info

Description	Получение информации о приложении, миниаппе и sdk

Namespace

moby.info.*

Description (d.ts)

```
export module info
{
  interface AppInfo {
    locale?: LocaleDetails;
    url: string;
    version: string; // Номер версии
    build: string; // Номер сборки
    bundle: string;
    id: string; // Идентификатор
  }

  interface MiniappInfo {
    locale?: LocaleDetails;
    id: string; // Идентификатор
    name: string;
    version: string; // Номер версии
    tags?: string[];
  }

  interface LocaleDetails {
    currencyCode: string; // Код валюты
    currencySymbol: string; // Символ валюты
    decimalSeparator: string; // Десятичный разделитель
    groupingSeparator: string; // Разделитель групп
    identifier: string; // Идентификатор
    languageCode: string; // Код языка
    regionCode: string; // Код региона
  }


  export function miniapp(): Promise<MiniappInfo>;
```

```
  export function app(): Promise<AppInfo>;
  export function sdk(): {
              version: string;
          };
}
```

Example

```
const f = async() => {
  const app = await moby.info.app();
  const miniapp = await moby.info.miniapp();
  const sdk = moby.info.sdk();

  console.log('bundle: ' + app.bundle);
  console.log('miniapp: ' + miniapp.id);
  console.log('sdk v' + sdk.version);
}
```

camera

Description	Работа с камерой

Namespace

moby.camera.*

Description (d.ts)

```
export module camera {
    export function scan_qr(): Promise<string>; // Сканирование QR-кода
    interface CardDetails {
        pan: string; // PAN карты
        month?: string; // Месяц, в котором истекает срок действия карты
        year?: string; // Год, в котором истекает срок действия карты
        name?: string; // Имя владельца карты
    }
    export function scan_emv(): Promise<CardDetails>;
}
```

Example

```

```

sqlite

Description	Работа с базами данных SQLite

Namespace

moby.sqlite.*

Description (d.ts)

```
export module sqlite {
      function run(sql: string): Promise<any>;
      function get(sql: string): Promise<any>;
      function all(sql: string): Promise<any>;
  }
```

Example

```

```

auth

Description	Авторизация пользователя на платформе moby и внешних сервисах

Namespace

moby.auth.*

Description (d.ts)

```
  export module auth {
        interface MobyToken {
            jwt?: JwtToken; // JSON Web Token
            info: TokenInfo; // Информация о токене
            expire?: Date; // Абсолютное время истечения токена по GMT+0
        }
        interface JwtToken {
            access_token: string; // Токен для получения данных ответа на запрос
            external_token: string; // Токен для доступа к внешним сервисам, опционален
            refresh_token: string; // Refresh токен для обновления основного токена доступ
a
            token_type: string; // Заголовок для токена, например 'Bearer'
            expire: string; // Относительное время истечения токена
        }
        interface TokenInfo {
            type: string | null; // Тип токена
            uid: string; // guid пользователя из токена
        }
        export module user {
            function get_token(): Promise<MobyToken>; // Получение текущего сессионного то
кена
            function is_token_exist(): Promise<boolean>;  // Проверка наличия токена
            function logout(): Promise<boolean>; // Логаут
            module sms {
                function login(number: string): Promise<boolean>; // Авторизация для moby
 пользователя через sms
                function request_code(): void; // Запрос OTP кода для 2FF авторизации
                function confirm_code(code: string): Promise<boolean>; // Подтверждение 2F
F авторизации
```

```
            }
            module openid {
                function login(url: string): Promise<boolean>; // Авторизация для moby пол
ьзователя через openid
            }
        }
        export module service {
            function is_token_exist(url: string): Promise<boolean>; // Проверка наличия то
кена для данного внешнего сервиса
            function login(url: string): Promise<boolean>;
            function get_token(url: string): Promise<MobyToken>; // Получение текущего сес
сионного токена
            function logout(url: string): Promise<boolean>; // Логаут
        }
        export {};
    }
```

Example

```
```

miniapp

Description	Реализация навигации между miniapps в приложении

Namespace

moby.miniapp.*

Description (d.ts)

```
    export module miniapp {
        function get_init_data(): Promise<string>; // Получить данные из инициализации окн
а. Используются при передаче данных из одного миниапп в другой.

        function open(uuid: string, init_data: string): Promise<any>; // Открыть модальное
окно

        function close(response_data: string): Promise<any>; // Закрыть модальное окно

        function show(uuid: string, init_data: string): Promise<any>; // Открыть новое окн
о в цепочке

        function pop(to_root: boolean, response_data: string): Promise<any>; // Закрыть ок
но в цепочке

        function get_installed_apps(): Promise<any>; // Получение списка установленных мин
иаппов на текущий момент

        function get_app_icon(uuid: string, type: string): Promise<any>; // Получение байт
ового массива иконки из miniapp. Типы: 'menu' / 'sidebar'
    }
```

Example

marketplace

Description	Установка и удаление дополнительных пользовательских miniapps

Namespace

moby.marketplace.*

Description (d.ts)

```
export module marketplace {
    interface LinkedInfo {
        wearableId: string; // Идентификатор носимого устройства
    }
    interface InstalledInfo {
        wearableId: string;
        installedVersionNumber: string; // Номер установленной версии
        updateAvailable: boolean; // Наличие обновления
    }
    interface ProviderInfo {
        name: string; // Имя провайдера
        email: string; // Адрес электронной почты провайдера
        site: string; // Адрес сайта провайдера
        id: string; // Идентификатор провайдера
        isDefault: boolean; // Значение по умолчанию
    }
    interface ProductTypes {
        id: number; // Идентификатор продукта
        name: string; // Наименование продукта
        iconUrl: string;
    }
    interface VirtualFeatures {
        isBoot: boolean;
        isDefault: boolean;
        menuIndex: number;
        sidebarIndex: number;
    }
    interface WearableFeatures {
        hasGeneralCommands: boolean; // Наличие общих команд
```

```
            hasKeys: boolean; // Наличие ключей
            hasSystemCommands: boolean; // Наличие системных команд
            hasJsCommands: boolean; // Наличие JS команд
            hasSeCommands: boolean; // Наличие SE команд
            hasFingerprint: boolean; // Наличие отпечатков пальцев
        }
        interface SupportedWearables {
            wearableId: string; // Идентификатор носимого устройства
            versionNumber: string; // Номер версии
        }
        interface InfoDetails {
            iconUrl: string;
            id: string; // Идентификатор
            created: string; // Дата создания
            provider: ProviderInfo; // Информация о провайдере
            productTypes: ProductTypes[]; // Типы продуктов
            comment: string; // Комментарий
            price: number; // Цена
            tags: string[]; // Тэги
            maxInstallationsCount: number; // Максимальное число установок
            isVirtual: boolean; // Проверка, является ли функция виртуальной
            wearableFeatures: WearableFeatures; // Носимые функции
            virtualFeatures: VirtualFeatures; // Виртуальные функции
        }
        interface ProductInfo {
            lastVersion: string; // Последняя версия
            availableInstallationSlotCount: number; // Количество доступных слотов
            linkedWith: LinkedInfo[]; // С чем слинкован
            installedOn: InstalledInfo[]; // Куда установлен
            info: InfoDetails; // Информация о продукте
            supportedWearables: SupportedWearables[]; // Поддерживаемые носимые устройства
        }
        interface Products {
            totalCount: number; // Общее количество
            list: ProductInfo[]; // Список продуктов
        }
        export function get_apps(from?: number, lenght?: number): Promise<Products>;
        export module app {
            module sub {
                function install(): Promise<unknown>; // Установка субпродукта на конкретн
ое физическое устройство пользователя
                function uninstall(): Promise<unknown>; // Удаление субпродукта с конкретн
ого физического устройства пользователя
            }
            function about(id: string): ProductInfo | undefined;
            function purchase(product: ProductInfo, device_id: string): Promise<unknown>;
 // Покупка и линковка продукта к конкретному устройству пользователя
            function install(product: ProductInfo, device_id: string): Promise<unknown>;
 // Установка продукта на конкретное устройство пользователя
            function update(product: ProductInfo, device_id: string): Promise<unknown>; //
Обновление продукта на конкретном устройстве пользователя до последней версии
            function uninstall(product: ProductInfo, device_id: string): Promise<unknown>;
// Удаление продукта с конкретного устройства пользователя
```

```
        }
    }
```

Example

```
```

share

Description	Передача пользовательских данных в другие приложения

Namespace

moby.share.*

Description (d.ts)

```
  export module share {
        function link(data: string): Promise<any>; // Позволяет пользователю поделиться сс
ылкой
        function text(data: string): Promise<any>; // Позволяет пользователю поделиться те
кстом
        function image(data: string): Promise<any>; // Позволяет пользователю поделиться и
зображением
        function file(data: string): Promise<any>; // Позволяет пользователю поделиться фа
йлом
    }
```

Example

firebase

Description	Отправка уведомлений пользователям

Namespace

moby.firebase.*

Description (d.ts)

```
  export module firebase {
       function get_token(): Promise<any>; // Возвращает токен пользователя для подписки
на персональную нотификацию (пуши) с использованием Firebase сервиса
   }
```

Example

```

```

vpn

Description	Создание клиентского VPN соединения

Namespace

moby.vpn.*

Description (d.ts)

```
 export module vpn {
        module ipsec {
            function open(server: string, user: string, pass: string, secret: string, just
_in_app: boolean): Promise<any>;
        }
        module ikev2 {
            function open(server: string, user: string, pass: string, remote_id: string, l
ocal_id: string, just_in_app: boolean): Promise<any>;
        }
        function close(): Promise<any>;
        function info(): Promise<any>;
    }
```

Example

```

```

nfc

Description	Работа с NFC

Namespace

moby.nfc.*

Description (d.ts)

```
  export module nfc {
        function open(): Promise<any>; // открытие (подготовка) NFC сессии на мобильном те
лефоне
        function close(): Promise<any>; // закрытие NFC сессии
        function exchange(capdu: string): Promise<any>; // обмен C-APDU / R-APDU командами
    }
```

Example

```

```

utils

Description	Преобразование данных

Namespace

moby.utils.*

Description (d.ts)

```
      export module utils {
          function hexstring_to_bytearray(hexString: string): any[]; // Преобразование шес
тнадцатеричной строки в массив байтов
          function bytearray_to_map(byteArray: number[]): any; // Преобразование массива б
айтов в градусы, минуты и секунды
          function hexstring_to_map(hexString: string): any; // Преобразование шестнадцате
ричной строки в градусы, минуты и секунды
          function to_hex(string: string): string; // Преобразование текста в шестнадцатер
ичные коды его символов
          function to_string(hex: string): string; // Преобразование шестнадцатеричной стр
оки в текст
      }
```

Example

```

```

devices

Description	Работа с moby устройствами

Namespace

moby.devices.*

Description (d.ts)

```
  export module devices {
          interface DeviceInfo {
          }
          export function list(): Promise<unknown>; // Получение списка текущих зарегистри
рованных устройств moby пользователя, включая virtual
          export function scan(timeout: number, handler: (list: DeviceInfo) => void): Prom
ise<DeviceInfo[]>; // Получение списка устройств с primary BLE service, соответствующему у
стройствам Moby. Таймаут задается в секундах
          export function add(data: any): Promise<unknown>; // Регистрация нового устройст
ва пользователя
          export function del(device_id: string): Promise<unknown>; // Удаление устройства
пользователя
          export function exchange(device_id: string, uuid: string, session_id: string, da
ta: any, timeout: number): Promise<unknown>; // нативный обмен с BLE устройством пользоват
еля
          export function script(device_id: string, product_id: string, script: any): Prom
ise<unknown>; // Выполнение серверного скрипта для указанного устройства
          export function connect(device_id: string, uuid: string): Promise<unknown>; // П
оключение к BLE устройству. Устройство не обязательно должно быть привязано к профилю поль
зователя
          export function disconnect(device_id: string, uuid: string): Promise<unknown>;
 // Отключение от BLE устройства.
          export {};
      }
```

Example