BizSpring Logger 웹 분석**: React** 환경 태깅 가이드 **(**개발자용**)**

목차

1. **(**필수**)** 기본 스크립트 설치
    - GTM 태그 생성 (맞춤 HTML)
    - React SPA 환경을 위한 트리거 설정 (History Change)
2. **(**선별**)** 환경변수 스크립트
    - Data Layer를 이용한 데이터 전달 (React -> GTM)
    - GTM 변수 설정 (Data Layer 변수)
    - GTM 트리거 설정 (맞춤 이벤트)
    - GTM 태그 설정 (맞춤 HTML - 이벤트 전송 스크립트)
    - 주요 환경변수별 태깅 가이드
        - `_TRK_PI` : 해당 웹페이지 성격 전달
        - `_TRK_PN` : 상품 코드 전달
        - `_TRK_PND` : 상품 명 전달
        - `_TRK_OA` : 주문 금액 전달
        - `_TRK_OE` : 주문 수량 전달
        - `_TRK_OP` : 주문 상품 코드 전달
        - `_TRK_CP` : 카테고리 전달
        - `_TRK_MF` : 상품의 브랜드명 전달
        - `_TRK_SX` : 회원성별 전달
        - `_TRK_AG` : 회원특성 전달
        - `_TRK_IK` : 내부 검색어 전달
3. **React** 애플리케이션 내 **Data Layer Push** 예시
4. 테스트 및 디버깅
5. **GTM** 미사용 시 직접 설치 방법 **(React)**
    - _BSTRK_LOADED 전역변수로 인한 스크립트 재실행 제한
    - 스크립트 실행 순서 조정 필요
    - 데이터 검증 방법

이 문서는 React 기반의 웹사이트에서 Google Tag Manager(GTM)를 사용하여 BizSpring Logger의 기본 스크립트 및 환경변수 스크립트의 태깅 방법을 안내합니다.

참고 자료 :

- **Logger** 기본 태깅 매뉴얼: https://logger.co.kr/images/200912_free/developer_manual.pdf
- **Logger** 환경변수 상세 매뉴얼: https://logger.co.kr/manual_FR/help.tsp?node=202 (및 하위 페이지)

사전 준비 :

1. **BizSpring Logger** 계정 : 분석 스크립트에 필요한 서비스번호 발급(서비스 생성시 자동 발급)
2. **Google Tag Manager** 계정 및 컨테이너 : GTM 계정과 웹사이트용 컨테이너 생성 및 기본 스니펫 설치 (GTM 시작하기 가이드 참고)
3. **React** 개발 환경 : 웹사이트 소스 코드 접근 및 수정 권한
4. **Data Layer** 활용 이해 : React 애플리케이션에서 GTM으로 데이터를 전달하기 위한 `dataLayer` 객체 사용법 숙지

1. (필수) 기본 스크립트 설치

React와 같은 SPA(Single Page Application)는 페이지 이동 시 전체 페이지를 새로 로드하지 않고 내용만 동적으로 변경합니다. 따라서 기본적인 '페이지 뷰' 트리거 외에 'History Change' 트리거를 사용하여 가상 페이지뷰를 추적해야 합니다.

다만, 모든 SPA 구조에서 'History Change' 트리거가 동작하는게 아니기 때문에 GTM 상에서 테스트를 진행하여 'History Change' 트리거 동작 여부에 따라 dataLayer 를 활용하여 가상 페이지뷰를 추적해야 하는 경우도 있습니다.

1.1. GTM 태그 생성 **(맞춤 HTML)**

1. GTM 작업 공간 > 태그 > 새로 만들기 클릭
2. 태그 구성 클릭 > 맞춤 **HTML** 선택
3. HTML 필드에 아래 적용 스크립트 붙여넣기
4. 서비스번호는 로거 서비스 생성시 발급된 서비스번호로 교체

적용 스크립트 :

None

```
<script id="pre_서비스번호_script">
  if (document.getElementById("logger_서비스번호_tracker_id")) {
    document.getElementById("logger_서비스번호_tracker_id").remove();
  }
  if (document.getElementById("logger_서비스번호_tracking_script")) {
    document.getElementById("logger_서비스번호_tracking_script").remove();
  }
  if (document.getElementById("logger_서비스번호_noscript_tracker")) {

document.getElementById("logger_서비스번호_noscript_tracker").remove();
  }
  if
(document.querySelector('script[src="//fs.bizspring.net/fs4/bstrk.1.js"]
')) {

document.querySelector('script[src="//fs.bizspring.net/fs4/bstrk.1.js"]'
).remove();
  }
```

```
  if
(document.querySelector('script[src="//fs.bizspring.net/fs4/logger.v4.1.
js"]')) {

document.querySelector('script[src="//fs.bizspring.net/fs4/logger.v4.1.j
s"]').remove();
  }
  if (document.getElementById("pre_서비스번호_script")) {
    document.getElementById("pre_서비스번호_script").remove();
  }
</script>

<!-- LOGGER(TM) TRACKING SCRIPT V.40 FOR logger.co.kr / 서비스번호 :
COMBINE TYPE / DO NOT ALTER THIS SCRIPT. -->
<script id="logger_서비스번호_tracker_id" type="text/javascript">
  var _TRK_LID = "서비스번호";
  var _L_TD = "ssl.logger.co.kr";
  var _TRK_CDMN = "";
</script>
<script id="logger_서비스번호_tracking_script" type="text/javascript">
  if (typeof(_BSTRK_LOADED) != 'undefined') {
    _BSTRK_LOADED = undefined;
  }

  var _CDN_DOMAIN = location.protocol == "https:" ?
"https://fs.bizspring.net" : "http://fs.bizspring.net";
    (function(b, s) {
      var f = b.getElementsByTagName(s)[0],
      j = b.createElement(s);
      j.async = true;
      j.src = "//fs.bizspring.net/fs4/bstrk.1.js";
      f.parentNode.insertBefore(j, f);
    })(document, "script");
</script>
<noscript id="logger_서비스번호_noscript_tracker">
  <img alt="Logger Script" width="1" height="1"
src="http://ssl.logger.co.kr/tracker.1.tsp?u=서비스번호&amp;js=N" />
</noscript>
```

```
<!-- END OF LOGGER TRACKING SCRIPT -->
```

1.2. React SPA 환경을 위한 트리거 설정 (History Change)

React Router 등 라우팅 라이브러리를 사용하면 브라우저의 History API (`pushState`, `replaceState`)를 통해 URL이 변경됩니다. GTM은 이 변경을 감지할 수 있습니다.

1. **GTM** 변수 활성화:
    - GTM 작업 공간 > 변수 > 기본 제공 변수 섹션의 구성 클릭
    - 페이지 섹션에서 `Page URL`, `Page Hostname`, `Page Path` 등이 선택되어 있는지 확인합니다.
    - 기록 섹션에서 `History Source`, `New History Fragment`, `New History State`, `Old History Fragment`, `Old History State` 를 모두 체크하여 활성화합니다.
2. **GTM** 트리거 생성:
    - GTM 작업 공간 > 트리거 > 새로 만들기 클릭
    - 트리거 구성 클릭 > 기록 변경 선택
    - 이 트리거 실행 시점: `모든 기록 변경` 선택 (또는 필요에 따라 특정 조건 추가 가능)

    트리거 이름 지정 (예: `History Change - Page View`) 후 저장
3. 태그에 트리거 연결:
    - 앞서 만든 Logger 기본 분석 스크립트 태그 (맞춤 HTML) 편집 화면으로 돌아갑니다.
    - 트리거링 섹션 클릭
    - 기존에 'All Pages' 트리거가 있다면 제거합니다.
      다만, 환경에 따라 초기 로드 시 페이지뷰 트리거만 동작하는 경우
      페이지뷰 트리거와 기록 변경 트리거를 같이 설정해야 합니다.
      (SPA에서는 초기 로드 외 페이지 이동 시 작동하지 않음)
    - 방금 만든 `History Change - Page View` 트리거를 선택하고 추가합니다.

- 중요**:** 초기 페이지 로드 시에도 스크립트가 실행되어야 하므로, **'All Pages'** 트리거도 함께 추가해주는 것이 좋습니다. 즉, `History Change - Page View` 와 `All Pages` 두 개의 트리거를 설정합니다.
- 태그 이름 지정 (예: `HTML - Logger Base Script`) 후 저장

이제 GTM 컨테이너를 게시하면 React 사이트의 모든 페이지뷰(초기 로드 및 가상 페이지뷰)가 Logger로 전송됩니다.

2. (선별) 환경변수 스크립트

특정 사용자 행동(버튼 클릭, 회원가입 완료, 구매 완료 등)을 추적하고 관련 데이터를 Logger로 보내려면 환경변수를 사용합니다. React 환경에서는 `dataLayer`를 통해 동적 데이터를 GTM으로 전달하는 것이 가장 효율적입니다.

환경변수 스크립트는 각 변수값이 필요한 웹페이지에만 선별적으로 삽입하며, 기본 스크립트보다 위쪽에 위치되도록 삽입합니다.

2.1. Data Layer를 이용한 데이터 전달 **(React -> GTM)**

React 컴포넌트 내에서 특정 이벤트가 발생했을 때, `window.dataLayer.push()` 함수를 사용하여 GTM으로 데이터를 보냅니다. GTM에서 dataLayer 변수로 바인딩해서 사용합니다.

2.1.1 (전환) 인기 상품 측정

적용 스크립트 :

None

```
// ProductDetail.js (React Component)
import React, { useEffect } from 'react';

const ProductDetail = ({ product }) => {
  useEffect(() => {
    // 컴포넌트 마운트 시 (또는 상품 정보 로드 완료 시) 데이터 전송
    if (!product) return;
    window.dataLayer = window.dataLayer || [];
    window.dataLayer.push({
      'event': 'loggerProductView', // GTM 트리거 이벤트 이름
      'loggerPI': 'PDV', // Logger _TRK_PI 변수 값
      'loggerPN': product.productId,   // Logger _TRK_PN 변수 값
      'loggerPND': product.productName     // Logger _TRK_PND 변수 값
    });
  }, [product]); // product 정보가 변경될 때마다 실행

  return (
    <div>
      <h1>{product.productName}</h1>
      <p>카테고리: {product.categoryName}</p>
```

```
      {/* ... 상품 상세 정보 ... */}
    </div>
  );
}

export default ProductDetail;
```

2.1.2 (전환) 구매정보 측정

적용 스크립트 :

None

```
// CheckoutComplete.js (React Component)
import React from 'react';

const CheckoutComplete = ({ orderDetails }) => {
  const handlePurchaseComplete = () => {
    if (!orderDetails || !orderDetails.items ||
orderDetails.items.length === 0) return;

    // Logger에서 기대하는 형식: 세미콜론으로 이어진 값
    const loggerOA = orderDetails.items.map(item => item.price *
item.quantity).join(';');
    const loggerOE = orderDetails.items.map(item =>
item.quantity).join(';');
    const loggerOP = orderDetails.items.map(item =>
item.productId).join(';');

    // dataLayer에 이벤트 push
    window.dataLayer = window.dataLayer || [];
    window.dataLayer.push({
      event: 'loggerOrderResult', // GTM 트리거 이벤트 이름
      loggerPI: 'ODR',            // Logger _TRK_PI 변수 값
```

```
        loggerOA,                    // Logger _TRK_OA 변수 값
        loggerOE,                    // Logger _TRK_OE 변수 값
        loggerOP                     // Logger _TRK_OP 변수 값
      });
    };

    // 컴포넌트 렌더링 후 또는 특정 액션 후 handlePurchaseComplete 호출
    useEffect(() => {
        handlePurchaseComplete();
    }, [orderDetails]);


    return (
      <div>
        <h2>구매 완료</h2>
        <p>주문 번호: {orderDetails.orderId}</p>
        {/* ... 주문 완료 정보 ... */}
      </div>
    );
}

export default CheckoutComplete;
```

- `event`: GTM에서 이벤트를 감지하는 데 사용될 고유한 이름입니다.
- `loggerOA`, `loggerOP` 등: Logger 환경변수에 매핑될 데이터 키입니다. 명확하게 이름을 지정하는 것이 좋습니다.

2.1.3 (전환) 회원가입수 측정

적용 스크립트 :

None

```
import React, { useEffect } from 'react';

const JoinResult = ({ user }) => {
  useEffect(() => {
```

```
    window.dataLayer = window.dataLayer || [];
    window.dataLayer.push({
      event: 'loggerRegisterComplete', // GTM 트리거 이벤트 이름
      loggerPI: 'RGR',                 // Logger _TRK_PI 변수 값
      loggerSX: user?.gender || '',    // 성별 (예: 'M', 'F', 'U')
      loggerAG: user?.ageGroup || '',  // 연령대 (예: 'A', 'B', 'C', ...)
    });
  }, [user]);

  return (
    <div>
      <h2>회원가입 완료</h2>
      {/* ... 회원 정보 ... */}
    </div>
  );
};

export default JoinResult;
```

2.1.3 (전환) 상담신청수 측정

적용 스크립트 :

```
None

import React, { useEffect } from 'react';

const ReservationComplete = () => {
  useEffect(() => {
    window.dataLayer = window.dataLayer || [];
    window.dataLayer.push({
      event: 'loggerReservationComplete', // GTM 트리거 이벤트 이름
      loggerPI: 'RES'                     // Logger _TRK_PI 변수 값
    });
```

```
  }, []);

  return (
    <div>
      <h2>상담 신청이 완료되었습니다.</h2>
    </div>
  );
};

export default ReservationComplete;
```

2.1.4 (전환) 버튼 클릭수 전환 측정

적용 스크립트 :

None

```
import React from 'react';

const InquiryButton = () => {
  const handleClick = () => {
    try {
      if (typeof _trk_flashEnvView === 'function') {
        _trk_flashEnvView(
          '_TRK_PI=ODR',
          '_TRK_OP=ASAN',
          '_TRK_OE=1',
          '_TRK_OA=1'
        );
      }
    } catch (e) {
      console.error('Tracking failed:', e);
    }
  };
```

```
  return (
    <button onClick={handleClick}>
      문의하기
    </button>
  );
};

export default InquiryButton;
```

2.1.5 (전환) 인기 컨텐츠 측정

적용 스크립트 :

None

```
import React, { useEffect } from 'react';

const ExamplePage = () => {
  useEffect(() => {
    const contentCategory = '/회사소개/제휴안내/'; // 카테고리 경로 (필요에
따라 변경)

    window.dataLayer = window.dataLayer || [];
    window.dataLayer.push({
      event: 'loggerContentView',  // GTM 트리거 이벤트 이름
      loggerCP: contentCategory    // Logger _TRK_CP (ex.
/회사소개/제휴안내/)
    });
  }, []);

  return (
    <div>
      <h1>컨텐츠 페이지</h1>
```

```
      <p>이 페이지는 컨텐츠 페이지입니다.</p>
    </div>
  );
};

export default ExamplePage;
```

2.1.6 회원 특성 분류

적용 스크립트 :

```
None
```

```
import React, { useEffect } from 'react';

const SignupComplete = ({ user }) => {
  useEffect(() => {
    window.dataLayer = window.dataLayer || [];

    window.dataLayer.push({
      event: 'loggerMemberAttribute', // GTM 트리거 이벤트 이름
      loggerPI: 'RGR',                // Logger _TRK_PI
      loggerSX: user?.gender || 'U',  // 성별: 'M' | 'F' | 'U'
      loggerAG: user?.ageCode || ''   // 연령 코드: A~Z 중 리포트에 매핑된
값
    });
  }, [user]);

  return (
    <div>
      <h2>회원가입이 완료되었습니다.</h2>
    </div>
  );
};
```

```
export default SignupComplete;
```

2.1.7 내부 검색어 측정

적용 스크립트 :

None

```
import React, { useEffect } from 'react';

const SearchResultsPage = ({ keyword }) => {
  useEffect(() => {
    if (!keyword) return;

    window.dataLayer = window.dataLayer || [];
    window.dataLayer.push({
      event: 'loggerInternalSearch', // GTM 트리거용 이벤트 이름
      loggerIK: keyword              // 검색어 변수
    });
  }, [keyword]);

  return (
    <div>
      <h2>“{keyword}” 검색 결과</h2>
      {/* 검색 결과 목록 렌더링 */}
    </div>
  );
};

export default SearchResultsPage;
```

2.1.8 특정 링크의 클릭수 측정

측정이 필요한 버튼에 설치합니다.

기본 인자값 정의 :

인자값	용도
EVT	이벤트, SNS 등 링크 클릭 수
DNL	자료별 다운로드 클릭 수
RFO	외부 사이트 연결 링크의 클릭 수
SCI	상품별 장바구니에 담긴 수
SCO	상품별 장바구니에서 삭제된 수

적용 스크립트 :

None

```
import React from 'react';

const TrackableLink = ({ type, label, children, ...props }) => {
  const handleClick = () => {
    try {
      if (typeof _trk_clickTrace === 'function') {
        _trk_clickTrace(type, label); // 예: 'EVT', '메인/봄기획전'
      }
    } catch (e) {
      console.error('Tracking error:', e);
    }
  };

  return (
    <a onClick={handleClick} {...props}>
      {children}
    </a>
```

```
  );
};

export default TrackableLink;
```

2.1.9 방문자 이동경로 시나리오 측정

시나리오로 정의할 특정 웹페이지에 설치합니다.

기본 코드값 정의 :

코드값	용도
RGI	회원가입 약관 화면 (Register Information)
RGF	회원가입 양식 화면 (Register Form)
RGR	회원가입 결과 화면 (Register Result)
LIF	로그인 화면 (Log In Form)
LIR	로그인 결과 화면 (Log In Result)
PLV	상품 목록 화면 (Product List View)
PDV	상품 상세 화면 (Product Detail View)
OCV	장바구니 보기 화면 (Order Cart View)
ODF	주문 입력 화면 (Order Form)
ODR	주문 완료 화면 (Order Result)
PNF	찾을 수 없는 페이지 (404 Not Found Error)

적용 스크립트 :

None

```
import React, { useEffect } from 'react';
```

```
const NotFoundPage = () => {
  useEffect(() => {
    window.dataLayer = window.dataLayer || [];
    window.dataLayer.push({
      event: 'loggerScenarioStep', // GTM 트리거 이벤트 이름
      loggerPI: 'PNF'              // 404 Not Found 페이지 코드
    });
  }, []);

  return (
    <div>
      <h1>페이지를 찾을 수 없습니다</h1>
    </div>
  );
};

export default NotFoundPage;
```

2.2. GTM 변수 설정 (Data Layer 변수)

React에서 `dataLayer`로 보낸 데이터를 GTM 태그에서 사용하기 위해 변수를 생성합니다.

1. GTM 작업 공간 > 변수 > 사용자 정의 변수 섹션의 새로 만들기 클릭
2. 변수 구성 클릭 > 데이터 영역 변수 선택
3. 데이터 영역 변수 이름**:** React 코드에서 `dataLayer.push` 시 사용한 키 이름 입력 (예: `loggerPI`)
4. 데이터 영역 버전**:** `버전 2` 선택 (일반적인 경우)
5. 변수 이름 지정 (예: `DLV - Logger PI`) 후 저장
6. 추적하려는 모든 **Logger** 환경변수에 대해 위 과정을 반복하여 데이터 영역 변수를 생성합니다. (예: `DLV - Logger Product`, `DLV - Logger Conversion`, `DLV - Logger Revenue` 등)

2.3. GTM 트리거 설정 (맞춤 이벤트)

`dataLayer.push` 시 함께 보낸 `event` 키 값을 사용하여 특정 이벤트가 발생했을 때만 태그가 실행되도록 트리거를 설정합니다.

1. GTM 작업 공간 > 트리거 > 새로 만들기 클릭
2. 트리거 구성 클릭 > 맞춤 이벤트 선택
3. 이벤트 이름: React 코드에서 `dataLayer.push` 시 사용한 `event` 값 입력 (예: `loggerProductView`)
4. 이 트리거 실행 시점: `모든 맞춤 이벤트` 선택 (또는 특정 조건 추가 가능)
5. 트리거 이름 지정 (예: `Custom Event - Logger Product View`) 후 저장
6. 추적하려는 모든 이벤트 종류에 대해 위 과정을 반복하여 트리거를 생성합니다. (예: `Custom Event - Logger Purchase`)

2.4. GTM 태그 설정 (맞춤 HTML - 이벤트 전송 스크립트)

이제 데이터 영역 변수와 맞춤 이벤트 트리거를 사용하여 Logger 환경변수 값을 전송하는 태그를 만듭니다.

1. GTM 작업 공간 > 태그 > 새로 만들기 클릭
2. 태그 구성 클릭 > 맞춤 **HTML** 선택
3. HTML 필드에 아래 이벤트 전송 스크립트 기본 구조를 붙여넣고, 필요한 환경변수에 맞게 수정합니다.

```
None
<script type="text/javascript">
  _TRK_PI = "{{loggerPI}}";
  _TRK_OA = "{{loggerOA}}";
  _TRK_OE = "{{loggerOE}}";
  _TRK_OP = "{{loggerOP}}";
//  if 구문 사용 시
//  if ("{{loggerPI}}" && "{{loggerPI}}" !== "undefined") _TRK_PI =
"{{loggerPI}}";
  console.log('[Logger] 전환 전송됨:', _TRK_PI, _TRK_OA, _TRK_OE,
_TRK_OP); </script>
```

- 주의:
  - `{{loggerPI}}`, `{{loggerOA}}` 등은 실제로 생성한 **GTM** 데이터 영역 변수 이름으로 바꿔야 합니다. `{{` 와 `}}` 를 포함해야 합니다.

    - if ({{...}}) 구문은 해당 데이터 영역 변수에 값이 있을 때만 Logger 변수를 설정하여 오류를 방지합니다.
    - _OA 등 숫자 값은 Logger 매뉴얼에 따라 문자열로 변환(String(...))하는 것이 안전할 수 있습니다. (매뉴얼 확인 필요)
    - 주문 상품 정보(_OP, _OE, _OA)는 여러 상품일 경우 세미콜론(;)로 구분된 문자열 형태로 만들어야 합니다. (위 예시 코드 주석 참고). 단일 상품인 경우 직접 매핑할 수 있습니다.
4. 트리거링 설정**:**
    - 트리거링 섹션 클릭
    - 이 태그를 실행시킬 맞춤 이벤트 트리거 선택 (예: Custom Event - Logger Product View 또는 Custom Event - Logger Purchase)
    - 추가 클릭
5. 태그 이름 지정 (예: HTML - Logger Event - Product View 또는 HTML - Logger Event - Purchase) 후 저장
6. 필요한 모든 이벤트 종류에 대해 위 태그 생성 과정을 반복합니다. (각 이벤트에 필요한 환경변수만 포함하도록 스크립트 수정)

2.5. 주요 환경변수별 태깅 가이드 **(GTM** 기준 요약**)**

- _TRK_PI**:** 해당 웹페이지 성격 전달
    - **React:** dataLayer.push({'event': 'eventName', 'loggerPI': 'ODR'});
    - **GTM** 변수**:** loggerPI
    - **GTM** 태그 스크립트**:** _TRK_PI = "{{loggerPI}}";
    - 설명**:** 이벤트의 전환 유형을 지정합니다. (예: 'ODR'(구매 완료), 'PDV'(상품상세))
- _TRK_PN**:** 상품 코드 전달
    - **React:** dataLayer.push({'event': 'eventName', 'loggerPN': 'P1234'});
    - **GTM** 변수**:** loggerPN
    - **GTM** 태그 스크립트**:** _TRK_PN = "{{loggerPN}}";
    - 설명**:** 이벤트 대상의 고유 ID(상품번호 등).
- _TRK_PND**:** 상품 명 전달

    - **React:** `dataLayer.push({'event': 'eventName', 'loggerProduct': '스마트폰 V'});`
    - **GTM 변수:** `loggerPND`
    - **GTM 태그 스크립트:** `_TRK_PND = "{{loggerPND}}";`
    - 설명: 이벤트 대상의 이름 또는 제목
- `_TRK_OA` : 주문 금액 전달
    - **React:** `dataLayer.push({'event': 'addToCart', 'loggerOA': '3000;800;500'});`
    - **GTM 변수:** `loggerOA`
    - **GTM 태그 스크립트:** `_TRK_OA = "{{loggerOA}}";`
    - 설명: 상품별 주문 금액 (여러 상품은 ;로 구분)
- `_TRK_OE` : 주문 수량 전달
    - **React:** `dataLayer.push({'event': 'addToCart', 'loggerOE': '10;4;1'});`
    - **GTM 변수:** `loggerOE`
    - **GTM 태그 스크립트:** `_TRK_OE = "{{loggerOE}}";`
    - 설명: 상품별 수량 (여러 상품은 ;로 구분)
- `_TRK_OP` : 주문 상품코드 전달
    - **React:** `dataLayer.push({'event': 'purchase', 'loggerOP': 'A001;B002;C003''});`
    - **GTM 변수:** `loggerOP`
    - **GTM 태그 스크립트:** `_TRK_OP = "{{loggerOP}}";`
    - 설명: 상품별 코드 (여러 상품은 ;로 구분)
- `_TRK_CP` : 카테고리 전달
    - **React:** `dataLayer.push({'event': 'loggerContentView', 'loggerCP': '/회사소개/제휴안내'});`
    - **GTM 변수:** `loggerCP`
    - **GTM 태그 스크립트:** `_TRK_CP = "{{loggerCP}}";`
    - 설명: 페이지 카테고리 전달
- `_TRK_MF` : 상품의 브랜드명 전달
    - **React:** `dataLayer.push({'event': 'loggerProductView', 'loggerMF': 'ABC 상사'});`
    - **GTM 변수:** `loggerMF`
    - **GTM 태그 스크립트:** `_TRK_MF = "{{loggerMF}}";`
    - 설명: 특정 제조사 상품이 노출된 횟수
- `_TRK_SX` : 회원성별 전달
    - **React:** `dataLayer.push({'event': 'loggerRegistration', 'loggerSX': 'M'});`

    - **GTM 변수:** `'loggerSX'`
    - **GTM 태그 스크립트:** `_TRK_SX = "{{'loggerSX'}}";`
    - 설명: 회원성별 전달
- `_TRK_AG` : 회원특성 전달
    - **React:** `dataLayer.push({'event': 'loggerRegistration',  'loggerAG': 'A'});`
    - **GTM 변수:** `loggerAG`
    - **GTM 태그 스크립트:** `_TRK_AG= "{{loggerAG}}";`
    - 설명: 회원특성 전달
- `_TRK_IK` : 내부 검색어 전달
    - **React:** `dataLayer.push({'event': 'loggerInternalSearch',  'loggerIK': 'keyword'});`
    - **GTM 변수:** `loggerIK`
    - **GTM 태그 스크립트:** `_TRK_IK= "{{loggerIK}}";`
    - 설명: 내부 검색어 전달

3. React 애플리케이션 내 **Data Layer Push** 예시 **(**버튼 클릭**)**

JavaScript

None

```
import React from 'react';

function MyButtonComponent({ productId }) {
  const handleAddToCart = () => {
    try {
      if (typeof _trk_clickTrace === 'function') {
        _trk_clickTrace('SCI', productId);
      }
    } catch (e) {
      console.error('Logger clickTrace error:', e);
    }

    // 실제 장바구니 로직
    alert('장바구니에 추가되었습니다!');
  };

  return (
    <button onClick={handleAddToCart}>
      장바구니 담기
    </button>
  );
}

export default MyButtonComponent;
```

이 버튼을 클릭하면 Logger의 _trk_clickTrace() 함수가 호출되어, 해당 클릭 이벤트(장바구니 담기)가 Logger 서버로 직접 전송됩니다.

4. 테스트 및 디버깅

1. **GTM** 미리보기 모드**:** GTM 작업 공간 우측 상단의 미리보기 버튼을 클릭하고 React 웹사이트에 접속합니다.
2. **Tag Assistant** 확인**:** 브라우저 하단 또는 별도 창에 GTM Tag Assistant가 나타납니다.
    - 페이지 이동 시 `History` 이벤트가 발생하는지, 그리고 Logger 기본 스크립트 태그(`HTML - Logger Base Script`)가 실행되는지 확인합니다.
    - 이벤트(버튼 클릭 등)를 발생시켰을 때, `dataLayer`에 설정한 `event` 이름 (예: `loggerAddToCart`)이 좌측 이벤트 목록에 나타나는지 확인합니다.
    - 해당 이벤트를 클릭했을 때, 연결된 Logger 이벤트 태그 (예: `HTML - Logger Event - Add To Cart`)가 **Tags Fired** 섹션에 나타나는지 확인합니다.
    - 실행된 태그를 클릭하고 **Variables** 탭에서 데이터 영역 변수(`DLV - ...`)들이 React에서 보낸 값으로 제대로 채워졌는지 확인합니다.
    - **Data Layer** 탭에서 `dataLayer.push`로 전송된 데이터 구조를 직접 확인할 수 있습니다.
3. 브라우저 개발자 도구 콘솔**:** 태그 내에 추가한 `console.log` 메시지 (예: `Logger Event Sent: ...`)가 정상적으로 출력되는지 확인합니다. 오류 메시지가 없는지도 확인합니다.
4. **Logger** 실시간 로그**:** Logger 서비스에 로그인하여 실시간 로그 메뉴에서 페이지뷰 및 이벤트 데이터가 정상적으로 수집되는지 확인합니다. 환경변수 값이 제대로 들어오는지 상세히 확인합니다.

중요**:**

- 모든 GTM 변경사항은 제출 및 게시해야 실제 웹사이트에 적용됩니다.
- Logger 환경변수의 정확한 사용법과 제약사항은 Logger 공식 매뉴얼을 반드시 참고하십시오.
- 데이터 구조나 전송 방식은 웹사이트의 구체적인 구현에 따라 달라질 수 있습니다. 위 가이드는 일반적인 React + GTM 환경을 기준으로 작성되었습니다.

5. GTM 미사용 시 직접 설치 방법 (React)

React와 같은 SPA(Single Page Application) 환경에서는 페이지 이동 시 전체 페이지가 로드되지 않기 때문에, 스크립트가 최초 1회 이후 재실행되지 않는 문제가 발생할 수 있습니다.
이와 같은 문제들을 해결하기 위해 전역 변수 상태 및 스크립트 실행 시점에 대한 제어가 필요합니다.
React Router의 라우팅 변화 감지 및 React 훅을 활용한 적절한 처리 방식이 요구됩니다.

5.1. _BSTRK_LOADED 전역변수로 인한 스크립트 재실행 제한

기본 스크립트가 호출하는 bstrk.1 스크립트는 동일한 추적코드가 여러 번 실행되는 것을 방지하기 위해 window._BSTRK_LOADED가 정의되어 있지 않은 경우에만 로거 스크립트의 본문이 실행되도록 제한하고 있습니다.

React 환경에서는 페이지 전환 시 전체 페이지가 리로드 되지 않으므로 초기 페이지 진입 시에만 스크립트가 실행되고 이후 라우팅 변경 시 스크립트가 실행되지 않습니다.

라우팅 변경 시 React Router의 useEffect와 같은 훅을 이용해 _BSTRK_LOADED = undefined 로 초기화할 필요가 있습니다.

아래는 기본 스크립트에 대한 예시 코드입니다.

TypeScript
```
import { useEffect } from "react";
import { useLocation } from "react-router-dom";

const LoggerScript = () => {
  useEffect(() => {
    // 1. 전역 설정 변수 삽입
    const inlineConfig = document.createElement("script");
    inlineConfig.id = "logger_서비스번호_tracker_id";
    inlineConfig.type = "text/javascript";
    inlineConfig.text = `
      var _TRK_LID = "서비스번호";
      var _L_TD = "ssl.logger.co.kr";
      var _TRK_CDMN = "";
```

```
    `;
    document.head.appendChild(inlineConfig);

    // 2. _BSTRK_LOADED 초기화
    if (typeof window._BSTRK_LOADED !== "undefined") {
      window._BSTRK_LOADED = undefined;
    }

    // 3. 스크립트 동적 삽입
    const script = document.createElement("script");
    script.id = "logger_서비스번호_tracking_script";
    script.type = "text/javascript";
    script.async = true;
    script.src = "//fs.bizspring.net/fs4/bstrk.1.js";
    document.head.appendChild(script);

    // 4. 클린업
    return () => {
      document.getElementById("logger_서비스번호_tracker_id")?.remove();

document.getElementById("logger_서비스번호_tracking_script")?.remove();
    };
  }, [location.pathname]);

  return null;
};

export default LoggerScript;
```

5.2. 스크립트 실행 순서 조정 필요

로거 스크립트에는 기본 스크립트와 환경변수 스크립트가 존재합니다.
환경변수 스크립트는 각 변수값이 필요한 웹페이지에만 선별적으로 삽입하며 완료 페이지가 존재하는 경우 환경변수 스크립트가 기본 스크립트보다 먼저 실행되어야 합니다.
전환 정보가 기본 스크립트 로딩 시점에 이미 설정되어 있어야 정확한 수집이 가능하기

때문입니다.

완료 페이지가 없는 경우 버튼 클릭이나 fetch / axios 등의 비동기 응답 시점에 _trk_flashEnvView를 호출해 데이터를 수집합니다.

아래는 결제 완료 시 _trk_flashEnvView를 사용하는 경우에 대한 예시 코드입니다.

```typescript
const handleConversion = () => {
  try {
    if (typeof window._trk_flashEnvView === 'function') {
      window._trk_flashEnvView(
        '_TRK_PI=ODR',                    // 전환 유형 (예: 주문)
        '_TRK_OP=A001;B002;C003',         // 상품 코드 (세미콜론으로 구분)
        '_TRK_OE=2;1;5',                  // 각 상품 수량
        '_TRK_OA=1000;500;3000'           // 각 상품 금액
      );
    }
  } catch (e) {
    console.error('Tracking error:', e);
  }
};

<button onClick={handleConversion}>상담 신청</button>
```

5.3. 데이터 검증 방법

1. 리포트 로그인 후 [데이터검증TOOL™] 에서 데이터 검증할 사이트 선택 후 [조회] 클릭하여 작동 시킨다.





2. 사이트 방문 후 [데이터검증TOOL™] 표시되는 정보 확인한다.



· 방문정보

Session ID	방문일시	방문경로(Referrer)	CPC광고매체	CPC키워드	캠페인	브라우저
[redacted]	2014-10-20 17:14:59	http://search.naver.com/search.naver?sm=tab_hty.top&where=nexearch&ie=utf8&query=11AM				Microsoft Internet Explorer 11

· 페이지정보

Session ID	방문일시	페이지URL	카테고리(_CP)	페이지정의(전환단계)	주문상품명(_OP)	주문상품수량(_OE)	주문상품금액(_OA)	성별(_SX)	회원특성(_AG)	내부검색어(_IK)
[redacted]	2014-10-20 17:14:59	[redacted]								

[참고 사항]

- 위와 같이 표시되지 않는다면 ① 스크립트 설치 상태를 다시 확인해 주시거나, ② [분석제외ip]등록된 것이므로 각각의 내용을 확인합니다.
- [데이터검증tool]은 본인 IP에서만 실시간 검증이 가능합니다.
- 데이터 검증 시에는 시크릿모드 또는 쿠키를 삭제하신 후 검증을 진행을 해주시기 바랍니다.

이 가이드가 BizSpring Logger를 React 웹사이트에 성공적으로 연동하는 데 도움이 되기를 바랍니다.

LOGGER™ REFERENCE LIBRARY

Contact

발행처	㈜비즈스프링
발행인	로거 사업 운영 부문
제품	LOGGER™
문의	http://www.logger.co.kr
	service@logger.co.kr