

# 세종 로그인톡 2.1

## 연동 개발 가이드

[rnd@roumit.com](mailto:rnd@roumit.com)

02-6929-3299

## 목차

1. 요구사항 .....	p.3
2. 로그인톡 프로세스 .....	p.4
3. 로그인톡 클라이언트 호출 (클라이언트 연동) .....	p.5
4. 토큰 수신 및 유저 정보 교환 (서버 연동) .....	p.9
별첨1. 로그인톡에서 받은 유저 정보로 본인인증 처리하기 .....	p.14
별첨2. 로그인톡 버튼 가이드 .....	p.16

## 1. 요구사항

### (1) 제휴사 등록

- 로그인톡 연동을 위해서는 로움아이티에 제휴사로 등록되어 제휴사 키를 발급받아야 합니다.
- 현재 제휴사 등록 및 제휴사 키 발급은 로움아이티에서 수기로 처리하고 있습니다.
- 제휴사 등록 및 제휴사 키 발급은 로움아이티 영업 담당자 또는 연동 담당자에게 이메일 주세요.
- 제휴사 등록 및 제휴사 키 발급 요청 이메일은 아래의 내용이 필수로 포함되어 있어야 합니다.

서비스 이름: (알림톡에 표시될 서비스명)

대표 도메인: (알림톡에 표시될 대표 도메인)

운영 도메인: (테스트 및 실운영을 포함하는 복수의 도메인 - 제휴사 검증에 사용됨)

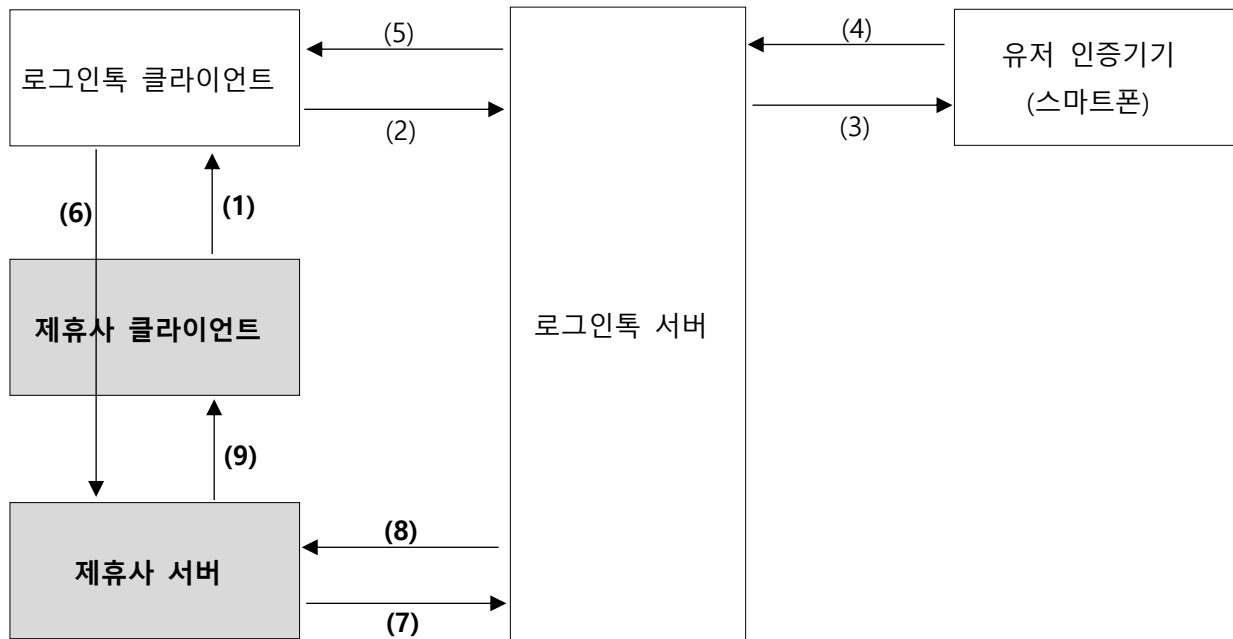
서버 아이피: (테스트 및 실운영을 포함하는 복수의 IP - 제휴사 검증에 사용됨)

- 동적 아이피를 사용하는 경우 로움아이티에 Secret Key 사용을 요청해 주세요.
- 제휴사 등록 및 제휴사 키 발급 이전에는 로그인톡 연동을 진행하실 수 없습니다.

### (2) 유저 개인정보

- 로그인톡을 통한 유저 인증(회원가입 제외)을 위해서는 제휴사 측에서도 인증을 위한 유저 개인정보를 별도 수집 및 보관하고 있어야 합니다.
- 특히 본인인증(점유인증 제외)의 경우 로그인톡은 제휴사 측에서 유저 CI(통신사에서 생성하는 개인 고유 연계정보)를 수집하여 로그인톡 인증에 사용할 것을 권장합니다. (CI는 생성 통신사 및 생성 시점에 관계없는 개인 고유 식별 값입니다.)
- 본인인증이 필요하나 CI를 수집하지 않는 경우 휴대폰 번호, 실명, 생년월일 중 최소 2개 이상의 개인정보를 수집 및 보관하고 있어야 합니다.
- 본인인증이 필요하나 CI, 휴대폰 번호, 실명, 생년월일 모두를 수집하지 않는 경우, 유저에게 "로그인톡 연동"을 요청하고 제휴사 측에서는 로그인톡 연동 시 수신한 해당 유저의 CI를 DB에 보관할 것을 권장합니다.

## 2. 로그인톡 프로세스



- (1) 제휴사 클라이언트에서 로그인톡 클라이언트를 호출합니다.
- (2) 로그인톡 클라이언트에서 로그인톡 서버로 유저 인증을 요청합니다.
- (3) 로그인톡 서버에서 유저 인증 기기로 인증 요청을 보냅니다. (예: 알림톡, ARS 아웃바운드)
- (4) 로그인톡 서버에서 유저 인증 승인 결과를 받습니다.
- (5) 로그인톡 서버에서 로그인톡 클라이언트로 인증 토큰을 발행합니다.
- (6) 로그인톡 클라이언트에서 제휴사 클라이언트를 통해 제휴사 서버로 인증 토큰을 전송합니다.
- (7) 제휴사 서버에서 로그인톡 서버로 인증 토큰을 전송합니다. (REST API 요청)
- (8) 로그인톡 서버에서 인증 토큰과 연계된 유저 정보를 제휴사 서버로 전송합니다. (REST API 응답)
- (9) 전송받은 유저 정보로 제휴사 서버에서 필요한 인증 처리를 진행합니다.

### 3. 로그인톡 클라이언트 호출 (클라이언트 연동)

#### (1) 버튼 생성을 생성해 주세요.

```
<button name="SERVICE_NAME">로그인톡 인증</button>
```

"name" attribute에는 이용하고자 하는 서비스에 맞는 SERVICE\_NAME을 입력해 주세요.

서비스	SERVICE_NAME
간편 로그인	logintalk_sejong_login
간편 본인 확인	logintalk_sejong_verify
간편 인증	logintalk_sejong_confirm

로그인톡 클라이언트로 유저 휴대폰 번호를 직접 전달하고 싶은 경우 유저 전화번호가 입력된 "data-tel" attribute를 추가해 주세요.

```
<button name="logintalk_ars2" data-tel="0212345678">ARS 인증</button>
```

#### (2) 로그인톡 스크립트를 삽입해 주세요.

```
<script src="https://auth.logintalk.io/js/logintalk.js"></script>
```

#### (3) logintalk 함수를 실행해 주세요.

```
<script>
  var options = {
    key: "YOUR_CLIENT_KEY",
    action: "YOUR_RETURN_URL"
  };
  logintalk(options);
</script>
```

#### options (Object)

- **key**: [필수, String] 로움아이티에서 발급받은 제휴사 키
- **action**: [필수, String] 유저 인증 완료 후 토큰을 받을 제휴사의 Return URL

로그인톡 라이브러리는 제휴사 페이지 내부에 form 태그를 생성합니다. 해당 form 태그의 "action" 및 "method" attribute에 options에서 입력한 "action" 및 "method" 값이 그대로 입력됩니다. 유저 인증이 완료되면 해당 form 태그 내부에 토큰이 입력된 input이 추가되고 form이 submit 됩니다.

- **method**: [옵션, String] 위 action의 요청 방법으로 "get" 또는 "post" (기본값: "get")
- **verify**: [옵션, Boolean] 본인인증 여부 (기본 점유인증)

점유인증	본인인증
<ul style="list-style-type: none"> <li>- 사용자가 특정 기기를 점유하고 있는지 확인하는 절차</li> <li>- 주로 2채널 인증에 쓰임</li> </ul>	<ul style="list-style-type: none"> <li>- 사용자가 특정기기를 점유함과 동시에 해당 기기의 명의자인지 확인하는 절차</li> <li>- 주로 회원가입, ID/PW 찾기 등에 쓰임</li> </ul>

- **id**: [옵션, String] 제휴사에서 생성한 요청 ID
- **redirect**: [옵션, Boolean] 인증 클라이언트에서 제휴사 Return URL로 redirection 여부

요청 클라이언트	인증 클라이언트
<ul style="list-style-type: none"> <li>- 사용자가 인증을 요청한 클라이언트</li> <li>- 로그인톡 클라이언트가 호출된 제휴사 페이지</li> </ul>	<ul style="list-style-type: none"> <li>- 사용자가 인증을 진행한 클라이언트</li> <li>- 카카오톡으로 인증하는 경우, 로그인톡의 인증 페이지가 열린 카카오톡 인앱 브라우저</li> </ul>

요청 클라이언트가 스마트폰에서 호출된 경우 요청 클라이언트와 인증 클라이언트의 기기가 동일(기기: 스마트폰)하기 때문에 사용자가 인증을 완료한 후 다시 요청 클라이언트로 돌아가야 하는 번거로움이 있습니다. options의 "redirect"를 true로 설정하시면 사용자가 요청 클라이언트로 돌아갈 필요 없이 인증 클라이언트에서 제휴사의 Return URL로 redirection 됩니다.

카카오톡 인앱 브라우저에서 요청 클라이언트가 호출된 경우 (카카오톡에서 제휴사 페이지가 열린 경우), 카카오톡에서 인증을 완료한 후 요청 클라이언트로 돌아갈 수가 없기 때문에 자동으로 인증 클라이언트 redirection이 실행됩니다.

#### (4) 로그인톡 클라이언트를 호출하는 전체 코드 예시입니다.

```
<!DOCTYPE html>
<head>
  <script src="https://auth.logintalk.io/js/logintalk.js"> </script>
  <script>
    logintalk({key: "YOUR_CLIENT_KEY", action: "YOUR_RETURN_URL"});
  </script>
</head>
<body>
  <button name="logintalk_sejong_login">로그인톡으로 로그인</button>
</body>
</html>
```

(5) 디버깅은 브라우저 콘솔 창의 "[logintalk] ..." 메시지를 확인해 주세요.

오류코드	설명
L501	등록된 제휴사 키가 없음
L502	제휴사가 해당 서비스를 계약하지 않음
L507	등록된 제휴사 도메인이 일치하지 않음
L944	비정상적인 방법으로 클라이언트가 호출됨

(6) [추가 기능] logintalk.main 함수를 이용하면 로그인톡 클라이언트를 직접 호출할 수 있습니다.

```
<!DOCTYPE html>
<head>
  <script src="https://auth.logintalk.io/js/logintalk.js"> </script>
  <script>
    logintalk({key: "YOUR_CLIENT_KEY", action: "YOUR_RETURN_URL"});
    var options = {service: 11};
    logintalk.main(options);
  </script>
</head>
<body>
  <!-- 로그인톡을 호출하는 버튼이 없어도 됩니다. -->
</body>
</html>
```

#### options (Object)

- **service**: [필수, Number] 이용하고자 하는 서비스의 서비스 코드

서비스	서비스 코드
간편 로그인	11
간편 본인 확인	13
간편 인증	12

- **user**: [옵션, String] 유저 전화번호  
"user" 값을 입력할 경우 해당 휴대폰 번호가 로그인톡 클라이언트로 전달됩니다. 그러면 유저가 로그인톡 클라이언트에서 휴대폰 번호를 입력하는 절차 없이 바로 인증 요청이 진행됩니다. (다시 말해, 로그인톡 클라이언트가 호출되자마자 해당 유저에게 카카오톡 알림톡이 발송됩니다.)
- **verify**: [옵션, Boolean] 본인인증 여부 (기본값: logintalk 함수에 등록된 option)
- **id**: [옵션, String] 제휴사에서 생성한 요청 ID (기본값: logintalk 함수에 등록된 option)

(7) [추가 기능] `logintalk.params` 함수를 이용해 Return URL에 전달할 패러미터를 추가할 수 있습니다.

```
<!DOCTYPE html>
<head>
  <script src="https://auth.logintalk.io/js/logintalk.js"> </script>
  <script>
    logintalk({key: "YOUR_CLIENT_KEY", action: "YOUR_RETURN_URL"});
    var params = {
      key1: "value1",
      key2: "value2"
    };
    logintalk.params(params);
  </script>
</head>
<body>
  <button name="logintalk_sejoing_login ">로그인톡으로 로그인</button>
</body>
</html>
```

(8) [추가 기능] `logintalk.callback`을 이용해 Return URL 대신 콜백 함수에서 토큰을 받을 수 있습니다.

```
<!DOCTYPE html>
<head>
  <script src="https://auth.logintalk.io/js/logintalk.js"> </script>
  <script>
    logintalk({key: "YOUR_CLIENT_KEY", action: "YOUR_RETURN_URL"});
    function cb(token){
      // 패러미터로 받으신 토큰을 서버로 전달하셔야 합니다.
    };
    logintalk.callback(cb);
  </script>
</head>
<body>
  <button name="logintalk_sejoing_login ">로그인톡으로 로그인</button>
</body>
</html>
```



#### 4. 토큰 수신 및 유저 정보 교환 (서버 연동)

(1) 클라이언트에서 등록한 Return URL로 토큰을 수신합니다.

Method	Data	Key	Type
POST	body	token	String
GET	query	token	String

(2) 수신한 토큰을 로그인톡 서버로 전송합니다.

URL	https://auth.logintalk.io/exchange	
Method	GET	POST
Query	?token=토큰	-
Body	-	token=토큰

(3) 토큰 전송에 대한 응답(Stringified JSON)으로 결과 코드와 유저 정보를 수신합니다.

Key	Value	인증 방법
result	결과 코드 (아래 표 참고)	점유/본인
id	제휴사에서 생성한 요청 ID	
mobile_number	유저 전화번호	
verify	본인인증 여부	
service	서비스 번호	
messenger	유저 인증 수단	
pw_force	유저 간편비밀번호 사용 강제 여부	
mobile	요청 클라이언트가 스마트폰인지 여부	
client_ip	토큰을 전송한 제휴사 서버 IP	
name	유저 이름	본인인증
CI	유저 CI (주민등록번호 대체 유저 고유 연계 정보 값)	
birthday	유저 생년월일	
sex	유저 성별 ("0": 여자)	
nation	유저 내외국인 구분 ("1": 내국인)	
telecom	유저 통신사 코드 (아래 표 참고)	
email	유저 이메일 (유저가 이메일을 입력한 경우만 존재)	

결과 코드	설명
L101	토큰 수신 및 유저 정보 전송이 정상적으로 처리됨
L301	토큰을 보내지 않음
L302	유효하지 않은 토큰임
L303	토큰과 연계된 정보를 찾을 수 없음 (예: 일회성인 토큰을 재전송한 경우)
L304	헤더의 비밀번호가 등록된 비밀번호와 일치하지 않음
L505	토큰을 전송한 서버의 IP가 등록된 제휴사의 IP와 일치하지 않음
L506	인증이 완료되지 않음 토큰이 전송됨 (해킹 시도로 분류함)
L101	토큰 수신 및 유저 정보 전송이 정상적으로 처리됨
L301	토큰을 보내지 않음

통신사 코드	통신사
01	SKT
02	KT
03	LG
04	SKT 알뜰폰
05	KT 알뜰폰
06	LG 알뜰폰

(4) 수신한 유저 정보로 제휴사에서 필요한 인증 처리를 진행합니다.

(5) 서버 언어별 REST API 호출 참고 코드입니다.

**PHP**

```
<?php
$url = "https://auth.logintalk.io/exchange?token=토큰값";
$is_post = false;
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, $is_post);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$headers = array();
$response = curl_exec ($ch);
$status_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
echo "status_code:".$status_code.";
curl_close ($ch);
if($status_code == 200) {
    echo $response;
} else {
    echo "Error 내용:".$response;
}
?>
```

**ASP.NET**

```
using System;
using System.Net.Http;

using (var client = new HttpClient())
{
    string apiURL = "https://auth.logintalk.io/exchange?token=토큰값";
    var res = await client.GetAsync(apiURL);
    var responseString = await res.Content.ReadAsStringAsync();
    Console.WriteLine("res.StatusCode = " + res.StatusCode);
    return "res.StatusCode=" + res.StatusCode + " ::: responseString" + responseString.ToString();
}
```

## JSP

```
<%@ page import="java.net.URLEncoder" %>
<%@ page import="java.net.URL" %>
<%@ page import="java.net.HttpURLConnection" %>
<%@ page import="java.io.BufferedReader" %>
<%@ page import="java.io.InputStreamReader" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
    try {
        String apiURL = "https://auth.logintalk.io/exchange?token=토큰값";
        URL url = new URL(apiURL);
        HttpURLConnection con = (HttpURLConnection)url.openConnection();
        con.setRequestMethod("GET");
        int responseCode = con.getResponseCode();
        BufferedReader br;
        System.out.print("responseCode="+responseCode);
        if(responseCode==200) {
            br = new BufferedReader(new InputStreamReader(con.getInputStream()));
        } else {
            br = new BufferedReader(new InputStreamReader(con.getErrorStream()));
        }
        String inputLine;
        StringBuffer res = new StringBuffer();
        while ((inputLine = br.readLine()) != null) {
            res.append(inputLine);
        }
        br.close();
        if(responseCode==200) {
            out.println(res.toString());
        }
    } catch (Exception e) {
        System.out.println(e);
    }
%>
```

### NodeJS

```
require('request').get('https://auth.logintalk.io/exchange?token=토큰값', (err, resp, data) => {  
  if(err || resp.statusCode !== 200) return console.log(err ? err : resp)  
  return console.log(data)  
})
```

## 별첨1. 로그인톡에서 받은 유저 정보로 본인인증 처리하기

### (1) 기존에 수집된 유저 CI 값이 있는 경우

- 로그인톡 서버로부터 수신 받은 유저 CI 값과 비교하여 인증 처리
- 로그인 세션 처리의 경우
  - ① 로그인톡 서버로부터 수신 받은 유저 CI 값으로 유저 DB를 검색
  - ② 검색된 유저 정보로 로그인 세션 처리
  - ③ 검색된 유저가 없는 경우 회원가입 페이지로 전환하여 회원가입 유도 (회원가입 폼에서 로그인 톡으로부터 수신 받은 정보로 채워줄 수 있는 부분은 자동입력 처리)

### (2) 기존에 수집된 유저 CI 값이 없으나 휴대폰 번호 및 실명 또는 생년월일이 있는 경우

- 로그인톡 서버로부터 수신 받은 휴대폰 번호 및 실명 또는 생년월일과 비교하여 인증 처리
- 로그인 세션 처리의 경우
  - ① 유저DB 테이블에 CI 컬럼 추가
  - ② 로그인톡 서버로부터 수신 받은 유저 CI 값으로 유저 DB를 검색
  - ③ 유저 검색에 성공한 경우 해당 유저 정보로 로그인 세션 처리
  - ④ 유저 검색에 실패한 경우 로그인톡 서버로부터 수신 받은 유저 휴대폰 번호와 함께 실명 또는 생년월일로 유저 DB 검색
  - ⑤ ④에서 유저 검색에 성공한 경우 로그인톡으로부터 수신 받은 CI 값을 DB에 수집 후 해당 유저 정보로 로그인 세션 처리
  - ⑥ ④에서 유저 검색에 실패한 경우 회원가입 페이지로 전환하여 회원가입 유도 (회원가입 폼에서 로그인톡으로부터 수신 받은 정보로 채워줄 수 있는 부분은 자동입력 처리)

### (3) 기존에 수집된 유저 CI, 실명 및 생년월일이 없으나 휴대폰 번호, 이메일이 있는 경우

- 일반 인증 처리의 경우
  - ① 유저DB 테이블에 CI 컬럼 추가
  - ② 로그인톡 서버로부터 수신 받은 휴대폰 번호 또는 이메일과 일치하는지 우선 비교
  - ③ 일치하지 않으면 인증 실패 처리
  - ④ 일치하면 해당 유저에게 비밀번호 입력을 통해 2차 인증을 요구하는 페이지(팝업) 렌더링
  - ⑤ 유저가 입력한 비밀번호가 일치하면 로그인톡으로부터 수신 받은 CI 값을 DB에 수집 후 해당 유저 정보로 인증 처리
  - ⑥ 이후 인증처리 시 수집한 CI 값 사용

- 로그인 세션 처리의 경우

- ① 유저DB 테이블에 CI 컬럼 추가
- ② 로그인톡 서버로부터 수신 받은 유저 CI 값으로 유저 DB를 검색
- ③ 유저 검색에 성공한 경우 해당 유저 정보로 로그인 세션 처리
- ④ 유저 검색에 실패한 경우 로그인톡 서버로부터 수신 받은 휴대폰 번호 또는 이메일로 유저 DB 검색
- ⑤ ④에서 유저 검색에 실패한 경우 회원가입 페이지로 전환하여 회원가입 유도 (회원가입 폼에서 로그인톡으로부터 수신 받은 정보로 채워줄 수 있는 부분은 자동입력 처리)
- ⑥ ④에서 유저 검색에 성공한 경우 해당 유저에게 비밀번호 입력을 통해 2차 인증을 요구하는 페이지(팝업) 렌더링
- ⑦ 유저가 입력한 비밀번호가 일치하면 로그인톡으로부터 수신 받은 CI 값을 DB에 수집 후 해당 유저 정보로 로그인 세션 처리

**(4) 기존에 수집된 유저 CI, 실명 및 생년월일, 휴대폰 번호, 이메일이 없는 경우**

- 일반 인증의 경우 로움아이티 담당자와 상의

- 로그인 세션 처리의 경우

- ① 아이디로 로그인 후 본인인증을 할 수 있는 기능 제공 (로그인톡을 통한 본인인증 권장)
- ② 최초 로그인 시도 시 아이디 로그인 후 (로그인톡) 본인인증을 하도록 유도
- ③ 아이디 로그인후 (로그인톡) 본인인증을 한 유저는 CI, 실명, 생년월일 등 정보를 수집
- ④ CI, 실명, 생년월일이 수집된 유저는 로그인톡으로 로그인 처리 가능하도록 처리

## 별첨2. 로그인톡 버튼 가이드

- 로그인톡 버튼은 제휴사의 디자인 형식에 맞추어 자유롭게 생성할 수 있습니다.
- 다만, 타 브랜드명(예: 카카오톡)을 표기함으로서 발생할 수 있는 문제를 최소화하기 위하여, 서비스 명칭을 표기할 시 서비스명 '로그인톡' 명시를 부탁드립니다.

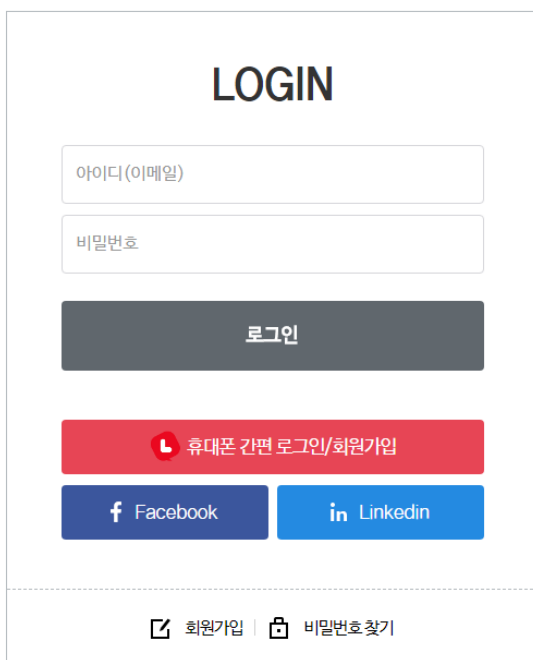
### (1) BI 가이드

- 로그인톡 BI는 제휴사의 디자인 형식에 맞추어 자유롭게 사용할 수 있습니다.
- BI 원본색상 이미지 주소: [https://static.logintalk.kr/img/logintalk\\_bi.png](https://static.logintalk.kr/img/logintalk_bi.png)
- BI 반전색상 이미지 주소: [https://static.logintalk.kr/img/logintalk\\_bi\\_r.png](https://static.logintalk.kr/img/logintalk_bi_r.png)

### (2) 색상 가이드

- 로그인톡 버튼 색상은 제휴사의 디자인 형식에 맞추어 자유롭게 변형할 수 있습니다.
- 로그인톡 BI 색상 코드: #ED1F35

### (3) 버튼 디자인 예시



LOGIN

아이디(이메일)

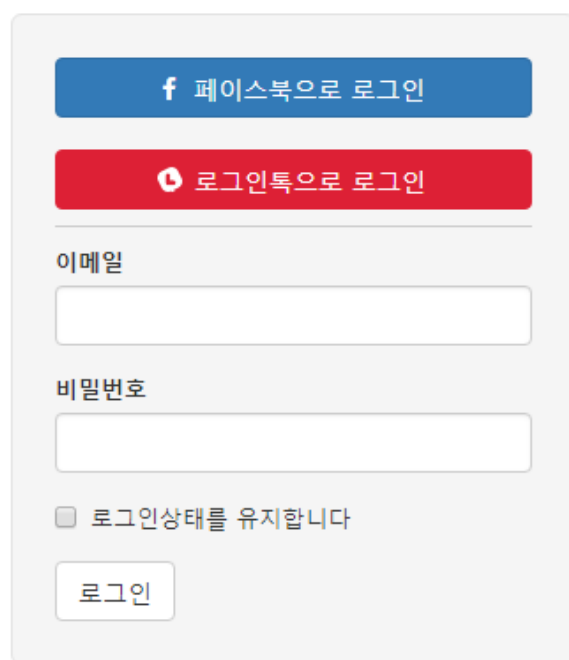
비밀번호

로그인

휴대폰 간편 로그인/회원가입

Facebook LinkedIn

회원가입 | 비밀번호찾기



f 페이스북으로 로그인

로그인톡으로 로그인

이메일

비밀번호

☐ 로그인상태를 유지합니다

로그인