P2P Subscription Transfers Extension

Architecture & Implementation Plan

Document Info	
Extension Name	`p2p`
Type	AppDirect Platform Extension
Target Location	Company Details > Vendor Information (Embedded)
Author	Neil Bolton
Created	January 13, 2026
Status	Planning

Table of Contents

1. Overview

1.1 Purpose

Build the P2P Subscription Transfers feature as an **AppDirect Platform Extension** that can be:

- Embedded within the marketplace UI
- Deployed independently from the core platform
- Reused across multiple marketplaces
- Maintained and updated without platform releases

1.2 Key Benefits

Benefit	Description
Modularity	P2P feature is self-contained and independently deployable
Reusability	Can be installed on any AppDirect marketplace
Maintainability	Updates don't require platform releases
Scalability	Extension can evolve independently
Consistency	Uses Mantine 7 UI library (same as AppDirect)

1.3 Reference Documentation

Resource	URL
Extensions Overview	https://developer.appdirect.com/user-guides/extensions/
Getting Started	https://developer.appdirect.com/user-guides/extensions/getting-started/
UI Toolkit	https://developer.appdirect.com/user-guides/extensions/toolkit-installation/
Marketplace APIs	https://developer.appdirect.com/user-guides/extensions/using-marketplace-apis/
Context Data	https://developer.appdirect.com/user-guides/extensions/accessing-context-data/

Resource	URL
External APIs	https://developer.appdirect.com/user-guides/extensions/integrate-external-apis/
GraphQL Reference	https://developer.appdirect.com/user-guides/graphql-dev-tools/

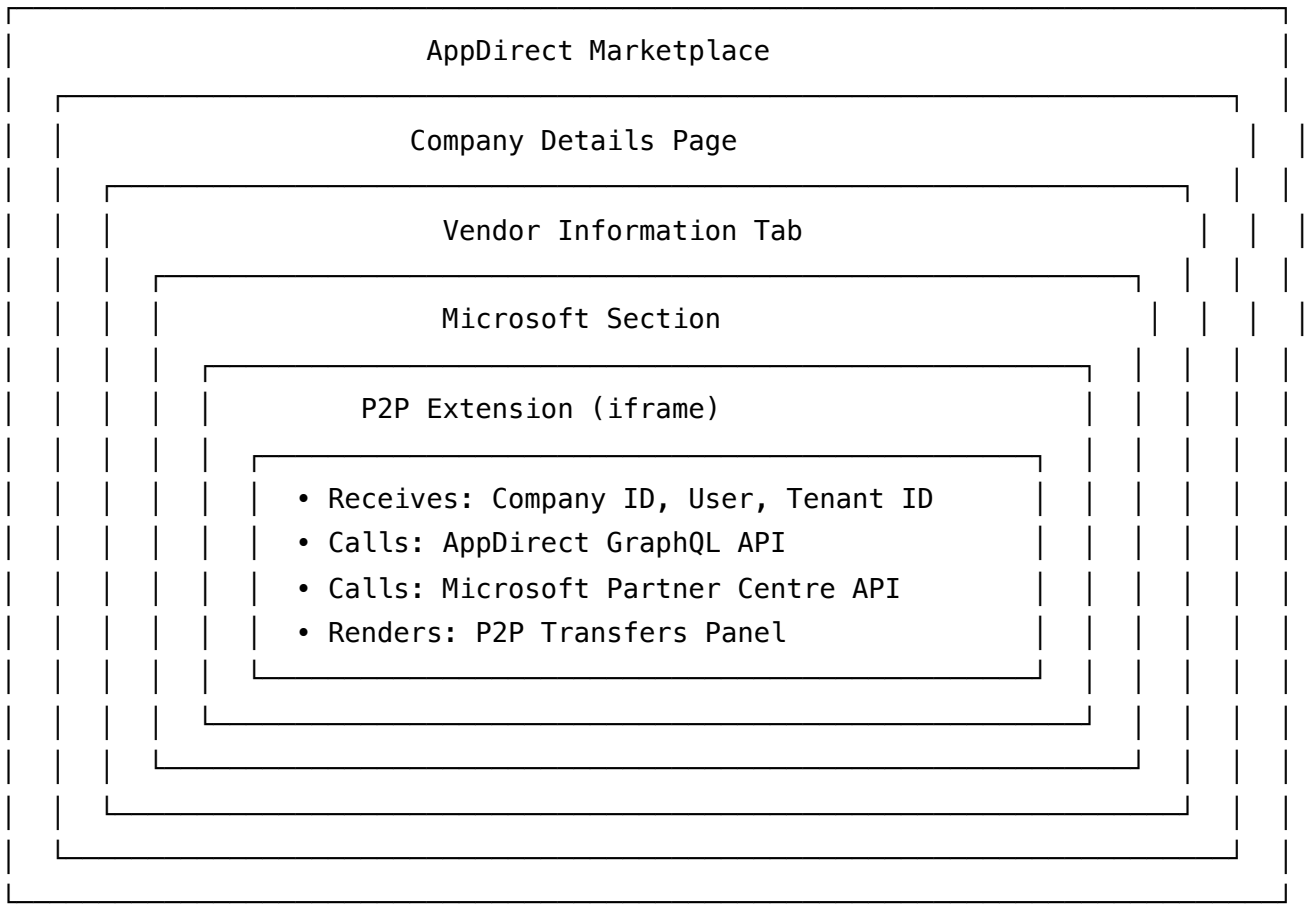
2. AppDirect Extensions Framework

2.1 What is an Extension?

An AppDirect Extension is an **embedded React application** that integrates into the marketplace platform. Extensions:

- Run within an iframe in the AppDirect UI
- Receive context data (user, company, settings) from the host
- Can call AppDirect APIs (GraphQL/REST) with inherited authentication
- Can integrate external APIs using secure token storage
- Use Mantine 7 for UI consistency

2.2 Extension Capabilities



2.3 Context Data Available

When the extension loads, it receives context from the host marketplace:

Context	Description	Use Case
`user`	Current logged-in user	Permissions, audit logging
`company`	Current company being viewed	Customer tenant ID lookup
`marketplace`	Marketplace configuration	API endpoints, branding
`settings`	Extension-specific settings	Partner credentials
`locale`	User's language preference	Localization

2.4 Technology Stack

Layer	Technology	Notes
UI Framework	React 18	Standard React app
UI Components	Mantine 7	Already integrated, same as platform
State Management	React Context / Zustand	Lightweight state
API Client	Apollo Client / Fetch	GraphQL + REST
Build Tool	Vite	Fast development
Testing	Vitest + RTL	Unit and integration

3. Extension Architecture

3.1 Directory Structure

```
extensions/p2p/
├── package.json                    # Extension dependencies
├── vite.config.ts                  # Build configuration
├── tsconfig.json                   # TypeScript config
├── index.html                      # Entry HTML
├── src/
│   ├── main.tsx                    # React entry point
│   ├── App.tsx                     # Root component
│   ├── extension.config.ts         # Extension metadata
│   │
│   ├── context/
│   │   ├── ExtensionContext.tsx    # Context from host marketplace
│   │   ├── P2PContext.tsx          # P2P-specific state
│   │   └── types.ts                # Context type definitions
│   │
│   ├── hooks/
│   │   ├── useExtensionContext.ts  # Access host context
│   │   ├── useTransfers.ts         # Transfer operations
│   │   ├── useSubscriptions.ts     # Subscription data
│   │   └── usePartnerCentre.ts     # Partner Centre API
│   │
│   ├── components/
│   │   ├── P2PTransfersPanel.tsx   # Main panel (from demo)
│   │   ├── TransferOverview.tsx    # Summary cards
│   │   ├── AvailableSubscriptions.tsx
│   │   ├── ActiveTransfers.tsx
│   │   ├── TransferHistory.tsx
│   │   ├── modals/
│   │   │   ├── CreateTransferModal.tsx
│   │   │   ├── ReviewTransferModal.tsx
│   │   │   └── TransferDetailsModal.tsx
│   │   └── shared/
│   │       ├── TransferStatusBadge.tsx
│   │       ├── SubscriptionCard.tsx
│   │       └── LoadingState.tsx
│   │
│   ├── api/
│   │   ├── graphql/
```

```
│   │   │   ├── client.ts           # Apollo client setup
│   │   │   ├── queries.ts          # GraphQL queries
│   │   │   └── mutations.ts        # GraphQL mutations
│   │   ├── partnerCentre/
│   │   │   ├── client.ts           # Partner Centre API client
│   │   │   ├── transfers.ts        # Transfer operations
│   │   │   └── subscriptions.ts    # Subscription queries
│   │   └── types.ts                # API response types
│   │
│   ├── services/
│   │   ├── transferService.ts      # Business logic
│   │   ├── notificationService.ts  # Toast notifications
│   │   └── auditService.ts         # Audit logging
│   │
│   ├── utils/
│   │   ├── formatters.ts           # Date, currency formatting
│   │   ├── validators.ts           # Input validation
│   │   └── constants.ts            # Static values
│   │
│   └── locales/
│       ├── en.json                 # English translations
│       └── index.ts                # i18n setup
│
├── public/
│   └── assets/                     # Static assets
│
└── tests/
    ├── components/                 # Component tests
    ├── hooks/                      # Hook tests
    └── api/                        # API integration tests
```

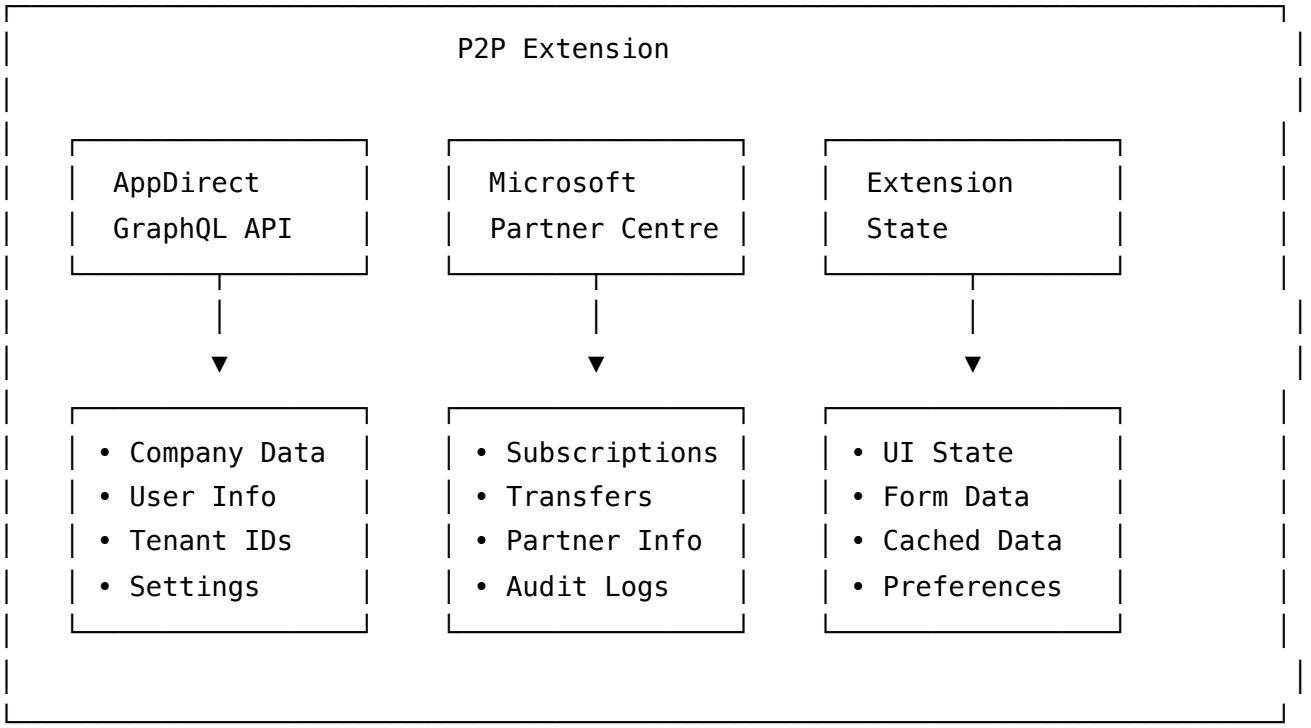
3.2 Extension Configuration

```
// src/extension.config.ts

export const extensionConfig = {
  id: 'p2p-subscription-transfers',
  name: 'P2P Subscription Transfers',
  version: '1.0.0',
  vendor: 'AppDirect',
  description: 'Manage Partner-to-Partner subscription transfers for Microsoft CSP',

  // Where this extension can be embedded
  placements: [
    {
      id: 'company-vendor-microsoft',
      location: 'company.details.vendor.microsoft',
      label: 'P2P Transfers',
    },
    {
      id: 'operations-companies',
      location: 'operations.companies.header',
      label: 'P2P Transfers',
      icon: 'arrow-left-right',
    }
  ],

  // Required permissions
  permissions: [
    'company.read',
    'company.subscriptions.read',
    'company.subscriptions.write',
    'vendor.microsoft.read',
    'vendor.microsoft.write',
  ],

  // Settings schema (configurable per marketplace)
  settings: {
    enableNotifications: {
      type: 'boolean',
      default: true,
      label: 'Enable in-app notifications',
    },
    transferExpiryDays: {
```

```
      type: 'number',
      default: 30,
      label: 'Transfer expiry period (days)',
    },
  },
};
```

4. Data Flow & Integration

4.1 Data Sources



4.2 AppDirect GraphQL Integration

The extension will use AppDirect's GraphQL API for:

Query/Mutation	Purpose
`company(id)`	Get company details including external IDs
`companySubscriptions(companyId)`	List company's Microsoft subscriptions

Query/Mutation	Purpose
`vendorIntegration(companyId, vendor)`	Get Microsoft tenant link status
`updateVendorIntegration()`	Store Partner Centre credentials

Example query:

```graphql
query GetCompanyMicrosoftData($companyId: ID!) {
  company(id: $companyId) {
    id
    name
    externalId
    vendorIntegrations(vendor: MICROSOFT) {
      tenantId
      mpnId
      gdapStatus
      lastSyncDate
    }
    subscriptions(vendor: MICROSOFT) {
      edges {
        node {
          id
          productName
          sku
          quantity
          term
          billingCycle
          status
          microsoftSubscriptionId
        }
      }
    }
  }
}
```

4.3 Microsoft Partner Centre Integration

The extension will call Partner Centre APIs via secure backend proxy:

API	Method	Purpose
`GET /customers/{customerId}/subscriptions`	Read	List transferable subscriptions
`POST /productTransfers`	Write	Create transfer request
`GET /productTransfers/{transferId}`	Read	Get transfer status
`POST /productTransfers/{transferId}/accept`	Write	Accept incoming transfer
`POST /productTransfers/{transferId}/reject`	Write	Reject incoming transfer
`DELETE /productTransfers/{transferId}`	Write	Cancel outgoing transfer

4.4 Secure Token Storage

Partner Centre credentials are stored securely using AppDirect's extension settings:

```
// Using secure token storage for Partner Centre auth
const partnerCentreConfig = await getSecureSettings('microsoft-partner-centre');

const client = new PartnerCentreClient({
  tenantId: partnerCentreConfig.tenantId,
  clientId: partnerCentreConfig.clientId,
  clientSecret: partnerCentreConfig.clientSecret, // Encrypted at rest
});
```

5. Component Structure

5.1 Component Hierarchy

```
<ExtensionProvider>                    # Provides host context
  <P2PProvider>                        # P2P-specific state
    <MantineProvider>                  # UI theming
      <NotificationsProvider>          # Toast notifications
        <App>
          <P2PTransfersPanel>          # Main container
            <TransferOverview />       # Summary cards
            <Accordion>
              <AvailableSubscriptions />
              <ActiveTransfers />
              <TransferHistory />
            </Accordion>
          </P2PTransfersPanel>

          <!-- Modals (portal) -->
          <CreateTransferModal />
          <ReviewTransferModal />
          <TransferDetailsModal />
        </App>
      </NotificationsProvider>
    </MantineProvider>
  </P2PProvider>
</ExtensionProvider>
```

5.2 Reusing Demo Components

The existing demo components will be adapted:

Demo Component	Extension Component	Changes Needed
`P2PTransfersPanel.tsx`	`P2PTransfersPanel.tsx`	Replace mock data with API hooks
`CreateTransferModal.tsx`	`modals/CreateTransferModal.tsx`	Add API mutations
`ReviewTransferModal.tsx`	`modals/ReviewTransferModal.tsx`	Add API mutations
`TransferDetailsModal.tsx`	`modals/TransferDetailsModal.tsx`	Add API queries

Demo Component	Extension Component	Changes Needed
`TransferStatusBadge.tsx`	`shared/TransferStatusBadge.tsx`	No changes
`mockData.ts`	Removed	Replaced with API calls

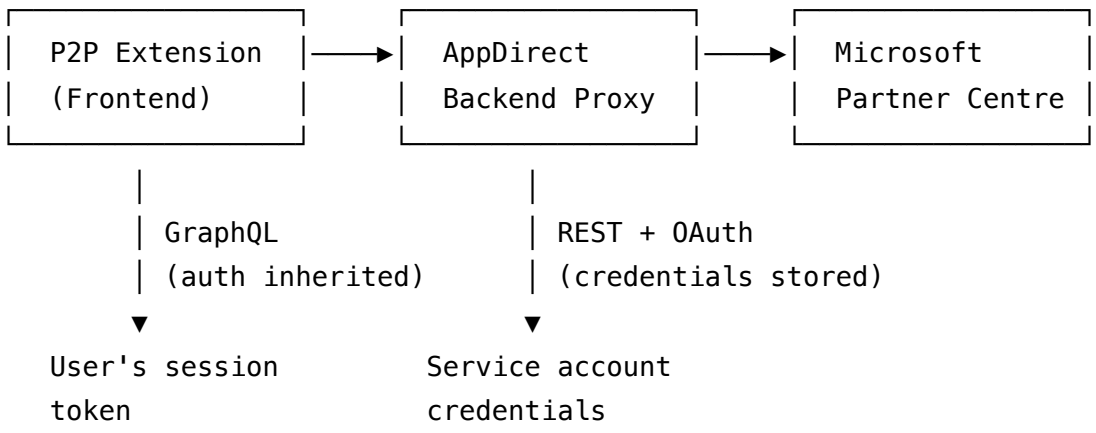
5.3 Custom Hooks

```
// useTransfers.ts - Main data hook
export function useTransfers(companyId: string) {
  const { data, loading, error, refetch } = useQuery(GET_TRANSFERS, {
    variables: { companyId },
  });

  const [createTransfer] = useMutation(CREATE_TRANSFER);
  const [acceptTransfer] = useMutation(ACCEPT_TRANSFER);
  const [rejectTransfer] = useMutation(REJECT_TRANSFER);
  const [cancelTransfer] = useMutation(CANCEL_TRANSFER);

  return {
    transfers: data?.transfers ?? [],
    incomingPending: data?.transfers.filter(t => t.direction === 'Incoming' && t.status
    outgoingPending: data?.transfers.filter(t => t.direction === 'Outgoing' && t.status
    loading,
    error,
    refetch,
    createTransfer,
    acceptTransfer,
    rejectTransfer,
    cancelTransfer,
  };
}
```

6. API Integration

6.1 Backend Proxy Architecture

For security, Partner Centre API calls go through a backend proxy:



6.2 GraphQL Schema Extensions

The extension may require GraphQL schema extensions:

```graphql
# Potential schema additions

type Transfer {
  id: ID!
  direction: TransferDirection!
  status: TransferStatus!
  sourcePartner: Partner!
  targetPartner: Partner!
  customer: Company!
  lineItems: [TransferLineItem!]!
  totalMonthlyValue: Float!
  createdDate: DateTime!
  expirationDate: DateTime!
  completedDate: DateTime
  rejectionReason: String
  auditLog: [AuditEntry!]!
}

enum TransferDirection {
  INCOMING
  OUTGOING
}

enum TransferStatus {
  PENDING
  IN_PROGRESS
  COMPLETED
  REJECTED
  CANCELLED
  EXPIRED
  FAILED
}

type Query {
  transfers(companyId: ID!, filters: TransferFilters): TransferConnection!
  transfer(id: ID!): Transfer
  transferableSubscriptions(companyId: ID!): [Subscription!]!
}

type Mutation {
  createTransfer(input: CreateTransferInput!): Transfer!
  acceptTransfer(id: ID!): Transfer!
  rejectTransfer(id: ID!, reason: String): Transfer!
```

```
  cancelTransfer(id: ID!): Transfer!
}
```

7. Security & Authentication

7.1 Authentication Flow

1. User logs into AppDirect Marketplace
2. User navigates to Company Details > Vendor Information
3. P2P Extension loads in iframe
4. Extension receives auth context from host
5. Extension calls AppDirect APIs with inherited session
6. Backend proxy calls Partner Centre with stored credentials

7.2 Permission Model

Action	Required Permission
View transfers	`company.subscriptions.read`
Create transfer	`company.subscriptions.write` + `vendor.microsoft.write`
Accept/Reject transfer	`company.subscriptions.write` + `vendor.microsoft.write`
Cancel transfer	`company.subscriptions.write` + `vendor.microsoft.write`
View audit log	`company.subscriptions.read`

7.3 Credential Storage

Partner Centre credentials are stored using AppDirect's secure settings:

- Encrypted at rest
- Scoped per marketplace
- Accessible only via backend proxy
- Never exposed to frontend

8. Development Workflow

8.1 Local Development

```
# Install AppDirect Extension Toolkit
npm install -g @appdirect/extension-toolkit

# Create extension from template
ad-ext create p2p --template react-mantine

# Start local development server
cd extensions/p2p
npm install
npm run dev

# Extension available at http://localhost:5173
# Use AppDirect dev tools to inject into marketplace
```

8.2 Testing in Marketplace

```
# Build extension for preview
npm run build

# Upload to AppDirect
ad-ext upload --preview

# Extension available in marketplace preview mode
```

8.3 Publishing

```
# Build production bundle
npm run build:prod

# Publish to marketplace
ad-ext publish --version 1.0.0

# Enable extension in marketplace settings
```

9. Implementation Phases

Phase 1: Foundation (Week 1)

Task	Description	Status
1.1	Set up extension project structure	☐
1.2	Configure Mantine theming	☐
1.3	Implement extension context provider	☐
1.4	Create base component structure	☐
1.5	Set up local development environment	☐

Phase 2: Core UI (Week 2)

Task	Description	Status
2.1	Port P2PTransfersPanel from demo	☐
2.2	Port all modal components	☐
2.3	Port shared components	☐
2.4	Implement loading states	☐
2.5	Add error handling UI	☐

Phase 3: AppDirect API Integration (Week 3)

Task	Description	Status
3.1	Set up Apollo Client	☐
3.2	Implement company data queries	☐
3.3	Implement subscription queries	☐
3.4	Connect UI to real data	☐
3.5	Test with marketplace sandbox	☐

Phase 4: Partner Centre Integration (Week 4)

Task	Description	Status
4.1	Implement backend proxy endpoints	⬜
4.2	Create Partner Centre API client	⬜
4.3	Implement transfer operations	⬜
4.4	Add secure credential storage	⬜
4.5	End-to-end testing	⬜

Phase 5: Polish & Launch (Week 5)

Task	Description	Status
5.1	Add localization support	⬜
5.2	Implement accessibility (a11y)	⬜
5.3	Performance optimization	⬜
5.4	Documentation	⬜
5.5	Publish to marketplace	⬜

10. Testing Strategy

10.1 Test Levels

Level	Tools	Coverage Target
Unit	Vitest	Components, hooks, utilities (80%+)
Integration	RTL + MSW	API interactions, state management
E2E	Playwright	Critical user flows
Visual	Chromatic	UI regression

10.2 Key Test Scenarios

Scenario	Type	Priority
Render P2P panel with transfers	Integration	High
Create outgoing transfer	Integration	High
Accept incoming transfer	Integration	High
Reject incoming transfer	Integration	High
Cancel outgoing transfer	Integration	High
Handle API errors gracefully	Integration	High
Display loading states	Unit	Medium
Validate form inputs	Unit	Medium
Permission denied handling	Integration	Medium

10.3 Mock Data Strategy

For development and testing, maintain mock data that mirrors Partner Centre responses:

```
// tests/mocks/handlers.ts
import { graphql, rest } from 'msw';

export const handlers = [
  // Mock AppDirect GraphQL
  graphql.query('GetTransfers', (req, res, ctx) => {
    return res(ctx.data({ transfers: mockTransfers }));
  }),

  // Mock Partner Centre REST
  rest.get('/api/partner-centre/transfers', (req, res, ctx) => {
    return res(ctx.json(mockPartnerCentreTransfers));
  }),
];
```

Appendix

A. Comparison: Demo vs Extension

Aspect	Current Demo	Extension
Data Source	Mock JSON	Live APIs
Authentication	None	OAuth/Session
Deployment	Standalone app	Embedded iframe
State	React useState	Context + API cache
Updates	App release	Extension publish
Reusability	Copy/paste	Install per marketplace

B. Dependencies

```
{
  "dependencies": {
    "@mantine/core": "^7.x",
    "@mantine/hooks": "^7.x",
    "@mantine/notifications": "^7.x",
    "@apollo/client": "^3.x",
    "graphql": "^16.x",
    "react": "^18.x",
    "react-dom": "^18.x",
    "lucide-react": "^0.x",
    "zustand": "^4.x",
    "i18next": "^23.x"
  },
  "devDependencies": {
    "@appdirect/extension-toolkit": "^1.x",
    "vite": "^5.x",
    "vitest": "^1.x",
    "@testing-library/react": "^14.x",
    "msw": "^2.x",
    "typescript": "^5.x"
  }
}
```

C. Revision History

Version	Date	Author	Changes
1.0	Jan 13, 2026	Neil Bolton	Initial plan

End of Document