프로토콜 바인딩 추가 가이드

본 문서에는 tinyIoT에서 새로운 프로토콜 바이딩을 추가하는 방법에 대해서 설명한다.
이를 통해 시스템이 다양한 네트워크 환경에서 유연하게 통신할 수 있도록 지원할 수 있다.
이 가이드는 웹소켓(WebSocket) 바인딩을 추가하는 예시를 기반으로 작성되었으며, 다른 프로토콜을 추가할 때도 동일한 절차를 따라 진행할 수 있다.

1. 프로토콜 바인딩의 개념 및 목적

tinyIoT는 다양한 프로토콜을 지원하여 IoT기기 간의 데이터 전송 및 제어를 원활하게 한다.
새로운 프로토콜 바인딩을 추가함으로써, 각 네트워크 환경에 최적화된 통신 방식을 사용할 수 있으며, 시스템의 확장성과 유연성이 향상된다.

2. 프로토콜 바인딩 추가를 위한 필수 수정 사항

새로운 프로토콜 바인딩을 추가할 때는 다음과 같은 주요 파일과 함수를 수정해야 한다.

- onem2mTypes.h 파일 수정 : 새로운 바인딩을 시스템에 인식하도록 추가함
- checkResourceAddressingTypes 함수 수정 : 새 바인딩 문자열을 인식할 수 있도록 로직을 추가
- parsePoa 함수 수정 : 새 바인딩의 포트 번호 설정 및 파싱 로직을 추가하여, poa(Protocol Option Addressing) 파싱 시 정상적으로 처리되도록 수정함.

3. 단계별 예시 : 웹소켓((WebSocket) 바인딩 추가

이 예시는 WebSocket 바인딩을 tinyIoT에 추가하는 과정을 보여준다. 각 단계는 다른 프로토콜을 추가할 때도 동일한 방식으로 적용할 수 있다.

① **onem2mTypes.h 파일 수정**

- 먼저, 새로운 프로토콜 바인딩 (예: WebSocket)을 typedef에 추가하여 시스템에서 바인딩을 인식할 수 있도록 한다
- onem2mTypes.h 파일을 열고, 지원하는 바인딩 리스트에 WebSocket 추가

```
typedef enum
{
    PROT_HTTP = 1,
    PROT_MQTT,
    PROT_COAP = 3,
    PROT_COAPS = 4,
    PROT_WEBSOCKET =5, // 추가
    PROT_WEBSOCKETS=6, // 추가
} Protocol;
```

② **checkResourceAddressingType 함수 수정**

- checkResourceAddressingType 함수에 새로 추가한 WebSocket 바인딩을 인식하는 로직을 추가함
- WebSocket 바인딩 문자열 (“ws”)을 처리하도록 해당 함수에 조건을 추가함

```
else if (strncmp(uri, "http://", 7) == 0 || strncmp(uri, "mqtt://", 7) == 0 || strcmp(uri, "coap://") == 0
|| strcmp(uri, "coaps://")  == 0|| strncmp(uri, "ws://", 5) == 0 || strncmp(uri, "wss://", 6) ==0)
{
    return PROTOCOL_BINDING;
}
```

③ parsePoa 함수 수정 (포트 번호 설정)

- WebSocket 바인딩을 추가한 후, parsePoa 함수에서 포트 번호와 관련된 설정을 추가하여 poa 파싱 시 WebSocket 바인딩이 올바르게 처리되도록 수정함

```
int parsePoa(char *poa_str, Protocol *prot, char **host, int *port, char **path)
{
    char *p = strdup(poa_str);
    char *ptr = NULL;
    if (!strncmp(poa_str, "http://", 7))
    {
        *prot = PROT_HTTP;
    }
    else if (!strncmp(poa_str, "mqtt://", 7))
    {
        *prot = PROT_MQTT;
    }
    else if (!strncmp(poa_str, "coap://", 7))
    {
        *prot = PROT_COAP;
    }
    else if (!strncmp(poa_str, "coaps://", 8))
    {
        *prot = PROT_COAPS;
    }
    else if (!strncmp(poa_str, "ws://", 5)) // 추가
    {
        *prot = PROT_WEBSOCKET;
    }
    else if (!strncmp(poa_str, "wss://", 6)){ // 추가
        *prot = PROT_WEBSOCKETS;
    }
```

```
if (*prot == PROT_HTTP && port == 0)
{
    *port = 80;
}
else if (*prot == PROT_MQTT && port == 0)
{
    *port = 1883;
}
else if (*prot == PROT_COAP && port == 0)
{
    *port = 5683;
}
else if (*prot == PROT_COAPS && port == 0)
{
    *port = 5684;
}
else if (*prot == PROT_WEBSOCKET && port ==0)
{
    *port = 8081;
}
else if (*prot == PROT_WEBSOCKETS && port ==0){
    *port = 8443;
}
return 0;
```

4. 다른 프로토콜 추가 시 참고사항

- ○ 각 프로토콜의 특성에 맞게 보안 설정, 데이터 형식, 포트 번호 등을 유연하게 구성할 수 있도록 코드 내에서 프로토콜 별 설정을 관리하는 것이 좋음
- ○ 예를 들어, 보안이 필요한 프로토콜의 경우 SSL/TLS 설정을 추가하고, 데이터 형식이 다를 경우 적절한 인코딩 및 디코딩 방법을 포함해야 함