ASMC Admin Panel - State Management Documentation

This document provides comprehensive documentation for the Redux state management system in the ASMC Admin Panel, including Redux Toolkit, RTK Query, and state patterns.

Table of Contents

- State Management Architecture
- Redux Store Configuration
- State Slices
- RTK Query APIs
- Container Pattern
- State Selectors
- Async Actions
- Error Handling
- Performance Optimization

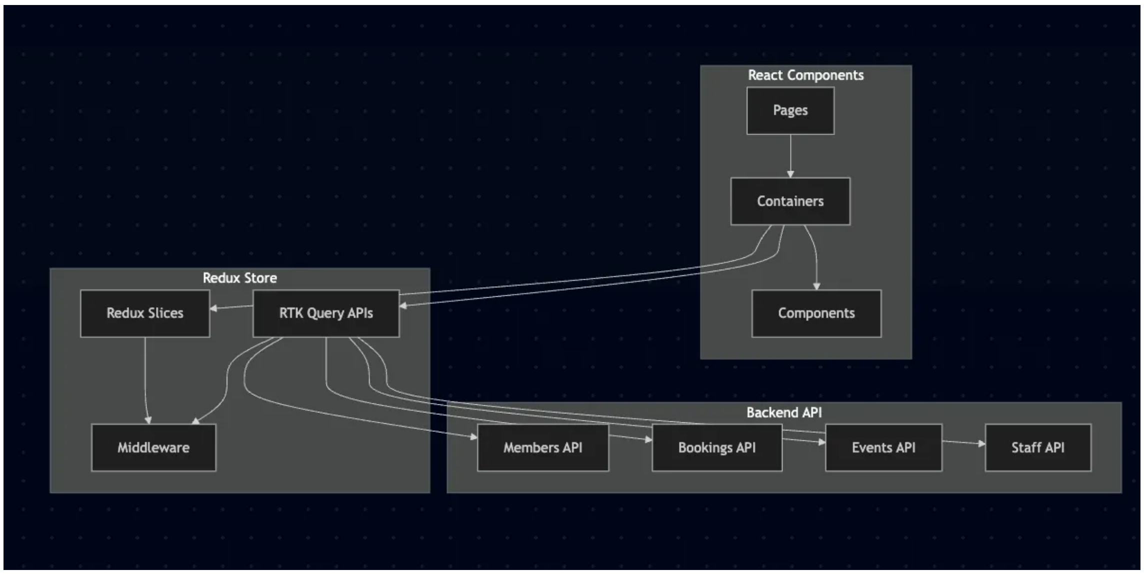
State Management Architecture

Overview

The ASMC Admin Panel uses Redux Toolkit with RTK Query for state management, providing:

- **Centralized State**: Global application state management
- **API Integration**: Seamless server state management with caching
- **Type Safety**: TypeScript integration for type-safe state
- **Developer Experience**: Redux DevTools integration
- **Performance**: Optimized re-renders and caching

Architecture Diagram



Redux Store Configuration

Store Setup

Location: `/src/store/index.js`

```
import { combineReducers, configureStore } from '@reduxjs/toolkit';
import { useSelector, useDispatch } from 'react-redux';

// RTK Query APIs
import membersApis from './members/membersApis';
import commonApis from './common/commonApis';
import plansApis from './plans/plansApis';
import facilityApis from './facility/facilityApis';
import activityApis from './activity/activityApis';
import mastersApis from './masters/mastersApis';
import bookingApis from './booking/bookingApis';
import eventsApis from './events/eventsApis';
import hallsApis from './halls/hallsApis';
import staffApis from './staff/staffApis';
import { documentationApi } from './documentation/documentationApis';

// Redux Slices
import commonSlice from './common/commonSlice';
import membersSlice from './members/membersSlice';
import plansSlice from './plans/plansSlice';
import facilitySlice from './facility/facilitySlice';
import activitySlice from './activity/activitySlice';
import mastersSlice from './masters/mastersSlice';
import bookingSlice from './booking/bookingSlice';
import eventsSlice from './events/eventsSlice';
import hallsSlice from './halls/hallsSlice';
import staffSlice from './staff/staffSlice';
import documentationSlice from './documentation/documentationSlice';

// Axios interceptor
import axiosInterceptor from '../helpers/axios';

const reducers = {
    // Redux Slices
    [commonSlice.name]: commonSlice.reducer,
    [membersSlice.name]: membersSlice.reducer,
    [plansSlice.name]: plansSlice.reducer,
    [facilitySlice.name]: facilitySlice.reducer,
    [activitySlice.name]: activitySlice.reducer,
    [mastersSlice.name]: mastersSlice.reducer,
    [bookingSlice.name]: bookingSlice.reducer,
    [eventsSlice.name]: eventsSlice.reducer,
    [hallsSlice.name]: hallsSlice.reducer,
    [staffSlice.name]: staffSlice.reducer,
    [documentationSlice.name]: documentationSlice.reducer,
    // RTK Query APIs
    [membersApis.reducerPath]: membersApis.reducer,
    [commonApis.reducerPath]: commonApis.reducer,
    [plansApis.reducerPath]: plansApis.reducer,
    [facilityApis.reducerPath]: facilityApis.reducer,
    [activityApis.reducerPath]: activityApis.reducer,
    [mastersApis.reducerPath]: mastersApis.reducer,
    [bookingApis.reducerPath]: bookingApis.reducer,
    [eventsApis.reducerPath]: eventsApis.reducer,
    [hallsApis.reducerPath]: hallsApis.reducer,
    [staffApis.reducerPath]: staffApis.reducer,
    [documentationApi.reducerPath]: documentationApi.reducer,
};

const initialState = {};

const rootReducer = combineReducers(reducers);

export const store = configureStore({
    reducer: rootReducer,
    middleware: (getDefaultMiddleware) => {
        return getDefaultMiddleware({
            serializableCheck: false,
        }).concat([
            membersApis.middleware,
            commonApis.middleware,
            plansApis.middleware,
            facilityApis.middleware,
            activityApis.middleware,
            mastersApis.middleware,
            bookingApis.middleware,
            eventsApis.middleware,
            hallsApis.middleware,
            staffApis.middleware,
            documentationApi.middleware,
        ]);
    },
    devTools: true,
    preloadedState: initialState,
    enhancers: (defaultEnhancers) => [...defaultEnhancers],
});

export const useAppSelector = () => useSelector(rootReducer);
export const useAppDispatch = () => useDispatch(store.dispatch);

axiosInterceptor(store.dispatch);

export default store;
```

Store Features

- **Redux DevTools**: Development debugging
- **Serializable Check**: Disabled for RTK Query

- **Middleware**: RTK Query middleware for all APIs
- **Axios Integration**: Automatic token management

State Slices

Common Slice

Location: `/src/store/common/commonSlice.js`

Global application state management.

```
import { createSlice } from '@reduxjs/toolkit';

const initialState = {
    loading: false,
    error: null,
    notification: {
        open: false,
        message: '',
        severity: 'info',
    },
    user: null,
    permissions: [],
    theme: 'light',
    sidebar: {
        open: true,
        collapsed: false,
    },
    filters: {
        global: {},
        members: {},
        bookings: {},
        events: {},
    },
};

const commonSlice = createSlice({
    name: 'common',
    initialState,
    reducers: {
        setLoading: (state, action) => {
            state.loading = action.payload;
        },
        setError: (state, action) => {
            state.error = action.payload;
        },
        setNotification: (state, action) => {
            state.notification = {
                open: true,
                message: action.payload.message,
                severity: action.payload.severity || 'info',
            };
        },
        clearNotification: (state) => {
            state.notification = {
                open: false,
                message: '',
                severity: 'info',
            };
        },
        setUser: (state, action) => {
            state.user = action.payload;
        },
        setPermissions: (state, action) => {
            state.permissions = action.payload;
        },
        setTheme: (state, action) => {
            state.theme = action.payload;
        },
        toggleSidebar: (state) => {
            state.sidebar.open = !state.sidebar.open;
        },
        setSidebarCollapsed: (state, action) => {
            state.sidebar.collapsed = action.payload;
        },
        setFilters: (state, action) => {
            const { module, filters } = action.payload;
            state.filters[module] = { ...state.filters[module], ...filters };
        },
        clearFilters: (state, action) => {
            state.filters[action.payload] = {};
        },
    },
});

export const {
    setLoading,
    setError,
    setNotification,
    clearNotification,
    setUser,
    setPermissions,
    setTheme,
    toggleSidebar,
    setSidebarCollapsed,
    setFilters,
    clearFilters,
} = commonSlice.actions;

export default commonSlice.reducer;
```

Members Slice

Location: `/src/store/members/membersSlice.js`

Member management state.

```
import { createSlice } from '@reduxjs/toolkit';

const initialState = {
    members: [],
    selectedMember: null,
    filters: {
        status: 'all',
        search: '',
        category: 'all',
        page: 1,
        limit: 10,
    },
    pagination: {
        currentPage: 1,
        totalPages: 1,
        totalItems: 0,
        itemsPerPage: 10,
    },
    loading: false,
    error: null,
    formData: {
        name: '',
        email: '',
        phone: '',
        status: 'active',
        category: '',
        joinDate: null,
    },
    validationErrors: {},
};

const membersSlice = createSlice({
    name: 'members',
    initialState,
    reducers: {
        setMembers: (state, action) => {
            state.members = action.payload;
        },
        setSelectedMember: (state, action) => {
            state.selectedMember = action.payload;
        },
        updateMember: (state, action) => {
            const index = state.members.findIndex(
                (member) => member.id === action.payload.id,
            );
            if (index !== -1) {
                state.members[index] = action.payload;
            }
        },
        removeMember: (state, action) => {
            state.members = state.members.filter(
                (member) => member.id !== action.payload,
            );
        },
        setFilters: (state, action) => {
            state.filters = { ...state.filters, ...action.payload };
        },
        setPagination: (state, action) => {
            state.pagination = { ...state.pagination, ...action.payload };
        },
        setLoading: (state, action) => {
            state.loading = action.payload;
        },
        setError: (state, action) => {
            state.error = action.payload;
        },
        setFormData: (state, action) => {
            state.formData = { ...state.formData, ...action.payload };
        },
        clearFormData: (state) => {
            state.formData = initialState.formData;
        },
        setValidationErrors: (state, action) => {
            state.validationErrors = action.payload;
        },
        clearValidationErrors: (state) => {
            state.validationErrors = {};
        },
    },
});

export const {
    setMembers,
    setSelectedMember,
    updateMember,
    removeMember,
    setFilters,
    setPagination,
    setLoading,
    setError,
    setFormData,
    clearFormData,
    setValidationErrors,
    clearValidationErrors,
} = membersSlice.actions;

export default membersSlice.reducer;
```

RTK Query APIs

Members API

Location: `/src/store/members/membersApis.js`

```
import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react';

export const membersApi = createApi({
    reducerPath: 'membersApi',
    baseQuery: fetchBaseQuery({
        baseUrl: '/api/members',
        prepareHeaders: (headers, { getState }) => {
            const token = getState().auth?.token;
            if (token) {
                headers.set('authorization', `Bearer ${token}`);
            }
            return headers;
        },
    }),
    tagTypes: ['Members', 'Member'],
    endpoints: (builder) => ({
        getMembers: builder.query({
            query: (params = {}) => ({
                url: '/',
                params: {
                    page: params.page || 1,
                    limit: params.limit || 10,
                    search: params.search,
                    status: params.status,
                    category: params.category,
                    sortBy: params.sortBy,
                    sortOrder: params.sortOrder,
                },
            }),
            providesTags: ['Members'],
        }),
        getMember: builder.query({
            query: (id) => `/${id}`,
            providesTags: (result, error, id) => [{ type: 'Member', id }],
        }),
        createMember: builder.mutation({
            query: (memberData) => ({
                url: '/',
                method: 'POST',
                body: memberData,
            }),
            invalidatesTags: ['Members'],
        }),
        updateMember: builder.mutation({
            query: ({ id, ...memberData }) => ({
                url: `/${id}`,
                method: 'PUT',
                body: memberData,
            }),
            invalidatesTags: (result, error, { id }) => [
                { type: 'Member', id },
                'Members',
            ],
        }),
        deleteMember: builder.mutation({
            query: (id) => ({
                url: `/${id}`,
                method: 'DELETE',
            }),
            invalidatesTags: ['Members'],
        }),
        getMemberHistory: builder.query({
            query: (id) => `/${id}/history`,
            providesTags: (result, error, id) => [{ type: 'Member', id, history: true
}],
        }),
        exportMembers: builder.query({
            query: (params) => ({
                url: '/export',
                params,
                responseHandler: (response) => response.blob(),
            }),
            providesTags: ['Members'],
        }),
    }),
});

export const {
    useGetMembersQuery,
    useGetMemberQuery,
    useCreateMemberMutation,
    useUpdateMemberMutation,
    useDeleteMemberMutation,
    useGetMemberHistoryQuery,
    useExportMembersQuery,
} = membersApi;
```

Common API

Location: `/src/store/common/commonApis.js`

```
import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react';

export const commonApi = createApi({
    reducerPath: 'commonApi',
    baseQuery: fetchBaseQuery({
        baseUrl: '/api/common',
        prepareHeaders: (headers, { getState }) => {
            const token = getState().auth?.token;
            if (token) {
                headers.set('authorization', `Bearer ${token}`);
            }
            return headers;
        },
    }),
    tagTypes: ['Categories', 'Statuses', 'Locations', 'Facilities'],
    endpoints: (builder) => ({
        getCategories: builder.query({
            query: () => '/categories',
            providesTags: ['Categories'],
        }),
        getStatuses: builder.query({
            query: () => '/statuses',
            providesTags: ['Statuses'],
        }),
        getLocations: builder.query({
            query: () => '/locations',
            providesTags: ['Locations'],
        }),
        getFacilities: builder.query({
            query: () => '/facilities',
            providesTags: ['Facilities'],
        }),
        uploadFile: builder.mutation({
            query: (formData) => ({
                url: '/upload',
                method: 'POST',
                body: formData,
            }),
        }),
        getDashboardStats: builder.query({
            query: () => '/dashboard/stats',
            providesTags: ['DashboardStats'],
        }),
    }),
});

export const {
    useGetCategoriesQuery,
    useGetStatusesQuery,
    useGetLocationsQuery,
    useGetFacilitiesQuery,
    useUploadFileMutation,
    useGetDashboardStatsQuery,
} = commonApi;
```

Container Pattern

Container Structure

Containers connect Redux state to components using the `connect` HOC pattern.

Location: `/src/container/admin/members-container/MembersContainer.jsx`

```
import React, { useEffect } from 'react';
import { connect } from 'react-redux';
import { bindActionCreators } from 'redux';
import {
    setMembers,
    setFilters,
    setPagination,
    setLoading,
    setError,
    setSelectedMember,
} from '../../../store/members/membersSlice';
import MembersComponent from '../../../components/admin/members-
manager/MembersComponent';

const MembersContainer = ({
    members,
    filters,
    pagination,
    loading,
    error,
    selectedMember,
    setMembers,
    setFilters,
    setPagination,
    setLoading,
    setError,
    setSelectedMember,
}) => {
    // Component logic and effects

    const handleFilterChange = (newFilters) => {
        setFilters(newFilters);
    };

    const handlePageChange = (newPagination) => {
        setPagination(newPagination);
    };

    const handleMemberSelect = (member) => {
        setSelectedMember(member);
    };

    return (
        <MembersComponent
            members={members}
            filters={filters}
            pagination={pagination}
            loading={loading}
            error={error}
            selectedMember={selectedMember}
            onFilterChange={handleFilterChange}
            onPageChange={handlePageChange}
            onMemberSelect={handleMemberSelect}
        />
    );
};
```

```
const mapStateToProps = (state) => ({
    members: state.members.members,
    filters: state.members.filters,
    pagination: state.members.pagination,
    loading: state.members.loading,
    error: state.members.error,
    selectedMember: state.members.selectedMember,
});

const mapDispatchToProps = (dispatch) =>
    bindActionCreators(
        {
            setMembers,
            setFilters,
            setPagination,
            setLoading,
            setError,
            setSelectedMember,
        },
        dispatch,
    );

export default connect(mapStateToProps, mapDispatchToProps)(MembersContainer);
```

Store Wrapper

Location: `/src/container/admin/members-container/MembersStore.js`

```
import { connect } from 'react-redux';
import MembersContainer from './MembersContainer';

const MembersStore = connect(
    (state) => ({
        // Redux state mapping
        members: state.members,
        common: state.common,
    }),
    (dispatch) => ({
        // Action creators
        dispatch,
    }),
)(MembersContainer);

export default MembersStore;
```

State Selectors

Selector Functions

Location: `/src/store/members/membersSelectors.js`

```javascript
// Basic selectors
export const selectMembers = (state) => state.members.members;
export const selectSelectedMember = (state) => state.members.selectedMember;
export const selectMembersLoading = (state) => state.members.loading;
export const selectMembersError = (state) => state.members.error;
export const selectMembersFilters = (state) => state.members.filters;
export const selectMembersPagination = (state) => state.members.pagination;

// Computed selectors
export const selectActiveMembers = (state) =>
    state.members.members.filter((member) => member.status === 'active');

export const selectMembersByCategory = (state) => {
    const members = state.members.members;
    const category = state.members.filters.category;

    if (category === 'all') return members;
    return members.filter((member) => member.category === category);
};

export const selectFilteredMembers = (state) => {
    const members = state.members.members;
    const filters = state.members.filters;

    return members.filter((member) => {
        if (
            filters.search &&
            !member.name.toLowerCase().includes(filters.search.toLowerCase())
        ) {
            return false;
        }
        if (filters.status !== 'all' && member.status !== filters.status) {
            return false;
        }
        if (filters.category !== 'all' && member.category !== filters.category) {
            return false;
        }
        return true;
    });
};

export const selectMemberById = (id) => (state) =>
    state.members.members.find((member) => member.id === id);

// Memoized selectors using reselect
import { createSelector } from '@reduxjs/toolkit';

export const selectMembersWithStats = createSelector([selectMembers], (members) => {
    const total = members.length;
    const active = members.filter((m) => m.status === 'active').length;
    const inactive = members.filter((m) => m.status === 'inactive').length;
    return {
        members,
        stats: {
            total,
            active,
            inactive,
            activePercentage: total > 0 ? (active / total) * 100 : 0,
        },
    };
});
```

Async Actions

Thunk Actions

Location: `/src/store/members/membersAsyncActions.js`

```
import { createAsyncThunk } from '@reduxjs/toolkit';
import { membersApi } from './membersApis';

// Fetch members with filters
export const fetchMembers = createAsyncThunk(
    'members/fetchMembers',
    async (params, { rejectWithValue }) => {
        try {
            const response = await membersApi.endpoints.getMembers.initiate(params);
            return response.data;
        } catch (error) {
            return rejectWithValue(
                error.response?.data?.message || 'Failed to fetch members',
            );
        }
    },
);

// Create member
export const createMember = createAsyncThunk(
    'members/createMember',
    async (memberData, { rejectWithValue }) => {
        try {
            const response = await
membersApi.endpoints.createMember.initiate(memberData);
            return response.data;
        } catch (error) {
            return rejectWithValue(
                error.response?.data?.message || 'Failed to create member',
            );
        }
    },
);

// Update member
```

```
export const updateMember = createAsyncThunk(
    'members/updateMember',
    async ({ id, memberData }, { rejectWithValue }) => {
        try {
            const response = await membersApi.endpoints.updateMember.initiate({
                id,
                ...memberData,
            });
            return response.data;
        } catch (error) {
            return rejectWithValue(
                error.response?.data?.message || 'Failed to update member',
            );
        }
    },
);

// Delete member
export const deleteMember = createAsyncThunk(
    'members/deleteMember',
    async (id, { rejectWithValue }) => {
        try {
            await membersApi.endpoints.deleteMember.initiate(id);
            return id;
        } catch (error) {
            return rejectWithValue(
                error.response?.data?.message || 'Failed to delete member',
            );
        }
    },
);
```

Slice with Async Actions

```
import { createSlice } from '@reduxjs/toolkit';
import {
    fetchMembers,
    createMember,
    updateMember,
    deleteMember,
} from './membersAsyncActions';

const membersSlice = createSlice({
    name: 'members',
    initialState,
    reducers: {
        // ... existing reducers
    },
    extraReducers: (builder) => {
        builder
            // Fetch members
            .addCase(fetchMembers.pending, (state) => {
                state.loading = true;
                state.error = null;
            })
            .addCase(fetchMembers.fulfilled, (state, action) => {
                state.loading = false;
                state.members = action.payload.data;
                state.pagination = action.payload.pagination;
            })
            .addCase(fetchMembers.rejected, (state, action) => {
                state.loading = false;
                state.error = action.payload;
            })

            // Create member
            .addCase(createMember.pending, (state) => {
                state.loading = true;
                state.error = null;
            })
            .addCase(createMember.fulfilled, (state, action) => {
                state.loading = false;
                state.members.push(action.payload);
                state.error = null;
            })
            .addCase(createMember.rejected, (state, action) => {
                state.loading = false;
                state.error = action.payload;
            })

            // Update member
            .addCase(updateMember.pending, (state) => {
                state.loading = true;
                state.error = null;
            })
            .addCase(updateMember.fulfilled, (state, action) => {
                state.loading = false;
                const index = state.members.findIndex(
                    (member) => member.id === action.payload.id,
                );
                if (index !== -1) {
                    state.members[index] = action.payload;
                }
                state.error = null;
            })
            .addCase(updateMember.rejected, (state, action) => {
                state.loading = false;
                state.error = action.payload;
            })

            // Delete member
            .addCase(deleteMember.pending, (state) => {
                state.loading = true;
                state.error = null;
            })
            .addCase(deleteMember.fulfilled, (state, action) => {
                state.loading = false;
                state.members = state.members.filter(
                    (member) => member.id !== action.payload,
                );
                state.error = null;
            })
            .addCase(deleteMember.rejected, (state, action) => {
                state.loading = false;
                state.error = action.payload;
            });
    },
});
```

Error Handling

Global Error Handling

Location: `/src/store/middleware/errorMiddleware.js`

```
import { createListenerMiddleware } from '@reduxjs/toolkit';

export const errorMiddleware = createListenerMiddleware();

// Listen for rejected async actions
errorMiddleware.startListening({
    predicate: (action) => action.type.endsWith('/rejected'),
    effect: async (action, listenerApi) => {
        const { dispatch } = listenerApi;

        // Extract error message
        const errorMessage = action.payload || 'An error occurred';

        // Show notification
        dispatch(
            setNotification({
                message: errorMessage,
                severity: 'error',
            }),
        );

        // Log error for debugging
        console.error('Redux Error:', action);

        // Handle specific error types
        if (action.type.includes('auth')) {
            // Handle authentication errors
            dispatch(logout());
        }
    },
});
```

Error Boundary

Location: `/src/components/Common/ErrorBoundary.jsx`

```
import React from 'react';
import { ErrorBoundary } from 'react-error-boundary';

function ErrorFallback({ error, resetErrorBoundary }) {
    return (
        <div role="alert">
            <h2>Something went wrong:</h2>
            <pre>{error.message}</pre>
            <button onClick={resetErrorBoundary}>Try again</button>
        </div>
    );
}

function ErrorBoundaryWrapper({ children }) {
    return (
        <ErrorBoundary
            FallbackComponent={ErrorFallback}
            onError={(error, errorInfo) => {
                console.error('Error Boundary caught an error:', error, errorInfo);
                // Log to error reporting service
            }}
        >
            {children}
        </ErrorBoundary>
    );
}

export default ErrorBoundaryWrapper;
```

Performance Optimization

Memoization

```
import { createSelector } from '@reduxjs/toolkit';

// Memoized selectors
export const selectExpensiveComputation = createSelector(
    [selectMembers, selectFilters],
    (members, filters) => {
        // Expensive computation
        return members.filter((member) => {
            // Complex filtering logic
            return complexFilterLogic(member, filters);
        });
    },
);

// Component memoization
```

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

const MembersList = memo(({ members, onEdit, onDelete }) => {
    return (
        <div>
            {members.map((member) => (
                <MemberItem
                    key={member.id}
                    member={member}
                    onEdit={onEdit}
                    onDelete={onDelete}
                />
            ))}
        </div>
    );
});
```

RTK Query Optimization

```
// Polling for real-time updates
export const membersApi = createApi({
    // ... other config
    endpoints: (builder) => ({
        getMembers: builder.query({
            query: (params) => ({
                url: '/',
                params,
            }),
            // Poll every 30 seconds
            pollingInterval: 30000,
            // Keep data for 60 seconds
            keepUnusedDataFor: 60,
            providesTags: ['Members'],
        }),
    }),
});

// Conditional queries
const { data: members, isLoading } = useGetMembersQuery(params, {
    skip: !shouldFetchMembers,
    refetchOnMountOrArgChange: true,
    refetchOnFocus: true,
});
```

State Normalization

```
// Normalized state structure
const initialState = {
    members: {
        byId: {},
        allIds: [],
    },
    // ... other state
};

// Normalized reducers
const membersSlice = createSlice({
    name: 'members',
    initialState,
    reducers: {
        addMember: (state, action) => {
            const member = action.payload;
            state.members.byId[member.id] = member;
            state.members.allIds.push(member.id);
        },
        updateMember: (state, action) => {
            const member = action.payload;
            state.members.byId[member.id] = member;
        },
        removeMember: (state, action) => {
            const id = action.payload;
            delete state.members.byId[id];
            state.members.allIds = state.members.allIds.filter(
                (memberId) => memberId !== id,
            );
        },
    },
});

// Selectors for normalized data
export const selectAllMembers = (state) =>
    state.members.members.allIds.map((id) => state.members.members.byId[id]);

export const selectMemberById = (id) => (state) => state.members.members.byId[id];
```

Summary

The ASMC Admin Panel's state management system provides:

- **Centralized State**: Redux Toolkit for predictable state management
- **API Integration**: RTK Query for efficient server state management
- **Type Safety**: Full TypeScript integration
- **Performance**: Optimized with memoization and normalization
- **Developer Experience**: Redux DevTools and error handling
- **Scalability**: Modular architecture for easy expansion

This architecture ensures maintainable, performant, and scalable state management for the admin panel.

Version: 1.0.0
Last Updated: January 2025
Maintainer: ASMC Development Team