# CSharp API Gaps

This document tracks possible gaps between the new admin interface requirements and the CSharp API capabilities. These gaps represent features or data fields that are needed in the new admin interface but are not currently supported by the CSharp API.

## Credit Union Information

### Sponsor Details Management

✅ Available:

- Sponsor ID (sponsorId)

- Sponsor Name (sponsorName)

❌ Missing:

- Sponsor Prefix

### Settlement Information

✅ Available:

- Routing Number (routingId)

❌ Missing:

- Settlement GL Code

- Settlement External ACH Number

- Clear definition of wrgAccountNumber purpose and relationship to settlement

### Operating Hours Configuration

❌ Missing:

- Primary business hours

- After hours contact information

### Staff Contact Information

❌ Missing:

- Staff contact list with roles

- Role-based contact management

## Holiday Management

❌ Missing Holiday Management Features:

- Complete holiday management system including:

- Holiday Type definitions (FEDERAL, STATE, BANK)

- Holiday CRUD operations

- Holiday status tracking

- Holiday date management with descriptions

- Creation and update timestamp tracking

### Required Types Not in API

```typescript

enum HolidayType {

FEDERAL = 'FEDERAL',

STATE = 'STATE',

BANK = 'BANK'

}

interface Holiday {

id: number;

name: string;

date: string;

type: HolidayType;

description?: string;

status: HolidayStatus;

createdAt?: string;

updatedAt?: string;

}

```

## Notifications

❌ Possible Missing Notification Features:

- Notification template management:

- Template CRUD operations

- Variable management

- Template activation/deactivation

- Delivery settings configuration:

- Email and SMS settings

- Default recipients

- Retry policy configuration

- Notification monitoring:

- Delivery statistics

- Failure tracking

- Performance metrics

### Required Types Not in API

```typescript

interface NotificationTemplate {

id: number;

name: string;

description?: string;

subject: string;

body: string;

type: 'email' | 'sms';

variables: string[];

isActive: boolean;

}

interface DeliverySettings {

emailEnabled: boolean;

smsEnabled: boolean;

defaultRecipients: string[];

retryAttempts: number;

retryInterval: number;

}

interface NotificationStats {

totalSent: number;

totalFailed: number;

deliveryRate: number;

averageDeliveryTime: number;

}

```

## Notification Templates

### Template Storage

❌ Missing:

- Storage location for notification templates

- Template retrieval by type

- Template versioning

### Required Types Not in API

```typescript

interface NotificationTemplate {

id: string;

type: NotificationType;

content: string;

version: string;

}

```

### Required Endpoints Not in API

```typescript

GET /api/v1/notification/templates/:type - Get template by type

PUT /api/v1/notification/templates/:type - Update template

```

## Authentication and Permission Management

❌ Missing Authentication Features:

- Comprehensive authentication system including:

- Role-based access control (RBAC)

- Permission management

- User session handling

- Token refresh mechanism

### Required Types Not in API

```typescript

interface AuthenticationRequest {

username: string;

password: string;

clientId: string;

}

interface AuthenticationResponse {

token: string;

refreshToken: string;

expiresIn: number;

tokenType: string;

}

```

## Configuration and Common Types

❌ Missing Configuration and Common Types Features:

- Configuration categories (BILL\_PAY, SYSTEM, SECURITY, NOTIFICATIONS)

- Type validation for configuration values

- Configuration audit tracking (lastUpdated, updatedBy)

- Configuration search and filtering by category

### Required Types Not in API

```typescript

interface SystemConfigurationItem {

key: string;

value: string | number | boolean;

category: string;

lastUpdated: string;

updatedBy: string;

}

interface ConfigurationUpdate {

key: string;

value: ConfigurationValue;

category: ConfigurationCategory;

}

```

Note: The core CBP API has a `/configuration/all` endpoint, but we need clarification from the C# team about how it should be used for bill pay settings.

❌ Missing Bill Pay Configuration Features:

- Comprehensive bill pay settings management including:

- Processing cutoff time configuration

- Daily and transaction limit settings

- Weekend processing configuration

- Dual approval requirements

- Retry attempt configuration

- Notification settings (email and general)

### Required Types Not in API

```typescript

interface BillPayConfig {

id: string;

cutoffTime: string;

maxDailyLimit: number;

maxTransactionLimit: number;

allowWeekendProcessing: boolean;

requireDualApproval: boolean;

retryAttempts: number;

notificationEmail: string;

enableEmailNotifications: boolean;

}

```

## Security

❌ Possible Missing Security Features:

- Comprehensive security settings management:

- Password policy configuration

- Login policy settings

- IP whitelist management

- MFA configuration

- Audit log settings

- Security alert configuration

- Security metrics and monitoring:

- Login attempt tracking

- MFA usage statistics

- Password reset metrics

- Risk level assessments

- Security event handling:

- Risk assessment endpoints

- Security alert management

- Access attempt logging

### Required Types Not in API

```typescript

interface SecuritySettings {

passwordPolicy: PasswordPolicy;

loginPolicy: LoginPolicy;

ipWhitelist: IPWhitelist;

mfaSettings: MFASettings;

auditSettings: AuditSettings;

alertSettings: AlertSettings;

}

interface SecurityMetrics {

loginAttempts: {

successful: number;

failed: number;

locked: number;

};

mfaUsage: {

enabled: number;

disabled: number;

byMethod: Record<MFAMethod, number>;

};

passwordResets: {

selfService: number;

adminInitiated: number;

forgotten: number;

};

}

```

## Security Settings API Gaps

The following TypeScript interfaces are defined but their corresponding C# models are not found in the C# APIs:

### BillPaySecuritySettings

```typescript

interface BillPaySecuritySettings {

passwordPolicy: BillPayPasswordPolicy;

loginPolicy: BillPayLoginPolicy;

ipWhitelist: BillPayIPWhitelist;

otpSettings: BillPayOTPSettings;

etag?: string;

}

```

### BillPayOTPMethod

```typescript

enum BillPayOTPMethod {

EMAIL = 'email',

SMS = 'sms'

}

```

### BillPaySecurityValidation

```typescript

interface BillPaySecurityValidation {

isValid: boolean;

errors: Record<string, unknown>;

}

```

### Authentication Interface

No corresponding C# interfaces or classes were found for the TypeScript authentication types defined in `api.types.ts` and `auth.types.ts`. This includes:

- Login request/response types

- Token refresh types

- Authentication state types

This suggests authentication may be handled by a separate service or through middleware rather than the core API.

\*\*Note\*\*: These security settings may be part of the `/configuration/all` endpoint's response. Need to verify the actual shape of the security configuration data returned by this endpoint.

\*\*Action Items\*\*:

1. Verify if these security settings exist in the `/configuration/all` endpoint response

2. If found, ensure the TypeScript types match the actual API response structure

3. If not found, determine if these are new features that need to be added to the API

## Reporting Documentation

❌ Missing Documentation for Report Endpoint:

- The `/api/v1/report/run` endpoint accepts stored procedures but lacks:

- List of available stored procedures

- Required arguments for each procedure

- Expected return types/schemas

- Argument validation rules

- Example requests and responses

### Required Documentation

```typescript

// Example of what the documentation should specify for each stored procedure

interface StoredProcedureSpec {

name: string;

description: string;

arguments: {

[key: string]: {

type: string;

required: boolean;

description: string;

validation?: string;

}

};

returnSchema: {

type: string;

properties: Record<string, unknown>;

};

}

```

Note: While the reporting endpoint exists, its documentation needs to be enhanced to specify the available stored procedures and their requirements.

### Analytics and Reporting

❌ Missing Analytics Features:

- Dedicated analytics endpoints for:

- Payment statistics with timeframe filtering

- Transaction trends analysis

- Performance metrics

- Current limitations:

- Stats available only through generic report runner

- Payment activity endpoint lacks proper analytics support

- No standardized date range handling

- Missing trend analysis capabilities

### Required Analytics Types Not in API

```typescript

interface PaymentStats {

timeframe: DateRange;

totalCount: number;

totalAmount: number;

successRate: number;

averageAmount: number;

categoryBreakdown: CategoryStats[];

}

interface TransactionTrend {

timeframe: DateRange;

interval: 'daily' | 'weekly' | 'monthly';

dataPoints: TrendPoint[];

}

interface DateRange {

startDate: string;

endDate: string;

timezone?: string;

}

interface CategoryStats {

category: string;

count: number;

amount: number;

percentage: number;

}

interface TrendPoint {

timestamp: string;

count: number;

amount: number;

successRate: number;

}

```

## Dashboard Metrics

❌ Missing Dashboard Features:

- Core transaction metrics including:

- Transaction counts and status

- Transaction volume over time

### Required Types Not in API

```typescript

interface DashboardMetrics {

transactions: {

successful: number;

failed: number;

pending: number;

total: number;

volume: {

daily: number;

weekly: number;

monthly: number;

};

};

}

```

### Required Endpoints Not in API

```typescript

// Dashboard Metrics

GET /api/v1/dashboard/metrics - Get all dashboard metrics

GET /api/v1/dashboard/transactions/stats - Get transaction statistics

```

### Notes

- Consider caching strategy for dashboard metrics

- Plan data aggregation approach (real-time vs pre-aggregated)

- Define metric calculation methods

- Chart visualizations will be handled client-side using the metrics data

## Payee Management Questions for Team Discussion

1. User Payee Management

- What CRUD operations are needed for user payees?

- How to handle user payee validation?

2. FIS Integration

- What FIS operations are required?

- How to handle FIS payee status changes?

## Payment Processing Questions for Team Discussion

1. Processing Requirements

- How is payment processing handled?

- What validation rules exist?

- How to handle reprocessing?

## Next Steps

1. Evaluate if these features should be added to the C# API

2. Determine if any of these settings are handled through other means (e.g., environment variables, config files)