XF AT 指令使用指南

文档版本：2.0.x

发布日期：2024-12-26

修改记录

文档版本	发布日期	修改说明
	2024-12-26	1. 修正 SLE 配置最大功率档位的描述信息。
	2024-12-24	1. 指令说明即示例中，增加显眼的 AT 数据通道（如串口）配置的指引 2. AT 配置指令描述中增加 json 处理可使用的外部工具 3. AT 配置指令各配置项中明确提供可直接使用的压缩描述与仅用于理解的格式化描述
2.0.x	2024-12-09	首次释放版本

前言

概述

本文介绍 XFusion 的 AT 指令及场景，为用户提供相应的指令格式和参数示例解释。

修改记录

指令说明

指令简介

> 如果没有特殊说明，AT 数据通道会默认配置为与固件下载的串口，配置项为波特率 115200，数据位 8，停止位 1，无校验位。
>
> **如需了解如何修改 AT 数据通道（如串口）配置，详情浏览：**
>
> 1. **"AT 配置指令" -> "AT 配置指令描述"**
> 2. **"AT 配置指令" -> "AT 配置指令描述" -> "AT 通用配置的配置描述说明" -> "AT 数据通道配置组"**

指令类型

AT 指令类型如表 1-1 所示

类型	格式	用途
测试指令	AT+<cmd>=?	用于查询设置指令 的参数以及取值范围等。
查询指令	AT+<cmd>?	用于信息的读取，一般是参数读取。
设置指令	AT+<cmd>=<parameter>, ...	用于设置参数值或执行。
执行指令	AT+<cmd>	用于执行本指令的功能。

注意事项

- 不是每一条指令都具备表 1-1 中的 4 种类型的命令。
- 如果存在当前软件版本不支持的 AT 指令，会返回 ERROR。
- 双引号表示字符串数据 "string"，例如：AT+SCANSSID="XXX"。
- 串口通信默认配置：波特率为 115200、8 个数据位、1 个停止位、无校验，无流控。
- <>为必选参数；[ ]为可选参数（可缺省）。
- 命令中的参数以“,”作为分隔符，除双引号括起来的字符串参数外，不支持参数本身带“,”。
- AT 指令中的参数不能有多余的空格。
- AT 指令必须大写，且必须以回车换行符作为结尾（CR LF），部分串口工具在用户敲击键盘回车键时只有回车符（CR）没有换行符（LF），导致 AT 指令无法识别，如需使用串口工具手动输入 AT 指令，需在串口工具中将回车键设置为回车符（CR）+换行符（LF）。
- 指令描述中的 `/**/` 类似C语言注释的标识，为注释说明，不是实际输出的字符。
- 指令描述中，返回响应时，"OK" 或 "ERROR" 前的部分结果可能会以 "INFO:" 或 "ERR:" 开头的行内容进行输出。"ERR:" 后为具体的错误内容；"INFO:" 后一般为提示内容。如 "INFO:ALREAD" 表示将要设置的配置或状态已经设置；"INFO:TIMEOUT" 表示执行超时提示；"ERR:TIMEOUT" 表示执行超时错误。

AT 基础指令

AT基础指令一览表

指令	描述
AT+HELP	查看当前可用 AT 命令。
AT+RST	重启设备指令。
AT+ECHO	回显设置指令。
AT+SWINFO	查询软件信息（sd版本、固件描述、作者、编译时间等）。

AT基础指令描述

AT+HELP - 查看当前可用 AT 命令

AT+HELP	
描述	查看当前可用 AT 命令
正常响应	`/* 显示当前支持的AT指令 */` OK
异常响应	ERROR
参数说明	-
示例	-
补充说明	-

AT+RST - 重启设备指令

AT+RST	
描述	重启设备
正常响应	`/* 进行重启 */`
异常响应	ERROR
参数说明	-
示例	`/* 发送的指令 */` AT+RST
补充说明	-

AT+ECHO - 回显设置指令

AT+ECHO=<state>	
描述	设置回显与否
响应	OK

AT+ECHO=<state>	
参数说明	state: 将要设置的回显状态

示例	`/* 发送的指令 */`
	AT+ECHO=1
	
	`/* 响应 */`
	OK

补充说明	默认回显不开启

AT+SWINFO - 查询软件信息

AT+SWINFO	
描述	对 GPIO 进行配置
正常响应	OK
异常响应	ERROR
参数说明	-

示例	`/* 发送的指令 */`
	AT+SWINFO
	
	`/* 响应 */`
	SDK Version:_
	SW Desc:
	SW ID:
	SW Create by:X
	Compiled:[Sep 6 2024]-[17:35:53]
	
	OK

补充说明	-

AT+GPIOCFG - GPIO配置指令

AT+GPIOCFG=<io_num>,<dir>,[pull_mode]	
描述	对 GPIO 进行配置
正常响应	OK
异常响应	ERROR
参数说明	io_num: 索引号（对应的值根据平台而定） dir: 方向

AT+GPIOCFG=<io_num>,<dir>,[pull_mode]	
	0: 输入 1: 输出（推挽） 2: 输出（开漏）（支持的情况取决于具体平台具体 IO） [pull_mode]: 上下拉模式（支持的情况取决于具体平台具体 IO） 0: 无上下拉（浮空） 1: 上拉 2: 下拉

示例	`/* 1. 配置 GPIO0 为输出 */`
	`/* 发送的指令 */`
	AT+GPIOCFG=0,1
	
	`/* 响应 */`
	OK

补充说明	-

AT+GPIOWR - GPIO 电平设置指令

AT+GPIOWR=<io_num>,<level>	
描述	GPIO 电平设置指令
正常响应	OK
异常响应	ERROR
参数说明	io_num: 索引号（对应的值根据平台而定） level: 设置的电平 0: 低电平 1: 高电平

示例	`/* 1. 设置 GPIO0 电平为高电平 */`
	`/* 发送的指令 */`
	AT+GPIOWR=0,1
	
	`/* 响应 */`
	OK

补充说明	-

AT+GPIORD - GPIO 电平读取指令

AT+GPIOWR=<io_num>	
描述	GPIO 电平读取指令
正常响应	`/* 显示 INFO:[读取到的电平值] */` OK
异常响应	ERROR
参数说明	io_num: 索引号（对应的值根据平台而定）

示例	`/* 1. 设置 GPIO0 电平为高电平 */`
	`/* 发送的指令 */`
	AT+GPIOWR=0,1
	
	`/* 响应 */`
	OK

补充说明	读取到的电平值： 0: 低电平 ·1:高电平

AT SLE 指令

AT SLE 指令一览表

指令	描述
AT+SLEENABLE	SLE 功能的开启、关闭与查询指令
AT+SLEOBJ	SLE 对象的创建或删除指令
AT+SLEADDR	SLE 本端地址的设置与查询指令
AT+SLENAME	SLE 本端设备名的设置与查询指令（全局）
AT+SLEADV	SLE 广播的开启与关闭指令
AT+SLEADVDATASET	SLE 广播数据的设置指令
AT+SLEADVDATAGET	SLE 广播数据的查询指令
AT+SLECONNSET	SLE 连接的设置指令
AT+SLECONNGET	SLE 连接的查询指令
AT+SLECONNBYNAME	SLE 尝试连接指定设备名的设备

指令	描述
AT+SLESCAN	SLE 扫描的设置指令
AT+SLESEND	SLE 数据发送指令
AT+SLERECV	SLE 数据接收指令

AT SLE 指令描述

AT+SLEENABLE - SLE 功能状态的设置（开或关）与查询

AT+SLEENABLE?	
描述	SLE 功能状态的查询
正常响应	`/* 显示 SLE 功能状态 */` OK
异常响应	ERROR
参数说明	-

示例	`/* 发送的指令 */`
	AT+SLEENABLE?
	
	`/* 响应 */`
	0
	
	OK

补充说明	-

AT+SLEENABLE=<state>	
描述	SLE 功能状态的设置（开或关）
正常响应	`/* 显示 SLE 功能状态 */` OK
异常响应	ERROR
参数说明	state: 将要设置的SLE 状态 1: 开启 SLE 功能 0: 关闭 SLE 功能
示例	`/* 发送的指令 */` AT+SLEENABLE=1

AT+SLEENABLE=<state>	
	/* 响应 */ OK
补充说明	-

AT+SLEOBJ - SLE 对象的设置（创建或删除）指令

AT+SLEOBJ=<cmd_type>,<arg>	
描述	SLE 对象的设置（创建或删除）指令
正常响应	/* 创建：显示分配到的 SLE OBJ 的 ID，如 `ID:0` */ /* 其他：无其他显示 */ OK
异常响应	/* 部分错误会显示具体错误原因，参考补充说明 */ ERROR
参数说明	cmd_type: 执行的命令类型 -1: 删除所有 SLE 对象 0: 定向删除 SLE 对象（指定 OBJ ID） 1: 创建指定 SLE 连接角色的对象 arg: 命令为删除所有时: 该参数可忽略不填 命令为定向删除时: 指定的 OBJ ID 命令为创建对象时: 指定的连接角色: 0: Slave，从机，可进行广播开启与关闭 1: Master，主机，可进行广播扫描、广播连接
示例	/* 1. 删除所有 SLE 对象 */ /* 发送的指令 */ AT+SLEOBJ=-1 /* 响应 */ OK /* 2. 定向删除 SLE 对象 */ /* 发送的指令 */ AT+SLEOBJ=0,0

AT+SLEOBJ=<cmd_type>,<arg>	

	`/* 响应 */`
	OK
	
	`/* 3. 创建 SLE 对象 */`
	`/* 发送的指令 */`
	AT+SLEOBJ=1,0
	
	`/* 响应 */`
	ID:0
	
	OK

补充说明	会显示具体错误原因的有： "ERR:MAX CNT [最大值]"：出现在创建时，表示当前 SLE OBJ 数量已经达到最大值 "ERR:INVALID ID"：出现在定向删除时，表示指定 ID 的无效

AT+SLEADDR - SLE 本端地址的设置与查询指令

AT+SLEADDR=<obj_id>,[addr],[type]	
描述	SLE 本端地址的设置与查询指令
正常响应	`/* 获取地址时：显示 SLE OBJ 的本端地址类型及地址，如 "0,00:11,22:33:44:55" */` OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	obj_id: 指定的 SLE OBJ 的 ID： 当只填入 1 个参数时，表示查询该 OBJ 的本端地址 当填入 2 个及以上的参数时，表示设置该 OBJ 的本端地址 [addr]: 地址值，以数组的形式传入，如001122334455 为查询指令时，该参数缺省 [type]: 地址类型 0: 公有地址 6: 私有地址 该参数可缺省，表示以默认地址类型（公有地址）设置 OBJ 的本端地址

AT+SLEADDR=<obj_id>,[addr],[type]	
示例	/* 1. 获取地址 */ /* 发送的指令 */ AT+SLEADDR=1 /* 响应 */ 0,00:11:22:33:44:55 OK /* 2. 设置地址。地址类型不填入时，以默认类型公有地址类型进行设置 */ /* 发送的指令 */ AT+SLEADDR=0,001122334455 /* 响应 */ OK
补充说明	会显示具体错误原因的有： "ERR:INVALID ID"：表示指定 ID 的无效

AT+SLENAME - SLE 本端设备名的设置与查询指令（全局）

AT+SLENAME?	
描述	SLE 本端设备名的查询指令（全局）
正常响应	/* 获取地址时：显示 SLE 本端设备名 */ OK
异常响应	ERROR
参数说明	-
示例	/* 发送的指令 */ AT+SLENAME? /* 响应 */ XF_SLE OK
补充说明	-

AT+SLENAME=<name>	
描述	SLE 本端设备名的设置指令（全局）
正常响应	OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	name: 指定设置本端设备名，该参数需要以字符串形式填入

示例	`/* 发送的指令 */`
	AT+SLENAME="XF_SLE"
	
	`/* 响应 */`
	OK

补充说明	该指令设置的本端设备名将会作用于所有 SLE OBJ，设备名默认情况下都将会加入到广播数据中（无论是否调用该指令） 会显示具体错误原因的有： "ERR:MAX LEN:[最大值]"：表示填入的字符串超出长度最大值

AT+SLEADV - SLE 广播的开启与关闭指令

AT+SLEADV=<obj_id>,<state>	
描述	SLE 本端设备名的查询指令
正常响应	`/* 获取地址时：显示 SLE 本端设备名 */` OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	obj_id: 指定的 SLE OBJ 的 ID state: 将要设置广播的状态: 0: 关闭 1: 开启

示例	`/* 发送的指令 */`
	AT+SLEADV=0,1
	
	`/* 响应 */`
	OK

AT+SLEADV=<obj_id>,<state>	
补充说明	"ERR:INVALID ID": 表示 OBJ ID 无效 "ERR:INVALID ROLE": 表示 OBJ 的角色无效，只有从机角色才可以进行广播

AT+SLEADVDATASET - SLE 广播数据的设置指令

AT+SLEADVDATASET=<obj_id>,<type>,<data_len>,[data]	
描述	SLE 本广播数据的设置指令（增加、修改或删除）
正常响应	OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	obj_id: 指定的 SLE OBJ 的 ID type: 广播数据的类型（详参 Xfusion SLE 抽象层 `xf_sle_adv_struct_type_t`） data_len: 广播数据的长度（仅计算广播数据主体的长度即可） 当广播数据长度为 0 时，表示移除该类型的广播数据 当广播数据长度不为 0 时，表示增加或修改该类型的广播数据 [data]: 广播数据主体 当 data_len 为 0，即将要移除指定类型广播数据时，该参数可缺省 当广播数据长度不为 0 时，需要填入要设置的广播数据，注意参数需为 16 进制数组形式: 如果是字符串类型的数据则需要先转成 16 进制数后再填入 如果是普通数值，则需表达为 两个（八位）字节的16 进制数传入，如想传入1时，填入的是01

示例	`/* 设置类型为 1 （发现等级）的广播数据 为 1 */`
	`/* 发送的指令 */`
	AT+SLEADVDATASET=0,1,1,01
	
	`/* 响应 */`
	OK

补充说明	"ERR:INVALID ID": 表示 OBJ ID 无效 "ERR:INVALID ROLE": 表示 OBJ 的角色无效，只有从机角色才可以进行广播 "ERR:INVALID ARG CNT": 表示参数个数无效，至少需要3个参数 "ERR:INVALID ARG": 表示参数无效，一般是只填入3个参数时，表示为删除指定的广播数据，第三个参数 data_len 需要为0

AT+SLEADVDATAGET - SLE 广播数据的查询指令

AT+SLEADVDATASET=<obj_id>	
描述	SLE 广播数据的查询指令
正常响应	`/* 输出设置的广播数据，一行显示一项广播数据，形式为 "type:[类型值],len:[数据长度],[数据]" */` OK

AT+SLEADVDATASET=<obj_id>	
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	obj_id: 指定的 SLE OBJ 的 ID

示例	`/* 发送的指令 */`
	AT+SLEADVDATAGET=0
	
	`/* 响应 */`
	type:1,len:1,┌ `/* 此处输出的广播数据为整形数值，所以在串口工具上显示为该数值对应的 ASCII 符号 */`
	type:11,len:6,XF_SLE
	
	OK

补充说明	"ERR:INVALID ID": 表示 OBJ ID 无效 "ERR:INVALID ROLE": 表示 OBJ 的角色无效，只有从机角色才可以进行广播

AT+SLECONNSET - SLE 连接的设置指令

AT+SLECONNSET=<cmd_type>,[obj_id],[addr],[addr_type]	
描述	SLE 连接的设置指令（定向创建、定向删除与删除所有）
正常响应	`/* 定向创建命令下，如果检测到已经建立连接时，则会报 "INFO:ALREADY" */` OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	cmd_type: 执行的命令类型 -1: 删除所有 SLE 连接 0: 定向删除 SLE 连接（指定 OBJ ID） 1: 创建指定 SLE 连接（指定 OBJ ID） 当命令类型为删除所有 SLE 连接时，后面的参数可缺省 [obj_id]: 指定的 SLE OBJ 的 ID 当命令类型为定向删除 SLE 连接时，后面的参数可缺省 [addr]: 创建指定 SLE 连接时，指定连接的地址 后面的参数可缺省，将会按默认地址类型（公有地址）进行连接 [addr_type]: 创建指定 SLE 连接时，指定连接的地址的类型
示例	`/* 发送的指令 */` AT+SLECONNSET=1,0,CACACACACACA

AT+SLECONNSET=<cmd_type>,[obj_id],[addr],[addr_type]	
	`/* 响应 */` OK
补充说明	"ERR:INVALID ID": 表示 OBJ ID 无效 "ERR:INVALID ARG CNT": 表示参数个数无效 "ERR:INVALID ROLE": 表示 OBJ 的角色无效。一般是创建连接的命令的报错，只有主机角色才可以进行主动连接 "ERR:FIND SVC FAILED": 表示发现服务结构失败

AT+SLECONNGET - SLE 连接的查询指令

AT+SLECONNGET=<obj_id>	
描述	SLE 连接的查询指令
正常响应	`/* 显示查询的 OBJ 的连接信息，形式如："0,AA:BB:CC:DD:EE:FF" */` OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	[obj_id]: 指定的 SLE OBJ 的 ID

示例	`/* 发送的指令 */`
	AT+SLECONNSET=1,0,CACACACACACA
	
	`/* 响应 */`
	OK

补充说明	"ERR:INVALID ID": 表示 OBJ ID 无效

AT+SLECONNBYNAME - SLE 尝试连接指定设备名的设备

AT+SLECONNBYNAME=<obj_id>,<peer_name>,<timeout_ms>	
描述	SLE 尝试连接指定设备名的设备
正常响应	`/* 定向创建命令下，如果检测到已经建立连接时，则会报 "INFO:ALREADY" */` OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	obj_id: 指定的 SLE OBJ 的 ID peer_name: 指定连接的设备名（将从广播数据中筛选设备名类型的数据）

AT+SLECONNBYNAME=<obj_id>,<peer_name>,<timeout_ms>	
	timeout_ms: 执行指令的最大超时时间，单位 ms

示例	`/* 发送的指令 */`
	AT+SLECONNBYNAME=0,"XF_SLE",3000
	
	`/* 响应 */`
	OK

补充说明	"ERR:INVALID ID": 表示 OBJ ID 无效 "ERR:INVALID ROLE": 表示 OBJ 的角色无效，只有主机角色才可以进行主动连接 "ERR:TIMEOUT:[超时时间] ms": 表示执行指令超时

AT+SLESCAN - SLE 扫描的设置指令

AT+SLESCAN=<obj_id>,<cnt_max>,<timeout_ms>	
描述	SLE 尝试连接指定设备名的设备

正常响应	`/* 显示扫描到的项，形式如 "RSSI:-60,  addr:0,AA:BB:CC:DD:EE:FF"*/`
	`/* 达到扫描最大显示项数时，输出 "CMPL"，达到超时时间时，输出 "TIMEOUT" */`
	
	OK

异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	obj_id: 指定的 SLE OBJ 的 ID cnt_max: 指定扫描结果最大显示项数 timeout_ms: 执行指令的最大超时时间，单位 ms

示例	`/* 发送的指令 */`
	AT+SLESCAN=0,1,3000
	
	`/* 响应 */`
	RSSI:-44，addr:CA:CA:CA:CA:CA:CA
	INFO:CMPL
	
	OK

补充说明	"ERR:INVALID ID": 表示 OBJ ID 无效 "ERR:INVALID CNT": 表示 cnt_max 无效，需要大于 0 "ERR:INVALID ROLE": 表示 OBJ 的角色无效，只有主机角色才可以进行主动连接

AT+SLESEND - SLE 数据发送指令

AT+SLESEND=<obj_id>,<type>,<data_size_max>,<idle_ms_max>,[dest_addr]	
描述	SLE 数据发送指令
正常响应	`/* 传入到AT通道的数据达到最大可发送的数据量时，输出 "INFO:CMPL"，达到超时时间时，输出 "INFO:TIMEOUT" */` OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	obj_id: 指定的 SLE OBJ 的 ID type: 指令发送的类型 0: 直接发送 data_size_max: 最大可发送的数据量 idle_ms_max: 最大可等待的空闲时间，单位 ms。每当有数据传入 AT 数据（读）通道时，空闲计时将会重新计时 [dest_addr]: 目标地址（可缺省）

示例	`/* 1. 发送的指令 */`
	AT+SLESEND=0,0,8,5000
	
	`/* 2. 直接传入要发送的数据 */`
	12345678
	
	`/* 3. 响应 */`
	INFO:CMPL,size:8
	
	OK

补充说明	SLE 数据发送的执行流程： 1. 发送 +SLESEND 指令 2. 其后直接向 AT 数据（读）通道传入数据 3.A. 当传入 AT 数据（读）通道的数据量到达最大可发送数据量时，退出指令阻塞状态，输出 "INFO:CMPL,size:" 提示，"size:"后紧跟着实际发送的数据量 3.B. 当 AT 数据（读）通道 空闲时间达到最大可等待的空闲时间时，退出指令阻塞状态，输出 "INFO:TIMEOUT,size" 提示，"size:"后紧跟着实际发送的数据量 "ERR:INVALID ID": 表示 OBJ ID 无效 "ERR:INVALID ARG CNT": 表示参数个数无效，需要至少 4 个参数

AT+SLERECV - SLE 数据接收指令

AT+SLERECV=<obj_id>,<type>,<data_size_max>,<idle_ms_max>,<repeat_cnt>,[src_addr]	
描述	SLE 数据接收指令
正常响应	`/* 当从 AT 写通道读出的数据达到最大可发送的数据量时，输出 "INFO:CMPL"，达到超时时间时，输出 "INFO:TIMEOUT,size:6" */` OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	obj_id: 指定的 SLE OBJ 的 ID type: 指令接收的类型 0: 直接接收 data_size_max: 最大可接收的数据量 0: 不定长接收（不以接收数据量作为指令退出标志） 0<: 定长接收 idle_ms_max: 最大可等待的空闲时间，单位 ms。每当有数据从 AT 数据（写）通道传出时，空闲计时将会重新计时 0>: 负值空闲时间表示当本次接收到至少一次有效数据时，才开启空闲计时功能 0: 不定时，关闭空闲计时功能 0<: 负值空闲时间表示立即开启空闲计时功能，当出现接收空闲立即计时 repeat_cnt: 自动重复执行指令的次数，当前版本固定为1 [src_addr]: 源地址（可缺省）

示例	`/* 1. 发送的指令 */`
	AT+SLERECV=0,0,8,5000,1
	
	`/* 2. 输出接收到的数据 */`
	12345678
	
	`/* 3. 响应 */`
	INFO:CMPL,size:8
	
	OK

补充说明	SLE 数据接收的执行流程： 1. 发送 +SLERECV 指令 2. 其后不断尝试从 AT （写）通道中读取接收到的数据 3.A. 当读出的数据量到达最大可接收数据量时，退出指令阻塞状态，输出 "INFO:CMPL,size:" 提示，"size:"后紧跟着实际接收到的数据量 3.B. 当 AT 数据（写）通道 空闲时间达到最大可等待的空闲时间时，退出指令阻塞状态，输出 "INFO:TIMEOUT,size:" 提示，"size:"后紧跟着实际接收到的数据量

AT+SLERECV=<obj_id>,<type>,<data_size_max>,<idle_ms_max>,<repeat_cnt>,[src_addr]	
	"ERR:INVALID ID": 表示 OBJ ID 无效 "ERR:INVALID ARG CNT": 表示参数个数无效，需要至少 5 个参数

AT 测试指令

AT 测试指令一览表

指令	描述
AT+SLETEST	SLE 快速测试指令

AT 测试指令描述

AT+SLETEST - SLE 快速测试指令（建议先浏览本节下的"注意事项"）

AT+SLETEST=<option>,[delay_ms],[pkt_size],[pkt_nums],[repeat_time]	
描述	SLE 快速测试指令
正常响应	`/* 主机侧连上从机后将会打印一系列测试信息（如测试子项信息），最终所有子项测试完成后，将会打印 "TEST CMPL(ALL):[已测试的子项数]" */` OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	option：测试选项 0：从机（服务端） 1：主机（客户端）（测试主体） -1：清除所有测试配置缓存（通过"CFG"指令配置的） [delay_ms]： 测试子项间的延迟时间，单位 ms [pkt_size]： 测试子项传输的每包大小（字节） [pkt_nums]： 测试子项传输的总包数 [repeat_time] 测试子项重复测试的次数（注意这并不是测试子项数，仅为一个测试子项中重复测试的次数）

AT+SLETEST=<option>,[delay_ms],[pkt_size],[pkt_nums],[repeat_time]	
示例	/*** 以下为主机侧测试（主动发起）示例 ***/ /* 发送的指令：配置指令，最大可接收128字节，最大等待传入数据的空闲时间为2000ms */ AT+CFG=128,2000 /* 传入自定义测试配置项：mtu.size 中有两个测试值，256 与 512，将分别进行测试 */ {"sletest":{"mtu":{"size":[256,512]}}} /* 响应：接收到传入的测试配置项数据，且解析正常 */ INFO:TIMEOUT,size:38 OK /* 发送的指令：SLE 测试，角色:主机，测试子项间不延时，包大小:236字节，包数:500，重复次数:2 */ AT+SLETEST=1,0,236,500,2 /* 响应：主机侧测试子项（每轮测试）的信息，包大小:236字节，包数:1000，重复测试次数:1 */ INFO:pkt size:236, pkt num: 1000, repeat:1 /* 第1个测试子项（第1轮测试）的结果 */ RESULT(SUB):times:1,pkt:236,nums:1000 Time:2301399 us 820370.56 bps 801.14 Kibps 0.78 Mibps 102546.32 Bps 100.14 KiBps 0.9 MiBps INFO:TEST CMPL(SUB):1 /* 第2个测试子项（第2轮测试）的结果 */ RESULT(SUB):times:1,pkt:236,nums:1000 Time:2294637 us 822788.16 bps 803.50 Kibps 0.78 Mibps 102848.52 Bps 100.43 KiBps 0.9 MiBps INFO:TEST CMPL(SUB):2 /* 表示所有子项测试完成 */ INFO:TEST CMPL(ALL):2 OK /*** 以下为从机侧测试（被动触发）示例 ***/

AT+SLETEST=<option>,[delay_ms],[pkt_size],[pkt_nums],[repeat_time]	

	`/* 发送的指令：SLE 测试，角色:从机 */`
	AT+SLETEST=0
	
	INFO:TEST CMPL(SUB):1
	
	INFO:TEST CMPL(SUB):2
	`/* 表示所有子项测试完成 */`
	INFO:TEST CMPL(ALL):2
	
	OK

补充说明	发起测试的流程: 从机（服务端）侧： 1. 开启 SLE 功能：AT+SLEENABLE=1 2. 启动从机测试程序：AT+SLETEST=0 3. 等待主机端连入后，进行测试信息同步及测试流程的执行 主机（客户端）侧：（发起测试请求的主体） 1. 开启 SLE 功能：AT+SLEENABLE=1 2. （可选）通过 "AT+CFG" 指令传入测试配置项的信息，进行自定义测试。 3. 启动主机测试程序：AT+SLETEST=1，主机将会尝试扫描连接从机，连接后以默认配置发起测试请求（可根据指令描述自定义测试的信息）。 4. 测试过程及结束时将会打印相关测试信息 报错说明： "ERR:TEST ABORT:[已测试的子项数]": 表示测试意外终止，通常为测试配置加载失败的问题 "ERR:RET:[错误码值]": 为测试流程中的错误，详情参考 XF AT 的 `sle_test_state_t` "ERR:INVALID ARG CNT": 表示参数个数无效，至少需要 2 个参数 "ERR:MAX CNT [最大值]"：出现在创建时，表示当前 SLE OBJ 数量已经达到最大值 更多补充说明请看本节的"注意事项"

注意事项

1. 测试主体是主机，测试请求将由主机发起，从机被动接收请求，其后同步信息（同步方向为：主机->从机），最后发起测试（测试子项的数据流向为：从机->主机）
2. 测试程序在开始测试前会进行从主机（客户端）到从机（服务端）的信息同步（所有配置、测试配置，测试子项信息，如包大小、包数等）
3. 测试配置项按类似C代码中的结构体进行划分，分为**测试配置组**与**测试配置成员**，配置组下包含多个配置成员，每个配置成员可能有一或多个**测试值。**

4. 每次整体正式测试前，都会先加载、更新一次所有测试配置组的所有测试配置成员的第0个测试值。
5. 测试程序会自动根据当前测试配置缓存，遍历所有测试值，每次只加载、更新一个测试值，然后进行一轮测试，每轮的测试又称**测试子项**。
6. 测试值遍历时，将会按测试配置传入时的顺序，逆向进行加载，而对应每个测试配置成员的测试值，将按值索引递增加载。

如当前传入配置的 json 为：mtu.size 2个测试值；mcs.mcs 2个测试值；payload_max.size 3个测试值

```
{
    "sletest": {
        "mtu": {
            "size": [
                512,
                256
            ]
        },
        "mcs": {
            "mcs": [
                5,
                10
            ]
        },
        "payload_max": {
            "size": [
                236,
                400,
                512
            ]
        }
    }
}
```

那自动测试遍历时，每轮的测试（测试子项）中各项测试配置值的索引及变更将如下：

测试子项数	mtu.size	mcs.mcs	payload_max.size
0	0	0	0
1	0	0	1
2	0	0	2

测试子项数	mtu.size	mcs.mcs	payload_max.size
3	0	1	0
4	0	1	1
5	0	1	2
6	1	0	0
7	1	0	1
8	1	0	2
9	1	1	0
10	1	1	1
11	1	1	2

那自动测试遍历时，每轮的测试（测试子项）中各项测试配置值及变更将如下：

测试子项数	mtu.size	mcs.mcs	payload_max.size
0	512	5	236
1	512	5	400
2	512	5	512
3	512	10	236
4	512	10	400
5	512	10	512
6	256	5	236
7	256	5	400
8	256	5	512
9	256	10	236
10	256	10	400
11	256	10	512

7. 从机（服务端）每次测试前都会从主机侧同步测试配置信息，并进行缓存，测试完成后自动销毁测试配置缓存。
8. 主机（客户端）的测试配置会进行缓存（如果通过"AT+CFG"传入了配置），直至掉电或主动调用清除测试配置缓存指令"AT+SLETEST=-1"。
9. 测试配置传入时，为增量更新，即缓存中存在相同值则忽略，不存在时新增值缓存。
10. 通常情况下，仅对主机（测试主体）传入测试配置，对从机传入测试配置将无效。

11. 如果测试出现意外，停滞不前的情况时，如显示长时间停留在更新连接参数，可复位重启再来。

AT 配置指令

AT 配置指令一览表

指令	描述
AT+CFG	AT 通用配置的设置指令

AT 配置指令描述

AT+CFG - AT 通用配置的设置指令

AT+CFG=<cfg_size_max>,<idle_ms_max>,[alt_val]	
描述	AT 通用配置的设置指令
正常响应	`/* 当从 AT 写通道读出的数据达到最大可发送的数据量时，输出 "INFO:CMPL"，达到超时时间时，输出 "INFO:TIMEOUT,size:6" */` OK
异常响应	`/* 部分错误会显示具体错误原因，参考补充说明 */` ERROR
参数说明	cfg_size_max: 将要传入的配置描述（ JSON 字符串）的最大大小，需按照将传入的配置描述实际大小进行设置，不可小于实际大小。 idle_ms_max: 最大可等待的空闲时间，单位 ms。每当有数据传入 AT 数据（读）通道时，空闲计时将会重新计时 [alt_val]: 配置描述项的替代值： 有需要时，可替代配置描述中某项允许替代的参数进行填入，这样可避免需要频繁修改某项的值（如 SLE 配置组中部分配置的 "sle_obj_id"配置项，在生成配置描述（JSON）时 此项可缺省，以表示使用替代值，然后通过本指令参数 alt_val 传入即可）

示例	`/* 发送的指令 */`
	AT+CFG=40,3000
	
	`/* 2. 直接传入配置描述的数据 */`
	{"sle":{"tx_pwr":10}}
	
	`/* 响应 */`

AT+CFG=<cfg_size_max>,<idle_ms_max>,[alt_val]	
	INFO:TIMEOUT,size:21 OK
补充说明	通用配置流程: 1. 发送 +CFG 指令 2. 其后直接向 AT 数据（读）通道传入配置描述的数据 3.A. 当传入 AT 数据（读）通道的数据量到达最大可发送数据量时，退出指令阻塞状态，输出 "CMPL" 提示 3.B. 当 AT 数据（读）通道 空闲时间达到最大可等待的空闲时间时，退出指令阻塞状态，输出 "TIMEOUT,size:[实际获取到的大小]" 提示 "ERR:INVALID ARG CNT": 表示参数个数无效，至少需要 2 个参数 "ERR:INVALID SIZE": 表示传入的 SIZE 参数无效 "ERR:PARSE ERR": 表示配置描述解析错误 "ERR:NOT SUPPORTED": 表示不支持 "ERR:INVALID SLE OBJ ID": 表示无效的 SLE OBJ ID

AT 通用配置的配置描述说明：

> 注意：
>
> 1. 以下配置中的配置项均不是必填项。缺省时，会按默认或上一次配置值进行配置。
> 2. **配置描述（JSON）传入时，需要去除所有注释等无效数据部分，并且 JSON 需要进行压缩，去除换行与空格。**

注意事项点 ② 中提及的JSON处理可使用外部工具：

json 处理工具：

- https://www.bejson.com/json/format
- https://www.json.cn/jsonzip/

注意，最好先点击格式化或格式化检验按钮对配置描述的数据进行格式校验（校验正确后再点压缩）

AT 数据通道配置组

AT 数据通道配置

配置 JSON 如下（当前版本仅支持配置为串口）：

如“AT 通用配置的配置描述说明”所言，可以只填入需要配置（修改）的项。

- 如果只是改变波特率不改其他参数（如串口号、IO等）的话，可以如下：
  `{"at_port":{"uart_cfg":{"baud":256000}}}` 当前使用的串口改波特率为 256000
- 如果想改变串口时，如果串口功能对应的 IO 引脚是固定（不能随意映射）的，则还需要填入对应的 IO 号，如下：
  - WS63 串口0（固定引脚）：`{"at_port":{"uart_cfg":{"num":0,"baud":115200,"io_tx":17,"io_rx":18}}}`
  - WS63 串口1（固定引脚）：`{"at_port":{"uart_cfg":{"num":1,"baud":115200,"io_tx":15,"io_rx":16}}}`

可配置项的压缩描述（可直接发送，能被解析）

```
{"at_port":{"uart_cfg":
{"num":1,"baud":115200,"data_bit":8,"stop_bit":1,"parity_bit":1,"io_tx":15,"io_r
x":16,"io_rts":-1,"io_dsr":-1}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "at_port":                          //  表示是 AT 数据通道配置组
    {
        "uart_cfg":                         // 表示配置为串口
        {                                   // 配置项:
            "num": 1,                           // 串口索引（号）
            "baud": 115200,                     // 波特率
            "data_bit": 8,                      // 数据位
            "stop_bit": 1,                      // 停止位
            "parity_bit": 1,                    // 校验位
            "io_tx": 15,                        // TX IO 号
            "io_rx": 16,                        // RX IO 号
            "io_rts": -1,                       // RTS IO 号
            "io_dsr": -1                        // DSR IO 号
        }
    }
}
```

AT SLE 配置组

功率配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sle":{"pwr":10}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                       // AT 配置描述的 ROOT
    "sle":                  //  表示是 SLE 配置组
    {
        "pwr":10                // 表示配置功率
    }
}
```

最大功率档位配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sle":{"max_pwr_level":{"pwr":20}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                       // AT 配置描述的 ROOT
    "sle":                  //  表示是 SLE 配置组
    {
        "max_pwr_level": {      // 表示是 SLE 最大功率档位的配置
            "pwr": 20               // 目标最大功率 (dBm)
        }
    }
}
```

MTU 配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sle":{"mtu":{"size":236,"version":1}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                               // AT 配置描述的 ROOT
    "sle":                          //  表示是 SLE 配置组
    {
        "mtu":                      // 表示是 SLE MTU 配置
        {                           // 配置项:
            "size": 236,                // MTU 大小
            "version": 1                // [版本] 可忽略
        }
    }
}
```

默认连接参数的配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sle":{"conn_param_def":
{"en_filter":true,"scan_phy":1,"en_gt_negotiate":true,"scan_intv":100,"scan_wind
ow":100,"conn_intv_min":100,"conn_intv_max":100,"conn_timeout":100}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sle":                              //  表示是 SLE 配置组
    {
        "conn_param_def":               // 表示是默认连接参数的配置
        {                               // 配置项:
            "en_filter": true,              // 是否开启过滤功能
            "scan_phy": 1,                  // 扫描频宽: 1:1M; 2:2M
            "en_gt_negotiate": true,        // 链路建立时是否进行 G 和 T 协商交
互
            "scan_intv": 100,               // 扫描间隔
            "scan_window": 100,             // 扫描窗口
            "conn_intv_min": 100,           // 连接（链路调度）间隔最小值
            "conn_intv_max": 100,           // 连接（链路调度）间隔最大值
```

```
            "conn_timeout": 100             // 连接（链路）超时时间
        }
    }
}
```

连接参数更新的配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sle":{"conn_param_upd":
{"sle_obj_id":0,"intv_min":100,"intv_max":100,"latency":100,"timeout":100}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sle":                              //  表示是 SLE 配置组
    {
        "conn_param_upd":                   // 表示是连接参数更新的配置
        {                                   // 配置项:
            "sle_obj_id": 0,                    // SLE OBJ ID
            "intv_min": 100,                    // 连接（链路调度）间隔最小值，单位
slot
            "intv_max": 100,                    // 连接（链路调度）间隔最大值，单位
slot
            "latency":100,                      // 连接（链路）延迟（休眠）时间，单位
slot
            "timeout": 100                      // 连接（链路）超时时间，单位 10 ms
        }
    }
}
```

PHY 配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sle":{"phy":
{"tx_fmt":1,"rx_fmt":1,"tx_phy":1,"rx_phy":1,"tx_pilot_density":1,"rx_pilot_dens
ity":1,"g_feedback":1,"t_feedback":1}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sle":                              //  表示是 SLE 配置组
    {
        "phy":                          // 表示是 PHY 配置
        {                               // 配置项:
            "tx_fmt": 1,                    // 发送无线帧类型 （详参 "SLE 无线帧类型"）
            "rx_fmt": 1,                    // 接收无线帧类型 （详参 "SLE 无线帧类型"）
            "tx_phy": 1,                    // 发送 PHY: 0:1M; 1:2M; 2:4M
            "rx_phy": 1,                    // 接收 PHY: 0:1M; 1:2M; 2:4M
            "tx_pilot_density": 1,          // 发送导频密度指示 （详参 "SLE 导频密度指示"）
            "rx_pilot_density": 1,          // 接收导频密度指示 （详参 "SLE 导频密度指示"）
            "g_feedback": 1,                // 先发链路反馈类型指示，取值范围 0-63
            "t_feedback": 1             // 后发链路反馈类型指示，取值范围 0-7
        }
    }
}
```

SLE 无线帧类型

```
    SLE_RADIO_FRAME_1    = 0,      /*!< 无线帧类型1 */
    SLE_RADIO_FRAME_2    = 1,      /*!< 无线帧类型2 */
    SLE_RADIO_FRAME_3_M0 = 2,      /*!< 无线帧类型3，m序列0 */
    SLE_RADIO_FRAME_3_M1 = 3,      /*!< 无线帧类型3，m序列1 */
    SLE_RADIO_FRAME_3_M2 = 4,      /*!< 无线帧类型3，m序列2 */
    SLE_RADIO_FRAME_3_M3 = 5,      /*!< 无线帧类型3，m序列3 */
    SLE_RADIO_FRAME_3_M4 = 6,      /*!< 无线帧类型3，m序列4 */
    SLE_RADIO_FRAME_3_M5 = 7,      /*!< 无线帧类型3，m序列5 */
    SLE_RADIO_FRAME_4_M0 = 8,      /*!< 无线帧类型4，m序列0 */
    SLE_RADIO_FRAME_4_M1 = 9,      /*!< 无线帧类型4，m序列1 */
    SLE_RADIO_FRAME_4_M2 = 10,     /*!< 无线帧类型4，m序列2 */
    SLE_RADIO_FRAME_4_M3 = 11,     /*!< 无线帧类型4，m序列3 */
    SLE_RADIO_FRAME_4_M4 = 12,     /*!< 无线帧类型4，m序列4 */
    SLE_RADIO_FRAME_4_M5 = 13,     /*!< 无线帧类型4，m序列5 */
```

SLE 导频密度指示

```
    SLE_PHY_PILOT_DENSITY_4_TO_1  = 0x0,    /*!< 导频密度为4:1 */
    SLE_PHY_PILOT_DENSITY_8_TO_1  = 0x1,    /*!< 导频密度为8:1 */
    SLE_PHY_PILOT_DENSITY_16_TO_1 = 0x2,    /*!< 导频密度为16:1 */
```

广播参数配置（配置完成后需要在下一次开启广播时生效）

可配置项的压缩描述（可直接发送，能被解析）

```
{"sle":{"adv_param":
{"sle_obj_id":0,"adv_type":1,"adv_gt_role":1,"adv_level":1,"adv_intv_min":100,"a
dv_intv_max":100,"adv_ch_map":1,"adv_tx_pwr":1,"conn_intv_min":1,"conn_intv_max"
:1,"conn_latency_max":1,"conn_timeout":1}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                       // AT 配置描述的 ROOT
    "sle":                                  //  表示是 SLE 配置组
    {
        "adv_param":                            // 表示是 广播参数 的配置
        {                                       // 配置项:
            "sle_obj_id": 0,                        // SLE OBJ ID
            "adv_type": 1,                          // 广播类型（详参: "SLE 广播类型"）
            "adv_gt_role": 1,                       // 广播 G/T 角色协商（详参: "G/T 角
色协商指示类型"）
            "adv_level": 1,                         // 广播（被发现）级别（详参: "SLE 广
播（被发现）级别"）
            "adv_intv_min": 100,                    // 广播间隔最小值，取值范围
[0x000020~0xffffff], 单位125us
            "adv_intv_max": 100,                    // 广播间隔最大值，取值范围
[0x000020~0xffffff], 单位125us
            "adv_ch_map": 1,                        // 广播信道 （详参: "SLE 广播信道类
型"）
            "adv_tx_pwr": 1,                        // 广播发射功率，单位dbm，取值范围
[-127, 20], 0x7F：不设置特定发送功率
            "conn_intv_min": 1,                     // 连接（链路调度）间隔最小值，取值范
围[0x001E,0x3E80]
            "conn_intv_max": 1,                     // 连接（链路调度）间隔最小值，取值范
围[0x001E,0x3E80]
```

```
            "conn_latency_max": 1,                  // 连接（链路调度）延迟（休眠）最大值，取值范围[0x0000,0x01F3]
            "conn_timeout": 1                       // 连接（链路）超时时间，取值范围[0x000A,0x0C80]
        }
    }
}
```

SLE 广播类型

```
    SLE_ADV_TYPE_NONCONN_NONSCAN      = 0x00, /*!< 不可连接不可扫描。*/
    SLE_ADV_TYPE_CONNECTABLE_NONSCAN  = 0x01, /*!< 可连接不可扫描。*/
    SLE_ADV_TYPE_NONCONN_SCANABLE     = 0x02, /*!< 不可连接可扫描。*/
    SLE_ADV_TYPE_CONNECTABLE_SCANABLE = 0x03, /*!< 可连接可扫描。*/
    SLE_ADV_TYPE_CONNECTABLE_DIRECTED = 0x07, /*!< 可连接可扫描定向。*/
```

G/T 角色协商指示类型

```
SLE_ANNOUNCE_ROLE_T_CAN_NEGO    = 0, /*!< 期望做T可协商 */
SLE_ANNOUNCE_ROLE_G_CAN_NEGO    = 1, /*!< 期望做G可协商 */
SLE_ANNOUNCE_ROLE_T_NO_NEGO     = 2, /*!< 期望做T不可协商 */
SLE_ANNOUNCE_ROLE_G_NO_NEGO     = 3, /*!< 期望做G不可协商 */
```

SLE 广播（被发现）级别

```
XF_SLE_ANNOUNCE_LEVEL_NONE      = 0, /*!< 不可见发现，预留 */
XF_SLE_ANNOUNCE_LEVEL_NORMAL    = 1, /*!< 一般可发现 */
XF_SLE_ANNOUNCE_LEVEL_PRIORITY  = 2, /*!< 优先可发现，预留 */
XF_SLE_ANNOUNCE_LEVEL_PAIRED    = 3, /*!< 被曾配对过的设备发现，预留 */
XF_SLE_ANNOUNCE_LEVEL_SPECIAL   = 4, /*!< 被指定设备发现 */
```

SLE 广播信道类型

```
    SLE_ADV_CHANNEL_MAP_77                  = 0x01,
    SLE_ADV_CHANNEL_MAP_78                  = 0x02,
    SLE_ADV_CHANNEL_MAP_79                  = 0x04,
    SLE_ADV_CHANNEL_MAP_DEFAULT             = 0x07, /*!< 表示所有信道 */
```

MCS （调制与编码策略） 配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sle":{"mcs":{"sle_obj_id":0,"mcs":1}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sle":                              //  表示是 SLE 配置组
    {
        "mcs":                          // 表示是 MCS （调制与编码策略） 配置
        {                               // 配置项:
            "sle_obj_id": 0,                // SLE OBJ ID
            "mcs": 1                        // 策略
        }
    }
}
```

负载偏好最大值（payload_max）配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sle":{"payload_max":{"sle_obj_id":0,"size":512}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sle":                              //  表示是 SLE 配置组
```

```
    {
        "payload_max":                          // 表示是负载偏好最大值
        {                                       // 配置项:
            "sle_obj_id": 0,                        // SLE OBJ ID
            "size": 512                             // 最大值
        }
    }
}
```

AT SLETEST 配置组

SLETEST 配置组信息与 AT SLE 配置组基本相似，SLETEST 不同之处主要为：

1. 配置项（成员）不是值，而是数组，表示可选择多个值
2. 移除 AT SLE 配置组下的 "sle_obj_id" 配置项（成员）
3. SLETEST 配置仅对主机侧有效，且配置传入解析时，为增量更新：
   如果测试配置缓存中存在相同配置项（成员）的测试值，则只保留一个；
   如无相同值则新增进缓存
4. 测试配置的缓存只会在掉电或主动调用清除指令 "AT+SLETEST=-1" 时，才会进行清除

发射功率配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sletest":{"tx_pwr":[10]}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                       // AT 配置描述的 ROOT
    "sletest":                  //  表示是 SLETEST 配置组
    {
        "tx_pwr":[10]               // 表示是发射功率配置
    }
}
```

MTU 配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sletest":{"mtu":{"size":[236]}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                           // AT 配置描述的 ROOT
    "sletest":                      //  表示是 SLETEST 测试配置组
    {
        "mtu":                      // 表示是 SLE MTU 配置
        {                           // 配置项:
            "size": [236]               // MTU 大小
        }
    }
}
```

默认连接参数的配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sletest":{"conn_param_def":{"en_filter":[true],"en_gt_negotiate":
[true],"scan_phy":[1],"scan_intv":[100],"scan_window":[100],"conn_intv_min":
[100],"conn_intv_max":[100],"conn_timeout":[100]}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sletest":                              //  表示是 SLETEST 配置组
    {
        "conn_param_def":                   // 表示是默认连接参数的配置
        {                                   // 配置项:
            "en_filter": [true],                // 是否开启过滤功能
            "en_gt_negotiate": [true],          // 链路建立时是否进行 G 和 T 协商交
互
            "scan_phy": [1],                    // 扫描频宽: 1:1M; 2:2M
            "scan_intv": [100],                 // 扫描间隔
            "scan_window": [100],               // 扫描窗口
```

```
            "conn_intv_min": [100],              // 连接（链路调度）间隔最小值
            "conn_intv_max": [100],              // 连接（链路调度）间隔最大值
            "conn_timeout": [100]                // 连接（链路）超时时间
        }
    }
}
```

连接参数更新的配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sletest":{"conn_param_upd":{"sle_obj_id":[0],"intv_min":[100],"intv_max":
[100],"latency":[100],"timeout":[100]}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sletest":                              //  表示是 SLETEST 配置组
    {
        "conn_param_upd":                   // 表示是连接参数更新的配置组
        {                                   // 配置成员：
            "sle_obj_id": [0],                  // SLE OBJ ID
            "intv_min": [100],                  // 连接（链路调度）间隔最小值，单位
slot
            "intv_max": [100],                  // 连接（链路调度）间隔最大值，单位
slot
            "latency":[100],                    // 连接（链路）延迟（休眠）时间，单位
slot
            "timeout": [100]                    // 连接（链路）超时时间，单位 10 ms
        }
    }
}
```

PHY 配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sletest":{"phy":{"tx_fmt":[1],"rx_fmt":[1],"tx_phy":[1],"rx_phy":
[1],"tx_pilot_density":[1],"rx_pilot_density":[1],"g_feedback":[1],"t_feedback":
[1]}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sletest":                              //  表示是 SLETEST 配置组
    {
        "phy":                              // 表示是 PHY 配置
        {                                   // 配置项:
            "tx_fmt": [1],                      // 发送无线帧类型 （详参 "SLE 无线帧
类型"）
            "rx_fmt": [1],                      // 接收无线帧类型 （详参 "SLE 无线帧
类型"）
            "tx_phy": [1],                      // 发送 PHY: 0:1M; 1:2M; 2:4M
            "rx_phy": [1],                      // 接收 PHY: 0:1M; 1:2M; 2:4M
            "tx_pilot_density": [1],            // 发送导频密度指示 （详参 "SLE 导频
密度指示"）
            "rx_pilot_density": [1],            // 接收导频密度指示 （详参 "SLE 导频
密度指示"）
            "g_feedback": [1],                  // 先发链路反馈类型指示，取值范围 0-
63
            "t_feedback": [1]                   // 后发链路反馈类型指示，取值范围 0-
7
        }
    }
}
```

SLE 无线帧类型

```
    SLE_RADIO_FRAME_1    = 0,       /*!< 无线帧类型1 */
    SLE_RADIO_FRAME_2    = 1,       /*!< 无线帧类型2 */
    SLE_RADIO_FRAME_3_M0 = 2,       /*!< 无线帧类型3，m序列0 */
    SLE_RADIO_FRAME_3_M1 = 3,       /*!< 无线帧类型3，m序列1 */
    SLE_RADIO_FRAME_3_M2 = 4,       /*!< 无线帧类型3，m序列2 */
    SLE_RADIO_FRAME_3_M3 = 5,       /*!< 无线帧类型3，m序列3 */
    SLE_RADIO_FRAME_3_M4 = 6,       /*!< 无线帧类型3，m序列4 */
    SLE_RADIO_FRAME_3_M5 = 7,       /*!< 无线帧类型3，m序列5 */
    SLE_RADIO_FRAME_4_M0 = 8,       /*!< 无线帧类型4，m序列0 */
    SLE_RADIO_FRAME_4_M1 = 9,       /*!< 无线帧类型4，m序列1 */
```

```
    SLE_RADIO_FRAME_4_M2 = 10,      /*!< 无线帧类型4，m序列2 */
    SLE_RADIO_FRAME_4_M3 = 11,      /*!< 无线帧类型4，m序列3 */
    SLE_RADIO_FRAME_4_M4 = 12,      /*!< 无线帧类型4，m序列4 */
    SLE_RADIO_FRAME_4_M5 = 13,      /*!< 无线帧类型4，m序列5 */
```

SLE 导频密度指示

```
    SLE_PHY_PILOT_DENSITY_4_TO_1  = 0x0,    /*!< 导频密度为4:1 */
    SLE_PHY_PILOT_DENSITY_8_TO_1  = 0x1,    /*!< 导频密度为8:1 */
    SLE_PHY_PILOT_DENSITY_16_TO_1 = 0x2,    /*!< 导频密度为16:1 */
```

广播参数配置（配置完成后需要在下一次开启广播时生效）

可配置项的压缩描述（可直接发送，能被解析）

```
{"sletest":{"adv_param":{"sle_obj_id":[0],"adv_type":[1],"adv_gt_role":
[1],"adv_level":[1],"adv_intv_min":[100],"adv_intv_max":[100],"adv_ch_map":
[1],"adv_tx_pwr":[1],"conn_intv_min":[1],"conn_intv_max":[1],"conn_latency_max":
[1],"conn_timeout":[1]}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sletest":                              //  表示是 SLETEST 配置组
    {
        "adv_param":                        // 表示是 广播参数 的配置
        {                                   // 配置项:
            "sle_obj_id": [0],                  // SLE OBJ ID
            "adv_type": [1],                    // 广播类型（详参: "SLE 广播类型"）
            "adv_gt_role": [1],                 // 广播 G/T 角色协商（详参: "G/T 角
色协商指示类型"）
            "adv_level": [1],                   // 广播（被发现）级别（详参: "SLE 广
播（被发现）级别"）
            "adv_intv_min": [100],              // 广播间隔最小值，取值范围
[0x000020~0xffffff], 单位125us
            "adv_intv_max": [100],              // 广播间隔最大值，取值范围
[0x000020~0xffffff], 单位125us
            "adv_ch_map": [1],                  // 广播信道 （详参: "SLE 广播信道类
型"）
```

```
            "adv_tx_pwr": [1],                  // 广播发射功率，单位dbm，取值范围
[-127, 20]，0x7F：不设置特定发送功率
            "conn_intv_min": [1],               // 连接（链路调度）间隔最小值，取值范
围[0x001E,0x3E80]
            "conn_intv_max": [1],               // 连接（链路调度）间隔最小值，取值范
围[0x001E,0x3E80]
            "conn_latency_max": [1],            // 连接（链路调度）延迟（休眠）最大
值，取值范围[0x0000,0x01F3]
            "conn_timeout": [1]                 // 连接（链路）超时时间，取值范围
[0x000A,0x0C80]
        }
    }
}
```

SLE 广播类型

```
    SLE_ADV_TYPE_NONCONN_NONSCAN      = 0x00, /*!< 不可连接不可扫描。*/
    SLE_ADV_TYPE_CONNECTABLE_NONSCAN  = 0x01, /*!< 可连接不可扫描。*/
    SLE_ADV_TYPE_NONCONN_SCANABLE     = 0x02, /*!< 不可连接可扫描。*/
    SLE_ADV_TYPE_CONNECTABLE_SCANABLE = 0x03, /*!< 可连接可扫描。*/
    SLE_ADV_TYPE_CONNECTABLE_DIRECTED = 0x07, /*!< 可连接可扫描定向。*/
```

G/T 角色协商指示类型

```
SLE_ANNOUNCE_ROLE_T_CAN_NEGO    = 0, /*!< 期望做T可协商 */
SLE_ANNOUNCE_ROLE_G_CAN_NEGO    = 1, /*!< 期望做G可协商 */
SLE_ANNOUNCE_ROLE_T_NO_NEGO     = 2, /*!< 期望做T不可协商 */
SLE_ANNOUNCE_ROLE_G_NO_NEGO     = 3, /*!< 期望做G不可协商 */
```

SLE 广播（被发现）级别

```
XF_SLE_ANNOUNCE_LEVEL_NONE      = 0, /*!< 不可见发现，预留 */
XF_SLE_ANNOUNCE_LEVEL_NORMAL    = 1, /*!< 一般可发现 */
XF_SLE_ANNOUNCE_LEVEL_PRIORITY  = 2, /*!< 优先可发现，预留 */
XF_SLE_ANNOUNCE_LEVEL_PAIRED    = 3, /*!< 被曾配对过的设备发现，预留 */
XF_SLE_ANNOUNCE_LEVEL_SPECIAL   = 4, /*!< 被指定设备发现 */
```

SLE 广播信道类型

```
    SLE_ADV_CHANNEL_MAP_77                  = 0x01,
    SLE_ADV_CHANNEL_MAP_78                  = 0x02,
    SLE_ADV_CHANNEL_MAP_79                  = 0x04,
    SLE_ADV_CHANNEL_MAP_DEFAULT             = 0x07, /*!< 表示所有信道 */
```

MCS （调制与编码策略） 配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sletest":{"mcs":{"mcs":[1]}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sletest":                              //  表示是 SLETEST 配置组
    {
        "mcs":                              // 表示是 MCS （调制与编码策略） 配置
        {                                   // 配置项:
            "mcs": [1]                          // 策略
        }
    }
}
```

负载偏好最大值（payload_max）配置

可配置项的压缩描述（可直接发送，能被解析）

```
{"sletest":{"payload_max":{"size":[512]}}}
```

可配置项的格式化描述（仅用于理解，不能直接发送，不能被解析）

```
{                                   // AT 配置描述的 ROOT
    "sletest":                              //  表示是 SLETEST 配置组
    {
```

```
        "payload_max":                          // 表示是负载偏好最大值
        {                                       // 配置项:
            "size": [512]                           // 最大值
        }
    }
}
```

示例

AT 数据通道（串口）的配置

> 如果没有特殊说明，AT 数据通道会默认配置为与固件下载的串口，配置项为波特率 115200，数据位 8，停止位 1，无校验位。
>
> **详情浏览：**
>
> 1. **"AT 配置指令" -> "AT 配置指令描述"**
> 2. **"AT 配置指令" -> "AT 配置指令描述" -> "AT 通用配置的配置描述说明" -> "AT 数据通道配置组"**

SLE 快速建立连接，进行通讯

服务端流程

1. 开启 SLE 功能

```
/* 发送指令 */
AT+SLEENABLE=1

/* 正确响应 */

OK
```

2. 创建 SLE 对象（服务端）

```
/* 发送指令 */
AT+SLEOBJ=1,0

/* 正确响应 */
ID:0
```

```
OK
```

3. 开启广播

```
/* 发送指令 */
AT+SLEADV=0,1

/* 正确响应 */

OK
```

4. 等待接收

```
/* 发送指令 */
AT+SLERECV=0,0,20,15000,1        // 直接接收类型，最大接收 20 字节数据，最大空闲时间为 15s，仅执行 1 次

/* 正确响应 */
/* 如果有收到数据则会直接输出，以下是本示例中的输出 */
1234QWER5678

/* 最后一次接收到数据起计时的 15s 后，达到最大空闲时间 */
INFO:TIMEOUT,size:12

OK
```

客户端流程

1. 开启 SLE 功能

```
/* 发送指令 */
AT+SLEENABLE=1

/* 正确响应 */

OK
```

2. 创建 SLE 对象（客户端）

```
/* 发送指令 */
AT+SLEOBJ=1,1
```

```
/* 正确响应 */
ID:0

OK
```

3. 通过名字进行连接（默认设备名为 "XF_SLE" ）

```
/* 发送指令 */
AT+SLECONNBYNAME=0,"XF_SLE",2000

/* 正确响应 */
peer:XF_SLE,RSSI:-37,addr:CA:CA:CA:CA:CA:CA

OK
```

4. 发送数据

```
/* 发送指令：第 1 次发送（演示达到最大发送字节数的效果） */
AT+SLESEND=0,0,8,5000        // 直接发送类型，最大发送 8 字节数据，最大空闲时间为 5s

/* 此时 AT 系统会阻塞，等待传入要发送的数据
    本示例中我们直接传入: 1234QWER
*/

/* 正确响应 */
INFO:CMPL

OK


/* 发送指令：第 2 次发送 （演示达到最大空闲时间的效果） */
AT+SLESEND=0,0,8,3000        // 直接发送类型，最大发送 8 字节数据，最大空闲时间为 5s

/* 此时 AT 系统会阻塞，等待传入要发送的数据
    本示例中我们直接传入: 5678
    然后等待 3s 左右的时间
*/

/* 正确响应 */
INFO:TIMEOUT

OK
```

SLETEST 测试

> 示例步骤此处不进行编写，请参考 "AT+SLETEST - SLE 快速测试指令" 指令描述中的示例
> 此处仅提供较全的 SLETEST 测试配置 json 样例，以便快速配置、测试

SLETEST 测试配置 json 样例

```
{
    "sletest": {
        "tx_pwr": [
            20
        ],
        "conn_param_def": {
            "en_filter": [
                false
            ],
            "en_gt_negotiate": [
                false
            ],
            "scan_phy": [
                1
            ],
            "scan_intv": [
                20
            ],
            "scan_window": [
                20
            ],
            "conn_intv_min": [
                20
            ],
            "conn_intv_max": [
                20
            ],
            "conn_timeout": [
                500
            ]
        },
        "mtu": {
            "size": [
                512
            ]
        },
```

```
        "conn_param_upd": {
            "intv_min": [
                20
            ],
            "intv_max": [
                20
            ],
            "latency": [
                0
            ],
            "timeout": [
                500
            ]
        },
        "phy": {
            "tx_fmt": [
                1
            ],
            "rx_fmt": [
                1
            ],
            "tx_phy": [
                2
            ],
            "rx_phy": [
                2
            ],
            "tx_pilot_density": [
                2
            ],
            "rx_pilot_density": [
                2
            ],
            "g_feedback": [
                0
            ],
            "t_feedback": [
                0
            ]
        },
        "mcs": {
            "mcs": [
                10
            ]
        },
        "payload_max": {
            "size": [
                512
```

```
            ]
        }
    }
}
```