WildFire API Automation - Usage Guide

This guide provides detailed examples of how to use the WildFire API Automation system, including API endpoints and direct service access.

Table of Contents

API Endpoint Usage

Health Check

Check system health and component status.

Endpoint: `GET /api/health`

Request:

```bash
curl -X GET http://localhost/api/health
```

Response:

json


```
{
  "success": true,
  "error": "",
  "data": {
    "status": "healthy",
    "timestamp": "2024-01-10 10:30:00",
    "version": "2.0.0",
    "checks": {
      "database": {
        "healthy": true,
        "message": "Database connection OK"
      },
      "logging": {
        "healthy": true,
        "message": "Logging system OK"
      },
      "memory": {
        "healthy": true,
        "message": "Memory usage: 45.2% (512 MB / 1 GB)"
      },
      "disk": {
        "healthy": true,
        "message": "Disk usage: 35.0% (Free: 65 GB)"
      }
    }
  }
}
```


WildFire Pre-Qualification

Process a complete WildFire application with 200+ fields.

Endpoint: `POST /api/wildfire/prequal`

Request Headers:

```
Content-Type: application/json
```

Request Body (abbreviated - actual payload has 200+ fields):

json

```
{
  "application_id": "APP-2024-001234",
  "applicant_first_name": "John",
  "applicant_last_name": "Doe",
  "applicant_ssn": "123-45-6789",
  "applicant_dob": "1985-06-15",
  "applicant_address": "123 Main St",
  "applicant_city": "Austin",
  "applicant_state": "TX",
  "applicant_zip": "78701",
  "applicant_employer_employment_type": "EMPLOYED",
  "total_primary_income": "5000",
  "total_debt": "1500",
  "vehicle_vin": "1HGBH41JXMN109186",
  "vehicle_year": "2020",
  "vehicle_make": "Honda",
  "vehicle_model": "Civic",
  "vehicle_miles": "35000",
  "data_requested_amt": "25000",
  "bureau_primary_active_bureau": "experian",
  "co_applicant_active": false
}
```


Response:

```
json
```

```
{
  "success": true,
  "data": {
    "bureau": "experian",
    "raw": "{...}",
    "hit": true,
    "pulled_at": "2024-01-10 10:30:00",
    "fico_score": 720,
    "has_coapp": false,
    "dti": 0.35,
    "open_trade_count": 8,
    "auto_trade_count": 2,
    "open_auto_trade_count": 1,
    "derogatory_marks": 0,
    "now_delinquent": 0,
    "was_delinquent": 1,
    "bankruptcies": 0,
    "past_due_amount": 0,
    "satisfactory_accounts": 7,
    "install_balance": 15000,
    "scheduled_payment": 850,
    "real_estate_balance": 180000,
    "real_estate_payment": 1200,
    "revolving_balance": 3500,
    "revolving_limit": 25000,
    "revolving_available": 14,
    "paid_accounts": 3,
    "inquiries_6mo": 2,
    "oldest_trade": "03/2015",
    "score_data": {
      "score": "720",
      "percentile": "75",
      "model": "FICO Score 8 based on Experian Data",
      "evaluation": "Good",
      "factors": [
        {
          "prio": "1",
          "code": "14",
          "desc": "LENGTH OF TIME ACCOUNTS HAVE BEEN ESTABLISHED"
        },
        {
          "prio": "2",
          "code": "10",
```

```
            "desc": "RATIO OF BALANCE TO LIMIT ON BANK REVOLVING ACCOUNTS"
          }
        ]
      },
      "identity_records": [
        {
          "first_name": "John",
          "last_name": "Doe",
          "middle_name": "",
          "dob": "",
          "ssn": ""
        }
      ],
      "address_records": [
        {
          "city": "Austin",
          "state": "TX",
          "zip": "78701",
          "first_reported": "2020-01-15",
          "source": "TransUnion",
          "address": "123 Main St",
          "current": true
        }
      ],
      "trade_lines": [
        {
          "bureau": "experian",
          "opened": "2018-03-15",
          "is_open": true,
          "closed": "",
          "type": "Auto Loan",
          "creditor": "HONDA FINANCIAL",
          "standing": 1,
          "balance": 12500,
          "payment": 350,
          "credit_limit": 0,
          "original_amount": 25000
        }
      ],
      "bureau_errors": [],
      "bureau_alerts": []
    },
```

```
  "error": ""
}
```

Direct Bureau Pull

Pull credit report using only PII data (no vehicle information).

Endpoint: `POST /api/wildfire/pull`

Request Body:

json

```
{
  "bureau": "equifax",
  "score_model": "FICO",
  "pii": {
    "primary": {
      "first": "Jane",
      "last": "Smith",
      "ssn": "987-65-4321",
      "dob": "1990-03-20",
      "address": "456 Oak Ave",
      "city": "Dallas",
      "state": "TX",
      "zip": "75201"
    },
    "secondary": {
      "first": "Robert",
      "last": "Smith",
      "ssn": "876-54-3210",
      "dob": "1988-07-15",
      "address": "456 Oak Ave",
      "city": "Dallas",
      "state": "TX",
      "zip": "75201"
    }
  }
}
```

Response: Same format as WildFire Pre-Qualification

Vehicle Valuation - VIN

Get vehicle valuation using VIN.

Endpoint: POST /api/valuation/vin

Request Body:

json

```
{
  "nada": {
    "vin": "1HGBH41JXMN109186",
    "state": "TX",
    "mileage": "35000"
  }
}
```

Response:

json

```
{
  "success": true,
  "error": "",
  "data": {
    "vehicle_year": 2020,
    "vehicle_make": "Honda",
    "vehicle_model": "Civic",
    "vehicle_trim": "LX Sedan 4D",
    "vehicle_vin_decoded": "true",
    "vehicle_msrp": 21050,
    "vehicle_retail_value": 19500,
    "vehicle_trade_value": 17800,
    "vehicle_curb_weight": 2771,
    "vehicle_doors": 4,
    "vehicle_drive_train": "FWD",
    "vehicle_transmission": "CVT",
    "vehicle_engine_config": "2.0L L4-16V (Port)",
    "vehicle_basecleantrade": 18200,
    "vehicle_baseaveragetrade": 16900,
    "vehicle_baseroughtrade": 15100,
    "vehicle_basecleanretail": 20100,
    "vehicle_adjustedcleantrade": 17800,
    "vehicle_adjustedaveragetrade": 16500,
    "vehicle_adjustedroughtrade": 14700,
    "vehicle_adjustedcleanretail": 19500,
    "vehicle_miles": 35000,
    "vehicle_miles_read_date": "2024-01-10",
    "vehicle_mileageadjustment": -400,
    "vehicle_options_json": "[...]"
  }
}
```


Vehicle Valuation - YMM

Get vehicle valuation using Year/Make/Model.

Endpoint: `POST /api/valuation/ymm`

Request Body:

```
json
```

```
{
  "nada": {
    "year": 2020,
    "make": "Honda",
    "model": "Civic",
    "trim": "LX",
    "state": "TX",
    "mileage": "35000"
  }
}
```


Response: Same format as VIN valuation

Modern Pre-Qualification

Modern JSON API with clean request/response format.

Endpoint: `POST /api/automation/prequal`

Request Body:

```
json
```

```
{
  "applicant": {
    "monthly_income": 5000,
    "monthly_debt": 1500,
    "employment_type": "W2",
    "state": "TX",
    "first_name": "John",
    "last_name": "Doe",
    "ssn": "123-45-6789",
    "address": "123 Main St",
    "city": "Austin",
    "zip_code": "78701",
    "date_of_birth": "1985-06-15",
    "co_applicant": null
  },
  "vehicle": {
    "vin": "1HGBH41JXMN109186",
    "year": 2020,
    "make": "Honda",
    "model": "Civic",
    "mileage": 35000,
    "loan_amount": 25000,
    "vehicle_value": null,
    "condition": "good"
  },
  "preferred_bureau": "experian",
  "use_cache": true
}
```


Response:

```
json
```

```
{
  "success": true,
  "data": {
    "approval_score": 0.725,
    "risk_tier": "B",
    "matched_lenders": [
      "Prime Auto Lender",
      "Regional Credit Union"
    ],
    "is_complete": true,
    "missing_reason": null,
    "fico_score": 720,
    "ltv": 1.28,
    "dti": 0.30,
    "is_approved": true,
    "metadata": {
      "bureau_used": "experian",
      "score_factors": {
        "fico": {
          "value": 720,
          "score": 0.764,
          "weight": 0.4
        },
        "dti": {
          "value": 0.30,
          "score": 0.8,
          "weight": 0.25
        },
        "ltv": {
          "value": 1.28,
          "score": 0.4,
          "weight": 0.20
        },
        "employment": {
          "value": "W2",
          "score": 1.0,
          "weight": 0.10
        },
        "credit_history": {
          "value": 8,
          "score": 1.0,
          "weight": 0.05
        }
```

```
        },
        "processing_time": 2.456,
        "from_cache": false
      }
    }
}
```


Modern Valuation

Modern valuation endpoint with provider selection.

Endpoint: POST /api/automation/valuation

Request Body:


```
json

{
  "vin": "1HGBH41JXMN109186",
  "mileage": 35000,
  "zip_code": "78701",
  "condition": "good",
  "provider": "nada"
}
```


Response:


```
json
```

```
{
  "success": true,
  "data": {
    "value": 19500,
    "trade_in": 17800,
    "loan_value": 17800,
    "source": "NADA",
    "success": true,
    "vehicle_details": {
      "year": 2020,
      "make": "Honda",
      "model": "Civic",
      "trim": "LX Sedan 4D",
      "msrp": 21050,
      "curb_weight": 2771,
      "doors": 4,
      "drive_train": "FWD",
      "transmission": "CVT",
      "engine_config": "2.0L L4-16V (Port)"
    }
  }
}
```


Direct Service Access

For custom integrations, you can access services directly through the DI container.

Using AutomationService

php

```php
<?php
require_once 'vendor/autoload.php';
require_once 'lib/logger.php';
require_once 'lib/cache.php';
require_once 'lib/curl.php';

// Load environment
if (file_exists('.env')) {
    $lines = file('.env', FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
    foreach ($lines as $line) {
        if (strpos($line, '=') !== false && !str_starts_with(trim($line), '#')) {
            [$key, $value] = explode('=', $line, 2);
            $_ENV[trim($key)] = trim($value, '"\'');
        }
    }
}

// Build container
$containerBuilder = new DI\ContainerBuilder();
$containerBuilder->addDefinitions(require 'config/di.php');
$container = $containerBuilder->build();

// Get AutomationService
$automationService = $container->get(WF\API\Automation\AutomationService::class);

// Prepare request data
$requestData = [
    'applicant' => [
        'monthly_income' => 5000,
        'employment_type' => 'W2',
        'state' => 'TX',
        'first_name' => 'John',
        'last_name' => 'Doe',
        'ssn' => '123-45-6789',
        'address' => '123 Main St',
        'city' => 'Austin',
        'zip_code' => '78701',
        'date_of_birth' => '1985-06-15'
    ],
    'vehicle' => [
        'vin' => '1HGBH41JXMN109186',
        'year' => 2020,
        'make' => 'Honda',
```

```php
        'model' => 'Civic',
        'mileage' => 35000,
        'loan_amount' => 25000
    ],
    'preferred_bureau' => 'experian'
];

// Process pre-qualification
try {
    $result = $automationService->processPreQual($requestData);

    echo "Approval Score: " . $result->approvalScore . "\n";
    echo "Risk Tier: " . $result->getRiskTier() . "\n";
    echo "FICO Score: " . $result->creditProfile->ficoScore . "\n";
    echo "Matched Lenders: " . implode(', ', $result->getMatchedLenders()) . "\n";

} catch (Exception $e) {
    echo "Error: " . $e->getMessage() . "\n";
}
```

Using Individual Services

Credit Bureau Pull

```
php
```

```php
// Get bureau factory
$bureauFactory = $container->get(WF\API\Automation\Factories\BureauClientFactory::class);
$parserFactory = $container->get(WF\API\Automation\Factories\CreditParserFactory::class);

// Create bureau client
$bureauClient = $bureauFactory->create('equifax');

// Prepare consumer data
$consumers = [
    [
        'firstName' => 'John',
        'lastName' => 'Doe',
        'ssn' => '123456789',
        'address' => '123 Main St',
        'city' => 'Austin',
        'state' => 'TX',
        'zip' => '78701',
        'dob' => '1985-06-15'
    ]
];

// Pull credit report
$rawResponse = $bureauClient->pullCreditReport($consumers);

// Parse response
$parser = $parserFactory->create('equifax');
$creditProfile = $parser->parse($rawResponse);

echo "FICO Score: " . $creditProfile->ficoScore . "\n";
echo "Open Accounts: " . $creditProfile->openTradeCount . "\n";
```

Vehicle Valuation

```
php
```

```php
// Get valuation factory
$valuationFactory = $container->get(WF\API\Automation\Factories\ValuationProviderFactory::class);

// Create NADA provider
$nadaProvider = $valuationFactory->create('nada');

// Get valuation
$valuation = $nadaProvider->getValuation(
    vin: '1HGBH41JXMN109186',
    mileage: 35000,
    zipCode: '78701',
    condition: 'good'
);

echo "Retail Value: $" . number_format($valuation['value'], 2) . "\n";
echo "Trade-In Value: $" . number_format($valuation['trade_in'], 2) . "\n";
```

Risk Scoring

php

```php
// Get risk scorer
$riskScorer = $container->get(WF\API\Automation\Contracts\RiskScorerInterface::class);

// Create models
$applicant = WF\API\Automation\Models\Applicant::fromArray([
    'monthly_income' => 5000,
    'monthly_debt' => 1500,
    'employment_type' => 'W2',
    'state' => 'TX',
    'first_name' => 'John',
    'last_name' => 'Doe',
    'ssn' => '123-45-6789',
    'address' => '123 Main St',
    'city' => 'Austin',
    'zip_code' => '78701',
    'date_of_birth' => '1985-06-15'
]);

$vehicle = WF\API\Automation\Models\Vehicle::fromArray([
    'vin' => '1HGBH41JXMN109186',
    'year' => 2020,
    'make' => 'Honda',
    'model' => 'Civic',
    'mileage' => 35000,
    'loan_amount' => 25000,
    'vehicle_value' => 19500
]);

// Calculate risk score
$score = $riskScorer->calculateScore($applicant, $vehicle, $creditProfile);
$tier = $riskScorer->getRiskTier($score);
$lenders = $riskScorer->matchLenders($applicant, $vehicle, $creditProfile);

echo "Risk Score: " . $score . "\n";
echo "Risk Tier: " . $tier . "\n";
echo "Matched Lenders: " . implode(', ', $lenders) . "\n";
```

Custom Integration Examples

Batch Processing

```
php
```

```php
// Process multiple applications
$applications = loadApplicationsFromDatabase();

foreach ($applications as $appData) {
    try {
        $result = $automationService->processPreQual($appData);

        // Save results
        saveResultToDatabase($appData['id'], $result);

        // Check if approved
        if ($result->isApproved()) {
            notifyApproval($appData['id'], $result->getMatchedLenders());
        }

    } catch (Exception $e) {
        logError($appData['id'], $e->getMessage());
    }
}
```

Caching Strategy

```php
// Get cache service
$cacheService = $container->get(WF\API\Automation\Services\BureauCacheService::class);

// Check cache before pulling
$cachedProfile = $cacheService->get($ssn, 'experian');

if ($cachedProfile !== null) {
    echo "Using cached profile\n";
    $creditProfile = $cachedProfile;
} else {
    // Pull fresh and cache
    $creditProfile = pullCreditReport($ssn);
    $cacheService->set($ssn, 'experian', $creditProfile, 1440); // 24 hour TTL
}
```

Custom Lender Rules

```php
```

```php
// Create custom lender repository
class CustomLenderRepository extends WF\API\Automation\Repositories\LenderRepository
{
    public function getActiveRules(): array
    {
        // Load from database
        $rules = $this->database->query("
            SELECT * FROM lender_rules
            WHERE active = 1
            AND effective_date <= NOW()
        ");

        return array_map(function($row) {
            return [
                'name' => $row['name'],
                'loan_types' => json_decode($row['loan_types']),
                'min_fico' => $row['min_fico'],
                'max_dti' => $row['max_dti'],
                'max_ltv' => $row['max_ltv'],
                'states' => json_decode($row['states']),
                'no_self_employed' => $row['no_self_employed']
            ];
        }, $rules);
    }
}

// Register in DI container
$container->set(
    WF\API\Automation\Repositories\LenderRepository::class,
    new CustomLenderRepository($database)
);
```

Response Formats

Success Response Structure

All API endpoints return a consistent response structure:

```
json
```

```
{
  "success": true,
  "data": { ... },
  "error": ""
}
```

Error Response Structure

json

```
{
  "success": false,
  "data": "",
  "error": "Detailed error message"
}
```

Common Error Messages

- `"Invalid JSON in request body"` - Malformed JSON
- `"Invalid SSN format"` - SSN validation failed
- `"VIN must be exactly 17 characters"` - Invalid VIN
- `"Bureau authentication failed"` - API credentials issue
- `"No valuation providers available"` - All providers offline
- `"Processing failed"` - Generic automation error

Error Handling

API Error Responses

The API returns appropriate HTTP status codes:

- `200 OK` - Successful request
- `400 Bad Request` - Invalid input data
- `401 Unauthorized` - Authentication required
- `404 Not Found` - Endpoint not found
- `500 Internal Server Error` - Server error

Exception Handling

php

```php
try {
    $result = $automationService->processPreQual($data);
} catch (WF\API\Automation\Exceptions\ValidationException $e) {
    // Handle validation errors
    echo "Validation Error: " . $e->getMessage();
} catch (WF\API\Automation\Exceptions\BureauApiException $e) {
    // Handle bureau API errors
    echo "Bureau Error: " . $e->getMessage();
} catch (WF\API\Automation\Exceptions\ValuationException $e) {
    // Handle valuation errors
    echo "Valuation Error: " . $e->getMessage();
} catch (Exception $e) {
    // Handle other errors
    echo "General Error: " . $e->getMessage();
}
```

Advanced Usage

Custom Bureau Configuration

```php
// Override bureau configuration at runtime
$container->set(WF\API\Automation\Clients\EquifaxClient::class,
    new WF\API\Automation\Clients\EquifaxClient([
        'client_id' => 'test_client_id',
        'client_secret' => 'test_secret',
        'token_endpoint' => 'https://sandbox.equifax.com/token',
        'report_endpoint' => 'https://sandbox.equifax.com/report',
        'member_number' => 'TEST123',
        'security_code' => 'TEST456'
    ])
);
```

Monitoring and Metrics

```php
```

```php
// Add custom monitoring
class MonitoredAutomationService extends WF\API\Automation\AutomationService
{
    private $metrics;

    public function processPreQual(array $requestData): PreQualResult
    {
        $start = microtime(true);

        try {
            $result = parent::processPreQual($requestData);

            // Record success metrics
            $this->metrics->increment('prequal.success');
            $this->metrics->timing('prequal.duration', microtime(true) - $start);
            $this->metrics->gauge('prequal.fico_score', $result->creditProfile->ficoScore);

            return $result;

        } catch (Exception $e) {
            // Record failure metrics
            $this->metrics->increment('prequal.failure');
            $this->metrics->increment('prequal.error.' . get_class($e));
            throw $e;
        }
    }
}
```

Async Processing

php

```php
// Queue application for async processing
$queue->push('prequal.process', [
    'application_id' => $applicationId,
    'request_data' => $requestData,
    'callback_url' => 'https://api.example.com/webhook/prequal'
]);

// Worker process
$queue->consume('prequal.process', function($job) use ($container) {
    $automationService = $container->get(AutomationService::class);

    try {
        $result = $automationService->processPreQual($job['request_data']);

        // Send callback
        postToWebhook($job['callback_url'], [
            'application_id' => $job['application_id'],
            'status' => 'completed',
            'result' => $result->toArray()
        ]);

    } catch (Exception $e) {
        // Handle error
        postToWebhook($job['callback_url'], [
            'application_id' => $job['application_id'],
            'status' => 'failed',
            'error' => $e->getMessage()
        ]);
    }
});
```

For more information, see the README.md or contact the development team.