设备的安全告警包后，向被测设备发送密钥更新请求包；

f) 被测设备向测试系统发送密钥更新响应包；

g) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 38 所示，具体测试说明如表 47 所示。

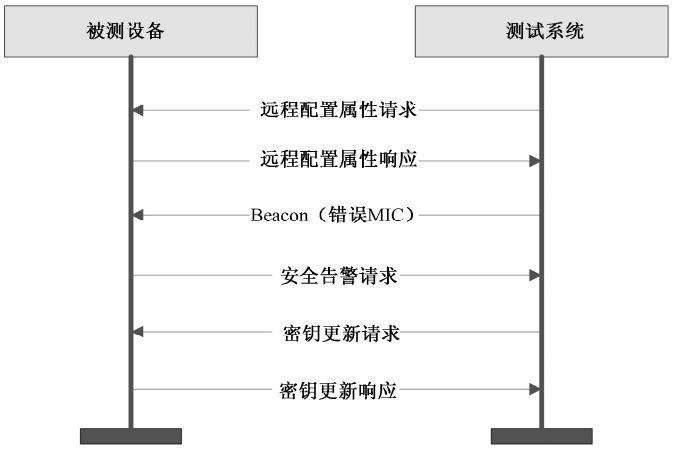

图 38 KEDB 攻击告警测试时序图

表 47 KEDB 攻击告警测试说明

用例名称	KEDB 攻击告警测试 [FD-RUN018]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥) KEDB 已激活 SecLevel=2,3,4,6,7,8
测试用例伪代码描述	TEST BODY: SendPacket1 = AttributeSettingRequest(MaxKeyAttackedCnt); SendPacket2 = broadcast data(with wrong MIC); SendPacket3 = KeyUpdataRequest; RefPacket1 = AttributeSettingResponse; RefPacket2 = SecurityAlarmRequest; RefPacket3 = KeyUpdateResponse;

表 47（续）

测试用例伪代码描述	Send(SendPacket1,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Response Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Send(SendPacket2,SecLevel,KEDB);
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for 2* AlarmRptDur;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all,SecLevel,KEDU)! = SUCCESS)
	{
	Printscreen("Security Attack Alarm Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Printscreen("Security Attack Alarm Received successful!");
	TestCaseResult = "SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Security Attack Alarm Request received!");
	TestCaseResult = "FAILED";
	}
	}
	}
	ELSE
	{
	Printscreen("No Attribute Setting Response received!");
	TestCaseResult = "FAILED";
	}
	IF (TestCaseResult == SUCCESS)
	{
	Send(SendPacket3, SecLevel,KEDU);
	WHILE(SecIn(Receive(RcvPacket),KEDU).type! =RefPacket3.type)forMaxWaitTime;
	IF(RcvPacket.type == RefPacket3.type)
	{

表 47（续）

<table>
<tr><td>测试用例伪代码描述</td><td>IF（Verify(RcvPacket.all，RefPacket3.all,SecLevel,KEDU）！＝SUCCESS)
{
Printscreen(“Key Updata Response Payload error!”)；
TestCaseResult ＝ “FAILED”；
}
ELSE
{
Printscreen(“Key Update finished，Security Attack Alarm Test success!”)；
TestCaseResult ＝ “SUCCESS”；
}
}
}
Printscreen(TestCaseResult)；

TEST RESULT：
SUCCESS or FAILED</td></tr>
<tr><td rowspan="2">参考数据包
SecLevel＝2，3，4，6，7，8</td><td>测试系统应发数据包：
数据包 1：
协议层：DLL
帧名称：Attribute Setting Request
帧：0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x02 | 0x13 | 0x** | 0x** 0x** | 0x00 0x00 | 0x01 | 0x?? … 0x?? | 0x?? 0x??
帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)|MIC(n)|FCS(2)
数据包 2：
协议层：DLL
帧名称：broadcast data eycrypted by KEDB with wrong MIC
帧：0x81 | 0xff| 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? … 0x?? |0x11 … 0x11| 0x?? 0x??
帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|载荷(n)|MIC(n)|FCS(2)
数据包 3：
协议层：DLL
帧名称：Key Update Request
帧：0x93 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x04| 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??… 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? … 0x?? | 0x?? 0x??
帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)密钥| MIC(4)|MIC(n)|FCS(2)</td></tr>
<tr><td>测试系统应收数据包：
数据包 1：
协议层：DLL
帧名称：Attribute SettingResponse
帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x02 | 0x13 | 0x** | 0x** 0x** | 0x00 0x00| 0x00| 0x?? … 0x?? | 0x?? 0x??</td></tr>
</table>

表 47（续）

参考数据包 SecLevel＝2，3，4，6，7，8	帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Security Attack Alarm Request 帧：0x95	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x?? 0x?? |0b*******? 1| 0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)|MIC(n)|FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x00| 0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|MIC(n)|FCS(2)

6.2.19 KEK 更新超时告警测试[FD-RUN019]

该测试用例用于测试启用安全的现场设备在 KEK 超时时，能否正确向网关进行安全告警。

测试过程为：

a） 被测设备安全加入网络后，测试系统向被测设备发送属性配置请求，设置被测设备的 KEK 的 KeyStatus 属性为 2；

b） 当被测设备接收到属性配置请求后，向测试系统返回属性配置响应；

c） 当被测设备在 AlarmRptDur 时间内，发送安全告警包；

d） 测试系统收到被测设备的安全告警包后，向被测设备发送密钥更新请求包；

e） 被测设备向测试系统发送密钥更新响应包；

f） 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 39 所示，具体测试说明如表 48 所示。

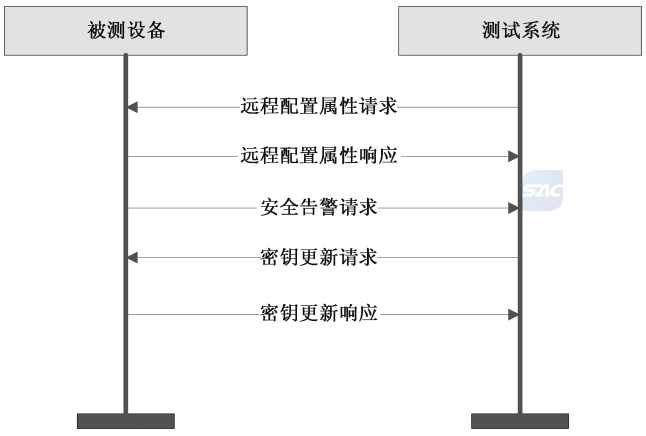

图 39 KEK 更新超时告警测试时序图

表 48　KEK 更新超时告警测试说明

用例名称	KEK 更新超时告警测试 [FD-RUN019]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥) KEK 已激活 SecLevel=2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = AttributeSettingRequest(Key Status);
	SendPacket2 =KeyUpdataRequest;
	RefPacket1 = AttributeSettingResponse;
	RefPacket2 = SecurityAlarmRequest;
	RefPacket3 = KeyUpdateResponse;
	Send(SendPacket1,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Response Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for 2 * AlarmRptDur;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Security Attack Alarm Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Printscreen("Security Attack Alarm Received successful!");
	TestCaseResult = "SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Security Attack Alarm Request received!");
	TestCaseResult = "FAILED";
	}
	}

表 48（续）

测试用例伪代码描述	}
	ELSE
	{
	Printscreen(“No Attribute Setting Response received!”);
	TestCaseResult = “FAILED”;
	}
	IF (TestCaseResult == SUCCESS)
	{
	Send(SendPacket2, ,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket3.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket3.type)
	{
	IF (Verify(RcvPacket.all, RefPacket3.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen(“Key Updata Response Payload error!”);
	TestCaseResult = “FAILED”;
	}
	ELSE
	{
	Printscreen(“Key Update finished, Security Attack Alarm Test success!”);
	TestCaseResult = “SUCCESS”;
	}
	}
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=5	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x02 | 0x84 | 0x07 | 0x?? 0x?? | 0x00 0x01 | 0x02 | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Key Update Request 包：0x93 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x02| 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)| FCS(2)

表 48（续）

参考数据包 SecLevel=5	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x02 | 0x84 | 0x07 | 0x?? 0x?? | 0x00 0x01 | 0x00| 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Security Attack Alarm Request 帧：0x95 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x01 | 0x?? 0x?? |0b******1? | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)|FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x03 | 0x?? 0x?? | 0x00| 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|FCS(2)
参考数据包 SecLevel=2,3,4,6,7,8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x02 | 0x84 | 0x07 | 0x?? 0x?? | 0x00 0x01 | 0x02 | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)|MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Key Update Request 包：0x93 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x1D | 0x?? 0x?? | 0x02| 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x??... 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|MIC(n)| FCS(2)

表 48（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x84	0x07	0x?? 0x??	0x00 0x01	0x00| 0x??... 0x??	0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|MIC(n)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Security Attack Alarm Request 帧：0x95	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x?? 0x?? |0b*******1?	0x??... 0x??	0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)|MIC(n)| FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x00| 0x??... 0x??	0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|MIC(n)| FCS(2)

6.2.20 KEDU 更新超时告警测试[FD-RUN020]

该测试用例用于测试启用安全的现场设备在 KEDU 超时时，能否正确向网关进行安全告警。

测试过程为：

a) 被测设备安全加入网络后，测试系统向被测设备发送属性配置请求，设置被测设备的 KEDU 的 KeyStatus 属性为 2；

b) 当被测设备接收到属性配置请求后，向测试系统返回属性配置响应；

c) 当被测设备在 AlarmRptDur 时间内，发送安全告警包；

d) 测试系统收到被测设备的安全告警包后，向被测设备发送密钥更新请求包；

e) 被测设备向测试系统发送密钥更新响应包；

f) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 40 所示，具体测试说明如表 49 所示。

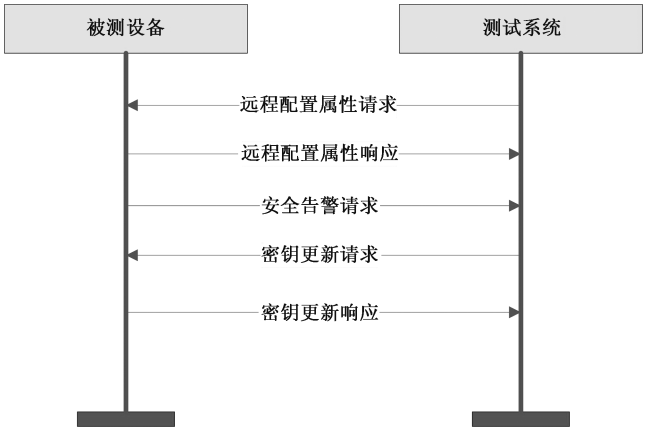

图 40　KEDU 更新超时告警测试时序图

表 49　KEDU 更新超时告警测试说明

用例名称	KEDU 更新超时告警测试[FD-RUN020]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥) KEDU 已激活 SecLevel=2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = AttributeSettingRequest(Key Status);
	SendPacket2 =KeyUpdataRequest;
	RefPacket1 = AttributeSettingResponse;
	RefPacket2 = SecurityAlarmRequest;
	RefPacket3 = KeyUpdateResponse;
	Send(SendPacket1,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Response Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	WHILE(Receive(RcvPacket).type! =RefPacket2.type)for 2 * AlarmRptDur;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all,SecLevel,KEDU) ! = SUCCESS)
	{

表 49（续）

测试用例伪代码描述	Printscreen("Security Attack Alarm Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Printscreen("Security Attack Alarm Received successful!");
	TestCaseResult = "SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Security Attack Alarm Request received!");
	TestCaseResult = "FAILED";
	}
	}
	}
	ELSE
	{
	Printscreen("No Attribute Setting Response received!");
	TestCaseResult = "FAILED";
	}
	IF (TestCaseResult == SUCCESS)
	{
	Send(SendPacket2, ,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket3.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket3.type)
	{
	IF (Verify(RcvPacket.all, RefPacket3.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Key Updata Response Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Printscreen("Key Update finished, Security Attack Alarm Test success!");
	TestCaseResult = "SUCCESS";
	}
	}
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 49（续）

	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x84	0x07	0x?? 0x??	0x00 0x01	0x02	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Key Update Request 包：0x93	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x03| 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)| FCS(2)
参考数据包 SecLevel=5	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Response 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x84	0x07	0x?? 0x??	0x00 0x01	0x00| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Security Attack Alarm Request 帧：0x95	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x?? 0x?? |0b*******1?	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)| FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x00| 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|FCS(2)

表 49（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x84	0x07	0x?? 0x??	0x00 0x01	0x02	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)|MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Key Update Request 包：0x93	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x03| 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)|MIC(n)| FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Response 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x84	0x07	0x?? 0x??	0x00 0x01	0x00| 0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Security Attack Alarm Request 帧：0x95	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x?? 0x?? |0b*******1?	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)| MIC(n)|FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x00| 0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|MIC(n)|FCS(2)

6.2.21 KEDB 更新超时告警测试[FD-RUN021]

该测试用例用于测试启用安全的现场设备在 KEDB 超时时，能否正确向网关进行安全告警。

测试过程为：

a) 被测设备安全加入网络后，测试系统向被测设备发送属性配置请求，设置被测设备的 KEDB 的 KeyStatus 属性为 2；

b) 当被测设备接收到属性配置请求后，向测试系统返回属性配置响应；

c) 当被测设备在 AlarmRptDur 时间内，发送安全告警包；

d) 测试系统收到被测设备的安全告警包后，向被测设备发送密钥更新请求包；

e) 被测设备向测试系统发送密钥更新响应包；

f) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过；

具体时序如图 41 所示，具体测试说明如表 50 所示。

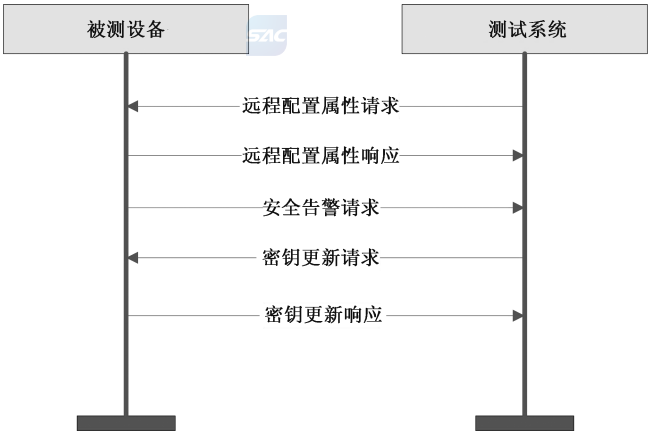

图 41 KEDB 更新超时告警测试时序图

表 50 KEDB 更新超时告警测试说明

用例名称	KEDB 更新超时告警测试[FD-RUN021]
被测设备	现场设备
依赖测试条件	被测设备已安全加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥) KEDB 已激活 SecLevel=2～8

测试用例伪代码描述	TEST BODY：
	SendPacket1 = AttributeSettingRequest(Key Status)；
	SendPacket2 = KeyUpdataRequest；
	RefPacket1 = AttributeSettingResponse；
	RefPacket2 = SecurityAlarmRequest；
	RefPacket3 = KeyUpdateResponse；
	Send(SendPacket1,SecLevel,KEDU)；
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime；
	IF(RcvPacket.type == RefPacket1.type)

表 50（续）

测试用例伪代码描述	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Response Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	WHILE(Receive(RcvPacket).type! =RefPacket2.type)for 2 * AlarmRptDur;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF(Verify(RcvPacket.all,RefPacket2.all,SecLevel,KEDU)! =SUCCESS)
	{
	Printscreen("Security Attack Alarm Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Printscreen("Security Attack Alarm Received successful!");
	TestCaseResult = "SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Security Attack Alarm Request received!");
	TestCaseResult = "FAILED";
	}
	}
	}
	ELSE
	{
	Printscreen("No Attribute Setting Response received!");
	TestCaseResult = "FAILED";
	}
	IF (TestCaseResult == SUCCESS)
	{
	Send(SendPacket2,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket3.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket3.type)
	{
	IF (Verify(RcvPacket.all, RefPacket3.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Key Updata Response Payload error!");
	TestCaseResult = "FAILED";

表 50（续）

测试用例伪代码描述	}
	ELSE
	{
	Printscreen(“Key Update finished, Security Attack Alarm Test success!”);
	TestCaseResult = “SUCCESS”;
	}
	}
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=5	测试系统应发数据包: 数据包 1: 协议层: DLL 帧名称:Attribute Setting Request 帧:0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x84	0x07	0x?? 0x??	0x00 0x01	0x02	0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)|FCS(2) 数据包 2: 协议层:DLL 帧名称: Key Update Request 包:0x93	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x04| 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)| FCS(2)
	测试系统应收数据包: 数据包 1: 协议层:DLL 帧名称:Attribute SettingResponse 帧: 0x90	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x84	0x07	0x?? 0x??	0x00 0x01	0x00| 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2) 数据包 2: 协议层:DLL 帧名称: Security Attack Alarm Request 帧: 0x95	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x?? 0x?? |0b*******1?	0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)|FCS(2)

表 50（续）

参考数据包 SecLevel＝5	数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x00	0x?? 0x?? 帧域说明：帧控制(1)	网络 ID(1)	源地址(1)	序列号(2)	帧长度(2)	密钥 ID(2)	密钥状态(1)	FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x84	0x07	0x?? 0x??	0x00 0x01	0x02	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(1)| MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Key Update Request 包：0x93	0xaa	0x03	0x?? 0x??	0x00 0x1D	0x?? 0x??	0x04| 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 0x?? 0x??	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)| MIC(n)| FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute SettingResponse 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x02	0x84	0x07	0x?? 0x??	0x00 0x01	0x00| 0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)| MIC(n)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Security Attack Alarm Request 帧：0x95	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x01	0x?? 0x?? |0b*******1?	0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1)| MIC(n)|FCS(2) 数据包 3： 协议层：DLL 帧名称：Key Update Response 包：0x94	0xaa	0x03	0x?? 0x??	0x00 0x03	0x?? 0x??	0x00| 0x??... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|密钥 ID(2)|密钥状态(1)|MIC(n)|FCS(2)

6.3 离开过程测试集

现场设备被动离开网络[FD-QUIT001]

该测试用例测试现场设备能否正确响应离开请求。

测试过程为：

a) 被测设备加入网络后，测试系统向被测设备发送离开请求；

b) 被测设备接收到请求后，向测试系统返回离开响应；

c) 测试系统将接收的离开响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 42 所示，具体测试说明如表 51 所示。

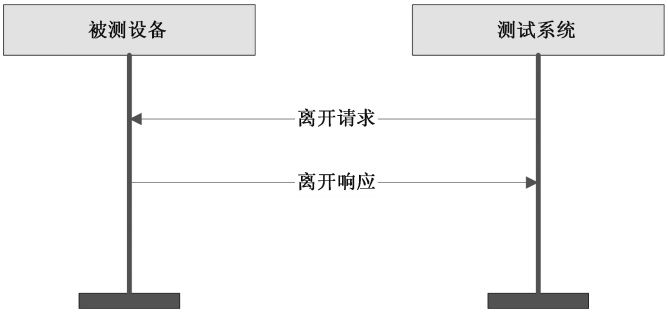

图 42 现场设备被动离开网络时序图

表 51 现场设备被动离开网络说明

用例名称	现场设备被动离开网络[FD-QUIT001]
被测设备	现场设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥) KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = LeaveRequest;
	RefPacket = LeaveResponse;
	Send(SendPacket,SecLevel, KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel, KEDU) ! = SUCCESS)
	{
	Printscreen("Leave Response Payload error!");
	TestCaseResult =FAILED;

表 51（续）

测试用例伪代码描述	}
	ELSE
	{
	Printscreen(“Leaving Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“NoLeave Response received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESSor FAILED

参考数据包 SecLevel=0，1，5	测试系统应发数据包： 协议层:DLL 帧名称：Leave Request 帧：0x87 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|FCS(2)
	测试系统应收数据包： 协议层:DLL 帧名称：Leave Response 帧：0x88 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 协议层:DLL 帧名称：Leave Request 帧：0x87 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x00 | 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)| MIC(n)| FCS(2)
	测试系统应收数据包： 协议层:DLL 帧名称：Leave Response 帧：0x88 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x00 | 0x??... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)| MIC(n)| FCS(2)

7 接入设备测试集

7.1 加入过程测试集

7.1.1 接入设备加入网络(正向测试)[AD-JOIN001]

该测试用例测试接入设备能否正确加入 WIA-FA 网络。

测试过程为:

a) 被测设备向测试系统发送接入设备加入请求;

b) 测试系统将接收的接入设备加入请求报文与期望的报文进行比对,如果比对匹配,则测试通过,并向被测设备返回接入设备加入响应。

具体时序如图 43 所示,具体测试说明如表 52 所示。

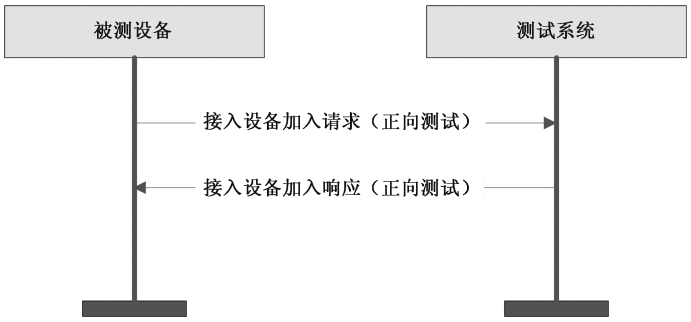

图 43 接入设备加入网络(正向测试)时序图

表 52 接入设备加入网络(正向测试)说明

用例名称	接入设备加入网络(正向测试)[AD-JOIN001]
被测设备	接入设备
依赖测试条件	被测设备未加入测试系统网络 配置被测设备网络 ID 测试系统已登记被测设备的长地址 设备上电启动

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredADJoinResponse;
	RefPacket =WiredADJoinRequest;
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("Wired ADJoinRequest Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{

表 52（续）

测试用例伪代码描述	wiredSend(SendPacket)；
	Printscreen(“Joining Network Test Success!”)；
	TestCaseResult =SUCCESS；
	}
	}
	ELSE
	{
	Printscreen(“No Wired ADJoinRequest received!”)；
	TestCaseResult = FAILED；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired ADJoin Response 帧：0x01 |0x?? … 0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AD 长地址(8)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired ADJoin Request 帧：0x00	0x?? … 0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AD 长地址(8)| 对端地址(1)|序列号(2)|长度(2)|载荷(3)

7.1.2 接入设备加入网络(反向测试)[AD-JOIN002]

该测试用例测试接入设备能否加入 WIA-FA 网络。

测试过程为：

a) 被测设备向测试系统发送接入设备加入请求；

b) 测试系统将接收的接入设备加入请求报文与期望的报文进行比对，如果比对匹配，则向被测设备返回错误的接入设备加入响应，具体情况如下；

1) 网络 ID 不匹配；

2) 认证失败(SecLevel 不为 0 时)；

3) 网络规模超限；

4) 加入超时。

参考数据包中具体载荷内容如表 53 所示。

表 53　接入设备加入网络(反向测试)参考数据包载荷

测试内容	测试系统应发加入响应数据载荷
网络 ID 不匹配	0x01 | 0x?? |0x02
	加入状态(1)|分配的 AdID(1)|分配 AD 地址
认证失败	0x02 | 0x?? |0x02
	加入状态(1)|分配的 AdID(1)|分配的 AD 地址
网络规模超限	0x03 | 0x?? |0x02
	加入状态(1)|分配的 AdID(1)|分配的 AD 地址
加入超时	0x04 | 0x?? |0x02
	加入状态(1)|分配的 AdID(1)|分配的 AD 地址

c)　测试系统向被测设备发送远程配置属性(超帧)请求,被测设备接收到请求后,不向测试系统返回远程配置属性(超帧)证实,则测试通过。

该测试用例用于接入设备入网的四种反向测试,测试体应循环执行,具体时序如图 44 所示,具体测试说明如表 54 所示。

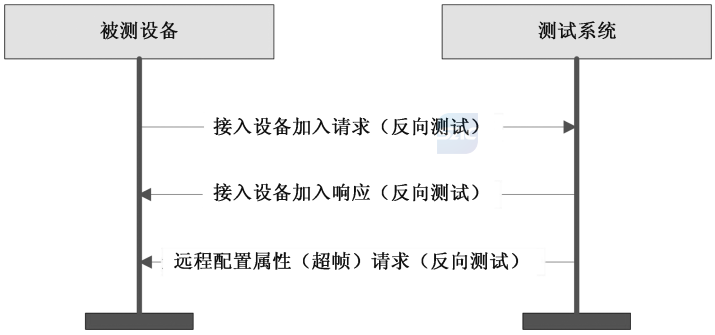

图 44　接入设备加入网络(反向测试)时序图

表 54　接入设备加入网络(反向测试)说明

用例名称	接入设备加入网络(反向测试)[AD-JOIN002]
被测设备	接入设备
依赖测试条件	被测设备未加入测试系统网络

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredADJoinResponse;
	SendPacket2 = WiredAttributeSettingRequest(Superframe);
	RefPacket1 =WiredADJoinRequest;
	RefPacket2 =WiredAttributeSettingConfirm;
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket .type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all) ! = SUCCESS)
	{

表 54（续）

测试用例伪代码描述	Printscreen(“Wired AD JoinRequest Payload error!”);
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket1);
	Printscreen(“Wired AD JoinRequest Payload correct!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No Wired AD JoinRequest received!”);
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	Send(SendPacket2);
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket .type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all) == SUCCESS)
	{
	Printscreen(“JoiningNetwork Test Failed!”);
	TestCaseResult =FAILED;
	}
	}
	ELSE
	{
	Printscreen(“JoiningNetwork Test Success!”);
	TestCaseResult = SUCCESS;
	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 54（续）

参考数据包	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired AD Join Response 帧：0x01 |0x?? … 0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AD 长地址(8) |对端地址(1)|序列号(2)|长度(2)|载荷(3) 数据包 2： 协议层：Wired DLL 帧名称：WiredAttribute Setting Request 帧：0x0c	0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired AD Join Request 帧：0x00	0x?? … 0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AD 长地址(8)|对端地址(1)|序列号(2)|长度(2)|载荷(3) 数据包 2： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d	0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

7.1.3 超帧分配测试[AD-JOIN003]

该测试用例测试接入设备能否正确响应远程配置属性(超帧)请求。

测试过程为：

a) 被测设备加入网络后，测试系统向被测设备发送远程配置属性(超帧)请求；

b) 被测设备接收到请求后，向测试系统返回远程配置属性(超帧)证实；

c) 测试系统将接收的远程配置属性(超帧)证实报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 45 所示，具体测试说明如表 55 所示。

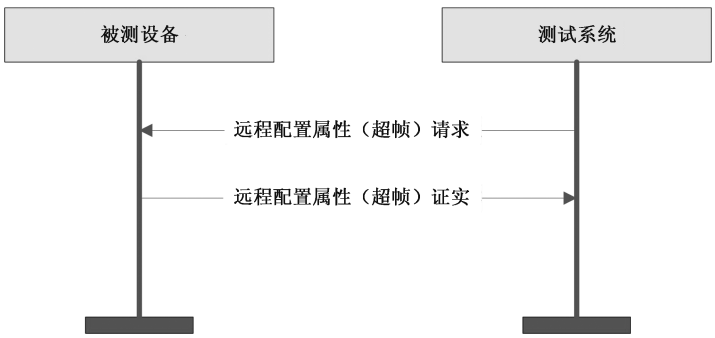

图 45 超帧分配测试时序图

表 55　超帧分配测试说明

用例名称	超帧分配测试[AD-JOIN003]
被测设备	接入设备
依赖测试条件	被测 AD 设备已加入测试系统网络

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredAttributeSettingRequest(Superframe);
	RefPacket =WiredAttributeSettingConfirm;
	wiredSend(SendPacket);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredAttribute Setting Confirm Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Superframe Resource Distributing Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredAttribute Setting Confirm received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 协议层:Wired DLL 帧名称:WiredAttribute Setting Request 帧:0x0c | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明:服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包: 协议层:Wired DLL 帧名称:Wired Attribute Setting Confirm 帧:0x0d | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明:服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

7.1.4 链路分配测试[AD-JOIN004]

该测试用例测试接入设备能否正确响应远程配置属性(链路)请求。

测试过程为:

a) 被测设备加入网络后,测试系统向被测设备发送远程配置属性(链路)请求;

b) 被测设备接收到请求后,向测试系统返回远程配置属性(链路)证实;

c) 测试系统将接收的远程配置属性(链路)证实报文与期望的报文进行比对,如果比对匹配,则测试通过。

具体时序如图 46 所示,具体测试说明如表 56 所示。

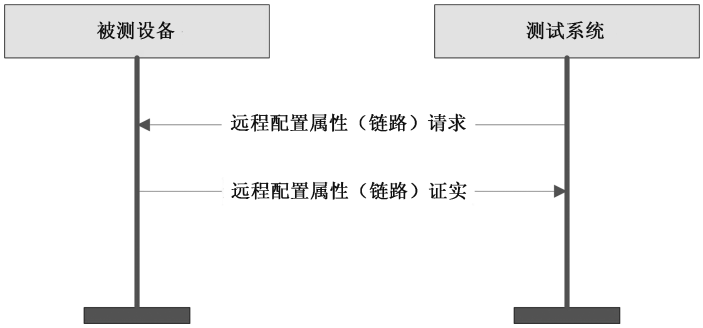

图 46 链路分配测试时序图

表 56 链路分配测试说明

用例名称	链路分配测试[AD-JOIN004]
被测设备	接入设备
依赖测试条件	被测设备已加入测试系统网络 测试系统已完成对被测设备的超帧配置

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredAttributeSettingRequest(Link);
	RefPacket =WiredAttributeSettingConfirm;
	wiredSend(SendPacket);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredAttribute Setting Confirm Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Link Resource Distributing Test Success!");
	TestCaseResult =SUCCESS;
	}

表 56（续）

测试用例伪代码描述	}
	ELSE
	{
	Printscreen(“No WiredAttribute Setting Confirm received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 协议层：Wired DLL 帧名称：WiredAttribute Setting Request 帧：0x0c | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

7.1.5 KEDU 密钥分发测试[AD-JOIN005]

该测试用例测试启用安全的 WIA-FA 的接入设备能否正确配置 KEDU 密钥。

测试过程为：

a） 被测设备加入网络后，测试系统向被测接入设备发送密钥分发请求(KEDU)；

b） 被测接入设备接收到请求后，向测试系统返回密钥分发证实；

c） 测试系统将接收的证实报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 47 所示，具体测试说明如表 57 所示。

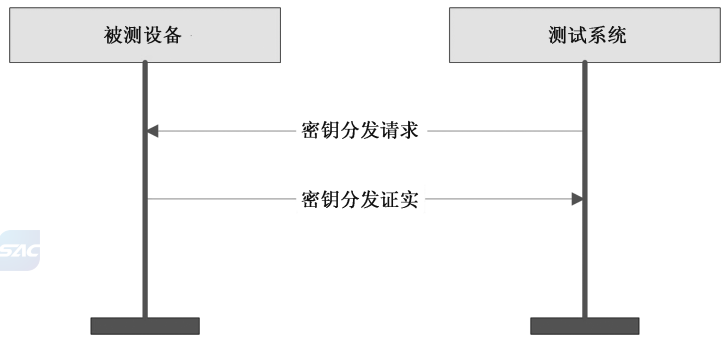

图 47 KEDU 密钥分发测试时序图

表 57　KEDU 密钥分发测试说明

用例名称	KEDU 密钥分发测试[AD-JOIN005]
被测设备	接入设备
依赖测试条件	被测设备已加入测试系统网络

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredKeyEstablishRequest(KEDU);
	RefPacket =WiredKeyEstablishConfirm;
	wiredSend(SendPacket);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredKey Establsih Confirm Payload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	Printscreen("KEDU Key Establish Test Success!");
	TestCaseResult ="SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No WiredKey Establish Confirm received!");
	TestCaseResult = "FAILED";
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 协议层:Wired DLL 帧名称:Wired Key Establish Request 帧:0x10 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x03 | 0x?? … 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明:服务标识符(1)|AdID(1)|目的地址(1)|包序列号(2)|包长度(2)|目的地址(2)|密钥 ID(2)|密钥类型(1)|密钥激活时间(6)|密钥(n)|密钥 MIC(4)|
	测试系统应收数据包: 协议层:Wired DLL 帧名称:WiredKey Establish Confirm 帧:0x11 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? |0x?? 0x?? 0x?? 0x?? | 0x00 帧域说明:服务标识符(1)|AdID(1)|源地址(1)|包序列号(2)|包长度(2)|源地址(2)|密钥 ID(4)|密钥状态(1)

7.1.6 KEDB 密钥分发测试[AD-JOIN006]

该测试用例测试启用安全的 WIA-FA 的接入设备能否正确配置 KEDB 密钥。

测试过程为：

a) 被测设备加入网络后，测试系统向被测接入设备发送密钥分发请求(KEDB)；

b) 被测接入设备接收到请求后，向测试系统返回密钥分发证实；

c) 测试系统将接收的证实报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 48 所示，具体测试说明如表 58 所示。

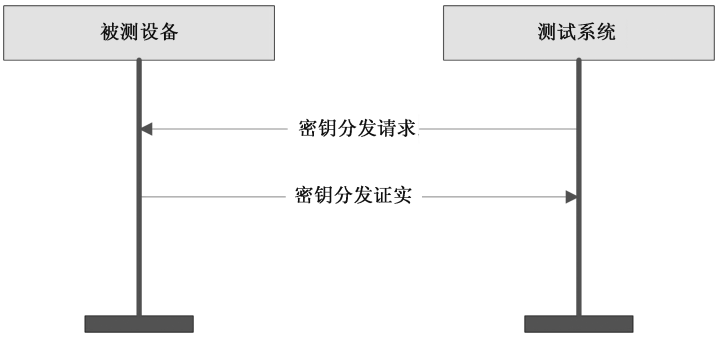

图 48 KEDB 密钥分发测试时序图

表 58 KEDB 密钥分发测试说明

用例名称	KEDB 密钥分发测试[AD-JOIN006]
被测设备	接入设备
依赖测试条件	被测设备已加入测试系统网络

测试用例伪代码描述	TEST BODY：
	SendPacket = WiredKeyEstablishRequest(KEDB)；
	RefPacket =WiredKeyEstablishConfirm；
	wiredSend(SendPacket)；
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxScanTime；
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredKey Establsih Confirm Payload error!")；
	TestCaseResult ="FAILED"；
	}
	ELSE
	{
	Printscreen("KEDU Key Establish Test Success!")；
	TestCaseResult ="SUCCESS"；
	}
	}
	ELSE
	{

表 58（续）

测试用例伪代码描述	Printscreen(“No WiredKey Establish Confirm received!”); TestCaseResult = “FAILED”; } Printscreen(TestCaseResult); TEST RESULT: SUCCESS or FAILED
参考数据包	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired Key Establish Request 帧：0x10 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x03 | 0x?? ... 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序列号(2)|包长度(2)|目的地址(2)|密钥 ID(2)|密钥类型(1)|密钥激活时间(6)|密钥(n)|密钥 MIC(4)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：WiredKey Establish Confirm 帧：0x11 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? |0x?? 0x?? 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序列号(2)|包长度(2)|源地址(2)|密钥 ID(4)|密钥状态(1)

7.2 运行过程测试集

7.2.1 发送信标测试[AD-RUN001]

该测试用例测试启用安全的接入设备能否周期性发送加密的信标帧。

测试过程为：

a) 在 4 个超帧时间内，测试系统(现场设备)应至少接收到 3 个信标帧；

b) 测试系统将接收的信标帧报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 49 所示，具体测试说明如表 59 所示。

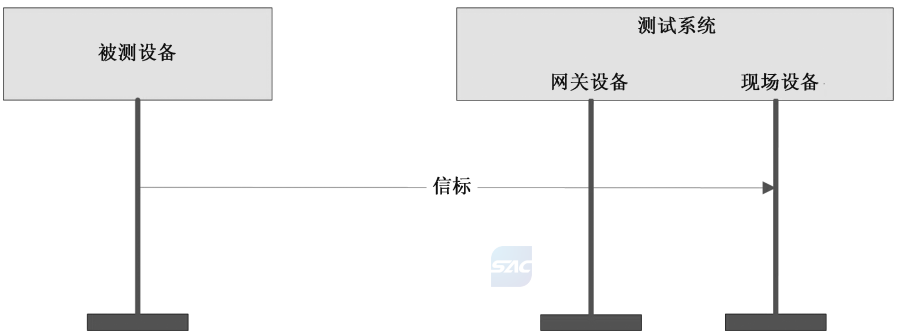

图 49 发送信标测试时序图

表 59　发送信标测试说明

用例名称	安全信标测试[AD-RUN001]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关)网络 测试系统已完成对被测设备的资源配置(超帧、链路) KS=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关)网络 测试系统已完成对被测设备的资源配置(超帧、链路、密钥)

测试用例伪代码描述	TEST BODY:
	RefPacket = Beacon;
	Count = 0;
	WHILE(1)|for 4 * SuperframeCycle
	{
	IF(Receive(RcvPacket).type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KS) ! = SUCCESS)
	{
	Printscreen ("Beacon Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	Count++;
	}
	}
	IF(Count < 3)
	{
	Printscreen ("Not EnoughBeacon received!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	Printscreen("Beacon Broadcast Test Success!");
	TestCaseResult = "SUCCESS";
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 59（续）

参考数据包 SecLevel = 0，1，5	测试系统应发数据包： 无
	测试系统应收数据包： 协议层：DLL 帧名称：Beacon 帧：0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|信标负载(1)|FCS(2)
参考数据包 SecLevel = 2，3，4，6，7，8	测试系统应发数据包： 无
	测试系统应收数据包： 协议层：DLL 帧名称：Beacon 帧：0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x?? | 0x?? … 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|信标负载(1)| MIC(n)| FCS(2)

7.2.2 现场设备加入网络[AD-RUN002]

该测试用例测试接入设备能否正确处理现场设备加入网络。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，向测试系统(现场设备)发送信标；

b) 测试系统(现场设备)将接收的信标与期望的报文进行比对，如果比对匹配，则向被测设备发送加入请求；

c) 被测设备接收到请求后，向测试系统(网关设备)发送设备加入指示；

d) 测试系统(网关设备)将接收的设备加入指示报文与期望的报文进行比对，如果比对匹配，则向被测设备返回设备加入响应；

e) 被测设备接收到响应后，向测试系统(现场设备)发送加入响应；

f) 测试系统(现场设备)将接收的加入响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 50 所示，具体测试说明如表 60 所示。

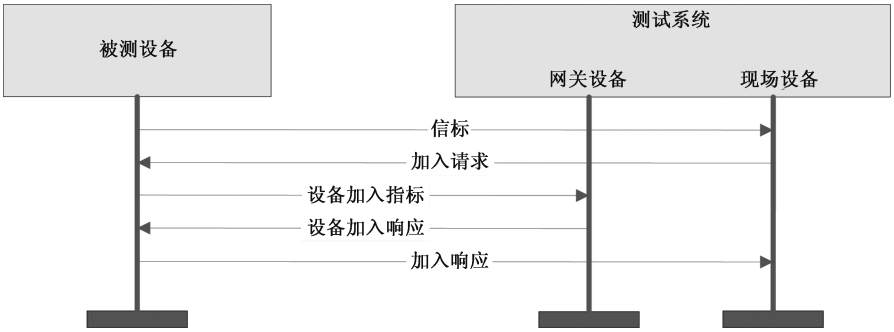

图 50　现场设备加入网络测试时序图

表 60　现场设备加入网络测试说明

用例名称	现场设备加入网络测试[AD-RUN002]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 KS=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置

测试用例伪代码描述	TEST BODY:
	SendPacket1 = JoinRequest;
	SendPacket2= WiredFDJoinResponse;
	RefPacket1 = Beacon;
	RefPacket2 = WiredFDJoinIndication;
	RefPacket3 = JoinResponse;
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxScanTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KS) ! = SUCCESS)
	{
	Printscreen("Beacon Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket1,SecLevel,KS);
	Printscreen("Beacon Payload correct!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Beacon received!");
	TestCaseResult = FAILED;

表 60（续）

测试用例伪代码描述	}
	IF(TestCaseResult == SUCCESS)
	{
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket2.all) ! = SUCCESS)
	{
	Printscreen("Wired FD JoinConfirm Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	wiredSend(SendPacket2);
	Printscreen("WiredFD JoinConfirm Payload correct!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredFD JoinConfirm received!");
	TestCaseResult = FAILED;
	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	WHILE(Receive(RcvPacket).type ! = RefPacket3.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket3.type)
	{
	IF (Verify(RcvPacket.all, RefPacket3.all,SecLevel,KS) ! = SUCCESS)
	{
	Printscreen("Join Response Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Joining Network Test Success!");
	TestCaseResult =SUCCESS;

表 60（续）

测试用例伪代码描述	}
	}
	ELSE
	{
	Printscreen(“No Join Response received!”);
	TestCaseResult = FAILED;
	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Join Request 帧：0x05 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(8)|序列号(2)|帧长度(2)|FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：WiredFD Join Response 帧：0x07 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x00 0x11 | 0x?? 0x?? | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired FD Join Indication 帧：0x06 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

表 60（续）

参考数据包 SecLevel=0	数据包 3： 协议层：DLL 帧名称：Join Response 帧：0x06 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x02 | 0x?? | 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(8)|序列号(2)|帧长度(2)|加入状态(1)|分配的短地址(1)|FCS(2)
参考数据包 SecLevel=1，5	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Join Request 帧：0x05 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x08| 0x?? ⋯ 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(8)|序列号(2)|帧长度(2)|安全材料(8)| FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired FD Join Response 帧：0x07 | 0x?? | 0x?? ⋯ 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? ⋯ 0x?? | 0x00 | 0x03 帧域说明：服务标识符(1)|AdID(1)|目的地址(8)|包序号(2)|服务参数长度(2)|AD 数量(1)|AD 列表(n)|状态(1)|短地址(1)|
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80 | 0xaa | 0xff | 0x?? 0x?? | 0x00 0x11 | 0x?? 0x?? | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|信标负载(1)|FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired FD Join Request 帧：0x06 | 0x?? | 0x?? ⋯ 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x?? |0x?? | 0x??… 0x?? 帧域说明：服务标识符(1)| AdID(1)|源地址(8)|包序号(2)|服务参数长度(2)|源地址(8)| AdID(1)|安全材料(8) 数据包 3： 协议层：DLL 帧名称：Join Response 帧：0x06 | 0xaa | 0x?? … 0x?? | 0x?? 0x?? | 0x00 0x02 | 0x00 | 0x03 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(8)|序列号(2)|帧长度(2)|加入状态(1)|分配的短地址(1)|FCS(2)

表 60（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：DLL 帧名称：Join Request 帧：0x05	0xaa	0x?? … 0x??	0x?? 0x??	0x00 0x08	0x?? … 0x??	0x??… 0x??	0x?? 0x?? 帧域说明：帧控制(1)	网络 ID(1)	源地址(8)	序列号(2)	帧长度(2)	安全材料(8)	MIC(n)	FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired FD Join Response 帧：0x07	0x??	0x?? … 0x??	0x?? 0x??	0x?? 0x??	0x??	0x?? … 0x??	0x00	0x03 帧域说明：服务标识符(1)	AdID(1)	目的地址(8)	包序号(2)	服务参数长度(2)	AD 数量(1)	AD 列表(n)	状态(1)	短地址(1)	
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80	0xaa	0xff	0x?? 0x??	0x00 0x11	0x?? 0x??	0x?? 0x?? |0x?? 0x??	0x?? 0x??	0x??	0x?? … 0x??	0x??…0x?? |0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|信标负载(1)| MIC(n)| FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired FD Join Request 帧：0x06	0x??	0x?? … 0x??	0x?? 0x??	0x?? 0x??	0x?? … 0x?? |0x??	0x??… 0x?? 帧域说明：服务标识符(1)|AdID(1)|源地址(8)|包序号(2)|服务参数长度(2)|源地址(8)| AdID(1)|安全材料(8) 数据包 3： 协议层：DLL 帧名称：Join Response 帧：0x06	0xaa	0x?? … 0x??	0x?? 0x??	0x00 0x02	0x00	0x03	0x??… 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(8)|序列号(2)|帧长度(2)|加入状态(1)|分配的短地址(1)|MIC(n)| FCS(2)

7.2.3 双向时间同步测试[AD-RUN003]

该测试用例测试接入设备能否正确进行双向时间同步。

测试过程为：

a） 被测设备向测试系统发送信标；

b） 测试系统将接收的信标与期望的报文进行比对，如果比对匹配，则向被测设备发送双向时间

同步请求；

c） 被测设备接收到请求后，向测试系统返回双向时间同步响应；

d） 测试系统将接收的双向时间同步响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 51 所示，具体测试说明如表 61 所示。

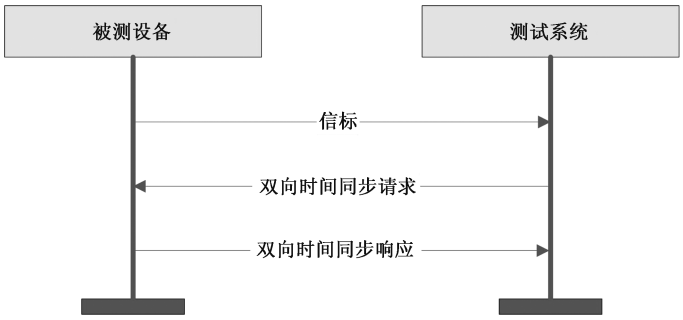

图 51 双向时间同步测试时序图

表 61 双向时间同步测试说明

用例名称	双向时间同步测试[AD-RUN003]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统网络(网关设备) 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已通过被测设备加入网络 KS=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备) 测试系统(网关设备)已完成对被测设备的资源配置(超帧、链路、密钥等) 测试系统(现场设备)已通过被测设备加入网络

测试用例伪代码描述	TEST BODY:
	SendPacket = TimeSynchronizeRequest;
	RefPacket1 = Beacon;
	RefPacket2 = TimeSynchronizeResponse;
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxScanTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all, SecLevel,KS) ! = SUCCESS)
	{
	Printscreen("Beacon Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket, SecLevel,KS);
	Printscreen("Beacon Payload correct!");
	TestCaseResult =SUCCESS;

表 61（续）

测试用例伪代码描述	}
	}
	ELSE
	{
	Printscreen(“No Beacon received!”);
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all, SecLevel,KS) ! = SUCCESS)
	{
	Printscreen(“Time Synchronize Response Payload error!”);
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen(“Time Synchronizing Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No Time Synchronize Response received!”);
	TestCaseResult = FAILED;
	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0, 1,5	测试系统应发数据包： 协议层:DLL 帧名称：Time Synchronize Request 帧：0x8b | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x08| 0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|现场设备发送时刻时间值(8)|FCS(2)

表 61（续）

参考数据包 SecLevel=0，1，5	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80	0xaa	0xff	0x?? 0x??	0x00 0x11	0x?? 0x??	0x?? 0x?? |0x?? 0x??	0x?? 0x??	0x??	0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|FCS(2) 数据包 2： 协议层：DLL 帧名称：Time Synchronize Response 帧：0x8c	0xaa	0x03	0x?? 0x??	0x000x10| 0x?? … 0x??	0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|现场设备发送时刻时间值(8)|接入设备接收时刻时间值(8)|FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 协议层：DLL 帧名称：Time Synchronize Request
	帧：0x8b	0xaa	0x03	0x?? 0x??	0x00 0x08	0x?? … 0x??	0x??… 0x??	0x?? 0x?? 帧域说明：帧控制(1)	网络 ID(1)	源地址(1)	序列号(2)	帧长度(2)	现场设备发送时刻时间值(8)	MIC(n)	FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Beacon 帧：0x80	0xaa	0xff	0x?? 0x??	0x00 0x11	0x?? 0x??	0x?? 0x?? |0x?? 0x??	0x?? 0x??	0x??	0x?? … 0x??	0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|超帧长度(2)|时隙长度(2)|信标帧相对时隙号(2)|共享时隙起始相对时隙号(2)|共享时隙数(1)|绝对时间值(8)|信标负载(1)|MIC(4)| FCS(2) 数据包 2： 协议层：DLL 帧名称：Time Synchronize Response 帧：0x8c	0xaa	0x03	0x?? 0x??	0x000x10| 0x?? … 0x??	0x?? … 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|现场设备发送时刻时间值(8)|接入设备接收时刻时间值(8)|MIC(4)| FCS(2)

7.2.4 超帧资源分配测试[AD-RUN004]

该测试用例测试接入设备能否正确进行现场设备超帧资源分配。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后,测试系统(网关设备)向被测设备发送远程配置属性(超帧)请求;
b) 被测设备接收到请求后,向测试系统(现场设备)发送远程配置属性(超帧)请求;
c) 测试系统(现场设备)将接收的远程配置属性(超帧)请求报文与期望的报文进行比对,如果比对匹配,则向被测设备返回远程配置属性(超帧)响应;
d) 被测设备接收到响应后,向测试系统(网关设备)发送远程配置属性(超帧)证实;
e) 测试系统(网关设备)将接收的远程配置属性(超帧)证实报文与期望的报文进行比对,如果比对匹配,则测试通过。

具体时序如图 52 所示,具体测试说明如表 62 所示。

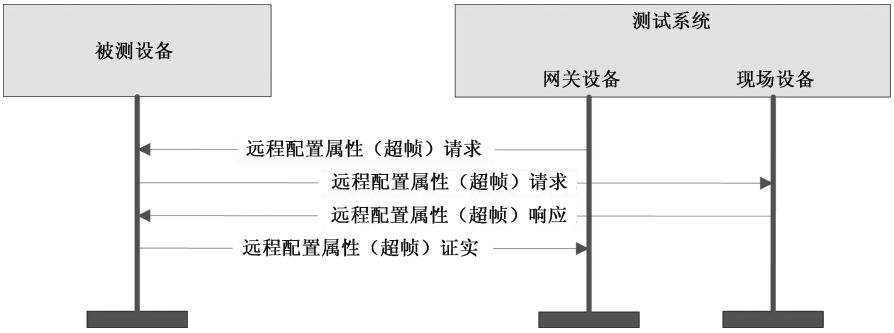

图 52 超帧资源分配测试时序图

表 62 超帧资源分配测试说明

用例名称	超帧资源分配测试[AD-RUN004]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已通过被测设备加入网络 KS=NULL
依赖测试条件 SecLevel=2~8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置(超帧、链路、密钥) 测试系统(现场设备)已通过被测设备加入网络

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredAttributeSettingRequest(Superframe);
	SendPacket2= AttributeSettingResponse;
	RefPacket1 = AttributeSettingRequest(Superframe);
	RefPacket2 = WiredAttributeSettingConfirm;
	wiredSend(SendPacket1);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KS) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Request (Superframe) Payload error!");

表 62（续）

测试用例伪代码描述	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket2,SecLevel,KS);
	Printscreen("Attribute Setting Request (Superframe) Payload correct!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Attribute Setting Request (Superframe) received!");
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket2.all) ! = SUCCESS)
	{
	Printscreen("WiredAttribute Setting Confirm Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Superframe Resource Distributing Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredAttribute Setting Confirm received!");
	TestCaseResult = FAILED;
	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 62（续）

参考数据包 SecLevel＝0， 1，5	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：WiredAttribute Setting Request 帧：0x0c	0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3) 数据包 2： 协议层：DLL 帧名称：Attribute Setting Response 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x80 |0x?? |0x?? 0x?? |0x?? 0x?? |0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x80 |0x?? |0x?? 0x?? |0x?? 0x?? |0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d	0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
参考数据包 SecLevel＝2，3， 4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Attribute Setting Request 帧：0x0c	0x?? |0x03	0x?? 0x??	0x?? 0x??	0x??	0x??	0x?? ... 0x??	0x03	0x00	0x80	0x??	0x?? 0x?? |0x?? 0x??	0x??... 0x?? 帧域说明：服务标识符(1)| AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|回调值(1) |AD 数量(1)| AD 列表(n)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n) 数据包 2： 协议层：DLL 帧名称：Attribute Setting Response 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x80 |0x?? |0x?? 0x?? |0x?? 0x?? |0x00| 0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|MIC(4)| FCS(2)

表 62（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x80 |0x?? |0x?? 0x?? |0x?? 0x?? |0x?? ... 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(4)| FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d	0x??	0x03	0x?? 0x?? |0x?? 0x??	0x?? 0x??	0x03 |0x00 |0x80	0x??	0x?? 0x?? |0x?? 0x?? |0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|源地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)

7.2.5 链路资源分配测试[AD-RUN005]

该测试用例测试接入设备能否正确进行现场设备链路资源分配。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(网关设备)向被测设备发送远程配置属性(链路)请求；

b) 被测设备接收到请求后，向测试系统(现场设备)发送远程配置属性(链路)请求；

c) 测试系统(现场设备)将接收的远程配置属性(链路)请求报文与期望的报文进行比对，如果比对匹配，则向被测设备返回远程配置属性(链路)响应；

d) 被测设备接收到响应后，向测试系统(网关设备)发送远程配置属性(链路)证实；

e) 测试系统(网关设备)将接收的远程配置属性(链路)证实报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 53 所示，具体测试说明如表 63 所示。

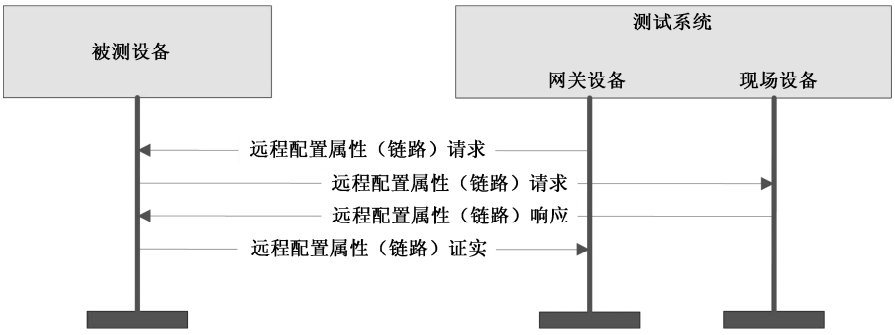

图 53 链路资源分配测试时序图

表 63　链路资源分配测试说明

用例名称	链路资源分配测试[AD-RUN005]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已通过被测设备加入网络 测试系统(网关设备)已通过被测设备完成对测试系统(现场设备)的超帧资源分配 KS=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置(超帧、链路、密钥) 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的超帧资源分配

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredAttributeSettingRequest(Link);
	SendPacket2= AttributeSettingResponse;
	RefPacket1 = AttributeSettingRequest(Link);
	RefPacket2 = WiredAttributeSettingConfirm;
	wiredSend(SendPacket1);
	WHILE(Receive(RcvPacket,SecLevel,KS).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Request (Link) Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket2,SecLevel,KS);
	Printscreen("Attribute Setting Request (Link) Payload correct!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Attribute Setting Request (Link) received!");
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	WHILE(wiredReceive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all) ! = SUCCESS)

表 63（续）

测试用例伪代码描述	{
	Printscreen(“WiredAttribute Setting Confirm Payload error!”);
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen(“Superframe Resource Distributing Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No WiredAttribute Setting Confirm received!”);
	TestCaseResult = FAILED;
	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0,1	测试系统应发数据包: 数据包 1: 协议层:Wired DLL 帧名称: Wired Attribute Setting Request 帧: 0x0c | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明: 服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3) 数据包 2: 协议层:DLL 帧名称:Attribute Setting Response 帧: 0x90 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x00 |0x81 |0x?? |0x?? 0x?? |0x?? 0x?? |0x?? | 0x?? 0x?? 帧域说明: 帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|FCS(2)
	测试系统应收数据包: 数据包 1: 协议层: DLL 帧名称:Attribute Setting Request 帧:0x8f | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x00 |0x81 |0x?? |0x?? 0x?? |0x?? 0x?? |0x?? ... 0x?? | 0x?? 0x??

表 63（续）

参考数据包 SecLevel＝0，1	帧域说明：帧控制(1)	网络 ID(1)	目的地址(1)	序列号(2)	帧长度(2)	远程属性操作(1)	属性标识符(1)	属性成员标识符(1)	多个属性值的第一个存储索引(2)	属性数目(2)	属性值(n)	FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d	0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)	AdID(1)	对端地址(1)	序列号(2)	长度(2)	载荷(3)
参考数据包 SecLevel＝2～8	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Attribute Setting Request 帧：0x0c	0x?? |0x03	0x?? 0x??	0x?? 0x??	0x??	0x??	0x?? … 0x??	0x03	0x00	0x81	0x??	0x?? 0x?? |0x?? 0x??	0x??… 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|回调值(1)| AD 数量(1)|AD 列表(n)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n) 数据包 2： 协议层：DLL 帧名称：Attribute Setting Response 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x81 |0x?? |0x?? 0x?? |0x?? 0x?? |0x00| 0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)| MIC(4)| FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8f	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x81 |0x?? |0x?? 0x?? |0x?? 0x?? |0x?? … 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)| MIC(4)| FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d	0x??	0x03	0x?? 0x?? |0x?? 0x??	0x?? 0x??	0x03 |0x00 |0x81	0x??	0x?? 0x?? |0x?? 0x?? |0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|源地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)

7.2.6 网关设备指示接入设备发送 NACK 测试[AD-RUN006]

该测试用例测试接入设备能否正确响应网关设备指示接入设备发送 NACK。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(网关设备)向被测设备发送网关设备指示接入设备发送 NACK；

b) 被测设备接收到报文后，向测试系统(现场设备)发送 NACK；

c) 测试系统(现场设备)将接收的 NACK 报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 54 所示，具体测试说明如表 64 所示。

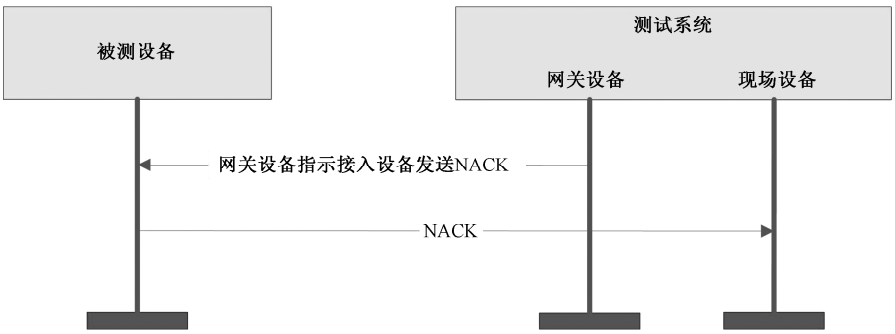

图 54 网关设备指示接入设备发送 NACK 测试时序图

表 64 网关设备指示接入设备发送 NACK 测试说明

用例名称	网关设备指示接入设备发送 NACK 测试[AD-RUN006]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已通过被测设备加入网络 测试系统(网关设备)已通过被测设备完成对测试系统(现场设备)的资源配置 KEDB=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDB 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredIndicatingNACK;
	RefPacket =NACK;
	wiredSend(SendPacket);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel,KEDB) ! = SUCCESS)
	{

表 64（续）

测试用例伪代码描述	Printscreen(“NACK Payload error!”);
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen(“Indicating NACK Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No NACK received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0，1，5	测试系统应发数据包： 协议层：Wired DLL 帧名称:Wired Indicating NACK 帧：0x03 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 协议层：DLL 帧名称:NACK 帧:0x84 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x02 |0x01 |0x03 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|重传设备数量(1)|设备短地址列表(1)|FCS(2)
参考数据包 SecLevel=2，3，4，6，7，8	测试系统应发数据包： 协议层：Wired DLL 帧名称:Wired Indicating NACK 帧：0x03 | 0x?? | 0xff| 0x?? 0x?? |0x?? 0x?? |0x01 | 0x03 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|重传设备数量(1)|设备短地址列表(1)
	测试系统应收数据包： 协议层：DLL 帧名称:NACK 帧:0x84 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x02 |0x01 | 0x03| 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|重传设备数量(1)|设备短地址列表(1)| MIC(4)|FCS(2)

7.2.7 网关设备指示接入设备发送 GACK 测试[AD-RUN007]

该测试用例测试接入设备能否正确响应网关设备指示接入设备发送 GACK。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(网关设备)向被测设备发送网关设备指示接入设备发送 GACK；

b) 被测设备接收到报文后，向测试系统(现场设备)发送 GACK；

c) 测试系统(现场设备)将接收的 GACK 报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 55 所示，具体测试说明如表 65 所示。

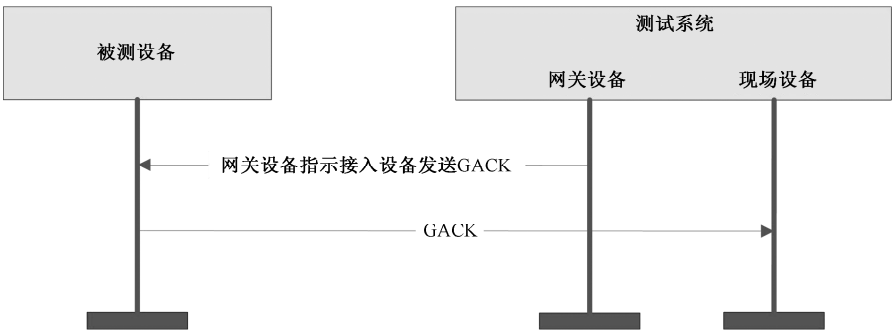

图 55 网关设备指示接入设备发送 GACK 测试时序图

表 65 网关设备指示接入设备发送 GACK 测试说明

用例名称	网关设备指示接入设备发送 GACK 测试[AD-RUN007]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已通过被测设备加入网络 测试系统(网关设备)已通过被测设备完成对测试系统(现场设备)的资源配置 KEDB=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDB 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredIndicatingGACK;
	RefPacket =GACK;
	wiredSend(SendPacket);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel, KEDB) ! = SUCCESS)
	{

表 65（续）

测试用例伪代码描述	Printscreen(“GACK Payload error!”);
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen(“Indicating GACKTest Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No GACK received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0,1	测试系统应发数据包： 协议层：Wired DLL 帧名称:Wired Indicating GACK 帧：0x02| 0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 协议层：DLL 帧名称:GACK 帧:0x83	0xaa	0x03	0x?? 0x??	0x00 0x04 |0x01	0x03| 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|设备数(1)|设备短地址(1)|帧序列号(2)|FCS(2)
参考数据包 SecLevel=2～8	测试系统应发数据包： 协议层：Wired DLL 帧名称:Wired Indicating GACK 帧:0x02	0x??	0x00 0x04	0x01 0x03 0x?? 0x?? |0x01 |0x03	0x?? 0x?? 服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|设备数(1)|设备短地址(1)|帧序列号(2)
	测试系统应收数据包： 协议层：DLL 帧名称:GACK 帧:0x83	0xaa	0x03	0x?? 0x??	0x00 0x04 |0x01	0x03| 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|设备数(1)|设备短地址(1)|帧序列号(2)|MIC(4)| FCS(2)

7.2.8 现场设备到网关设备数据传输测试[AD-RUN008]

该测试用例测试接入设备能否正确进行数据传输。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(现场设备)向被测设备发送数据帧；

b) 被测设备接收到数据帧后，向测试系统(网关设备)发送数据发送指示；

c) 测试系统(网关设备)将接收的数据发送指示报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 56 所示，具体测试说明如表 66 所示。

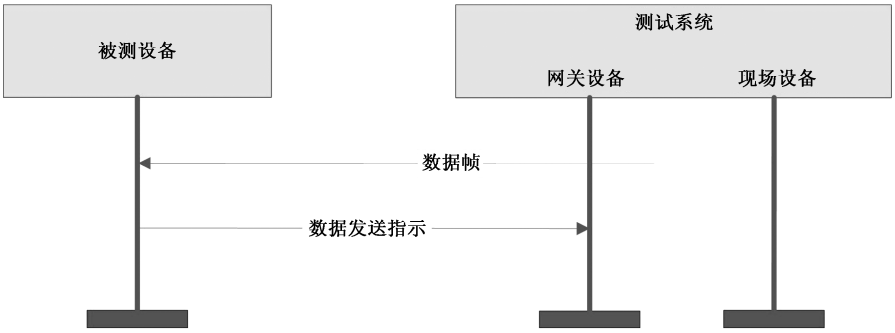

图 56 现场设备到网关设备数据传输测试时序图

表 66 现场设备到网关设备数据传输测试说明

用例名称	现场设备到网关设备数据传输测试[AD-RUN008]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已通过被测设备加入网络 测试系统(网关设备)已通过被测设备完成对测试系统(现场设备)的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已通过被测设备加入网络 测试系统(网关设备)已通过被测设备完成对测试系统(现场设备)的资源配置 KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = Data;
	RefPacket =WiredDataIndication;
	Send(SendPacket,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("Wired Data Indication Payload error!");

表 66（续）

测试用例伪代码描述	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Data Transmission Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Wire Data Indication received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0, 1,5	测试系统应发数据包： 协议层:DLL 帧名称：Data 帧：0x81 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|载荷(n)|FCS(2)
	测试系统应收数据包： 协议层：Wired DLL 帧名称:Wired Data Indication 帧：0x05 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
参考数据包 SecLevel=2,3, 4,6,7,8	测试系统应发数据包： 协议层:DLL 帧名称：Data 帧：0x81 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? ... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|载荷(n)| MIC(4)|FCS(2)
	测试系统应收数据包： 协议层：Wired DLL 帧名称:WiredData Indication 帧：0x05 | 0x?? | 0x03| 0x?? 0x?? |0x?? 0x?? | 0x03 | 0x00 | 0x?? 0x?? | 0x?? ... 0x?? 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|源地址(1)|数据包类型(1)|数据长度(2)|数据(n)

7.2.9 网关设备到现场设备数据传输测试[AD-RUN009]

该测试用例测试接入设备能否正确进行数据传输。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(网关设备)向被测设备发送数据发送请求；

b) 被测设备接收到请求后，向测试系统(现场设备)发送数据帧；

c) 测试系统(现场设备)将接收的数据帧报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 57 所示，具体测试说明如表 67 所示。

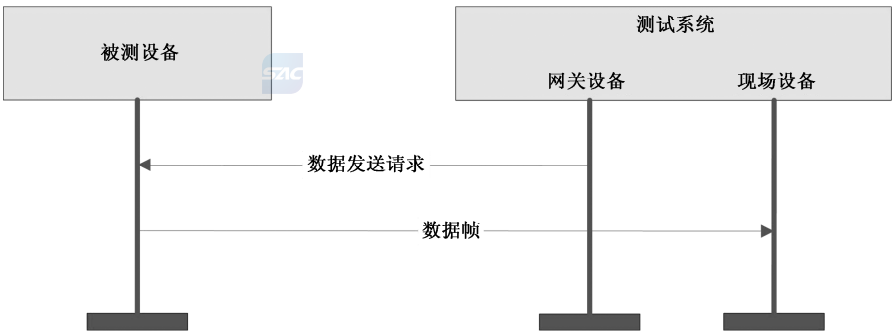

图 57 网关设备到现场设备数据传输测试时序图

表 67 网关设备到现场设备数据传输测试说明

用例名称	网关设备到现场设备数据传输测试[AD-RUN009]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 测试系统(现场设备)已通过被测设备加入网络 测试系统(网关设备)已通过被测设备完成对测试系统(现场设备)的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 测试系统(现场设备)已通过被测设备加入网络 测试系统(网关设备)已通过被测设备完成对测试系统(现场设备)的资源配置 KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredDataRequest;
	RefPacket =Data;
	wiredSend(SendPacket);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all,SecLevel, KEDU) ! = SUCCESS)
	{
	Printscreen("Data Payload error!");
	}

表 67（续）

测试用例伪代码描述	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Data Transmission Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Data received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0，1，5	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired Data Request 帧：0x04 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 协议层：DLL 帧名称：Data 帧：0x81 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? ... 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(n)|FCS(2)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired Data Request 帧：0x04 | 0x?? | 0x03| 0x?? 0x?? |0x?? 0x?? | 0x03 | 0x?? 0x?? | 0x00 | 0x?? | 0x?? 0x?? | 0x?? ... 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|目的地址(1)|VCR_ID(2)|数据包类型(1)|优先级(1)|数据长度(2)|数据(n)
	测试系统应收数据包： 协议层：DLL 帧名称：Data 帧：0x81 | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? ... 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|载荷(n)|MIC(4)| FCS(2)

7.2.10 设备状态报告测试[AD-RUN010]

该测试用例测试接入设备能否正确进行设备状态报告。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(现场设备)向被测设备发送设备状态报告；

b) 被测设备接收到设备状态报告后，向测试系统(网关设备)发送设备状态报告指示；

c) 测试系统(网关设备)将接收的设备状态报告指示报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 58 所示，具体测试说明如表 68 所示。

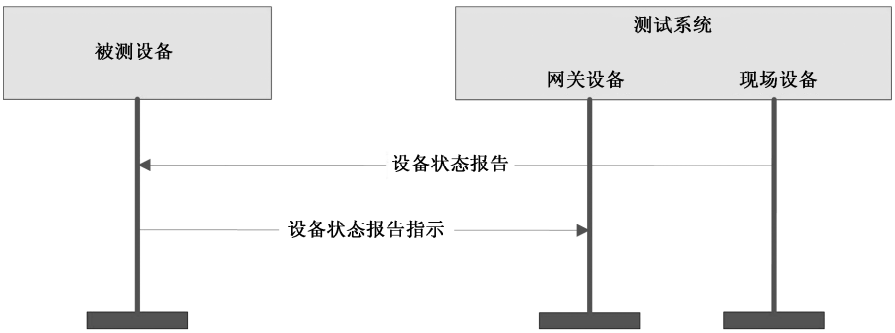

图 58 设备状态报告测试时序图

表 68 设备状态报告测试说明

用例名称	设备状态报告测试[AD-RUN010]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已通过被测设备加入网络 测试系统(网关设备)已通过被测设备完成对测试系统(现场设备)的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = DeviceConditionReport;
	RefPacket =WiredDeviceStatusIndication;
	Send(SendPacket,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{

表 68（续）

测试用例伪代码描述	Printscreen(“WiredDevice Status Indication Payload error!”);
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen(“Device Condition Report Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No WiredDevice Status Indication received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0，1，5	测试系统应发数据包： 协议层：DLL 帧名称：Device Condition Report 帧：0x89 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x01 |0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|设备状态信(1)|FCS(2)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Device Status Indication 帧：0x08 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
参考数据包 SecLevel = 2，3，4，6，7，8	测试系统应发数据包： 协议层：DLL 帧名称：Device Condition Report 帧：0x89 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x01 |0x?? | 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|设备状态信息(1)|MIC(4)|FCS(2)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Device Status Indication 帧：0x08 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x03 | 0x?? | 0x?? 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|源地址(1)|AdID(1)|设备状态信息(1)

7.2.11 信道状况报告测试[AD-RUN011]

该测试用例测试接入设备能否正确进行信道状况报告。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(现场设备)向被测设备发送信道状况报告；

b) 被测设备接收到信道状况报告后，向测试系统(网关设备)发送信道状况报告指示；

c) 测试系统(网关设备)将接收的信道状况报告指示报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 59 所示，具体测试说明如表 69 所示。

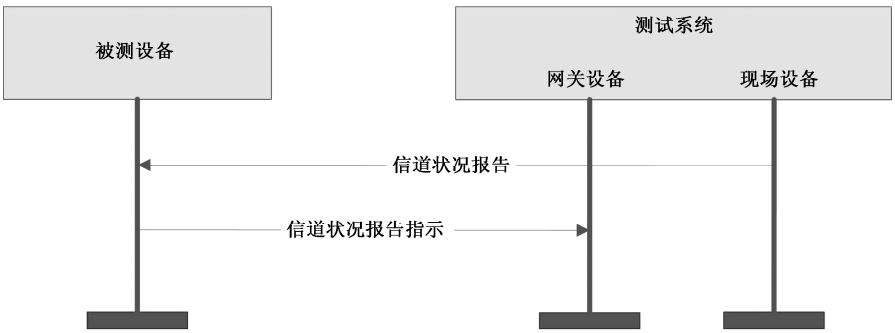

图 59 信道状况报告测试时序图

表 69 信道状况报告测试说明

用例名称	信道状况报告测试[AD-RUN011]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket = ChannelConditionReport;
	RefPacket = WiredChannelConditionIndication;
	Send(SendPacket,SecLevel,KEDU);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{

表 69（续）

测试用例伪代码描述	Printscreen(“WiredChannel Condition Indication Payload error!”);
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen(“Channel Condition Report Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No WiredChannel Condition Indication received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0，1，5	测试系统应发数据包： 协议层:DLL 帧名称：Channel Condition Report 帧：0x8a	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x?? … 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|信道状况信(n)|FCS(2)
	测试系统应收数据包： 协议层：Wired DLL 帧名称:WiredChannel Condition Indication 帧：0x09	0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 协议层:DLL 帧名称：Channel Condition Report 帧：0x8a	0xaa	0x03	0x?? 0x??	0x?? 0x??	0x?? |0x?? … 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|信道列表数量(1)|信道状况信息列表(n)|MIC(4)| FCS(2)
	测试系统应收数据包： 协议层：Wired DLL 帧名称:Wired Channel Condition Indication 帧：0x09	0x??	0x03	0x?? 0x?? |0x?? 0x??	0x03	0x??	0x?? |0x?? … 0x?? 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|源地址(1)|AdID(1)|信道列表数量(1)|信道状况信息列表(n)

7.2.12 远程读属性测试[AD-RUN012]

该测试用例测试接入设备能否正确进行远程读属性。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(网关设备)向被测设备发送远程读属性请求，具体情况如下：

1) 读非结构化属性；
2) 读结构化属性一个成员的一个记录；
3) 读结构化属性一个成员的多个记录；
4) 读结构化属性一个成员的全部记录；
5) 读结构化属性全部成员的一个记录；
6) 读结构化属性全部成员的多个记录；
7) 读结构化属性全部成员的全部记录。

参考数据包中具体服务参数内容如表 70 所示。

表 70 远程读属性测试参考数据包服务参数

测试内容	应发服务参数	应收服务参数
读非结构化属性	0x?? | 0x03 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x??	0x?? |0x03 | 0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性一个成员的一个记录	0x?? | 0x03 | 0x?? | 0x?? | 0x?? 0x?? | 0x00 0x01	0x?? |0x03 | 0x00 | 0x?? | 0x?? (不等于 0xff)| 0x?? 0x?? | 0x00 0x01 | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性一个成员的多个记录	0x?? | 0x03 | 0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x??	0x?? |0x03 | 0x00 | 0x?? | 0x?? (不等于 0xff) | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性一个成员的全部记录	0x?? | 0x03 | 0x?? | 0x?? | 0x?? 0x?? | 0x00 0x00	0x?? |0x03 | 0x00 | 0x?? | 0x?? (不等于 0xff) | 0x?? 0x?? | 0x00 0x00 | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性全部成员的一个记录	0x?? | 0x03 | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x01	0x?? |0x03 | 0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x01 | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)

表 70（续）

测试内容	应发服务参数	应收服务参数
读结构化属性全部成员的多个记录	0x?? | 0x03 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x??	0x?? |0x03 | 0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性全部成员的全部记录	0x?? | 0x03 | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x00	0x?? |0x03 | 0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x000x00 | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)

b) 被测设备接收到请求后，向测试系统（现场设备）发送远程读属性请求；

c) 测试系统（现场设备）将接收的远程读属性请求报文与期望的报文进行比对，如果比对匹配，则向被测设备返回远程读属性响应；

d) 被测设备接收到响应后，向测试系统（网关设备）发送远程读属性证实；

e) 测试系统（网关设备）将接收的远程读属性证实报文与期望的报文进行比对，如果比对匹配，则测试通过。

该测试用例用于读被测设备信息库中的所有属性，测试体应循环执行，具体时序如图 60 所示，具体测试说明如表 71 所示。

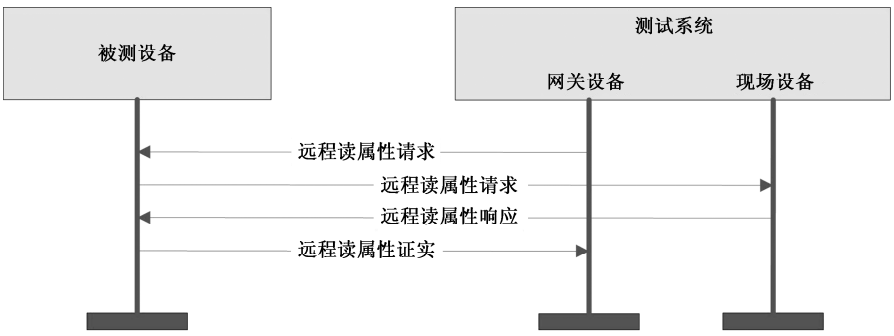

图 60 远程读属性测试时序图

表 71 远程读属性测试说明

用例名称	远程读属性测试［AD-RUN012］
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统（网关设备）网络 测试系统（网关设备）已完成对被测设备的资源配置 测试系统（现场设备）已加入被测设备网络 被测设备已完成对测试系统（现场设备）的资源配置

表 71（续）

依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredAttributeGettingRequest;
	SendPacket2= AttributeGettingResponse;
	RefPacket1 = AttributeGettingRequest;
	RefPacket2 = WiredAttributeGettingConfirm;
	wiredSend(SendPacket1);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Attribute Getting Request Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket2,SecLevel,KEDU);
	Printscreen("Attribute Getting Request Payload correct!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Attribute Getting Request received!");
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket2.all) ! = SUCCESS)
	{
	Printscreen("WiredAttribute Getting Confirm Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{

表 71（续）

测试用例伪代码描述	Printscreen(“Attribute Getting Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No WiredAttribute Getting Confirm received!”);
	TestCaseResult = FAILED;
	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Attribute Getting Request 帧：0x0a | 0x?? | 0x00 0x08 | 0x?? … 0x?? 帧域说明：服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(8) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8e | 0xaa | 0x03 | 0x?? 0x?? | 0x?? 0x?? |0x?? … 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|载荷(n)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Getting Request 帧：0x8d | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x06 |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Attribute Getting Confirm 帧：0x0b | 0x?? | 0x?? 0x?? | 0x?? … 0x?? 帧域说明：服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(n)

表 71（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Attribute Getting Request 帧：0x0a	0x??	0x03	0x?? 0x?? |0x?? 0x?? |0x?? ... 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|服务参数(n) 数据包 2： 协议层：DLL 帧名称：Attribute Getting Response 帧：0x8e	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x00 |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? |0x?? ... 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(4)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Getting Request 帧：0x8d	0xaa	0x03	0x?? 0x??	0x00 0x06 |0x?? |0x?? |0x?? 0x?? |0x?? 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|MIC(4)| FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Attribute Getting Confirm 帧：0x0b	0x??	0x03	0x?? 0x?? |0x?? 0x??	0x?? ... 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|服务参数(n)

7.2.13 远程配置属性测试[AD-RUN013]

该测试用例测试接入设备能否正确进行远程配置属性。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(网关设备)向被测设备发送远程配置属性请求，具体情况如下：
   1) 配置非结构化属性；
   2) 配置结构化属性一个成员的一个记录；
   3) 配置结构化属性一个成员的多个记录；
   4) 配置结构化属性一个成员的全部记录；
   5) 配置结构化属性全部成员的一个记录；
   6) 配置结构化属性全部成员的多个记录；
   7) 配置结构化属性全部成员的全部记录。

参考数据包中具体服务参数内容如表 72 所示。

表 72　远程配置属性测试参考数据包服务参数

测试内容	应发服务参数	应收服务参数
配置非结构化属性	0x??|0x03|0x??|0x??|0xff|0x?? 0x??|0x?? 0x??|0x?? ... 0x??	0x??|0x03|0x??|0x??|0xff|0x?? 0x??|0x?? 0x??|0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性一个成员的一个记录	0x??|0x03|0x??|0x??|0x??|0x?? 0x??|0x00 0x01|0x?? ... 0x??	0x??|0x03|0x??|0x??|0x??(不等于 0xff)|0x?? 0x??|0x00 0x01|0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性一个成员的多个记录	0x??|0x03|0x??|0x??|0x??|0x?? 0x??|0x?? 0x??|0x?? ... 0x??	0x??|0x03|0x??|0x??|0x??(不等于 0xff)|0x?? 0x??|0x?? 0x??|0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性一个成员的全部记录	0x??|0x03|0x??|0x??|0x??|0x?? 0x??|0x00 0x00|0x?? ... 0x??	0x??|0x03|0x??|0x??|0x??(不等于 0xff)|0x?? 0x??|0x00 0x00|0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性全部成员的一个记录	0x??|0x03|0x??|0x??|0xff|0x?? 0x??|0x00 0x01|0x?? ... 0x??	0x??|0x03|0x??|0x??|0xff|0x?? 0x??|0x00 0x01|0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性全部成员的全部记录	0x??|0x03|0x??|0x??|0xff|0x?? 0x??|0x00 0x00|0x?? ... 0x??	0x??|0x03|0x??|0x??|0xff|0x?? 0x??|0x00 0x00|0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(1)|执行结果(1)

b)　被测设备接收到请求后，向测试系统(现场设备)发送远程配置属性请求；

c)　测试系统(现场设备)将接收的远程配置属性请求报文与期望的报文进行比对，如果比对匹配，则向被测设备返回远程配置属性响应；

d)　被测设备接收到响应后，向测试系统(网关设备)发送远程配置属性证实；

e)　测试系统(网关设备)将接收的远程配置属性证实报文与期望的报文进行比对，如果比对匹配，则测试通过。

该测试用例用于配置被测设备信息库中的所有属性，测试体应循环执行，具体时序如图 61 所示，具体测试说明如表 73 所示。

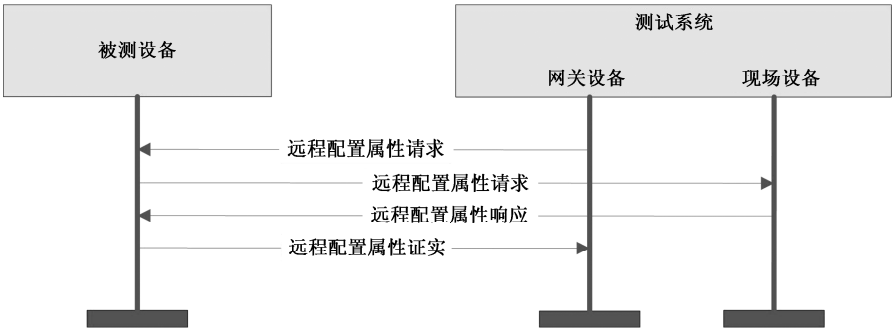

图 61　远程配置属性测试时序图

表 73　远程配置属性测试说明

用例名称	远程配置属性测试 [AD-RUN013]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDU 已启用

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredAttributeSettingRequest;
	SendPacket2= AttributeSettingResponse;
	RefPacket1 = AttributeSettingRequest;
	RefPacket2 = WiredAttributeSettingConfirm;
	wiredSend(SendPacket1);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Attribute Setting Request Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket2,SecLevel,KEDU);
	Printscreen("Attribute Setting Request Payload correct!");
	TestCaseResult =SUCCESS;
	}

表 73（续）

测试用例伪代码描述	}
	ELSE
	{
	Printscreen(“No Attribute Setting Request received!”);
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket2.all) ! = SUCCESS)
	{
	Printscreen(“WiredAttribute Setting Confirm Payload error!”);
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen(“Attribute Setting Test Success!”);
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen(“No WiredAttribute Setting Confirm received!”);
	TestCaseResult = FAILED;
	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0, 1,5	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Attribute Setting Request 帧：0x0c | 0x?? | 0x?? 0x?? | 0x?? … 0x?? 帧域说明：服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(n) 数据包 2：

表 73（续）

参考数据包 SecLevel = 0，1，5	协议层：DLL 帧名称：Attribute Setting Response 帧：0x8e	0xaa	0x03	0x?? 0x??	0x00 0x08	0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)	网络 ID(1)	源地址(1)	序列号(2)	帧长度(2)	载(8)	FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8d	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x?? |0x?? |0x?? |0x?? 0x?? |0x?? 0x??	0x?? ... 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d	0x??	0x00 0x02	0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(2)
参考数据包 SecLevel = 2，3，4，6，7，8	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Attribute Setting Request 帧：0x0c	0x?? |0x03	0x?? 0x??	0x?? 0x??	0x?? ... 0x?? | 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|服务参数(n) 数据包 2： 协议层：DLL 帧名称：Attribute Setting Response 帧：0x90	0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x?? |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? |0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)|MIC(4)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Attribute Setting Request 帧：0x8d| 0xaa	0x03	0x?? 0x??	0x?? 0x?? |0x?? |0x?? |0x?? |0x?? 0x?? |0x?? 0x?? |0x?? ... 0x??	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)|MIC(4)| FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d	0x?? |0x03	0x?? 0x??	0x?? 0x??	0x?? ... 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|服务参数(n)

7.3 离开过程测试集

7.3.1 现场设备被动离开网络[AD-QUIT001]

该测试用例测试接入设备能否正确处理现场设备离开网络。

测试过程为：

a) 被测设备加入测试系统(网关设备)网络后，测试系统(网关设备)向被测设备发送设备离开请求；

b) 被测设备接收到请求后，向测试系统(现场设备)发送离开请求；

c) 测试系统(现场设备)将接收的离开请求报文与期望的报文进行比对，如果比对匹配，则向被测设备返回离开响应；

d) 被测设备接收到响应后，向测试系统(网关设备)发送设备离开响应；

e) 测试系统(网关设备)将接收的设备离开响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 62 所示，具体测试说明如表 74 所示。

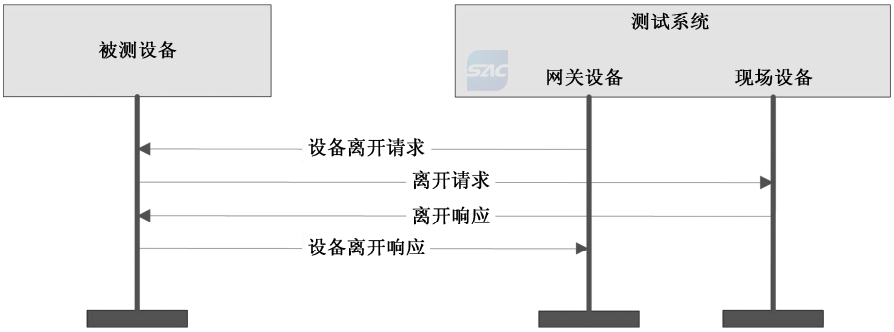

图 62 现场设备被动离开网络时序图

表 74 现场设备被动离开网络说明

用例名称	现场设备被动离开网络测试[AD-QUIT001]
被测设备	接入设备
依赖测试条件 SecLevel=0,1	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDU=NULL
依赖测试条件 SecLevel=2～8	被测设备已加入测试系统(网关设备)网络 测试系统(网关设备)已完成对被测设备的资源配置 测试系统(现场设备)已加入被测设备网络 被测设备已完成对测试系统(现场设备)的资源配置 KEDU 已启用
测试用例伪代码描述	TEST BODY： SendPacket1 = WiredLeaveRequest； SendPacket 2= LeaveResponse； RefPacket1 = LeaveRequest；

表 74（续）

测试用例伪代码描述	RefPacket2 = WiredLeaveResponse;
	wiredSend(SendPacket1);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Leave Request Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket2,SecLevel,KEDU);
	Printscreen("Leave Request Payload correct!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Leave Request received!");
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket2.all) ! = SUCCESS)
	{
	Printscreen("WiredLeave Response Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Leaving Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredLeave Response received!");
	TestCaseResult = FAILED;
	}
	}

表 74（续）

测试用例伪代码描述	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Leave Request 帧：0x0e | 0x?? | 0x00 0x01 | 0x?? 帧域说明：服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(1) 数据包 2： 协议层：DLL 帧名称：Leave Response 帧：0x88 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x00 | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Leave Request 帧：0x87 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x00 | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Leave Response 帧：0x0f | 0x?? |0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|AdID (1)|离开状态(1)
参考数据包 SecLevel = 2,3,4,6,7,8	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Leave Request 帧：0x0e | 0x?? |0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? ... 0x?? | 0x03 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|AdNum(1)|AdIDList(n)|设备短地址(1) 数据包 2： 协议层：DLL 帧名称：Leave Response 帧：0x88 | 0xaa | 0x03 | 0x?? 0x?? | 0x00 0x00| 0x?? 0x?? 0x?? 0x?? | 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源地址(1)|序列号(2)|帧长度(2)|MIC(4)| FCS(2)

表 74（续）

参考数据包 SecLevel＝2，3，4，6，7，8	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：Leave Request 帧：0x87	0xaa	0x03	0x?? 0x??	0x00 0x00	0x?? 0x?? 0x?? 0x??	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|目的地址(1)|序列号(2)|帧长度(2)|MIC(4)| FCS(2) 数据包 2： 协议层：Wired DLL 帧名称：Wired Leave Response 帧：0x0f	0x?? |0x03	0x?? 0x??	0x?? 0x??	0x??	0x00 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长(2)|AdID (1)|离开状态(1)

7.3.2 接入设备被动离开网络[AD-QUIT002]

该测试用例测试 AD 设备能否正确响应离开请求。

测试过程为：

a) 被测设备加入测试系统（网关设备）网络后，测试系统（网关设备）向被测设备发送设备离开请求；

b) 被测设备接收到请求后，向测试系统（网关设备）发送离开响应；

c) 测试系统（网关设备）将接收的离开响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 63 所示，具体测试说明如表 75 所示。

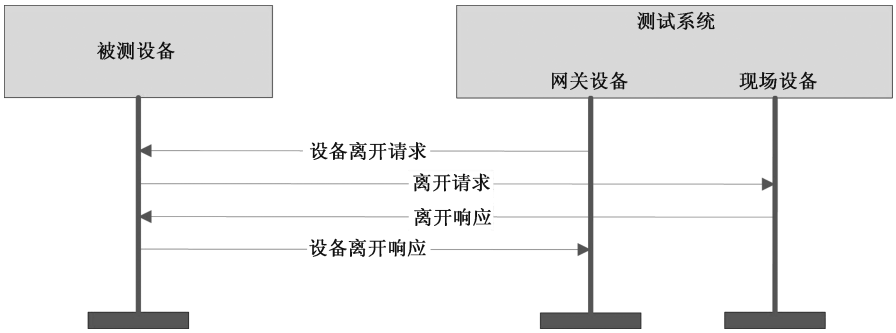

图 63 接入设备被动离开网络时序图

表 75 接入设备被动离开网络说明

用例名称	接入设备被动离开网络测试[AD-QUIT002]
被测设备	接入设备
依赖测试条件	被测设备已加入测试系统（网关设备）网络 测试系统（网关设备）已完成对被测设备的资源配置

表 75（续）

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredLeaveRequest;
	RefPacket2 = WiredLeaveResponse;
	wiredSend(SendPacket1);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all,SecLevel,KEDU) ! = SUCCESS)
	{
	Printscreen("Leave Request Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket2,SecLevel,KEDU);
	Printscreen("Leave Request Payload correct!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Leave Request received!");
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket2.all) ! = SUCCESS)
	{
	Printscreen("WiredLeave Response Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Leaving Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredLeave Response received!");
	TestCaseResult = FAILED;

表 75（续）

测试用例伪代码描述	}
	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel = 0,1,5	测试系统应发数据包: 数据包 1: 协议层:Wired DLL 帧名称: Wired Leave Request 帧: 0x0e | 0x?? | 0x00 0x01 | 0x?? 帧域说明: 服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(1)
	测试系统应收数据包: 数据包 1: 协议层:Wired DLL 帧名称: Wired Leave Response 帧: 0x0f | 0x?? |0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x00 帧域说明: 服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|AdID (1)|离开状态(1)

8 网关设备测试集

8.1 运行过程测试集

8.1.1 接入设备加入网络(正向测试)[GW-RUN001]

该测试用例测试网关设备能否正确处理接入设备加入 WIA-FA 网络。

测试过程为:

a) 测试系统向被测设备发送接入设备加入请求;

b) 被测设备接收到请求后,向测试系统返回接入设备加入响应;

c) 测试系统将接收的接入设备加入响应报文与期望的报文进行比对,如果比对匹配,则测试通过。

具体时序如图 64 所示,具体测试说明如表 76 所示。

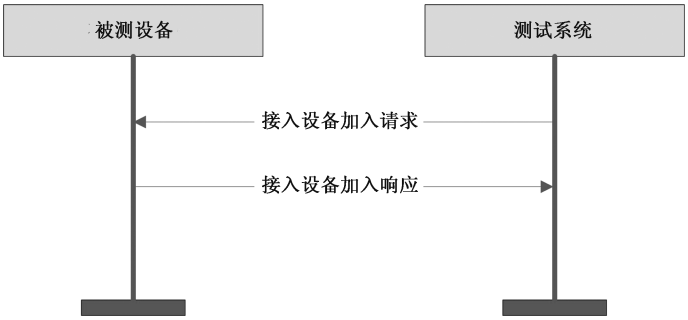

图 64　接入设备加入网络(正向测试)时序图

表 76　接入设备加入网络(正向测试)说明

用例名称	接入设备加入网络(正向测试)[GW-RUN001]
被测设备	网关设备
依赖测试条件	测试系统未加入被测设备网络

测试用例伪代码描述	TEST BODY:
	SendPacket =WiredADJoinRequest;
	RefPacket=WiredADJoinResponse;
	Send(SendPacket);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("Wire AD JoinResponse Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("JoiningTest Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Wired AD JoinResponse received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 76（续）

参考数据包	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired AD Join Request 帧：0x00 | 0x?? … 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AD 长地址(8)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired AD Join Response 帧：0x01 | 0x?? … 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AD 长地址(8)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

8.1.2 接入设备加入网络(反向测试)[GW-RUN002]

该测试用例测试网关设备能否处理接入设备加入 WIA-FA 网络。

测试过程为：

a) 测试系统向被测设备发送接入设备加入请求；

b) 被测设备将接收的接入设备加入请求报文与期望的报文进行比对，如果比对匹配，则向被测设备返回错误的接入设备加入响应，具体情况如下：

1) 网络 ID 不匹配；

2) 认证失败(SecLevel 不为 0 时)。

参考数据包中具体载荷内容如表 77 所示。

表 77 接入设备加入网络(反向测试)参考数据包载荷

测试内容	测试系统应发加入响应数据载荷
网络 ID 不匹配	0x01 | 0x?? |0x02
	加入状态(1)|分配的 AdID(1)|分配的 AD 地址
认证失败	0x02 | 0x?? |0x02
	加入状态(1)|分配的 AdID(1)|分配的 AD 地址

c) 被测设备向测试系统发送远程配置属性(超帧)请求，测试系统接收到请求后，不向被测设备返回远程配置属性(超帧)证实，则测试通过。

该测试用例用于网关设备处理接入设备入网的反向测试，测试体应循环执行，具体时序如图 65 所示，具体测试说明如表 78 所示。

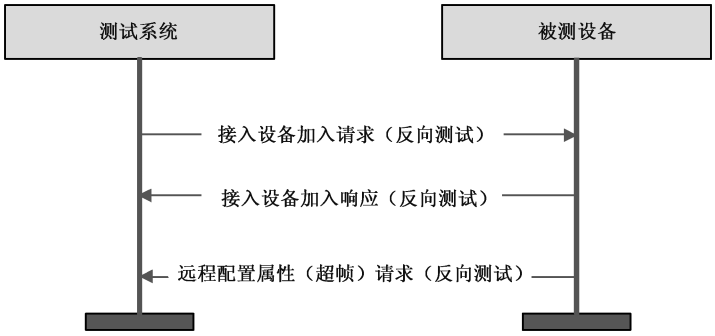

图 65 接入设备加入网络(反向测试)时序图

表 78　接入设备加入网络(反向测试)说明

用例名称	接入设备加入网络(反向测试)[GW-RUN002]
被测设备	网关设备
依赖测试条件	测试系统未加入测试系统网络

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredADJoinRequest;
	SendPacket2 = WiredAttributeSettingConfirm;
	RefPacket1 =WiredADJoinResponse;
	RefPacket2 =WiredAttributeSettingRequest(Superframe);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket .type == RefPacket1.type)
	{
	IF (Verify(RcvPacket.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Wired AD JoinRequest Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket1);
	Printscreen("Wired AD JoinRequest Payload correct!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Wired AD JoinRequest received!");
	TestCaseResult = FAILED;
	}
	IF(TestCaseResult == SUCCESS)
	{
	Send(SendPacket2);
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket .type == RefPacket2.type)
	{
	IF (Verify(RcvPacket.all, RefPacket2.all) == SUCCESS)
	{
	Printscreen("JoiningNetwork Test Failed!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("JoiningNetwork Test Success!");
	TestCaseResult = SUCCESS;
	}

表 78（续）

测试用例伪代码描述	}
	ELSE
	{
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 数据包 1: 协议层:Wired DLL 帧名称:Wired AD Join Request 帧:0x00 | 0x?? ... 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明:服务标识符(1)|AD 长地址(8)|对端地址(1)|序列号(2)|长度(2)|载荷(3) 数据包 2: 协议层:Wired DLL 帧名称:Wired Attribute Setting Confirm 帧:0x0d | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明:服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包: 数据包 1: 协议层:Wired DLL 帧名称:Wired AD Join Response 帧:0x01 |0x?? ... 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明:服务标识符(1)|AD 长地址(8)|对端地址(1)|序列号(2)|长度(2)|载荷(3) 数据包 2: 协议层:Wired DLL 帧名称:Wired Attribute Setting Request 帧:0x0c | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明:服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

8.1.3 接入设备超帧资源分配测试[GW-RUN003]

该测试用例测试网关设备能否正确对接入设备进行超帧资源分配。

测试过程为:

a) 测试系统加入被测设备网络后,被测设备向测试系统发送远程配置属性(超帧)请求;

b) 测试系统将接收的远程配置属性(超帧)请求报文与期望的报文进行比对,如果比对匹配,则测试通过,并向被测设备返回远程配置属性(超帧)证实。

具体时序如图 66 所示,具体测试说明如表 79 所示。

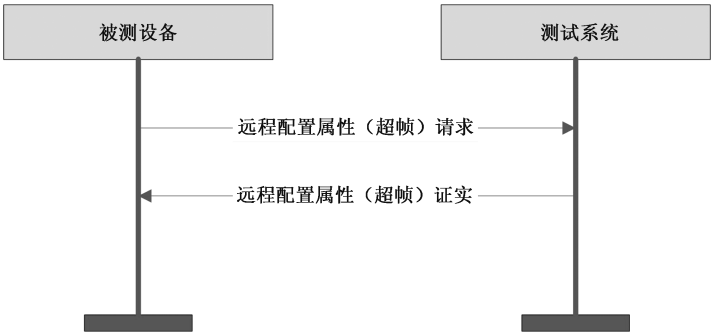

图 66　接入设备超帧资源分配测试时序图

表 79　接入设备超帧资源分配测试说明

用例名称	接入设备超帧资源分配测试[GW-RUN003]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredAttributeSettingConfirm;
	RefPacket =WiredAttributeSettingRequest(Superframe);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("Wire Attribute Setting Request (Superframe) Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket);
	Printscreen("Superframe Resource Distributing Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredAttribute Setting Request (Superframe) received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 79（续）

参考数据包	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d	0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Attribute Setting Request 帧：0x0c	0x??	0x02	0x?? 0x??	0x?? 0x??	0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

8.1.4 接入设备链路资源分配测试[GW-RUN004]

该测试用例测试网关设备能否正确对接入设备进行链路资源分配。

测试过程为：

a) 测试系统加入被测设备网络后，被测设备向测试系统发送远程配置属性(链路)请求；

b) 测试系统将接收的远程配置属性(链路)请求报文与期望的报文进行比对，如果比对匹配，则测试通过，并向被测设备返回远程配置属性(链路)证实。

具体时序如图 67 所示，具体测试说明如表 80 所示。

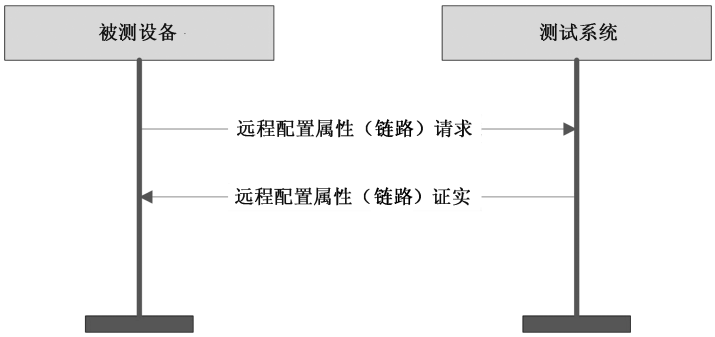

图 67 接入设备链路资源分配测试时序图

表 80 接入设备链路资源分配测试说明

用例名称	接入设备链路资源分配测试[GW-RUN004]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络

测试用例伪代码描述	TEST BODY：
	SendPacket = WiredAttributeSettingConfirm；
	RefPacket =WiredAttributeSettingRequest(Link)；
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime；
	IF(RcvPacket.type == RefPacket.type)

表 80（续）

测试用例伪代码描述	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredAttribute Setting Request (Link) Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	wiredSend(SendPacket);
	Printscreen("Link Resource Distributing Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredAttribute Setting Request (Link) received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Attribute Setting Request 帧：0x0c | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

8.1.5 现场设备加入网络(正向测试)[GW-RUN005]

该测试用例测试网关设备能否正确处理现场设备加入 WIA-FA 网络。

测试过程为：

a） 测试系统加入被测设备网络后，向被测设备发送设备加入指示；

b） 被测设备接收到指示后，向测试系统返回设备加入响应；

c） 测试系统将接收的设备加入响应报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 68 所示，具体测试说明如表 81 所示。

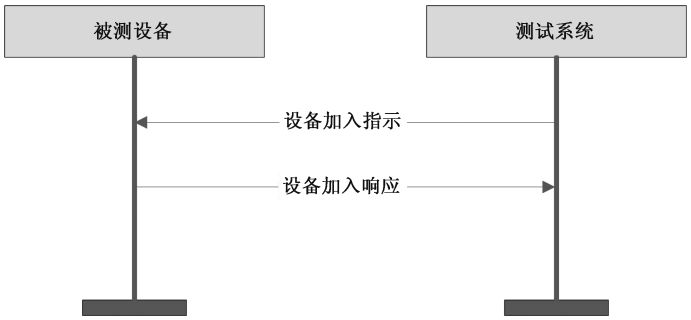

图 68　现场设备加入网络(正向测试)时序图

表 81　现场设备加入网络(正向测试)说明

用例名称	现场设备加入网络测试[GW-RUN005]
被测设备	网关设备
依赖测试条件 SecLevel=0	测试系统(接入设备)已加入被测设备网络 被测设备已完成对测试系统(接入设备)的资源配置
依赖测试条件 SecLevel=1～8	测试系统(接入设备)已加入被测设备网络 被测设备已完成对测试系统(接入设备)的资源配置 被测设备已完成对测试系统(现场设备)的长地址登记

测试用例伪代码描述	TEST BODY:
	SendPacket =WiredFDJoinRequest;
	RefPacket= WiredFDJoinResponse;
	wiredSend(SendPacket);
	WHILE(wiredReceive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("Wired FD JoinResponse Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("FD JoiningTest Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Wired FD JoinResponse received!");
	TestCaseResult = FAILED;
	}

表 81（续）

测试用例伪代码描述	Printscreen(TestCaseResult)； TEST RESULT： SUCCESS or FAILED
参考数据包 SecLevel＝0	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired FD Join Request 帧：0x06 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x?? |0x?? Field：服务标识符(1)|AdID(1)|源地址(8)|包序号(2)|服务参数长度(2)|源地址(8)| AdID(1)|
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired FD Join Response 帧：0x07 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x00 | 0x03 Field：服务标识符(1)|AdID(1)|目的地址(8)|包序号(2)|服务参数长度(2)|AD 数量(1)|AD 列表(n)|状态(1)|短地址(1)
参考数据包 SecLevel＝1～8	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wire FD Join Request 帧：0x06 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x?? |0x?? | 0x??… 0x?? Field：服务标识符(1)|AdID(1)|源地址(8)|包序号(2)|服务参数长度(2)|源地址(8)| AdID(1)|安全材料(8)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired FD Join Response 帧：0x07 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x00 | 0x03 Field：服务标识符(1)|AdID(1)|目的地址(8)|包序号(2)|服务参数长度(2)|AD 数量(1)|AD 列表(n)|状态(1)|短地址(1)

8.1.6 现场设备加入网络(反向测试)[GW-RUN006]

该测试用例测试网关设备能否正确响应现场设备的加入请求。

测试过程为：

a) 测试系统向被测设备通过有线发送错误的加入请求；

1) 错误的网络 ID；

2) 错误的 KJ 计算的 SecMaterial（SecLevel 不为 0 时）。

b) 被测设备向测试系统返回响应，具体情况如下：

1) 网络 ID 不匹配；

2) 认证失败(SecLevel 不为 0 时)。

参考数据包中具体载荷内容如表 82 所示。

c) 测试系统将接收的加入响应报文与期望的报文进行比对，如果比对匹配则测试通过

表 82　现场设备加入网络(反向测试)参考数据包载荷

测试内容	测试系统应发加入响应数据载荷
网络 ID 不匹配	0x01 | 0x03
	加入状态(1)|分配的短地址(1)
认证失败	0x02 | 0x03
	加入状态(1)|分配的短地址(1)

该测试用例用于现场设备入网的两种反向测试,测试体应循环执行,具体时序如图 69 所示,具体测试说明如表 83 所示。

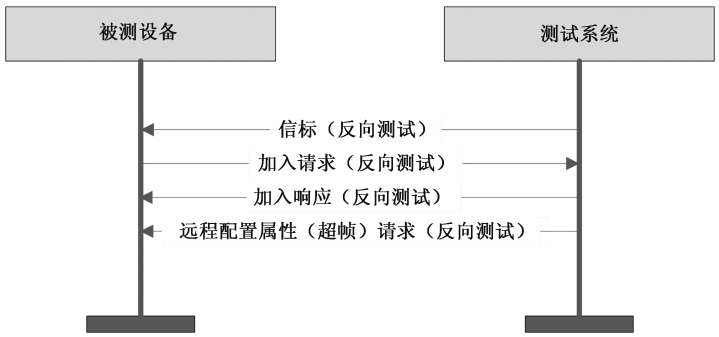

图 69　现场设备加入网络(反向测试)时序图

表 83　现场设备加入网络(反向测试)说明

用例名称	现场设备加入网络(反向测试)[GW-RUN006]
被测设备	网关设备
依赖测试条件 SecLevel=0	测试系统(接入设备)已加入被测设备网络 测试系统(现场设备)未加入被测设备网络 KJ=NULL
依赖测试条件 SecLevel=1～8	被测设备已登记测试系统(现场设备)的长地址 测试系统已完成现场设备的网络 ID 和 SecLevel 设置 测试系统已完成现场设备的加入密钥 KJ 配置 测试系统(接入设备)已加入被测设备网络 测试系统(现场设备)未加入被测设备网络 KJ={0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7 0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf} Wrong KJ={0xc1 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7 0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf}

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredFDJoinRequest;
	RefPacket= WiredFDJoinResponse;
	wiredSend(SendPacket);
	WHILE(wiredReceive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{

表 83（续）

测试用例伪代码描述	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("Wired FD JoinResponse Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("FD JoiningTest Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Wired FD JoinResponse received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包 SecLevel=0	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired FD Join Request 帧：0x06 | 0x?? | 0x?? ... 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? |0x?? Field：服务标识符(1)|AdID(1)|源地址(8)|包序号(2)|服务参数长度(2)|源地址(8)| AdID(1)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired FD Join Response 帧：0x07 | 0x?? | 0x?? ... 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? ... 0x?? | 0x?? 0x?? Field：服务标识符(1)|AdID(1)|目的地址(8)|包序号(2)|服务参数长度(2)|AD 数量(1)|AD 列表(n)|载荷(2)
参考数据包 SecLevel=1～8	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired FD Join Request 帧：0x06 | 0x?? | 0x?? ... 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x?? |0x?? | 0x??... 0x?? Field：服务标识符(1)|AdID(1)|源地址(8)|包序号(2)|服务参数长度(2)|源地址(8)| AdID(1)|安全材料(8)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired FD Join Response 帧：0x07 | 0x?? | 0x?? ... 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? ... 0x?? | 0x?? 0x?? Field：服务标识符(1)|AdID(1)|目的地址(8)|包序号(2)|服务参数长度(2)|AD 数量(1)|AD 列表(n)|载荷(2)

8.1.7 现场设备超帧资源分配测试[GW-RUN007]

该测试用例测试网关设备能否正确对现场设备进行超帧资源分配。

测试过程为：

a) 测试系统加入被测设备网络后，被测设备向测试系统发送远程配置属性(超帧)请求；

b) 测试系统将接收的远程配置属性(超帧)请求报文与期望的报文进行比对，如果比对匹配，则测试通过，并向被测设备返回远程配置属性(超帧)证实。

具体时序如图 70 所示，具体测试说明如表 84 所示。

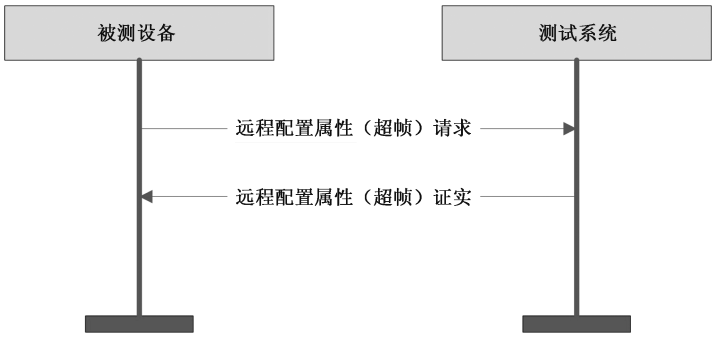

图 70 现场设备超帧资源分配测试时序图

表 84 现场设备超帧资源分配测试说明

用例名称	现场设备超帧资源分配测试[GW-RUN007]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络

测试用例伪代码描述	TEST BODY：
	SendPacket = WiredAttributeSettingConfirm；
	RefPacket =WiredAttributeSettingRequest(Superframe)；
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime；
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredAttribute Setting Request (Superframe) Payload error!")；
	TestCaseResult =FAILED；
	}
	ELSE
	{
	wiredSend(SendPacket)；
	Printscreen("Superframe Resource Distributing Test Success!")；
	TestCaseResult =SUCCESS；
	}
	}
	ELSE
	{

表 84（续）

测试用例伪代码描述	Printscreen(“No WiredAttribute Setting Request (Superframe) received!”); TestCaseResult = FAILED; } Printscreen(TestCaseResult); TEST RESULT: SUCCESS or FAILED
参考数据包	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Attribute Setting Request 帧：0x0c | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

8.1.8 现场设备链路资源分配测试[GW-RUN008]

该测试用例测试网关设备能否正确对现场设备进行链路资源分配。

测试过程为：

a) 测试系统加入被测设备网络后，被测设备向测试系统发送远程配置属性(链路)请求；

b) 测试系统将接收的远程配置属性(链路)请求报文与期望的报文进行比对，如果比对匹配，则测试通过，并向被测设备返回远程配置属性(链路)证实。

具体时序如图 71 所示，具体测试说明如表 85 所示。

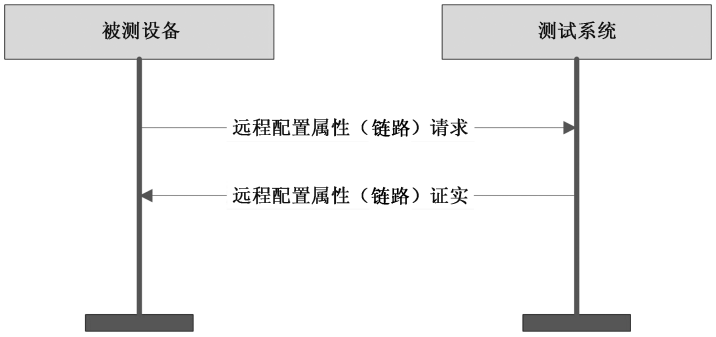

图 71 现场设备链路资源分配测试时序图

表 85　现场设备链路资源分配测试说明

用例名称	现场设备链路资源分配测试[GW-RUN008]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredAttributeSettingConfirm;
	RefPacket =WiredAttributeSettingRequest(Link);
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredAttribute Setting Request (Link) Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	wiredSend(SendPacket);
	Printscreen("Link Resource Distributing Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredAttribute Setting Request (Link) received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 协议层:Wired DLL 帧名称:Wired Attribute Setting Confirm 帧:0x0d | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明:服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)
	测试系统应收数据包: 协议层:Wired DLL 帧名称:Wired Attribute Setting Request 帧:0x0c | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明:服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

8.1.9 网关设备指示接入设备发送 NACK 测试[GW-RUN009]

该测试用例测试网关设备能否正确指示接入设备发送 NACK。

测试过程为：

a) 测试系统加入被测设备网络后，测试系统(接入设备)停止向网关发送设备状态汇报帧，被测设备在设备状态回报周期后向测试系统发送网关设备指示接入设备发送 NACK；

b) 测试系统将接收的网关设备指示接入设备发送 NACK 报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 72 所示，具体测试说明如表 86 所示。

图 72 网关设备指示接入设备发送 NACK 测试时序图

表 86 网关设备指示接入设备发送 NACK 测试说明

用例名称	网关设备指示接入设备发送 NACK 测试[GW-RUN009]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测设备已完成对测试系统的资源配置(超帧、链路、设备状态汇报周期等)

测试用例伪代码描述	TEST BODY：
	RefPacket ＝WiredIndicatingNACK；
	WHILE(Receive(RcvPacket).type ！ ＝ RefPacket.type) for MaxWaitTime；
	IF(RcvPacket.type ＝＝ RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ！ ＝ SUCCESS)
	{
	Printscreen(“WiredIndicating NACKPayload error!”)；
	TestCaseResult ＝FAILED；
	}
	ELSE
	{
	Printscreen(“Indicating NACK Test Success!”)；
	TestCaseResult ＝SUCCESS；
	}
	}
	ELSE
	{

表 86（续）

测试用例伪代码描述	Printscreen(“No Wired Indicating NACKreceived!”); TestCaseResult = FAILED; } Printscreen(TestCaseResult); TEST RESULT: SUCCESS or FAILED 测试系统应收数据包: 协议层: Wired DLL 帧名称:Wired Indicating NACK 帧: 0x03 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明: 服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

8.1.10 网关设备指示接入设备发送 GACK 测试[GW-RUN010]

该测试用例测试网关设备能否正确指示接入设备发送 GACK。

测试过程为:

a) 测试系统加入被测设备网络后,通过有线向被测设备发送非周期性数据帧;

b) 被测设备通过有线向测试系统发送网关设备指示接入设备发送 GACK 帧;

c) 测试系统将接收的网关设备指示接入设备发送 GACK 报文与期望的报文进行比对,如果比对匹配,则测试通过。

具体时序如图 73 所示,具体测试说明如表 87 所示。

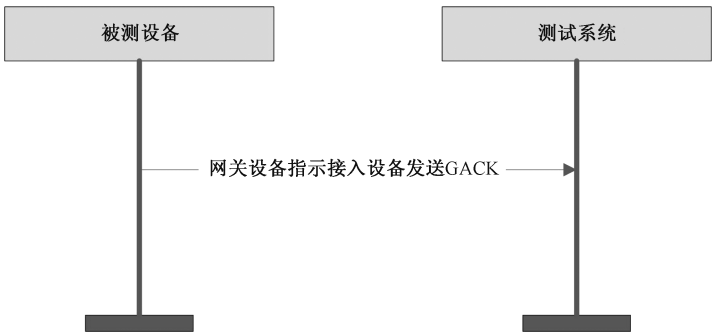

图 73 网关设备指示接入设备发送 GACK 测试时序图

表 87 网关设备指示接入设备发送 GACK 测试说明

用例名称	网关设备指示接入设备发送 GACK 测试[GW-RUN010]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测设备已完成对测试系统的资源配置 被测设备已完成非周期数据组态

表 87（续）

测试用例伪代码描述	TEST BODY：
	SendPacket = wiredData
	RefPacket =WiredIndicatingGACK；
	wiredSend(SendPacket)；
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime；
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen(“WiredIndicating GACK Payload error!”)；
	TestCaseResult =FAILED；
	}
	ELSE
	{
	Printscreen(“Indicating GACK Test Success!”)；
	TestCaseResult =SUCCESS；
	}
	}
	ELSE
	{
	Printscreen(“No Wired Indicating GACKreceived!”)；
	TestCaseResult = FAILED；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 无
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Indicating GACK 帧：0x02 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

8.1.11 网关发送数据测试[GW-RUN011]

该测试用例测试网关设备能否正确进行数据传输。

测试过程为：

a） 测试系统加入被测设备网络后，被测设备向测试系统发送数据发送请求；

b） 测试系统将接收的数据发送请求报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 74 所示，具体测试说明如表 88 所示。

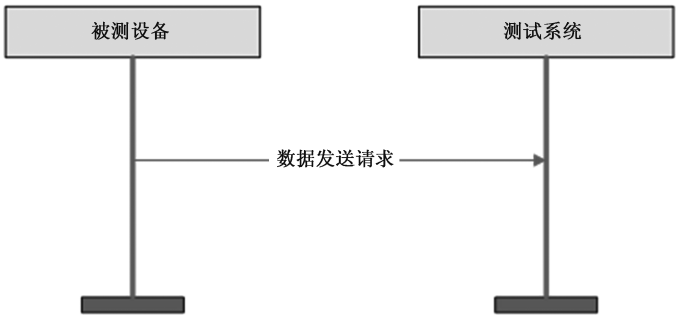

图 74　网关发送数据测试时序图

表 88　网关发送数据测试说明

用例名称	网关发送数据测试[GW-RUN011]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测设备已完成对测试系统的资源配置 被测设备和测试系统已完成应用组态

测试用例伪代码描述	TEST BODY:
	RefPacket =WiredDataRequest;
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("Wired Data RequestPayload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("Data Transmission Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No Wired Data Requestreceived!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 88（续）

参考数据包	测试系统应发数据包： 无
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Data Request 帧：0x04 | 0x?? | 0x02 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)| AdID(1)|对端地址(1)|序列号(2)|长度(2)|载荷(3)

8.1.12 远程读属性测试[GW-RUN012]

该测试用例测试网关设备能否正确发送远程读属性请求。

测试过程为：

a) 测试系统加入被测设备网络后，被测设备向测试系统发送远程读属性请求，具体情况如下：

1) 读非结构化属性；
2) 读结构化属性一个成员的一个记录；
3) 读结构化属性一个成员的多个记录；
4) 读结构化属性一个成员的全部记录；
5) 读结构化属性全部成员的一个记录；
6) 读结构化属性全部成员的多个记录；
7) 读结构化属性全部成员的全部记录。

参考数据包中具体服务参数内容如表 89 所示。

表 89 远程读属性测试参考数据包服务参数

测试内容	应收服务参数	应发服务参数
读非结构化属性	0x?? | 0x03 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x??	0x?? |0x03 | 0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性一个成员的一个记录	0x?? | 0x03 | 0x?? | 0x?? | 0x?? 0x?? | 0x00 0x01	0x?? |0x03 | 0x00 | 0x?? | 0x?? (不等于 0xff)| 0x?? 0x?? | 0x00 0x01 | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性一个成员的多个记录	0x?? | 0x03 | 0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x??	0x?? |0x03 | 0x00 | 0x?? | 0x?? (不等于 0xff) | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)

表 89（续）

测试内容	应收服务参数	应发服务参数
读结构化属性一个成员的全部记录	0x?? | 0x03 | 0x?? | 0x?? | 0x?? 0x?? | 0x00 0x00	0x?? |0x03 | 0x00 | 0x?? | 0x?? (不等于 0xff) | 0x?? 0x?? | 0x00 0x00 | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性全部成员的一个记录	0x?? | 0x03 | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x01	0x?? |0x03 | 0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x01 | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性全部成员的多个记录	0x?? | 0x03 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x??	0x?? |0x03 | 0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)
读结构化属性全部成员的全部记录	0x?? | 0x03 | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x00	0x?? |0x03 | 0x00 | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x00 | 0x?? … 0x??
	句柄(1)|目的地址(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)	句柄(1)|目的地址(1)|执行结果(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)

b） 测试系统将接收的远程读属性请求报文与期望的报文进行比对，如果比对匹配，则测试通过，并向被测设备返回远程读属性证实。

该测试用例用于读测试系统信息库中的所有属性，测试体应循环执行，具体时序如图 75 所示，具体测试说明如表 90 所示。

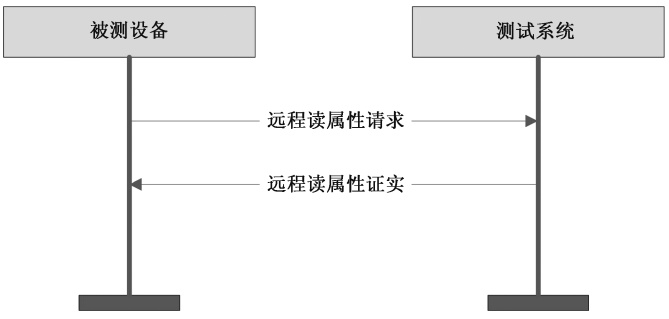

图 75 远程读属性测试时序图

表 90　远程读属性测试说明

用例名称	远程读属性测试 [GW-RUN012]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测设备已完成对测试系统的资源配置

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredAttributeGettingConfirm;
	RefPacket =WiredAttributeGettingRequest;
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredAttribute Getting Request Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	wiredSend(SendPacket);
	Printscreen("Attribute Getting Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredAttribute Getting Request received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 协议层: Wired DLL 帧名称: Wired Attribute Getting Confirm 帧: 0x0b | 0x?? | 0x?? 0x?? | 0x?? ... 0x?? 帧域说明: 服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(n)
	测试系统应收数据包: 协议层: Wired DLL 帧名称: Wired Attribute Getting Request 帧: 0x0a | 0x?? | 0x00 0x08 | 0x?? ... 0x?? 帧域说明: 服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(8)

8.1.13 远程配置属性测试[GW-RUN013]

该测试用例测试网关设备能否正确发送远程配置属性请求。

测试过程为：

a) 测试系统加入被测设备网络后，被测设备向测试系统发送远程配置属性请求，具体情况如下：

1) 配置非结构化属性；
2) 配置结构化属性一个成员的一个记录；
3) 配置结构化属性一个成员的多个记录；
4) 配置结构化属性一个成员的全部记录；
5) 配置结构化属性全部成员的一个记录；
6) 配置结构化属性全部成员的多个记录；
7) 配置结构化属性全部成员的全部记录。

参考数据包中具体服务参数内容如表91所示。

表91 远程配置属性测试参考数据包服务参数

测试内容	应收服务参数	应发服务参数
配置非结构化属性	0x?? |0x03 | 0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x??	0x?? |0x03 | 0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x?? 0x?? | 0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性一个成员的一个记录	0x?? |0x03 | 0x?? | 0x?? | 0x?? | 0x?? 0x?? | 0x00 0x01 | 0x?? ... 0x??	0x?? |0x03 | 0x?? | 0x?? | 0x?? (不等于0xff)| 0x?? 0x?? | 0x00 0x01 | 0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性一个成员的多个记录	0x?? |0x03 | 0x?? | 0x?? | 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? ... 0x??	0x?? |0x03 | 0x?? | 0x?? | 0x?? (不等于0xff)| 0x?? 0x?? | 0x?? 0x?? | 0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性一个成员的全部记录	0x?? |0x03 | 0x?? | 0x?? | 0x?? | 0x?? 0x?? | 0x00 0x00 | 0x?? ... 0x??	0x?? |0x03 | 0x?? | 0x?? | 0x?? (不等于0xff)| 0x?? 0x?? | 0x00 0x00 | 0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)
配置结构化属性全部成员的一个记录	0x?? |0x03 | 0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x01 | 0x?? ... 0x??	0x?? |0x03 | 0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x01 | 0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|执行结果(1)

表 91（续）

测试内容	应收服务参数	应发服务参数
配置结构化属性全部成员的全部记录	0x?? |0x03 | 0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x00 | 0x?? … 0x??	0x?? |0x03 | 0x?? | 0x?? | 0xff | 0x?? 0x?? | 0x00 0x00 | 0x00
	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(2)|属性值(n)	句柄(1)|目的地址(1)|远程属性操作(1)|属性标识符(1)|属性成员标识符(1)|多个属性值的第一个存储索引(2)|属性数目(1)|执行结果(1)

b） 测试系统将接收的远程配置属性请求报文与期望的报文进行比对，如果比对匹配，则测试通过，并向被测设备返回远程配置属性证实。

该测试用例用于配置测试系统信息库中的所有属性，测试体应循环执行，具体时序如图 76 所示，具体测试说明如表 92 所示。

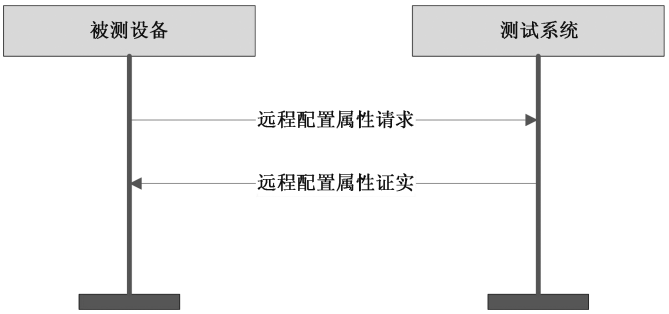

图 76　远程配置属性测试时序图

表 92　远程配置属性测试说明

用例名称	远程配置属性测试 [GW-RUN013]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测设备已完成对测试系统的资源配置

测试用例伪代码描述	TEST BODY：
	SendPacket = WiredAttributeSettingConfirm；
	RefPacket =WiredAttributeSettingRequest；
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime；
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen(“WiredAttribute Setting Request Payload error!”)；
	TestCaseResult =FAILED；
	}
	ELSE

表 92（续）

测试用例伪代码描述	{
	wiredSend(SendPacket);
	Printscreen("Attribute Setting Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredAttribute Setting Request received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 协议层：Wired DLL 帧名称：Wired Attribute Setting Confirm 帧：0x0d | 0x?? | 0x00 0x02 | 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(2)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Attribute Setting Request 帧：0x0c | 0x?? | 0x?? 0x?? | 0x?? ... 0x?? 帧域说明：服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(n)

8.1.14 现场设备 KEK 密钥分发测试[GW-RUN014]

该测试用例测试网关设备能否正确为现场设备分发 KEK 密钥。

测试过程为：

a) 测试系统加入被测设备网络后，通过有线向被测设备发送现场设备加入请求帧；

b) 被测设备接收到现场设备加入请求后，通过有线返回现场设备加入响应，并发起 KEK 密钥分发请求；

c) 测试系统接收到密钥分发请求后，通过有线向被测设备发送密钥分发响应；

d) 测试系统对接收到的报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 77 所示，具体测试说明如表 93 所示。

图 77　现场设备 KEK 密钥分发测试时序图

表 93　现场设备 KEK 密钥分发测试说明

用例名称	现场设备 KEK 密钥分发测试[GW-RUN014]
被测设备	网关设备
依赖测试条件	测试系统(接入设备)已加入被测设备网络 被测设备已完成对测试系统(接入设备)的资源配置(链路、超帧、密钥) SecLevel = 2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredFDJoinIndication;
	SendPacket2 = WiredKeyEstablishRequest(KEK);
	RefPacket1 =WiredFDJoinResponse;
	RefPacket2 =WiredKeyEstablishConfirm;
	wiredSend(SendPacke1);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (wiredVerify(RcvPacket1.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Wired FD Join ResponsePayload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	Printscreen("FD Join Success!");
	TestCaseResult ="FAILED";
	}
	}
	ELSE
	{
	Printscreen("No WiredFD Join Responsereceived!");
	TestCaseResult = "FAILED";
	}
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)

表 93（续）

测试用例伪代码描述	{
	IF（wiredVerify(RcvPacket.all，RefPacket2.all) ！ = SUCCESS)
	{
	Printscreen(“Wired Key Establish Request Payload error!”)；
	TestCaseResult =“FAILED”；
	}
	ELSE
	{
	wiredSend(SendPacket2)；
	Printscreen(“KEK Establish Test Success!”)；
	TestCaseResult =“SUCCESS”；
	}
	}
	ELSE
	{
	Printscreen(“No Wired Key Establish Request received!”)；
	TestCaseResult = “FAILED”；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired FD Join Indication 帧：0x06 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x?? |0x?? | 0x??… 0x?? 帧域说明：服务标识符(1)|AdID(1)|源地址(8)|包序号(2)|服务参数长度(2)|源地址(8)| AdID(1)|安全材料(8) 数据包 2： 协议层：Wired DLL 帧名称：Wired Key Establish Confirm 帧：0x11 | 0x?? | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序列号(2)|包长度(2)|源地址(2)|密钥 ID(4)|密钥状态(1)
	测试系统应收数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired FD Join Response 帧：0x07 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x00 | 0x03 帧域说明：服务标识符(1)|AdID(1)|目的地址(8)|包序号(2)|服务参数长度(2)|AD 数量(1)|AD 列表(n)|状态(1)|短地址(1)| 数据包 2：

表 93（续）

参考数据包	协议层：Wired DLL 帧名称：Wired Key Establish Request 帧：0x10 | 0x?? | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x02 | 0x?? … 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序列号(2)|包长度(2)|目的地址(2)|密钥 ID(2)|密钥类型(1)|密钥激活时间(6)|密钥(n)|密钥 MIC(4)|

8.1.15 现场设备 KEDU 密钥分发测试[GW-RUN015]

该测试用例测试网关设备能否正确为现场设备分发 KEDU 密钥。

测试过程为：

a) 测试系统加入被测设备网络后，通过有线向被测设备发送现场设备加入请求帧；

b) 被测设备接收到现场设备加入请求后，通过有线返回现场设备加入响应，并发起 KEDU 密钥分发请求；

c) 测试系统接收到密钥分发请求后，通过有线向被测设备发送密钥分发响应；

d) 测试系统对接收到的报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 78 所示，具体测试说明如表 94 所示。

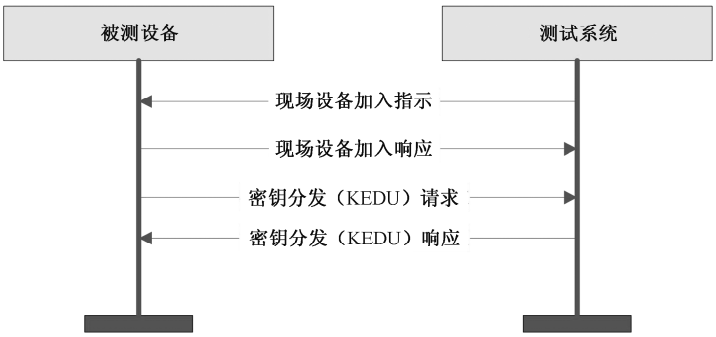

图 78 现场设备 KEDU 密钥分发测试时序图

表 94 现场设备 KEDU 密钥分发测试说明

用例名称	现场设备 KEDU 密钥分发测试[GW-RUN015]
被测设备	网关设备
依赖测试条件	测试系统(接入设备)已加入被测设备网络 被测设备已完成对测试系统(接入设备)的资源配置(链路、超帧、密钥) SecLevel = 2～8
测试用例伪代码描述	TEST BODY： SendPacket1 = WiredFDJoinIndication； SendPacket2 = WiredKeyEstablishRequest(KEDU)； RefPacket1 =WiredFDJoinResponse； RefPacket2 =WiredKeyEstablishConfirm； wiredSend(SendPacke1)；

表 94（续）

测试用例伪代码描述	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (wiredVerify(RcvPacket1.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Wired FD Join ResponsePayload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	Printscreen("FD Join Success!");
	TestCaseResult ="FAILED";
	}
	}
	ELSE
	{
	Printscreen("No WiredFD Join Responsereceived!");
	TestCaseResult = "FAILED";
	}
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket2.all) ! = SUCCESS)
	{
	Printscreen("Wired Key Establish Request Payload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	wiredSend(SendPacket2);
	Printscreen("KEK Establish Test Success!");
	TestCaseResult ="SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Wired Key Establish Request received!");
	TestCaseResult = "FAILED";
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 94（续）

参考数据包	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired FD Join Indication 帧：0x06 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x?? |0x?? | 0x??… 0x?? 帧域说明：服务标识符(1)|AdID(1)|源地址(8)|包序号(2)|服务参数长度(2)|源地址(8)| AdID(1)|安全材料(8) 数据包 2： 协议层：Wired DLL 帧名称：Wired Key Establish Confirm 帧：0x11 | 0x?? | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序列号(2)|包长度(2)|源地址(2)|密钥 ID(4)|密钥状态(1)
	测试系统应收数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired FD Join Response 帧：0x07 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x00 | 0x03 帧域说明：服务标识符(1)|AdID(1)|目的地址(8)|包序号(2)|服务参数长度(2)|AD 数量(1)|AD 列表(n)|状态(1)|短地址(1) 数据包 2： 协议层：Wired DLL 帧名称：Wired Key Establish Request 帧：0x10 | 0x?? | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x03 | 0x?? … 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序列号(2)|包长度(2)|目的地址(2)|密钥 ID(2)|密钥类型(1)|密钥激活时间(6)|密钥(n)|密钥 MIC(4)

8.1.16 现场设备 KEDB 密钥分发测试[GW-RUN016]

该测试用例测试网关设备能否正确为现场设备分发 KEDB 密钥。

测试过程为：

a） 测试系统加入被测设备网络后，通过有线向被测设备发送现场设备加入请求帧；

b） 被测设备接收到现场设备加入请求后，通过有线返回现场设备加入响应，并发起 KEDB 密钥分发请求；

c） 测试系统接收到密钥分发请求后，通过有线向被测设备发送密钥分发响应；

d） 测试系统对接收到的报文与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 79 所示，具体测试说明如表 95 所示。

图 79　现场设备 KEDB 密钥分发测试时序图

表 95　现场设备 KEDB 密钥分发测试说明

用例名称	现场设备 KEDB 密钥分发测试[GW-RUN016]
被测设备	网关设备
依赖测试条件	测试系统(接入设备)已加入被测设备网络 被测设备已完成对测试系统(接入设备)的资源配置(链路、超帧、密钥) SecLevel = 2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredFDJoinIndication;
	SendPacket2 = WiredKeyEstablishRequest(KEDB);
	RefPacket1 =WiredFDJoinResponse;
	RefPacket2 =WiredKeyEstablishConfirm;
	wiredSend(SendPacke1);
	WHILE(Receive(RcvPacket).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (wiredVerify(RcvPacket1.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Wired FD Join ResponsePayload error!");
	TestCaseResult ="FAILED";
	}
	ELSE
	{
	Printscreen("FD Join Success!");
	TestCaseResult ="FAILED";
	}
	}
	ELSE
	{
	Printscreen("No WiredFD Join Responsereceived!");
	TestCaseResult = "FAILED";
	}
	WHILE(Receive(RcvPacket).type ! = RefPacket2.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket2.type)

表 95（续）

测试用例伪代码描述	{
	IF（wiredVerify(RcvPacket.all，RefPacket2.all) ！ = SUCCESS)
	{
	Printscreen(“Wired Key Establish Request Payload error!”)；
	TestCaseResult =“FAILED”；
	}
	ELSE
	{
	wiredSend(SendPacket2)；
	Printscreen(“KEK Establish Test Success!”)；
	TestCaseResult =“SUCCESS”；
	}
	}
	ELSE
	{
	Printscreen(“No Wired Key Establish Request received!”)；
	TestCaseResult = “FAILED”；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired FD Join Indication 帧：0x06 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? … 0x?? |0x?? | 0x??… 0x?? 帧域说明：服务标识符(1)|AdID(1)|源地址(8)|包序号(2)|服务参数长度(2)|源地址(8)| AdID(1)|安全材料(8)
	测试系统应收数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired FD Join Response 帧：0x07 | 0x?? | 0x?? … 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x?? … 0x?? | 0x00 | 0x03 帧域说明：服务标识符(1)|AdID(1)|目的地址(8)|包序号(2)|服务参数长度(2)|AD 数量(1)|AD 列表(n)|状态(1)|短地址(1)
	测试系统应发数据包： 数据包 2： 协议层：Wired DLL 帧名称：Wired Key Establish Confirm 帧：0x11 | 0x?? | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序列号(2)|包长度(2)|源地址(2)|密钥 ID(4)|密钥状态(1)

表 95（续）

参考数据包	数据包 2： 协议层：Wired DLL 帧名称：Wired Key Establish Request 帧：0x10 | 0x?? | 0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x?? 0x?? | 0x04 | 0x?? … 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序列号(2)|包长度(2)|目的地址(2)|密钥 ID(2)|密钥类型(1)|密钥激活时间(6)|密钥(n)|密钥 MIC(4)

8.1.17 现场设备 KEK 攻击告警测试[GW-RUN017]

该测试用例用于测试被测网关设备能否正确处理现场设备的 KEK 密钥攻击告警。

测试过程为：

a) 测试系统安全加入被测设备网络后，测试系统（接入设备）通过有线向被测设备发送 KEK 安全告警指示包；

b) 被测设备收到安全告警指示后，向测试系统通过有线发送 KEK 密钥更新请求；

c) 测试系统通过有线向被测设备发送密钥更新确认；

d) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 80 所示，具体测试说明如表 96 所示。

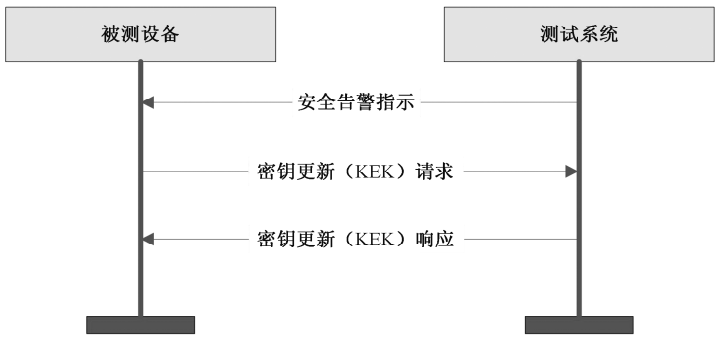

图 80 现场设备 KEK 攻击告警测试时序图

表 96 现场设备 KEK 攻击告警测试说明

用例名称	现场设备 KEK 攻击告警测试[GW-RUN017]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测识别已完成对测试系统的资源配置（超帧、链路、密钥） KEK 已启用 SecLevel=2～8
测试用例伪代码描述	TEST BODY： SendPacket1 = WiredKeyAttackAlarmIndication SendPacket2 = WiredKeyUpdateResponse； RefPacket = WiredKeyUpdateRequest（KEK）；

表 96（续）

测试用例伪代码描述	wiredSend(SendPacket1)；
	WHILE(Receive(RcvPacket)).type ！ = RefPacket1.type) for MaxWaitTime；
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket1.all) ！ = SUCCESS)
	{
	Printscreen(“Wired Key Update Request Payload error!”)；
	TestCaseResult = “FAILED”；
	}
	ELSE
	{
	wiredSend(SendPacket2)；
	TestCaseResult = “SUCCESS”；
	}
	}
	ELSE
	{
	Printscreen(“No Wired KeyUpdate Request received!”)；
	TestCaseResult = “FAILED”；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Security Attack Alarm Indication 帧：0x14 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? |0x01 | 0x?? 0x?? | 0b******？1 | 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1) 数据包 2： 协议层：Wired DLL 帧名称：Wired Key Update Response 帧：0x13 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|密钥 ID(2)|密钥状态(1)
	测试系统应收数据包： 协议层：Wired DLL 帧名称：Wired Key Update Request 帧：0x12 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x03 | 0x?? 0x?? | 0x02 | 0x?? … 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|目的地址(1)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)

8.1.18 现场设备 KEDU 攻击告警测试[GW-RUN018]

该测试用例用于测试被测网关设备能否正确处理现场设备的 KEDU 密钥攻击告警。

测试过程为：

a) 测试系统安全加入被测设备网络后，测试系统（接入设备）通过有线向被测设备发送 KEDU 安全告警指示包；

b) 被测设备收到安全告警指示后，向测试系统通过有线发送 KEDU 密钥更新请求；

c) 测试系统通过有线向被测设备发送密钥更新确认；

d) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 81 所示，具体测试说明如表 97 所示。

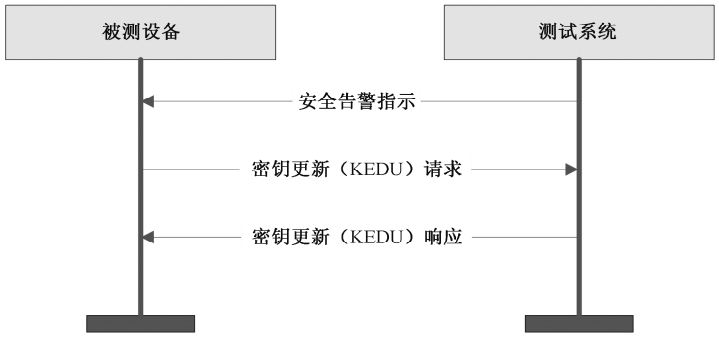

图 81 现场设备 KEDU 攻击告警测试时序图

表 97 现场设备 KEDU 攻击告警测试说明

用例名称	现场设备 KEDU 攻击告警测试[GW-RUN018]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测识别已完成对测试系统的资源配置（超帧、链路、密钥） KEDU 已启用 SecLevel=2～8

测试用例伪代码描述	TEST BODY：
	SendPacket1 = WiredKeyAttackAlarmIndication
	SendPacket2 = WiredKeyUpdateResponse；
	RefPacket = WiredKeyUpdateRequest（KEDU）；
	wiredSend(SendPacket1)；
	WHILE(Receive(RcvPacket)).type ! = RefPacket1.type) for MaxWaitTime；
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Wired Key Update Request Payload error!")；
	TestCaseResult = "FAILED"；
	}
	ELSE
	{
	wiredSend(SendPacket2)；

表 97（续）

测试用例伪代码描述	TestCaseResult = “SUCCESS”;
	}
	}
	ELSE
	{
	Printscreen(“No Wired KeyUpdate Request received!”);
	TestCaseResult = “FAILED”;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Security Attack Alarm Indication 帧：0x14 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? |0x01 | 0x?? 0x?? | 0b*******? 1 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1) 数据包 2： 协议层：Wired DLL 帧名称：Wired Key Update Response 帧：0x13 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|密钥 ID(2)|密钥状态(1)
	测试系统应收数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Key Update Request 帧：0x12 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x03 | 0x?? 0x?? | 0x03 | 0x?? … 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|目的地址(1)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)

8.1.19 现场设备 KEDB 攻击告警测试[GW-RUN019]

该测试用例用于测试被测网关设备能否正确处理现场设备的 KEDB 密钥攻击告警。

测试过程为：

a) 测试系统安全加入被测设备网络后，测试系统（接入设备）通过有线向被测设备发送 KEDB 安全告警指示包；

b) 被测设备收到安全告警指示后，向测试系统通过有线发送 KEDB 密钥更新请求；

c) 测试系统通过有线向被测设备发送密钥更新确认；

d) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 82 所示，具体测试说明如表 98 所示。

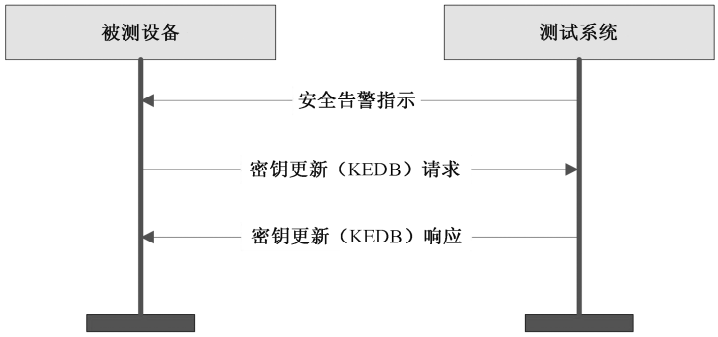

图 82　现场设备 KEDB 攻击告警测试时序图

表 98　现场设备 KEDB 攻击告警测试说明

用例名称	现场设备 KEDB 攻击告警测试[GW-RUN019]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测识别已完成对测试系统的资源配置(超帧、链路、密钥) KEDB 已启用 SecLevel=2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredKeyAttackAlarmIndication
	SendPacket2 = WiredKeyUpdateResponse;
	RefPacket = WiredKeyUpdateRequest (KEDB);
	wiredSend(SendPacket1);
	WHILE(Receive(RcvPacket)).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Wired Key Update Request Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	wiredSend(SendPacket2);
	TestCaseResult = "SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Wired KeyUpdate Request received!");
	TestCaseResult = "FAILED";
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

表 98（续）

参考数据包	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Security Attack Alarm Indication 帧：0x14 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? |0x01 | 0x?? 0x?? | 0b*******? 1 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1) 数据包 2： 协议层：Wired DLL 帧名称：Wired Key Update Response 帧：0x13 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|密钥 ID(2)|密钥状态(1)
	测试系统应收数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Key Update Request 帧：0x12 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x03 | 0x?? 0x?? | 0x04 | 0x?? … 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|目的地址(1)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)

8.1.20 现场设备 KEK 更新超时告警测试[GW-RUN020]

该测试用例用于测试被测网关设备能否正确处理现场设备的 KEK 密钥更新超时告警。

测试过程为：

a） 测试系统安全加入被测设备网络后，测试系统（接入设备）通过有线向被测设备发送 KEK 安全告警指示包；

b） 被测设备收到安全告警指示后，向测试系统通过有线发送 KEK 密钥更新请求；

c） 测试系统通过有线向被测设备发送密钥更新确认；

d） 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 83 所示，具体测试说明如表 99 所示。

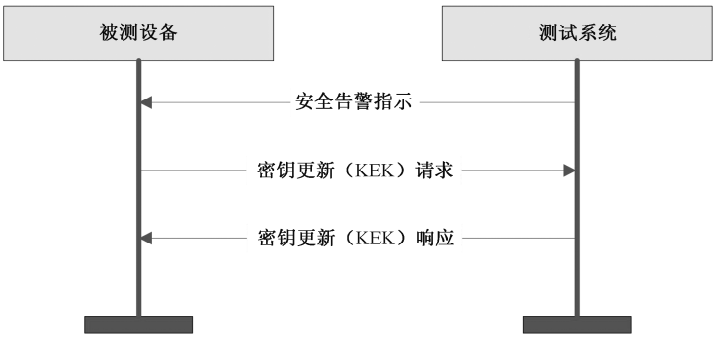

图 83 现场设备 KEK 更新超时告警测试时序图

表 99　现场设备 KEK 更新超时告警测试说明

用例名称	现场设备 KEK 更新超时告警测试[GW-RUN020]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测识别已完成对测试系统的资源配置(超帧、链路、密钥) KEK 已启用 SecLevel=2～8

测试用例伪代码描述	TEST BODY:
	SendPacket1 = WiredKeyAttackAlarmIndication
	SendPacket2 = WiredKeyUpdateResponse;
	RefPacket = WiredKeyUpdateRequest (KEK);
	wiredSend(SendPacket1);
	WHILE(Receive(RcvPacket)).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Wired Key Update Request Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	wiredSend(SendPacket2);
	TestCaseResult = "SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Wired KeyUpdate Request received!");
	TestCaseResult = "FAILED";
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 数据包 1: 协议层: Wired DLL 帧名称: Wired Security Attack Alarm Indication 帧: 0x14 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? |0x01 | 0x?? 0x?? | 0b******1? 帧域说明: 服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1) 数据包 2: 协议层:Wired DLL

表 99（续）

参考数据包	帧名称：Wired Key Update Response 帧：0x13 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|密钥 ID(2)|密钥状态(1)
	测试系统应收数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Key Update Request 帧：0x12 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x03 | 0x?? 0x?? | 0x02 | 0x?? … 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|目的地址(1)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)

8.1.21 现场设备 KEDU 更新超时告警测试[GW-RUN021]

该测试用例用于测试被测网关设备能否正确处理现场设备的 KEDU 密钥更新超时告警。

测试过程为：

a) 测试系统安全加入被测设备网络后，测试系统（接入设备）通过有线向被测设备发送 KEDU 安全告警指示包；

b) 被测设备收到安全告警指示后，向测试系统通过有线发送 KEDU 密钥更新请求；

c) 测试系统通过有线向被测设备发送密钥更新确认；

d) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 84 所示，具体测试说明如表 100 所示。

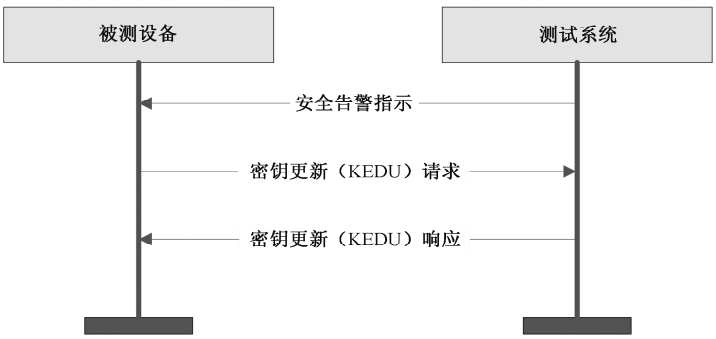

图 84 现场设备 KEDU 更新超时告警测试时序图

表 100 现场设备 KEDU 更新超时告警测试说明

用例名称	现场设备 KEDU 更新超时告警测试[GW-RUN021]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测识别已完成对测试系统的资源配置（超帧、链路、密钥） KEDU 已启用 SecLevel=2～8

表 100（续）

测试用例伪代码描述	TEST BODY：
	SendPacket1 = WiredKeyAttackAlarmIndication
	SendPacket2 = WiredKeyUpdateResponse；
	RefPacket = WiredKeyUpdateRequest (KEDU)；
	wiredSend(SendPacket1)；
	WHILE(Receive(RcvPacket)).type ! = RefPacket1.type) for MaxWaitTime；
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Wired Key Update Request Payload error!")；
	TestCaseResult = "FAILED"；
	}
	ELSE
	{
	wiredSend(SendPacket2)；
	TestCaseResult = "SUCCESS"；
	}
	}
	ELSE
	{
	Printscreen("No Wired KeyUpdate Request received!")；
	TestCaseResult = "FAILED"；
	}
	Printscreen(TestCaseResult)；
	
	TEST RESULT：
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Security Attack Alarm Indication 帧：0x14 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? |0x01 | 0x?? 0x?? | 0b******1? 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1) 数据包 2： 协议层：Wired DLL 帧名称：Wired Key Update Response 帧：0x13 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|密钥 ID(2)|密钥状态(1)

表 100（续）

参考数据包	测试系统应收数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Key Update Request 帧：0x12 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x03 | 0x?? 0x?? | 0x03 | 0x?? … 0x?? | 0x?? … 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|目的地址(1)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)

8.1.22 现场设备 KEDB 更新超时告警测试[GW-RUN022]

该测试用例用于测试被测网关设备能否正确处理现场设备的 KEDB 密钥更新超时告警。

测试过程为：

a) 测试系统安全加入被测设备网络后，测试系统（接入设备）通过有线向被测设备发送 KEDB 安全告警指示包；

b) 被测设备收到安全告警指示后，向测试系统通过有线发送 KEDB 密钥更新请求；

c) 测试系统通过有线向被测设备发送密钥更新确认；

d) 测试系统将接收到的报文与期望的报文进行对比，如果比对匹配，则认为测试通过。

具体时序如图 85 所示，具体测试说明如表 101 所示。

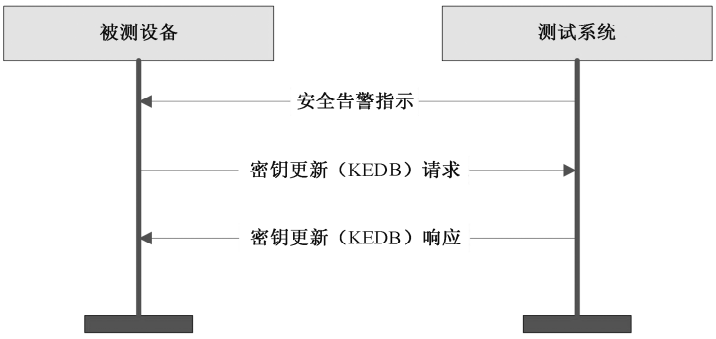

图 85 现场设备 KEDB 更新超时告警测试时序图

表 101 现场设备 KEDB 更新超时告警测试说明

用例名称	现场设备 KEDB 更新超时告警测试[GW-RUN022]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测识别已完成对测试系统的资源配置（超帧、链路、密钥） KEDB 已启用 SecLevel=2～8
测试用例伪代码描述	TEST BODY： SendPacket1 = WiredKeyAttackAlarmIndication SendPacket2 = WiredKeyUpdateResponse； RefPacket = WiredKeyUpdateRequest（KEDB）；

表 101（续）

测试用例伪代码描述	wiredSend(SendPacket1);
	WHILE(Receive(RcvPacket)).type ! = RefPacket1.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket1.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("Wired Key Update Request Payload error!");
	TestCaseResult = "FAILED";
	}
	ELSE
	{
	wiredSend(SendPacket2);
	TestCaseResult = "SUCCESS";
	}
	}
	ELSE
	{
	Printscreen("No Wired KeyUpdate Request received!");
	TestCaseResult = "FAILED";
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Security Attack Alarm Indication 帧：0x14 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? |0x01 | 0x?? 0x?? | 0b******1? 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|安全告警数量(1)|密钥 ID(2)|告警标志(1) 数据包 2： 协议层：Wired DLL 帧名称：Wired Key Update Response 帧：0x13 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x?? 0x?? | 0x00 帧域说明：服务标识符(1)|AdID(1)|源地址(1)|包序号(2)|服务参数长度(2)|密钥 ID(2)|密钥状态(1)
	测试系统应收数据包： 数据包 1： 协议层：Wired DLL 帧名称：Wired Key Update Request 帧：0x12 | 0x?? | 0x03 | 0x?? 0x?? |0x?? 0x?? | 0x03 | 0x?? 0x?? | 0x04 | 0x?? ... 0x?? | 0x?? ... 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|目的地址(1)|密钥 ID(2)|密钥类型(1)|密钥激活时隙(6)|密钥值(16)|密钥 MIC(4)

8.1.23 P/S 通信测试[GW-RUN023]

该测试用例测试网关设备能否周期性发送输出数据。

测试过程为：

a) 被测设备配置 VCR 结束后，其 AL 运行 AMPB 状态机，向测试系统发送输入数据；

b) 在 4 个超帧时间内，测试系统应至少接收到 3 个数据，被测设备发出数据的周期的最大误差为 1×TimeSlot；

c) 测试系统将接收的数据帧报文与期望的报文进行比对，如果比对匹配且周期误差不超过最大误差，则测试通过。

具体时序如图 86 所示，具体测试说明如表 102 所示。

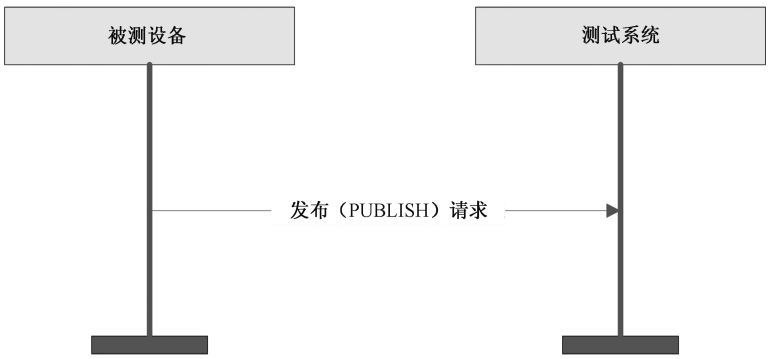

图 86 P/S 通信测试时序图

表 102 P/S 通信测试说明

用例名称	P/S 通信测试[GW-RUN023]
被测设备	网关设备
依赖测试条件	被测设备已加入

测试用例伪代码描述	TEST BODY：
	RefPacket =UAPPublishRequestMessage；
	Count = 0；
	WHILE(1)|for 4 * SuperframeCycle
	{
	IF (Receive (RcvPacket). DLL 帧 Header. 帧 Control. 帧 Type = = 0b00001 &&Receive (RcvPacket).ASLHeader. PacketControl. MessageType = =0b011 &&Receive(RcvPacket).ASL-Header. PacketControl. ServiceIdentifier = =0b00)
	{
	Count++；
	IF (Verify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen ("Publish Request Message error!")；
	TestCaseResult = FAILED；
	}
	}
	}

表 102（续）

测试用例伪代码描述	IF(Count < 3)
	{
	Printscreen ("Not Enough Data received!");
	TestCaseResult = FAILED;
	}
	ELSE IF(CycleError >1* TimeSlot)
	{
	Printscreen ("Data Cycle beyonds tolerance!");
	TestCaseResult = FAILED;
	}
	ELSE
	{
	Printscreen("Publish Request Message Transmission Test Success!");
	TestCaseResult = SUCCESS;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包： 无
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：DLLData 帧：0x90 | 0xaa | 0x?? | 0x?? 0x?? | 0x?? 0x?? |0x?? ... 0x?? | 0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)|DLL 载荷(n)|FCS(2) 数据包 2： 协议层：ASL 包名称：ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包：0x06 | 0x?? | 0x?? | 0x?? ... 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(n) 数据包 3： 协议层：UAP 消息名称：UAPPublishRequestMessage 消息：0x?? ... 0x?? 帧域说明：数据(n)

8.1.24 R/S 通信测试(启动/停止)[GW-RUN024]

该测试用例测试网关设备能否正确发送启动/停止命令。

测试过程为：

a) 被测设备 AL 运行 AMRS 状态机，向测试系统发送启动/停止命令；

b) 在 AlarmRptDur 时间内，测试系统应至少接收到 1 个启动/停止命令，被测设备发出启动/停止命令的周期的最大误差为 1×TimeSlot；

c) 测试系统将接收的报文与期望报文进行比对，如果比对匹配，则测试通过。

具体时序如图 87 所示，具体测试说明如表 103 所示。

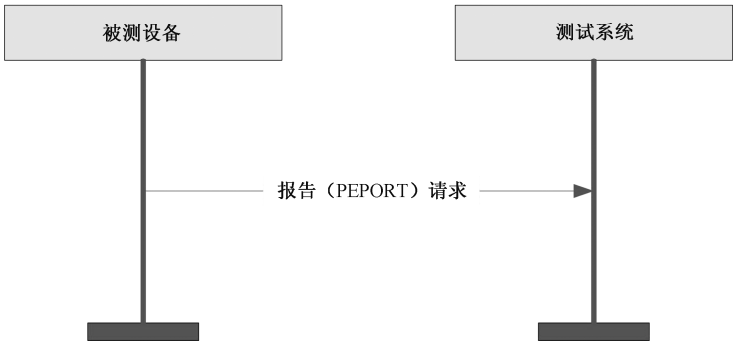

图 87 R/S 通信测试(启动/停止)时序图

表 103 R/S 通信测试(启动/停止)说明

用例名称	R/S 通信测试(启动/停止)[GW-RUN024]
被测设备	网关设备
依赖测试条件	被测设备已加入网络

测试用例伪代码描述	TEST BODY：
	RefPacket= UAPReportRequestMessage；
	WHILE(1)|forAlarmRptDur
	{
	IF(Receive(RcvPacket). DLL 帧 Header.帧 Control. 帧 Type ==0b00001
	&&Receive(RcvPacket).ASLHeader. PacketControl. MessageType
	==0b100&&Receive(RcvPacket).ASLHeader. PacketControl. ServiceIdentifier ==0b00)
	{
	IF (Verify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen (“Report Request Message error!”)；
	TestCaseResult = FAILED；
	}
	}
	}
	ELSE IF(CycleError >1 * TimeSlot)
	{
	Printscreen (“ReportData Cycle beyonds tolerance!”)；
	TestCaseResult = FAILED；
	}
	ELSE
	{
	Printscreen(“Report Request Message Transmission Test Success!”)；
	TestCaseResult = SUCCESS；
	}

表 103（续）

测试用例伪代码描述	Printscreen(TestCaseResult)； TEST RESULT： SUCCESS or FAILED
参考数据包	测试系统应发数据包： 无
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：DLLData 帧：0x90| 0xaa	0x??	0x?? 0x??	0x000x0b| 0x010x?? 0x070x?? ... 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)|载荷(11)FCS(2) 数据包 2： 协议层：ASL 包名称：ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包：0x01	0x??	0x00 0x07| 0x?? ... 0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(7) 数据包 3： 协议层：UAP 消息名称：UAPReportRequestMessage 消息：0x?? 0x??	0x?? 0x?? 0x?? 0x??	0x?? 帧域说明：UAO 标识符(2)|事件(4)|附件信息(1)

8.1.25 R/S 通信测试（报告确认）[GW-RUN025]

该测试用例测试网关设备能否正确发送报告确认请求。

测试过程为：

a） 测试系统向被测设备发送正确的报告请求；

b） 被测设备 AL 运行 AMCL 状态机，向测试系统发送报告确认请求；

c） 在 AlarmRptDur 时间内，测试系统应至少接收到 1 个报告确认请求，被测设备发出报告确认请求的周期的最大误差为 1×TimeSlot；

d） 测试系统将接收的报告确认请求与期望报文进行比对，如果比对匹配，则测试通过。

具体时序如图 88 所示，具体测试说明如表 104 所示。

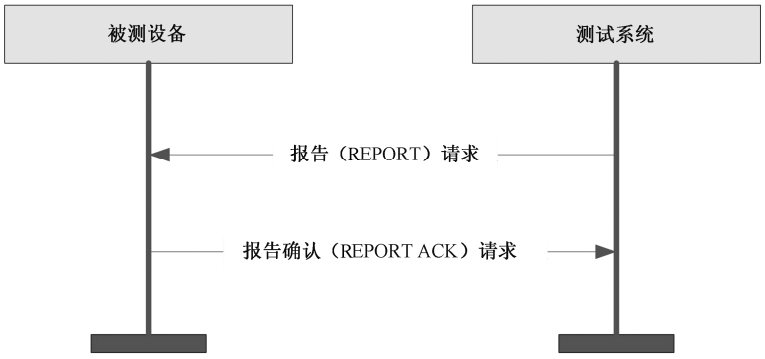

图 88 R/S 通信测试（报告确认）时序图

表 104　R/S 通信测试(报告确认)说明

用例名称	R/S 通信测试(报告确认)[GW-RUN025]
被测设备	网关设备
依赖测试条件	被测设备已加入网络

测试用例伪代码描述	TEST BODY:
	RefPacket = UAOReportAckRequest;
	WHILE(1)|for AlarmRptDur
	{
	IF(Receive(RcvPacket). DLLframeHeader.frameControl. frameType ==0b00001 &&Receive(RcvPacket).ASLHeader. PacketControl. MessageType ==0b101&&Receive(RcvPacket).ASL-Header. PacketControl. ServiceIdentifier ==0b00)
	{
	IF (Verify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen ("Report Ack RequestMessage error!");
	TestCaseResult = FAILED;
	}
	}
	}
	ELSE IF(CycleError >1 * TimeSlot)
	{
	Printscreen ("Report Ack Data Cycle beyonds tolerance!");
	TestCaseResult = FAILED;
	}
	ELSE
	{
	Printscreen("Report Ack RequestMessage Transmission Test Success!");
	TestCaseResult = SUCCESS;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 1.协议层: UAP 消息名称:UAPReportRequestMessage 消息: 0x?? 0x?? | 0x00 0x00 0x000x01 | 0x?? 帧域说明: UAO 标识符(2)|事件(4)|附加信息(1) 数据包 2: 协议层: ASL 包名称: ASLDataReqeust(PacketControl, UAP_ID, Asdulength, Asdu) 包:0x04 | 0x?? | 0x00 0x07 | 0x?? 0x?? 0x00 0x00 0x00 0x010x?? 帧域说明: 包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(7)

表 104（续）

参考数据包	数据包 3： 协议层：DLL 帧名称：DLLData 帧：0x90| 0xaa	0x??	0x?? 0x??	0x000x0b| 0x040x?? 0x00 0x070x?? 0x?? 0x00 0x00 0x00 0x010x??	0x?? 0x?? 0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)DLL 载荷(11)FCS(2)
	测试系统应收数据包： 数据包 1： 协议层：DLL 帧名称：DLLData 帧：0x90| 0xaa	0x??	0x?? 0x??	0x00 0x08| 0x05 0x?? 0x00 0x04 0x?? 0x?? 0x?? 0x01	0x?? 0x?? 帧域说明：帧控制(1)|网络 ID(1)|源或目的地址(1)|序列号(2)|帧长度(2)DLL 载荷(8)|FCS(2) 数据包 2： 协议层：ASL 包名称：ASLDataResponse(PacketControl，UAP_ID，Asdulength，Asdu) 包：0x05	0x??	0x00 0x04| 0x?? 0x?? 0x?? 0x01 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|载荷(4) 数据包 3： 协议层：UAP 消息名称：UAPReportAckRequestMessage 消息：0x?? 0x?? 0x?? 0x01 帧域说明：UAO 标识符(2)|确认事件(2)

8.1.26 C/S 通信测试(读属性)[GW-RUN026]

该测试用例测试网关设备 AL 能否正确发送读/写 MIB 属性请求。

测试过程为：

a) 被测设备加入网络后，被测设备运行 AMCL 状态机，向测试系统发送读(READ)请求，以读取设备表 DeviceList 为例进行测试；

b) 测试系统将接收的读(READ)请求与期望的报文进行比对，如果比对匹配，则测试通过。

具体时序如图 89 所示，具体测试说明如表 105 所示。

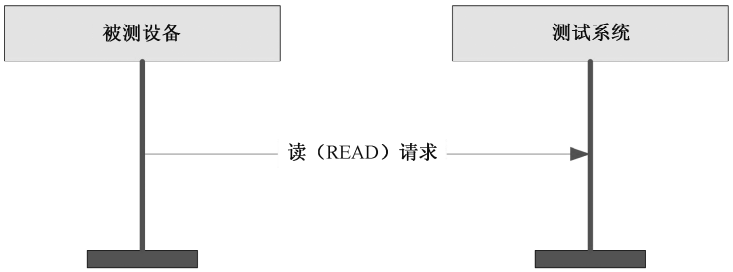

图 89 C/S 通信测试(读属性)时序图

表 105　C/S 通信测试(读属性)说明

用例名称	C/S 通信测试(读属性)[GW-RUN026]
被测设备	网关设备
依赖测试条件	被测设备已加入测试系统网络

测试用例伪代码描述	TEST BODY:
	RefPacket= UAPReadRequest(DeviceList);
	WHILE(Receive(ASLDataRequest).PacketControl&0x1F! =0x04)forLossConnectDuration;
	IF(Receive(ASLDataRequest).PacketControl & 0x1F== 0x04)
	{
	IF (Verify(Receive(ASLDataRequest). Asdu,RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("ALRead MIB Request error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("ALRead MIB RequestTest Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("NoALReadMIB Request received!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESSor FAILED

参考数据包	测试系统应发包: 无
	测试系统应收包: 数据包 1: 协议层:UAP 包名称:UAPReadRequest(DeviceList) 包:0x00 0x00 | 0x83 | 0x?? | 0xff 帧域说明:UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1) 数据包 2: 协议层:ASL 包名称:ASLDataRequest(DstAddr, ServiceID, UAP_ID, Priority, AsduLength, Asdu) 包:0x04 | 0x00 | 0x00 0x?? | 0x00 0x00 | 0x83 |0x?? | 0xff 帧域说明:包控制(1)|UAP 标识符(1)|载荷长度(2)|UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)

8.1.27 C/S 通信测试(写属性)[GW-RUN027]

该测试用例测试网关设备 AL 能否正确发送写 MIB 属性请求。

测试过程为:

a) 被测设备加入网络后,被测设备运行 AMCL 状态机,向测试系统发送写(WRITE)请求,以写设备表 DeviceList 为例进行测试;

b) 测试系统将接收的写(WRITE)请求与期望的报文进行比对,如果比对匹配,则测试通过。

具体时序如图 90 所示,具体测试说明如表 106 所示。

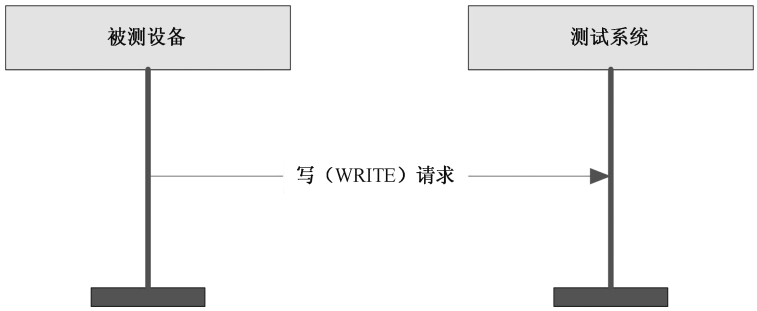

图 90 C/S 通信测试(写属性)时序图

表 106 C/S 通信测试(写属性)说明

用例名称	C/S 通信测试(写属性)[GW-RUN027]
被测设备	网关设备
依赖测试条件	被测设备已加入测试系统网络

测试用例伪代码描述	TEST BODY:
	RefPacket = UAPWriteRequest(DeviceList);
	WHILE(Receive(ASLDataRequest).PacketControl&0x1F! =0x02)forLossConnectDuration;
	IF(Receive(ASLDataRequest).PacketControl & 0x1F== 0x02)
	{
	IF (Verify(Receive(ASLDataRequest). Asdu,RefPacket1.all) ! = SUCCESS)
	{
	Printscreen("ALWrite MIB Request error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Printscreen("ALWrite MIB Request Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("NoALWrite MIB Request received!");
	TestCaseResult = FAILED;
	}

表 106（续）

测试用例伪代码描述	Printscreen(TestCaseResult)； TEST RESULT： SUCCESSor FAILED
参考数据包	测试系统应发包： 无
	测试系统应发包： 数据包 1： 协议层：UAP 包名称：UAPWriteRequest(DeviceList) 包：0x00 0x00 | 0x83 | 0x?? | 0xff | 0x??...0x?? 帧域说明：UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)数据(n) 数据包 2： 协议层：ASL 包名称：ASLDataRequest(DstAddr，ServiceID，UAP_ID，Priority，AsduLength，Asdu) 包：0x02 |0x00 | 0x?? 0x?? | 0x00 0x00 | 0x83 |0x?? | 0xff | 0x??...0x?? 帧域说明：包控制(1)|UAP 标识符(1)|载荷长度(2)|UAO 标识符(2)|属性标识符(1)|存储索引(1)|成员标识符(1)|数据(n)

8.2 离开过程测试集

8.2.1 现场设备被动离开网络[GW-QUIT001]

该测试用例测试网关设备能否正确发送设备离开请求。

测试过程为：

a） 测试系统加入被测设备网络后，被测设备向测试系统发送设备离开请求；

b） 测试系统将接收的设备离开请求报文与期望的报文进行比对，如果比对匹配，则测试通过，并向被测设备返回设备离开响应。

具体时序如图 91 所示，具体测试说明如表 107 所示。

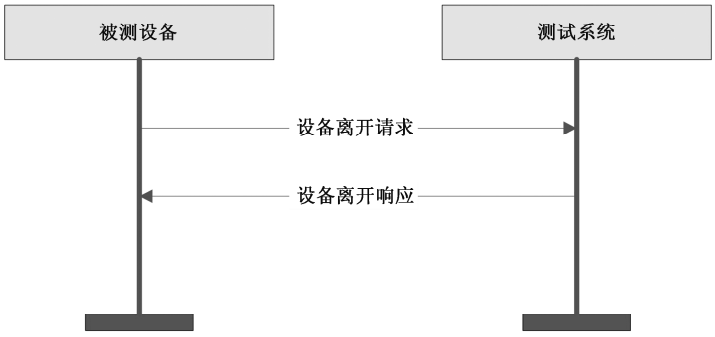

图 91 现场设备被动离开网络时序图

表 107　现场设备被动离开网络说明

用例名称	现场设备被动离开网络[GW-QUIT001]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测设备已完成对测试系统的资源配置

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredLeaveResponse;
	RefPacket =WiredLeaveRequest;
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (Verify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredLeave RequestPayload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	Send(SendPacket);
	Printscreen("Leaving Test Success!");
	TestCaseResult =SUCCESS;
	}
	}
	ELSE
	{
	Printscreen("No WiredLeave Requestreceived!");
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 协议层: Wired DLL 帧名称: WiredLeave Response 帧: 0x0f | 0x?? |0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x00 帧域说明:服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|AdID (1)|离开状态(1)
	测试系统应收数据包: 协议层: Wired DLL 帧名称: WiredLeave Request 帧: 0x0e | 0x?? | 0x00 0x01 | 0x?? 帧域说明: 服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(1)

8.2.2 接入设备被动离开网络[GW-QUIT002]

该测试用例测试网关设备能否正确发送接入设备离开请求。

测试过程为：

a) 测试系统(接入设备)加入被测设备网络后，被测设备通过有线向测试系统(接入设备)发送设备离开请求；

b) 测试系统将接收的设备离开请求报文与期望的报文进行比对，如果比对匹配，则测试通过，并向被测设备返回设备离开响应。

具体时序如图 92 所示，具体测试说明如表 108 所示。

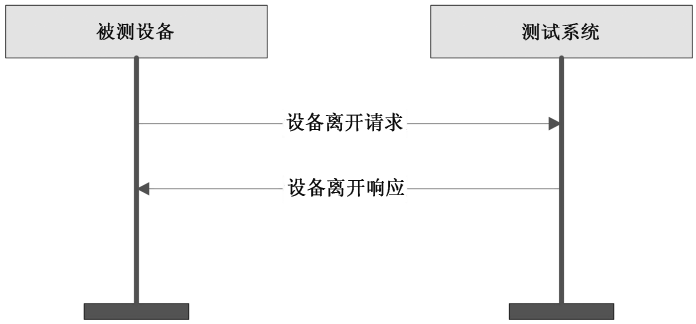

图 92 接入设备被动离开网络时序图

表 108 接入设备被动离开网络说明

用例名称	接入设备被动离开网络[GW-QUIT002]
被测设备	网关设备
依赖测试条件	测试系统已加入被测设备网络 被测设备已完成对测试系统的资源配置

测试用例伪代码描述	TEST BODY:
	SendPacket = WiredLeaveResponse;
	RefPacket =WiredLeaveRequest;
	WHILE(Receive(RcvPacket).type ! = RefPacket.type) for MaxWaitTime;
	IF(RcvPacket.type == RefPacket.type)
	{
	IF (wiredVerify(RcvPacket.all, RefPacket.all) ! = SUCCESS)
	{
	Printscreen("WiredLeave Request Payload error!");
	TestCaseResult =FAILED;
	}
	ELSE
	{
	wiredSend(SendPacket);
	Printscreen("Leaving Test Success!");
	TestCaseResult =SUCCESS;
	}
	}

表 108（续）

测试用例伪代码描述	ELSE
	{
	Printscreen(“No WiredLeave Request received!”);
	TestCaseResult = FAILED;
	}
	Printscreen(TestCaseResult);
	
	TEST RESULT:
	SUCCESS or FAILED

参考数据包	测试系统应发数据包: 协议层: Wired DLL 帧名称: WiredLeave Response 帧: 0x0f | 0x?? |0x03 | 0x?? 0x?? | 0x?? 0x?? | 0x?? | 0x00 帧域说明:服务标识符(1)|AdID(1)|目的地址(1)|包序号(2)|服务参数长度(2)|AdID (1)|离开状态(1)
	测试系统应收数据包: 协议层: Wired DLL 帧名称: WiredLeave Request 帧: 0x0e | 0x?? | 0x00 0x01 | 0x?? 帧域说明: 服务标识符(1)|AdID(1)|服务参数长度(2)|服务参数(1)

参 考 文 献

[1] IEEE 802.15.4:2011, Low-Rate Wireless Personal Area Networks (LR-WPANs).