readme.md for @social.io/interfaces

An interface package for social.io.

Install

To install the `@social.io/interfaces` package, you can use npm. Make sure you have Node.js installed, then run the following command in your terminal:

```
npm install @social.io/interfaces
```

This will fetch the latest package from the npm registry and add it to your project's dependencies.

Usage

The `@social.io/interfaces` package provides a set of interfaces to be used with the social.io application, designed to standardize the communication and data exchange within the application. To effectively use this package, it's recommended to employ TypeScript, which provides advanced features like intellisense and type safety, promoting a better development experience.

Setting Up Your Project

Firstly, ensure that you set up your project to work with ES Modules and optionally TypeScript. If you are working with JavaScript, ensure your `package.json` file contains the line `"type": "module"`. For TypeScript, you might want to add configuration in `tsconfig.json` as follows:

```
{
  "compilerOptions": {
    "module": "ESNext",
    "target": "ESNext",
    "moduleResolution": "Node",
```

```
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true
  }
}
```

Importing Interfaces

The package exports various interfaces for managing social.io data and operations. You can import them using:

```
import {
  ISioConversation,
  ISioSession,
  IRequest_GetSocialSession,
  IRequest_AttachProfileId,
  IRequest_GetConversations
} from '@social.io/interfaces';
```

Managing Sessions

The package provides an interface `IRequest_GetSocialSession` which models a request to get a social session. This can be useful when you need to manage user sessions:

```
const getSessionRequest: IRequest_GetSocialSession = {
  method: 'getSocialSession',
  request: {
    existingSessionId: 'currentSessionId123'
  },
  response: {
    newSessionId: 'newSessionId456'
  }
};

// Utilize getSessionRequest in your session management logic
```

This follows a pattern of detailing a request/response structure, allowing you to handle sessions more seamlessly.

Attaching Profile IDs

The interaction with user profiles is essential in most social applications. Use `IRequest_AttachProfileId` to attach a profile to a session:

```
const attachProfileRequest: IRequest_AttachProfileId = {
  method: 'attachProfileId',
  request: {
    sessionId: 'newSessionId456',
    profileId: 'userProfileId789'
  },
  response: {
    newSessionId: 'updatedSessionId012'
  }
};

// Implement logic to leverage attachProfileRequest for profile operations
```

These types ensure strict adherence to your backend contract, reducing runtime errors and improving code quality.

Handling Conversations

Conversations are a critical part of any social platform. The package defines various interfaces to accommodate complex conversation structures:

```
const conversation: ISioConversation = {
  subject: 'Chat about TypeScript best practices',
  parties: [
    { id: 'user1', name: 'Alice', description: 'TypeScript enthusiast' },
    { id: 'user2', name: 'Bob', description: 'Node.js guru' }
  ],
  conversationBlocks: [
    { partyId: 'user1', text: 'I love using TypeScript with Node.js!' },
    { partyId: 'user2', text: 'Same here! It makes development so much safer and robust.' }
  ]
};

// These conversations can be leveraged in your business logic to create engaging user
```

```
experiences
```

This interface allows for a detailed conversational structure including multiple participants and conversation blocks.

Extending Sessions and Conversations

For further extensions, the `ISioSession` interface is key, providing session-level information like tenant IDs, activity status, and user profiles:

```
const session: ISioSession = {
  tenantId: 'socialTenant123',
  active: true,
  abandoned: false,
  markedForDeletion: false,
  profileInfo: {
    profileId: 'profileId1234',
    name: 'Alice Johnson',
    email: 'alice@example.com',
    mobilePhone: '+1234567890'
  },
  conversations: [conversation]
};

// Handle session operations and data management through this detailed session structure
```

This level of detail in interfaces encourages robust software design by promoting thorough type usage and better communication through shared types.

Requests Pattern

The package uses a consistent pattern by inheriting from a base typed request interface, promoting consistency and reusability:

```
import * as plugins from '@social.io/interfaces';

export interface IRequest_GetArrayExample
  extends plugins.typedrequestInterfaces.implementsTR<
    plugins.typedrequestInterfaces.ITypedRequest,
    IRequest_GetArrayExample
```

```
  > {
  method: 'getArrayExample';
  request: {
    itemCount: number;
  };
  response: {
    items: Array<string>;
  };
}
```

Such patterns simplify request and response management due to their predictability and standardization across different modules.

By following these patterns, you can build scalable, maintainable, and highly reliable applications centered around the social.io interfaces. Always remember to check the comprehensive module documentation and integrate the above interfaces into your TypeScript-first, node-based projects. undefined

---

Revision #3
Created 2026-03-28 11:14:52 UTC by foss.global Team
Updated 2026-03-28 12:21:38 UTC by foss.global Team